



---

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**  
**MB90460/465 Series**  
**HARDWARE MANUAL**



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

## 16-BIT MICROCONTROLLER

# MB90460/465 Series

# HARDWARE MANUAL

The information for microcontroller supports is shown in the following homepage.  
Be sure to refer to the "Check Sheet" for the latest cautions on development.

**"Check Sheet" is seen at the following support page**

"Check Sheet" lists the minimal requirement items to be checked to prevent problems beforehand in system development.  
<http://edevic.fujitsu.com/micom/en-support/>

**FUJITSU MICROELECTRONICS LIMITED**



# PREFACE

## ■ Objectives and intended reader

Thank you for purchasing Fujitsu Microelectronics products.

The MB90460/465 series was developed as a group of general-purpose models in the F<sup>2</sup>MC-16LX Family, which is a family of original 16-bit single-chip microcontrollers that can be used for application specific ICs (ASICs).

This manual is intended for engineers who design products using the MB90460/465 series of microcontrollers. The manual describes the functions and operation of the MB90460/465 series.

Note: F<sup>2</sup>MC is a registered of Fujitsu Microelectronics Limited and stands for FUJITSU Flexible Microcontroller.

## ■ Trademarks

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

## ■ Organization of this manual

This manual consists of the following 24 chapters and 1 appendix:

### **Chapter 1 Overview**

This chapter describes the features and basic specifications of the MB90460/465 series.

### **Chapter 2 Notes on Handling Devices**

This chapter describes notes on Handling Devices.

### **Chapter 3 CPU**

This chapter describes the memory space of the MB90460/465 series.

### **Chapter 4 Reset**

This chapter describes the reset function of the MB90460/465 series.

### **Chapter 5 Clock**

This chapter describes the clocks of the MB90460/465 series.

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

This chapter describes the energy-saving mode of the MB90460/465 series.

### **Chapter 7 Interrupt**

This chapter describes the interrupts and extended intelligent I/O services of the MB90460/465 series.

### **Chapter 8 Mode Setting**

This chapter describes the operating modes and memory access mode of the MB90460/465 series.

### **Chapter 9 I/O Port**

This chapter describes the functions and operation of the MB90460/465 series I/O ports.

### **Chapter 10 Time-base Timer**

This chapter describes the functions and operation of the MB90460/465 series time-base timer.

### **Chapter 11 Watchdog Timer**

This chapter describes the functions and operation of the MB90460/465 series watchdog timer.

## **Chapter 12 16-bit Reload Timer**

This chapter describes the functions and operation of the MB90460/465 series 16-bit reload timer.

## **Chapter 13 16-bit PPG Timer**

This chapter describes the functions and operation of the MB90460/465 series 16-bit PPG timer.

## **Chapter 14 Multi-functional Timer**

This chapter describes the functions and operation of the MB90460/465 series multi-functional timer.

## **Chapter 15 Multi-pulse Generator**

This chapter describes the functions and operation of the MB90460/465 series multi-pulse generator.

## **Chapter 16 PWC Timer**

This chapter describes the functions and operation of the MB90460/465 series PWC timer.

## **Chapter 17 UART**

This chapter describes the functions and operation of the MB90460/465 series UART.

## **Chapter 18 DTP/External Interrupt Circuit**

This chapter describes the functions and operation of the MB90460/465 series DTP/external interrupt circuit.

## **Chapter 19 Delayed Interrupt Generator Module**

This chapter describes the functions and operation of the MB90460/465 series delayed interrupt generator module.

## **Chapter 20 8/10-bit A/D Converter**

This chapter describes the functions and operation of the MB90460/465 series 8/10-bit A/D Converter.

## **Chapter 21 ROM Correction Function**

This chapter describes the functions and operation of the MB90460/465 series ROM correction function.

## **Chapter 22 ROM Mirroring Function Selection Module**

This chapter describes the functions and operation of the MB90460/465 series ROM mirroring function selection module.

## **Chapter 23 512K / 1024K bit Flash Memory**

This chapter describes the functions and operation of the MB90460/465 series 512K / 1024K bit flash memory.

## **Chapter 24 EXAMPLE OF F<sup>2</sup>MC-16LX MB90F462/F462A/F463A CONNECTION FOR SERIAL WRITING**

This chapter describes examples of F<sup>2</sup>MC-16LX MB90F462/F462A/F463A connections for serial writing.

## **Appendix**

### **Appendix A I/O Map**

The appendix A contains an I/O map and instruction overview.

### **Appendix B Instructions**

The appendix B contains an instruction overview.

- 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	MB90460/465 Series Features .....	2
1.2	MB90460/465 Series Product line-up .....	5
1.3	Block Diagram of MB90460/465 Series .....	7
1.4	Pin Assignment .....	8
1.5	Package Dimensions .....	11
1.6	I/O Pins and Pin Functions .....	14
1.7	I/O Circuit Types .....	19
<b>CHAPTER 2</b>	<b>NOTES ON HANDLING DEVICES .....</b>	<b>23</b>
2.1	Notes on Handling Devices .....	24
<b>CHAPTER 3</b>	<b>CPU .....</b>	<b>27</b>
3.1	CPU .....	28
3.2	Memory Space .....	29
3.3	Memory Maps .....	31
3.4	Addressing .....	33
3.4.1	Address Specification by Linear Addressing .....	34
3.4.2	Address Specification by Bank Addressing .....	35
3.5	Memory Location of Multi-byte Data .....	37
3.6	Registers .....	39
3.7	Dedicated Registers .....	40
3.7.1	Accumulator (A) .....	42
3.7.2	Stack Pointers (USP, SSP) .....	45
3.7.3	Processor Status (PS) .....	47
3.7.4	Condition Code Register (PS: CCR) .....	48
3.7.5	Register Bank Pointer (PS: RP) .....	50
3.7.6	Interrupt Level Mask Register (PS: ILM) .....	51
3.7.7	Program Counter (PC) .....	52
3.7.8	Direct Page Register (DPR) .....	53
3.7.9	Bank Registers (PCB, DTB, USB, SSB, ADB) .....	54
3.8	General-purpose Registers .....	55
3.9	Prefix Codes .....	57
3.9.1	Bank Select Prefix (PCB, DTB, ADB, SPB) .....	58
3.9.2	Common Register Bank Prefix (CMR) .....	60
3.9.3	Flag Change Suppression Prefix (NCC) .....	61
3.9.4	Restrictions on Prefix Codes .....	62
<b>CHAPTER 4</b>	<b>RESET .....</b>	<b>65</b>
4.1	Reset .....	66
4.2	Reset Causes and Oscillation Stabilization Wait Intervals .....	68
4.3	External Reset Pin .....	69
4.4	Reset Operation .....	71

4.5	Reset Cause Bits .....	73
4.6	Status of Pins in a Reset .....	75
<b>CHAPTER 5 CLOCK .....</b>		<b>77</b>
5.1	Clock .....	78
5.2	Block Diagram of the Clock Generation Block .....	80
5.3	Clock Selection Register (CKSCR) .....	82
5.4	Clock Mode .....	84
5.5	Oscillation Stabilization Wait Interval .....	86
5.6	Connection of an Oscillator or an External Clock to the Microcontroller .....	87
<b>CHAPTER 6 LOW POWER CONSUMPTION MODE .....</b>		<b>89</b>
6.1	Low Power Consumption Mode .....	90
6.2	Block Diagram of the Low Power Consumption Control Circuit .....	92
6.3	Low Power Consumption Mode Control Register (LPMCR) .....	94
6.4	CPU Intermittent Operation Mode .....	97
6.5	Standby Mode .....	98
6.5.1	Sleep Mode .....	99
6.5.2	Time-base Timer Mode .....	102
6.5.3	Stop Mode .....	104
6.6	State Change Diagram .....	106
6.7	State of Pins in Standby Mode and during Reset .....	109
6.8	Usage Notes on Low Power Consumption Mode .....	110
<b>CHAPTER 7 INTERRUPT .....</b>		<b>113</b>
7.1	Interrupt .....	114
7.2	Interrupt Causes and Interrupt Vectors .....	116
7.3	Interrupt Control Registers and Peripheral Functions .....	119
7.3.1	Interrupt Control Registers (ICR00 to ICR15) .....	121
7.3.2	Interrupt Control Register Functions .....	123
7.4	Hardware Interrupt .....	126
7.4.1	Operation of Hardware Interrupt .....	129
7.4.2	Processing for Interrupt Operation .....	131
7.4.3	Procedure for using Hardware Interrupt .....	132
7.4.4	Multiple Interrupts .....	133
7.4.5	Hardware Interrupt Processing Time .....	135
7.5	Software Interrupt .....	137
7.6	Interrupt of Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	139
7.6.1	Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	141
7.6.2	Registers of EI <sup>2</sup> OS Descriptor (ISD) .....	142
7.6.3	Operation of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	145
7.6.4	Procedure for using the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	146
7.6.5	Processing Time of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	147
7.7	Exception Processing Interrupt .....	149
7.8	Stack Operations for Interrupt Processing .....	150
7.9	Sample Programs for Interrupt Processing .....	152

<b>CHAPTER 8</b>	<b>MODE SETTING .....</b>	<b>157</b>
8.1	Mode Setting .....	158
8.2	Mode Pins (MD2 to MD0) .....	159
8.3	Mode Data .....	160
<b>CHAPTER 9</b>	<b>I/O PORT .....</b>	<b>163</b>
9.1	Overview of I/O Port .....	164
9.2	Registers of I/O Port .....	166
9.3	Port 0 .....	167
9.3.1	Port 0 Registers (PDR0, DDR0 and RDR0) .....	169
9.3.2	Operation of Port 0 .....	171
9.4	Port 1 .....	173
9.4.1	Port 1 Registers (PDR1, DDR1 and RDR1) .....	175
9.4.2	Operation of Port 1 .....	176
9.5	Port 2 .....	178
9.5.1	Port 2 Registers (PDR2 and DDR2) .....	180
9.5.2	Operation of Port 2 .....	181
9.6	Port 3 .....	183
9.6.1	Port 3 Registers (PDR3 and DDR3) .....	185
9.6.2	Operation of Port 3 .....	186
9.7	Port 4 .....	188
9.7.1	Port 4 Registers (PDR4 and DDR4) .....	190
9.7.2	Operation of Port 4 .....	191
9.8	Port 5 .....	193
9.8.1	Port 5 Registers (PDR5, DDR5 and ADER) .....	195
9.8.2	Operation of Port 5 .....	196
9.9	Port 6 .....	198
9.9.1	Port 6 Registers (PDR6 and DDR6) .....	200
9.9.2	Operation of Port 6 .....	201
9.10	Sample I/O Port Program .....	203
<b>CHAPTER 10</b>	<b>TIME-BASE TIMER .....</b>	<b>205</b>
10.1	Overview of the Time-base Timer .....	206
10.2	Configuration of the Time-base Timer .....	208
10.3	Time-base Timer Control Register (TBTC) .....	209
10.4	Time-base Timer Interrupts .....	211
10.5	Operation of the Time-base Timer .....	212
10.6	Usage Notes on the Time-base Timer .....	214
10.7	Sample Program for the Time-base Timer Program .....	216
<b>CHAPTER 11</b>	<b>WATCHDOG TIMER .....</b>	<b>219</b>
11.1	Overview of the Watchdog Timer .....	220
11.2	Configuration of the Watchdog Timer .....	221
11.3	Watchdog Timer Control Register (WDTC) .....	222
11.4	Operation of the Watchdog Timer .....	224
11.5	Usage Notes on the Watchdog Timer .....	226
11.6	Sample Program for the Watchdog Timer .....	227

<b>CHAPTER 12 16-BIT RELOAD TIMER .....</b>	<b>229</b>
12.1 Overview of the 16-bit Reload Timer .....	230
12.2 Block Diagram of the 16-bit Reload Timer .....	233
12.3 16-bit Reload Timer Pins .....	235
12.4 16-bit Reload Timer Registers .....	236
12.4.1 Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1) .....	237
12.4.2 Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1) .....	239
12.4.3 16-bit Timer Register (TMR0/TMR1) .....	241
12.4.4 16-bit Reload Register (TMRD0/TMRD1) .....	242
12.5 16-Bit Reload Timer Interrupts .....	243
12.6 Operation of the 16-bit Reload Timer .....	244
12.6.1 Internal Clock Mode (reload mode) .....	246
12.6.2 Internal Clock Mode (single-shot mode) .....	248
12.6.3 Event Count Mode .....	250
12.7 Usage Notes on the 16-bit Reload Timer .....	252
12.8 Sample Programs for the 16-bit Reload Timer .....	253
 <b>CHAPTER 13 16-BIT PPG TIMER .....</b>	 <b>257</b>
13.1 Overview of 16-bit PPG Timer .....	258
13.2 Block Diagram of 16-bit PPG Timer .....	259
13.3 16-bit PPG Timer Pins .....	260
13.4 16-bit PPG Timer Registers .....	262
13.4.1 PPG Down Counter Register (PDCR0 to PDCR2) .....	264
13.4.2 PPG Period Setting Buffer Register (PCSR0 to PCSR2) .....	265
13.4.3 PPG Duty Setting Buffer Register (PDUT0 to PDUT2) .....	266
13.4.4 PPG Control Status Register (PCNTL0 to PCNTL2, PCNTH0 to PCNTH2) .....	267
13.5 16-bit PPG Timer Interrupts .....	271
13.6 Operation of 16-bit PPG Timer .....	273
13.7 Usage Notes on the 16-bit PPG Timer .....	276
13.8 Sample Programs for the 16-bit PPG Timer .....	277
 <b>CHAPTER 14 MULTI-FUNCTIONAL TIMER .....</b>	 <b>279</b>
14.1 Overview of Multi-functional Timer .....	280
14.2 Block Diagram of Multi-functional Timer .....	282
14.3 Multi-functional Timer Pins .....	286
14.4 Registers of Multi-functional Timer .....	289
14.4.1 Compare Clear Buffer Register (CPCLRB) and Compare Clear Register (CPCLR) .....	293
14.4.2 Timer Data Register (TCDT) .....	294
14.4.3 Timer Control Status Register (TCCSH, TCCSL) .....	295
14.4.4 Output Compare Buffer Registers (OCCPB0 to OCCPB5) / Output Compare Registers (OCCP0 to OCCP5) .....	299
14.4.5 Compare Control Registers (OCS0 to OCS5) .....	301
14.4.6 Input Capture Register (IPCP0 to IPCP3) .....	305
14.4.7 Input Capture Control Status Registers (ICS23, PICS01) .....	306
14.4.8 16-bit Timer Register (TMRR0/TMRR1/TMRR2) .....	313
14.4.9 16-bit Timer Control Register (DTCR0/DTCR1/DTCR2) .....	314
14.4.10 Waveform Control Register (SIGCR) .....	318

14.5	Multi-functional Timer Interrupts .....	320
14.6	Operation of Multi-functional Timer .....	324
14.6.1	Operation of 16-bit free-run timer .....	325
14.6.2	Operation of 16-bit Output Compare .....	332
14.6.3	Operation of 16-bit Input Capture .....	337
14.6.4	Operation of Waveform Generator .....	339
14.7	Usage Notes on the Multi-functional Timer .....	349
14.8	Sample Programs for the Multi-functional Timer .....	351
<b>CHAPTER 15</b>	<b>MULTI-PULSE GENERATOR .....</b>	<b>355</b>
15.1	Overview of Multi-pulse Generator .....	356
15.2	Block Diagram of Multi-pulse Generator .....	359
15.3	Multi-pulse Generator Pins .....	367
15.4	Registers of Multi-pulse Generator .....	369
15.4.1	Output Control Register (OPCR) .....	372
15.4.2	Output Data Register (OPDR) .....	376
15.4.3	Output Data Buffer Register (OPDBR) .....	380
15.4.4	Input Control Register (IPCR) .....	384
15.4.5	Compare Clear Register (CPCR) .....	388
15.4.6	Timer Buffer Register (TMBR) .....	389
15.4.7	Timer Control Status Register (TCSR) .....	390
15.4.8	Noise Cancellation Control Register (NCCR) .....	392
15.5	Multi-pulse Generator Interrupts .....	394
15.6	Operation of Multi-pulse Generator .....	397
15.6.1	Operation of Position Detection .....	399
15.6.2	Operation of Data Write Control Unit .....	401
15.6.3	Operation of Output Data Buffer Register .....	405
15.6.4	Operation of Data Transfer of Output Data Register .....	407
15.6.5	Operation of DTT11 Input Control .....	421
15.6.6	Operation of Noise Cancellation Function .....	424
15.6.7	Operation of 16-bit Timer .....	425
15.7	Usage Notes on the Multi-pulse Generator .....	429
15.8	Sample Programs for the Multi-pulse Generator .....	431
<b>CHAPTER 16</b>	<b>PWC Timer .....</b>	<b>433</b>
16.1	Overview of the PWC Timer .....	434
16.2	Block Diagram of the PWC Timer .....	435
16.3	PWC Timer Pins .....	436
16.4	PWC Timer Registers .....	438
16.4.1	PWC Control Status Register (PWCSH0/PWCSH1, PWCSL0/PWCSL1) .....	439
16.4.2	PWC Data Buffer Register (PWC0/PWC1) .....	443
16.4.3	Division Rate Control Register (DIV0/DIV1) .....	444
16.5	PWC Timer Interrupts .....	445
16.6	Operation of the PWC Timer .....	447
16.6.1	Operation Mode Selection .....	450
16.6.2	Starting and Stopping the Timer and Pulse-width Measurement and Clearing the Timer .....	451
16.6.3	Timer Mode Operation .....	453

16.6.4	Pulse Width Measurement Mode Operation .....	456
16.7	Usage Notes on the PWC Timer .....	461
16.8	Sample Programs for the PWC Timer .....	464
<b>CHAPTER 17</b>	<b>UART .....</b>	<b>467</b>
17.1	Overview of UART .....	468
17.2	Block Diagram of UART .....	470
17.3	UART Pins .....	473
17.4	UART Registers .....	475
17.4.1	Serial Control Register (SCR0/SCR1) .....	476
17.4.2	Serial Mode Register (SMR0/SMR1) .....	478
17.4.3	Serial Status Register (SSR0/SSR1) .....	480
17.4.4	Input Data Register (SIDR0/SIDR1) and Output Data Register (SOR0/SOR1) .....	482
17.4.5	Communication Prescaler Control Register (CDCR) .....	484
17.5	UART Interrupts .....	486
17.5.1	Reception Interrupt Generation and Flag Set Timing .....	488
17.5.2	Transmission Interrupt Generation and Flag Set Timing .....	489
17.6	UART Baud Rates .....	490
17.6.1	Baud Rates Determined Using the Dedicated Baud Rate Generator .....	492
17.6.2	Baud Rates Determined Using the Internal Timer (16-bit Reload Timer 0) .....	495
17.6.3	Baud Rates Determined Using the External Clock .....	497
17.7	Operation of UART .....	498
17.7.1	Operation in Asynchronous Mode (Operation Modes 0 and 1) .....	500
17.7.2	Operation in Synchronous Mode (Operation Mode 2) .....	502
17.7.3	Bidirectional Communication Function (Normal Mode) .....	504
17.7.4	Master-slave Communication Function (Multiprocessor Mode) .....	506
17.8	Usage Notes on UART .....	509
17.9	Sample Program for UART .....	510
<b>CHAPTER 18</b>	<b>DTP/EXTERNAL INTERRUPT CIRCUIT .....</b>	<b>513</b>
18.1	Overview of the DTP/External Interrupt Circuit .....	514
18.2	Block Diagram of the DTP/External Interrupt Circuit .....	516
18.3	DTP/External Interrupt Circuit Pins .....	518
18.4	DTP/External Interrupt Circuit Registers .....	520
18.4.1	DTP/interrupt Cause Register (EIRR) .....	521
18.4.2	DTP/interrupt Enable Register (ENIR) .....	522
18.4.3	Request Level Setting Register (ELVR) .....	524
18.5	Operation of the DTP/External Interrupt Circuit .....	525
18.5.1	External Interrupt Function .....	528
18.5.2	DTP Function .....	529
18.6	Usage Notes on the DTP/External Interrupt Circuit .....	530
18.7	Sample Programs for the DTP/External Interrupt Circuit .....	532
<b>CHAPTER 19</b>	<b>DELAYED INTERRUPT GENERATOR MODULE .....</b>	<b>535</b>
19.1	Overview of the Delayed Interrupt Generator Module .....	536
19.2	Delayed Interrupt Generator Module Register .....	537
19.3	Operation of the Delayed Interrupt Generator Module .....	538

19.4	Usage Notes on the Delayed Interrupt Generator Module .....	539
<b>CHAPTER 20</b>	<b>8/10-BIT A/D CONVERTER .....</b>	<b>541</b>
20.1	Overview of the 8/10-bit A/D Converter .....	542
20.2	Block Diagram of the 8/10-bit A/D Converter .....	544
20.3	8/10-bit A/D Converter Pins .....	546
20.4	8/10-bit A/D Converter Registers .....	548
20.4.1	A/D Control Status Register 1 (ADCS1) .....	549
20.4.2	A/D Control Status Register 0 (ADCS0) .....	551
20.4.3	A/D Data Register (ADCR0/ADCR1) .....	554
20.5	8/10-bit A/D Converter Interrupts .....	556
20.6	Operation of the 8/10-bit A/D Converter .....	557
20.6.1	Conversion using EI <sup>2</sup> OS .....	560
20.6.2	A/D Conversion Data Protection Function .....	561
20.7	Usage Notes on the 8/10-bit A/D Converter .....	563
20.8	Sample Program 1 for the 8/10-bit A/D Converter (Single Conversion Mode Using EI <sup>2</sup> OS) .....	564
20.9	Sample Program 2 for the 8/10-bit A/D Converter (Continuous Conversion Mode Using EI <sup>2</sup> OS) ..	566
20.10	Sample Program 3 for the 8/10-bit A/D Converter (Stop Conversion Mode Using EI <sup>2</sup> OS) .....	568
<b>CHAPTER 21</b>	<b>ROM CORRECTION FUNCTION .....</b>	<b>571</b>
21.1	Overview of the ROM Correction Function .....	572
21.2	Block Diagram of ROM Correction Function .....	573
21.3	ROM Correction Function Registers .....	574
21.3.1	Program Address Detection Register (PADR0/PADR1) .....	575
21.3.2	Program Address Detection Control Status Register (PACSR) .....	576
21.4	Operation of the ROM Correction Function .....	578
21.5	Example of Using ROM Correction Function .....	579
<b>CHAPTER 22</b>	<b>ROM MIRRORING FUNCTION SELECTION MODULE .....</b>	<b>583</b>
22.1	Overview of the ROM Mirroring Function Selection Module .....	584
22.2	ROM Mirroring Function Selection Register (ROMM) .....	585
<b>CHAPTER 23</b>	<b>512K / 1024K BIT FLASH MEMORY .....</b>	<b>587</b>
23.1	Overview of the 512K / 1024K Bit Flash Memory .....	588
23.2	512K / 1024K Bit Flash Memory Sector Configuration .....	589
23.3	Flash Memory Control Status Register (FMCS) .....	590
23.4	Method of Starting the Automatic Algorithm in Flash Memory .....	592
23.5	Verifying Automatic Algorithm Execution Status .....	593
23.5.1	Data Polling Flag (DQ7) .....	595
23.5.2	Toggle Bit Flag (DQ6) .....	597
23.5.3	Time limit Exceeded Flag (DQ5) .....	598
23.5.4	Sector Deletion Timer Flag (DQ3) .....	599
23.6	Detailed Explanation on the Flash Memory Write/Delete .....	600
23.6.1	Setting the Read/Reset Status .....	601
23.6.2	Writing the Data .....	602
23.6.3	Deleting the Data (Chip Deletion) .....	604
23.6.4	Deleting the Data (Sector Deletion) .....	605



23.6.5	Temporarily Stopping the Sector Deletion .....	607
23.6.6	Restarting the Sector Deletion .....	608
23.7	Flash Security Feature .....	609
23.8	Programming Example of 512K Bit Flash Memory .....	610

## **CHAPTER 24 EXAMPLE OF F<sup>2</sup>MC-16LX MB90F462/F462A/F463A CONNECTION FOR SERIAL WRITING ..... 615**

24.1	Standard Configuration for Serial On-board Writing (Fujitsu Standard) .....	616
24.2	Example of Connection for Serial Writing (When Power Supplied by User) .....	618
24.3	Example of Connection for Serial Writing (When Power Supplied from Writer) .....	620
24.4	Example of Minimum Connection with Flash Microcontroller Programmer (When Power Supplied by User) .....	622
24.5	Example of Minimum Connection with Flash Microcontroller Programmer (When Power Supplied from Writer) .....	624

## **APPENDIX ..... 627**

APPENDIX A	I/O MAP .....	628
APPENDIX B	Instructions .....	635
B.1	Instruction Types .....	636
B.2	Addressing .....	637
B.3	Direct Addressing .....	639
B.4	Indirect Addressing .....	645
B.5	Execution Cycle Count .....	653
B.6	Effective address field .....	656
B.7	How to Read the Instruction List .....	657
B.8	F <sup>2</sup> MC-16LX Instruction List .....	660
B.9	Instruction Map .....	674

## **INDEX..... 697**

# Main changes in this edition

Page	Section	Change Results
	-	The product name is changed to "MB90460/465 series".
83	CHAPTER 5 CLOCK 5.3 Clock Selection Register (CKSCR)	The function column of bit 14 in Table 5.3-1 is changed. ("· Writing has no effect on the operation." is added.)
106	CHAPTER 6 LOW POWER CONSUMPTION MODE 6.6 State Change Diagram	Figure 6.6-1 is changed.
109	CHAPTER 6 LOW POWER CONSUMPTION MODE 6.7 State of Pins in Standby Mode and during Reset	"*3" under Table 6.7-1 is changed.
123	CHAPTER 7 INTERRUPT 7.3.2 Interrupt Control Register Functions	The summary is changed.
230	CHAPTER 12 16-BIT RELOAD TIMER 12.1 Overview of the 16-bit Reload Timer	"● External trigger operation" is changed.
253	CHAPTER 12 16-BIT RELOAD TIMER 12.8 Sample Programs for the 16-bit Reload Timer	"● Coding example" is changed.
298	CHAPTER 14 MULTI-FUNCTIONAL TIMER 14.4.3 Timer Control Status Register (TCCSH, TCCSL)	The function column of bit 3 in Table 14.4-2 is changed. ("· Even after "1" is written, the counter value is not initialized if "0" is written to this bit before the next count clock." is added.)
521	CHAPTER 18 DTP/EXTERNAL INTERRUPT CIRCUIT 18.4.1 DTP/interrupt Cause Register (EIRR)	"■ DTP/interrupt Cause Register (EIRR)" is changed. ("Notes:" is added.)
522	CHAPTER 18 DTP/EXTERNAL INTERRUPT CIRCUIT 18.4.2 DTP/interrupt Enable Register (ENIR)	"■ DTP/interrupt Enable Register (ENIR)" is changed. ("Note:" is added.)
525	CHAPTER 18 DTP/EXTERNAL INTERRUPT CIRCUIT 18.5 Operation of the DTP/External Interrupt Circuit	"■ Setting the DTP/external Interrupt Circuit" is changed. ("1. Set as an input port the general I/O port to be used also as a pin to input external interrupts." is added.)
530	CHAPTER 18 DTP/EXTERNAL INTERRUPT CIRCUIT 18.6 Usage Notes on the DTP/External Interrupt Circuit	"● Input polarities of external interrupts" is changed. ("If the request input level is level setting, the pulse width requires a longer period than the minimum pulse width stated on the data sheet. Also, as long as the interrupt input pin retains the active level, interrupt requests continue to be generated to the interrupt controller, even if the DTP/external interrupt cause register is cleared." is added.)
552	CHAPTER 20 8/10-BIT A/D CONVERTER 20.4.2 A/D Control Status Register 0 (ADCS0)	The function column of bit5 to bit3 in Table 20.4-2 is changed. ("Notes:" is added.)

Page	Section	Change Results
607	CHAPTER 23 512K / 1024K BIT FLASH MEMORY 23.6.5 Temporarily Stopping the Sector Deletion	"■ Temporarily Stopping the Sector Deletion" is changed. (· up to 15 $\mu$ s $\rightarrow$ up to 20 $\mu$ s · The sector deletion temporary stop command must be executed 20 $\mu$ s or more after the sector deletion command or sector deletion restart command is issued.)
616	CHAPTER 24 EXAMPLE OF F2MC-16LX MB90F462/ F462A/F463A CONNECTION FOR SERIAL WRITING 24.1 Standard Configuration for Serial On-board Writing (Fujitsu Standard)	Table 24.1-1 is changed.

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

# ***CHAPTER 1***

---

# ***OVERVIEW***

**This chapter describes the main features and basic specifications of the MB90460/465 series.**

- 1.1 MB90460/465 Series Features
- 1.2 MB90460/465 Series Product line-up
- 1.3 Block Diagram of MB90460/465 Series
- 1.4 Pin Assignment
- 1.5 Package Dimensions
- 1.6 I/O Pins and Pin Functions
- 1.7 I/O Circuit Types

## 1.1 MB90460/465 Series Features

---

The MB90460/465 series is a line of general-purpose, 16-bit microcontrollers designed for those applications which require high-speed real-time processing, proving to be suitable for various industrial machines and motor control (AC induction motor and brushless DC motor). These microcontrollers consist of a multi-functional timer for AC/DC motor control and a multi-pulse generator for DC motor control, which can generate various type of waveform.

The instruction set is designed to be optimized for controller applications which inheriting the AT architecture of F<sup>2</sup>MC-16LX family and allow a wide range of control tasks to be processed efficiently at high speed.

---

### ■ MB90460/465 Series Features

- Clock
  - Embedded PLL clock multiplication circuit
  - Operating clock (PLL clock) can selected from divided-by-2 of oscillation or one to four times the oscillation (at oscillation of 4 MHz, 4 MHz to 16 MHz)
  - Minimum instruction execution time of 62.5 ns (at oscillation of 4 MHz, four times the PLL clock, operation at V<sub>cc</sub> of 5.0 V)
- CPU addressing space of 16 Mbytes
  - Internal 24-bit addressing
- Instruction set optimized for controller applications
  - Rich data types (bit, byte, word, long word)
  - Rich addressing mode (23 types)
  - High code efficiency
  - Enhanced precision calculation realized by the 32-bit accumulator
- Instruction set designed for high level language (C) and multi-task operations
  - Adoption of system stack pointer
  - Enhanced pointer indirect instructions
  - Barrel shift instructions
- Program patch function (2 address pointer)
- Improved execution speed
  - 4-byte instruction queue

- Powerful interrupt function
  - Priority level programmable: 8 levels
  - 32 factors of stronger interrupt function
- Automatic data transmission function independent of CPU operation
  - Extended intelligent I/O service function (EI<sup>2</sup>OS)
  - Maximum 16 channels
- Low-power consumption (standby) mode
  - Sleep mode (mode in which CPU operating clock is stopped)
  - Time-base timer mode (mode in which other than oscillation and time-base timer are stopped)
  - Stop mode (mode in which oscillation is stopped)
  - CPU intermittent operation mode
- Package
  - LQFP-64 (FPT-64P-M09: 0.65 mm pitch)
  - QFP-64 (FPT-64P-M06: 1.00 mm pitch)
  - SDIP-64 (DIP-64P-M01: 1.78 mm pitch)
- Process
  - CMOS technology

## ■ Internal Peripheral Features

- I/O port
  - Maximum of 51 ports
- 18-bit time-base counter/watchdog timer: 1 channel
- Watchdog timer: 1 channel
- PWC: 2 channels (MB90460 series), 1 channel (MB90465 series)
- 16-bit reload timer: 1 channel
- 16-bit PPG timer: 1 channel
- Multi-functional timer (for AC/DC motor control): 1 channel
  - 16-bit free-run timer with up or up-down mode selection and buffer: 1 channel
  - 16-bit output compare with buffer: 6 channels
  - 16-bit input capture: 4 channels
  - 16-bit PPG timer: 1 channel

## CHAPTER 1 OVERVIEW

- Waveform generator (16-bit timer: 3 channels, 3-phase waveform or dead time)
- Multi-pulse generator (for DC motor control): 1 channel (not present in MB90465 series)
  - 16-bit reload timer: 1 channel (can be used individually in MB90465 series)
  - 16-bit PPG timer: 1 channel (not present in MB90465 series)
  - Waveform sequencer (includes 16-bit timer with buffer and compare clear function) (not present in MB90465 series)
- UART: 2 channels
  - With full-duplex double buffer (8-bit length)
  - Clock asynchronous or clock synchronized transmission (with start and stop bits) can be selectively used
- DTP/External interrupt circuit: 8 channels
  - A module for starting extended intelligent I/O service (EI<sup>2</sup>OS) and generating an external interrupt triggered by an external input
- Delayed interrupt generation module
  - Generates an interrupt request for switching tasks
- 8/10-bit A/D converter: 8 channels
  - Selectable 8/10-bit resolution

## 1.2 MB90460/465 Series Product line-up

The MB90460/465 series contains 6 different devices. Table 1.2-1 lists the product line-up.

### ■ MB90460/465 Series Product Line-up

Table 1.2-1 MB90460/465 Series Product Line-up (1/2)

Part number Parameter	MB90V460	MB90F462	MB90F462A	MB90F463A	MB90462	MB90467
Classification	—	Flash type ROM with security			Mask ROM	
ROM size	—	64K Bytes		128K Bytes	64K Bytes	
RAM size	8K Bytes	2K Bytes				
CPU function	Number of instruction: 351 Minimum execution time: 62.5 ns / 4 MHz (PLL x 4) Addressing mode: 23 Data bit length: 1, 8, 16 bits Maximum memory space: 16 MBytes					
I/O port	I/O port (CMOS): 51					
PWC	Pulse width counter timer: 2 channels					1 channel
UART	With full-duplex double buffer (8-bit length) Clock asynchronous or clock synchronized transmission (with start and stop bits) can be selectively used					
16-bit reload timer	Reload timer: 2 channels Reload mode, single-shot mode or event count mode selectable Can be worked with multi-pulse generator (MB90460 series only) or individually					
16-bit PPG timer	PPG timer: 3 channels					2 channels
	PWM mode or single-shot mode selectable Can be worked with multi-functional timer / multi-pulse generator (MB90460 series only) or individually					
Multi-functional timer (for AC/DC motor control)	16-bit free-run timer with up or up/down mode selection and buffer: 1 channel 16-bit output compare: 6 channels 16-bit input capture: 4 channels 16-bit PPG timer: 1 channel Waveform generator (16-bit timer: 3 channels, 3-phase waveform or dead time)					
Multi-pulse generator (for DC motor control)	16-bit PPG timer: 1 channel 16-bit reload timer operation (toggle output, one shot output selectable) Event counter function, 1 channel built-in Waveform sequencer (includes 16-bit timer with buffer and compare clear function)					Not present
8/10-bit A/D converter	8/10-bit resolution:8 channels Conversion time: Min. 6.13 μs (16 MHz internal clock)					
External interrupt	8 independent channels Selectable causes: Rising edge, falling edge, “L” level or “H” level					
Low-power consumption	Stop mode / Sleep mode / CPU intermittent operation mode					
Process	CMOS					



**Table 1.2-1 MB90460/465 Series Product Line-up (2/2)**

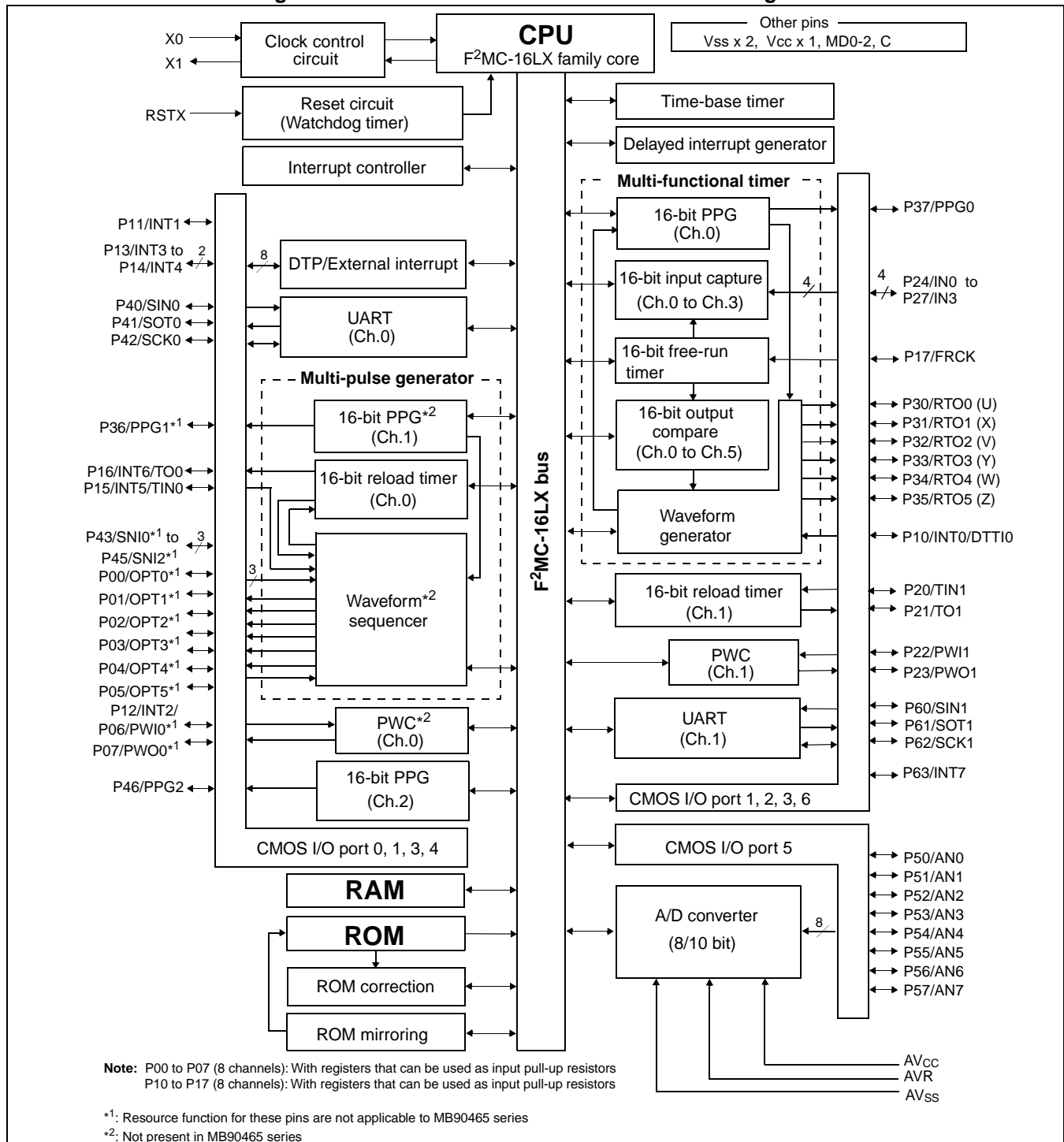
Part number Parameter	MB90V460	MB90F462	MB90F462A	MB90F463A	MB90462	MB90467
Package	PGA256	LQFP-64 (FPT-64P-M09: 0.65 mm pitch) QFP-64 (FPT-64P-M06: 1.00 mm pitch) SDIP-64 (DIP-64P-M01: 1.78 mm pitch)				
Operating voltage	5V ± 10% @16 MHz					

## 1.3 Block Diagram of MB90460/465 Series

Figure 1.3-1 shows a overall block diagram of the MB90460/465 series.

### ■ MB90460/465 Series Block Diagram

Figure 1.3-1 MB90460/465 Series Overall Block Diagram

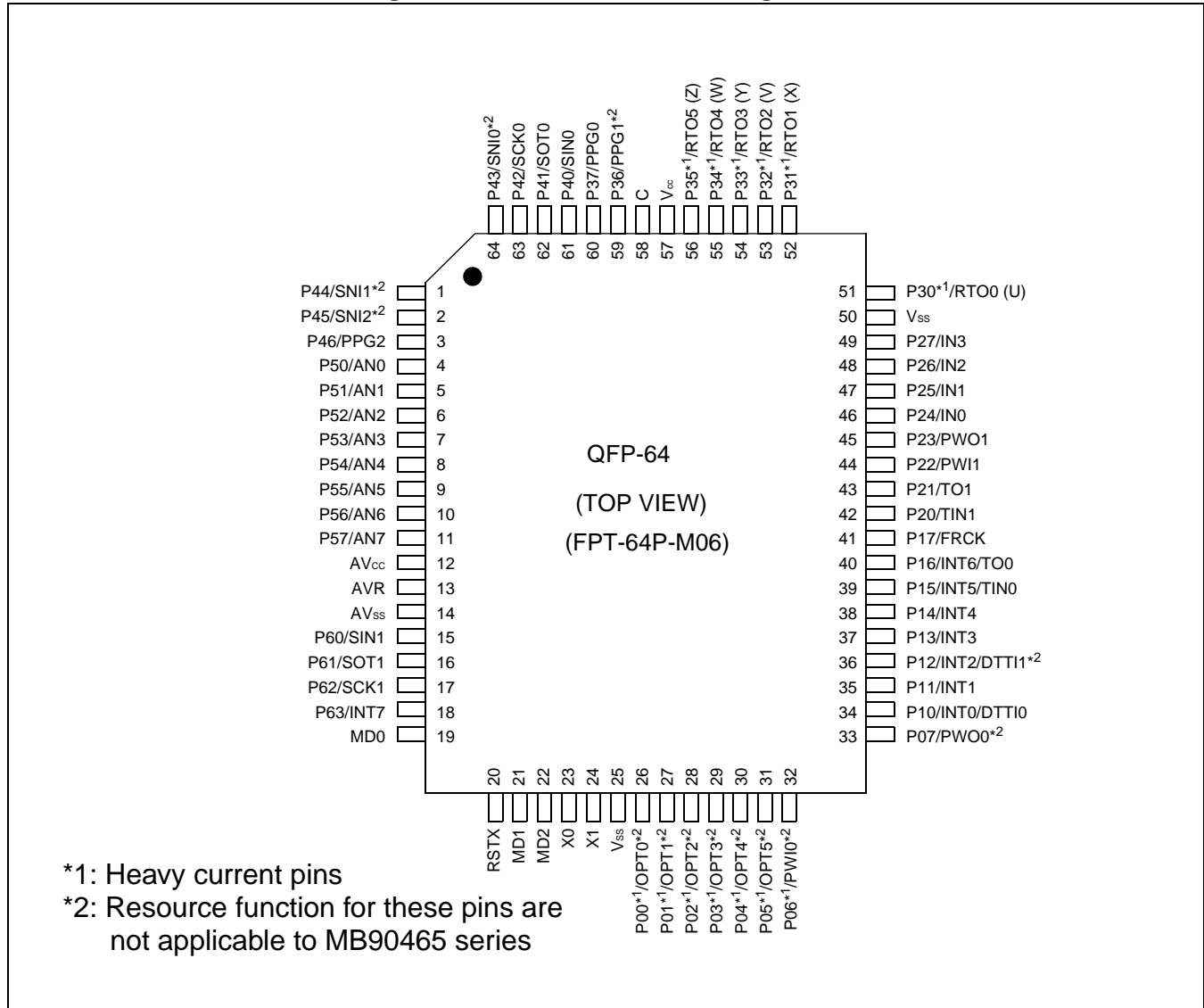


## 1.4 Pin Assignment

Figure 1.4-1 to Figure 1.4-3 show the pin assignment diagrams for the MB90460/465 series.

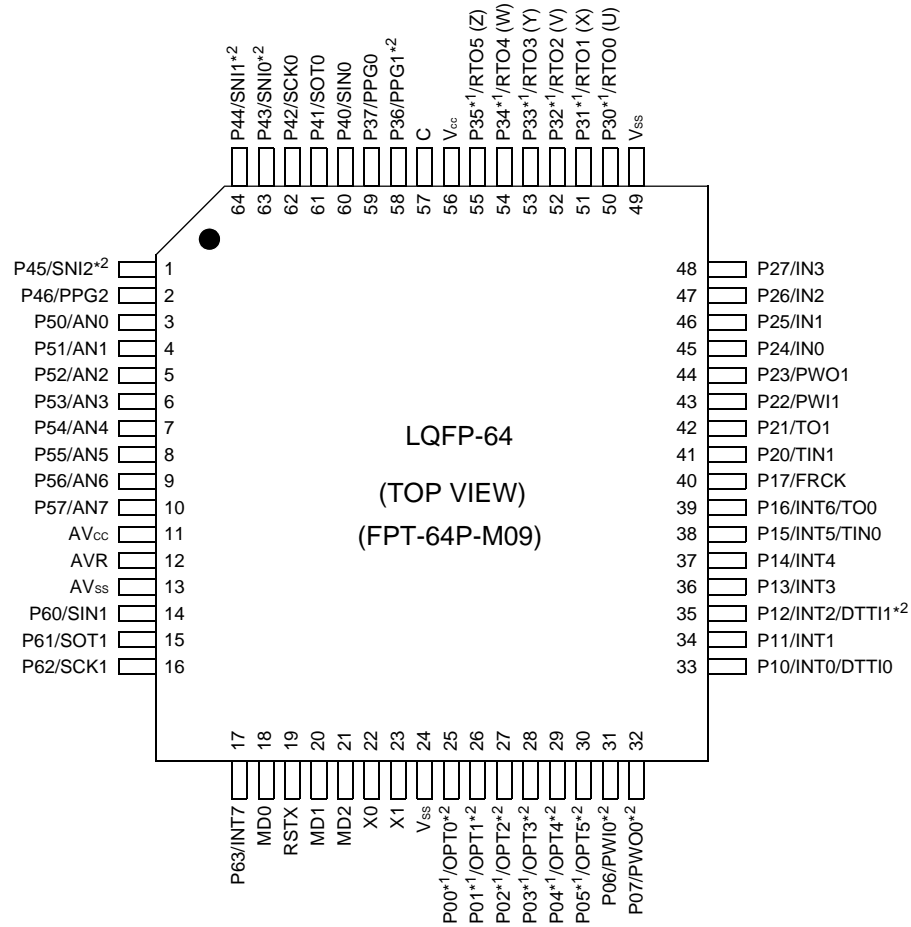
### ■ FPT-64P-M06 Pin Assignment

Figure 1.4-1 FPT-64P-M06 Pin Assignment



## ■ FPT-64P-M09 Pin Assignment

Figure 1.4-2 FPT-64P-M09 Pin Assignment

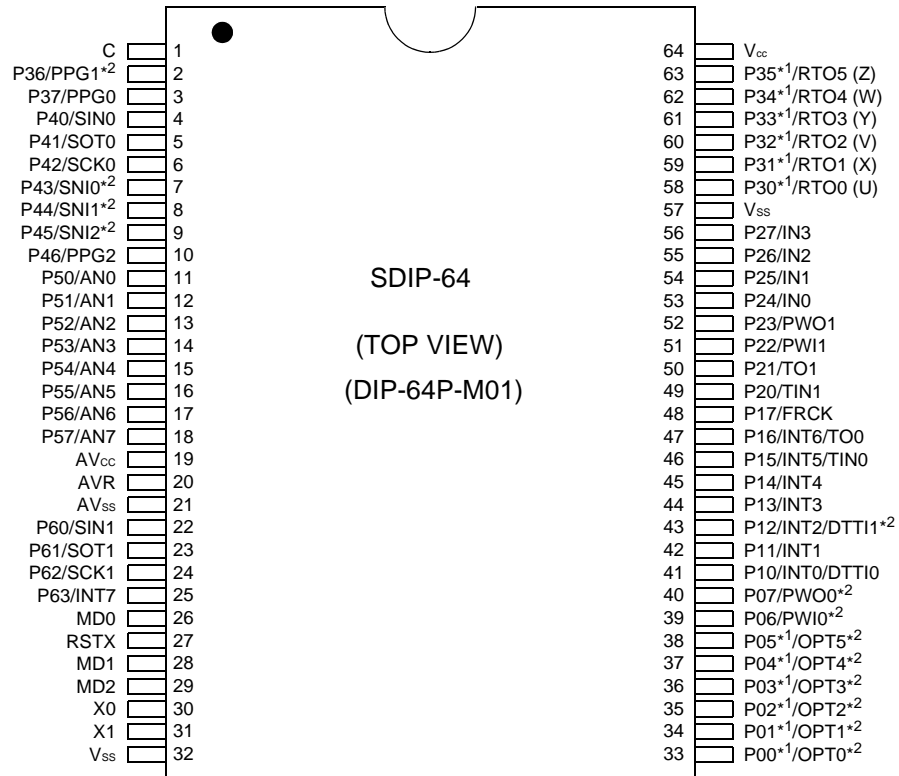


\*1: Heavy current pins

\*2: Resource function for these pins  
are not applicable to MB90465 series

## ■ DIP-64P-M01 Pin Assignment

Figure 1.4-3 DIP-64P-M01 Pin Assignment



\*1: Heavy current pins

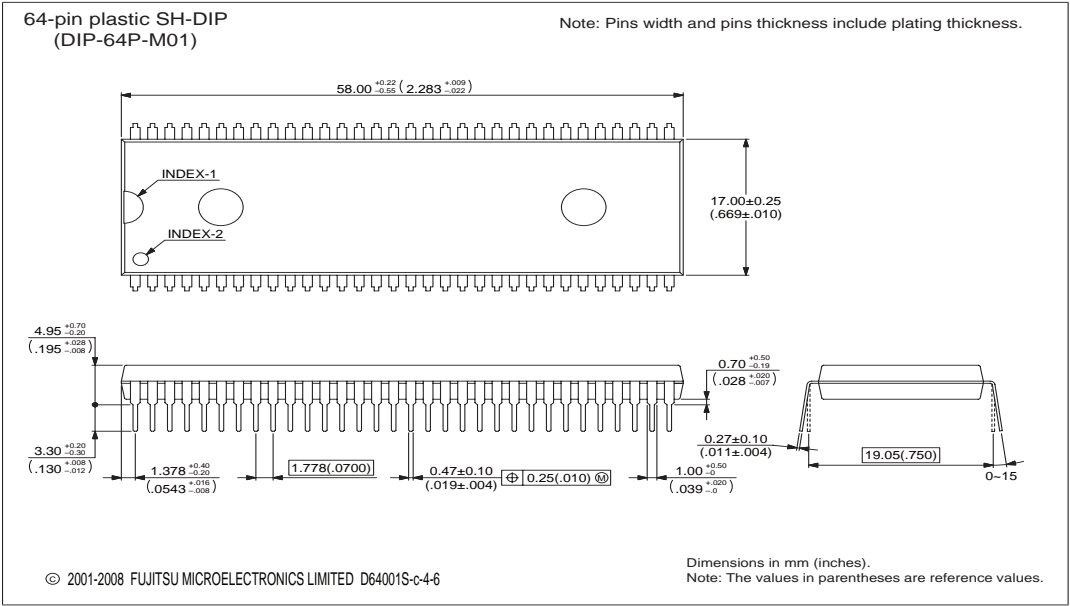
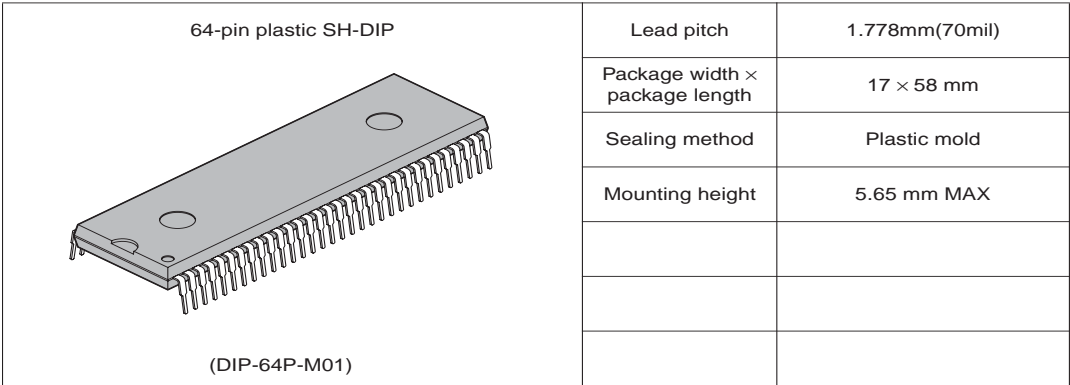
\*2: Resource function for these pins  
are not applicable to MB90465 series

# 1.5 Package Dimensions

Three types of packages are available for MB90460/465 series. Figure 1.5-1 to Figure 1.5-3 show the package dimensions.

## ■ DIP-64P-M01 Package Dimensions

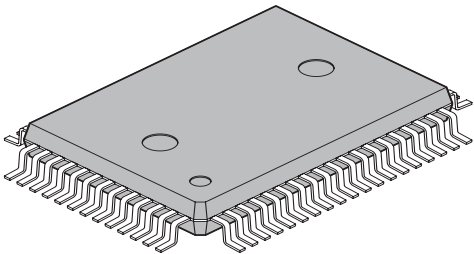
Figure 1.5-1 DIP-64P-M01 Package Dimensions

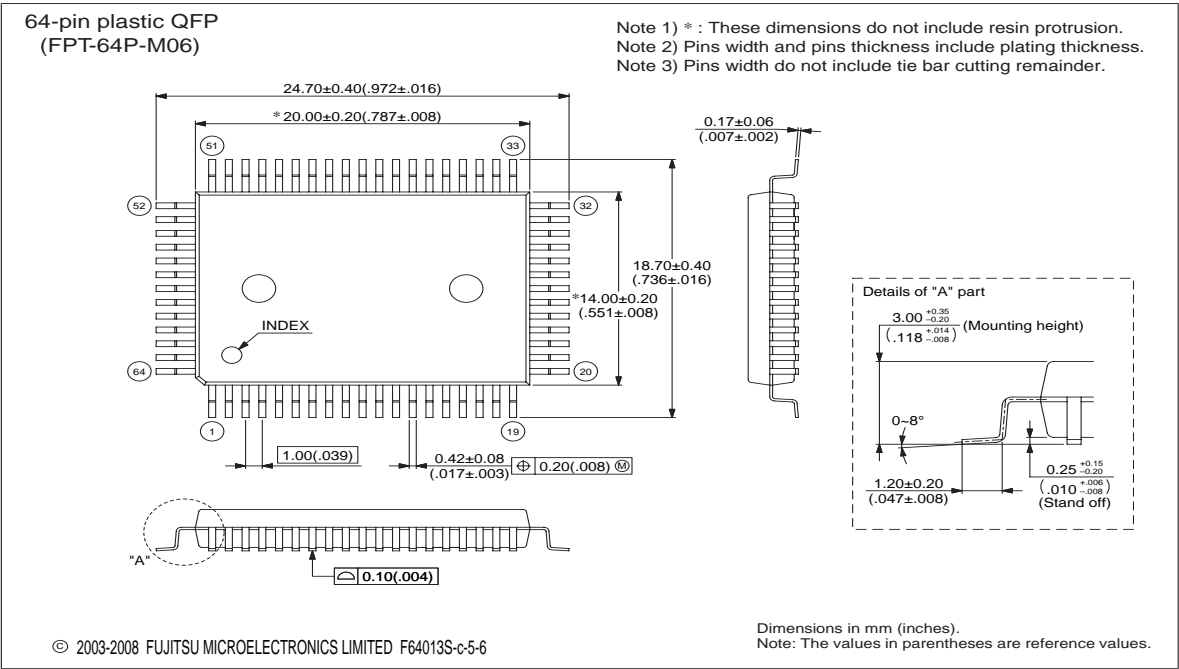


Please confirm the latest Package dimension by following URL.  
<http://edevic.fujitsu.com/package/en-search/>

■ FPT-64P-M06 Package Dimensions

Figure 1.5-2 FPT-64P-M06 Package Dimensions

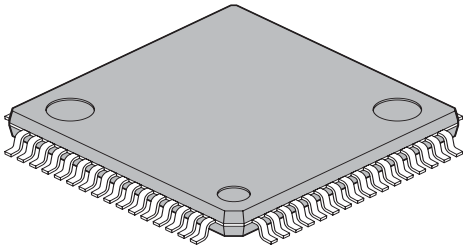
<div>64-pin plastic QFP</div>  <div>(FPT-64P-M06)</div>	Lead pitch	1.00 mm
	Package width × package length	14 × 20 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	3.35 mm MAX
	Code (Reference)	P-QFP64-14×20-1.00

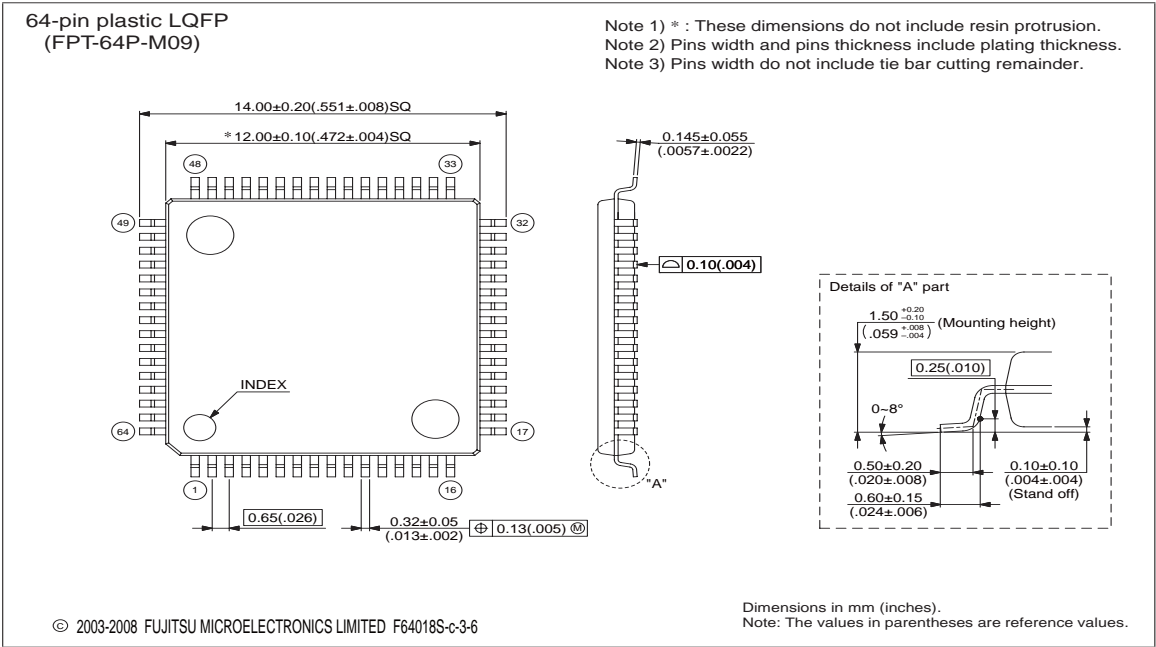


Please confirm the latest Package dimension by following URL.  
<http://edevic.fujitsu.com/package/en-search/>

■ FPT-64P-M09 Package Dimensions

Figure 1.5-3 FPT-64P-M09 Package Dimensions

<div>64-pin plastic LQFP</div>  <div>(FPT-64P-M09)</div>	Lead pitch	0.65 mm
	Package width × package length	12 × 12 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	1.70 mm MAX
	Code (Reference)	P-LQFP64-12×12-0.65



Please confirm the latest Package dimension by following URL.  
<http://edevic.fujitsu.com/package/en-search/>



## 1.6 I/O Pins and Pin Functions

Table 1.6-1 lists the MB90460/465 series I/O pins and their functions. Table 1.7-1 lists the I/O circuit types.

The letter in the "I/O circuit type" column in Table 1.6-1 refers to the letter in the "Type" column Table 1.7-1.

### ■ I/O Pins and Pin Functions

Table 1.6-1 Pin Description (1/5)

Pin no.			Pin name	I/O circuit	Pin status during reset	Function
QFP-M09*1	QFP-M06*2	SDIP*3				
22,23	23,24	30,31	X0,X1	A	Oscillating	Oscillation input pins.
19	20	27	RSTX	B	Reset input	External reset input pin.
25 to 30	26 to 31	33 to 38	P00 to P05	D	Port input	General-purpose I/O ports.
			OPT0 to OPT5*4			Output terminals OPT0 to OPT5 of the waveform sequencer. These pins output the waveforms specified at the output data buffer register of the waveform sequencer circuit. Output is generated when OPE0 to OPE5 of OPCR is enabled.
31	32	39	P06	E		General-purpose I/O ports.
			PWI0*4			PWC 0 signal input pin.
32	33	40	P07	E		General-purpose I/O ports.
			PWO0			PWC 0 signal output pin.
33	34 to 35	41 to 42	P10	C		General-purpose I/O ports.
			INT0			Can be used as interrupt request input channel 0. Input is enabled when "1" is set in EN0 in standby mode.
			DTTI0			RTO0 to RTO5 pins for fixed-level input. This function is enabled when the waveform generator enables its input bits.
34	35	42	P11	C		General-purpose I/O ports.
			INT1			Can be used as interrupt request input channel 1. Input is enabled when "1" is set in EN1 in standby mode.

Table 1.6-1 Pin Description (2/5)

Pin no.			Pin name	I/O circuit	Pin status during reset	Function
QFP-M09*1	QFP-M06*2	SDIP*3				
35	36	43	P12	C	Port input	General-purpose I/O ports.
			INT2			Can be used as interrupt request input channel 2. Input is enabled when "1" is set in EN2 in standby mode.
			DTTI1*4			OPT0 to OPT5 pins for fixed-level input. This function is enabled when the waveform sequencer enables its input bits.
36, 37	37, 38	44, 45	P13, P14	C		General-purpose I/O ports.
			INT3, INT4			Can be used as interrupt request input channels 3 to 4. Input is enabled when "1" is set in EN3 to EN4 in standby mode.
38	39	46	P15	C		General-purpose I/O ports.
			INT5			Can be used as interrupt request input channel 5. Input is enabled when "1" is set in EN5 in standby mode.
			TIN0			External clock input pin for reload timer 0.
39	40	47	P16	C		Port input
			INT6		Can be used as interrupt request input channel 6. Input is enabled when "1" is set in EN6 in standby mode.	
			TO0		Event output pin for reload timer 0.	
40	41	48	P17	C	General-purpose I/O ports.	
			FRCK		External clock input pin for free-run timer.	
41	42	49	P20	F	General-purpose I/O ports.	
			TIN1		External clock input pin for reload timer 1.	
42	43	50	P21	F	General-purpose I/O ports.	
			TO1		Event output pin for reload timer 1.	
43	44	51	P22	F	General-purpose I/O ports.	
			PWI1		PWC 1 signal input pin.	
44	45	52	P23	F	General-purpose I/O ports.	
			PWO1		PWC 1 signal output pin.	

**Table 1.6-1 Pin Description (3/5)**

Pin no.			Pin name	I/O circuit	Pin status during reset	Function	
QFP-M09*1	QFP-M06*2	SDIP*3					
45 to 48	46 to 49	53 to 56	P24 to P27	F	Port input	General-purpose I/O ports.	
			IN0 to IN3			Trigger input pins for input capture channels 0 to 3. When input capture channels 0 to 3 are used for input operation, these pins are enabled as required and must not be used for any other I/P.	
50 to 55	51 to 56	58 to 63	P30 to P35	G		General-purpose I/O ports.	
			RTO0 to RTO5			Waveform generator output pins. These pins output the waveforms specified at the waveform generator. Output is generated when waveform generator output is enabled.	
58,59	59, 60	2, 3	P36, 37	H		General-purpose I/O ports.	
			PPG1*4, PPG0			Output pins for PPG channels 1, 0. This function is enabled when PPG channels 1, 0 enable output.	
60	61	4	P40	F		General-purpose I/O ports.	
			SIN0			Serial data input pin for UART channel 0. While UART channel 0 is operating for input, the input of this pin is used as required and must not be used for any other input.	
61	62	5	P41	F		General-purpose I/O ports.	
			SOT0			Serial data output pin for UART channel 0. This function is enabled when UART channel 0 enables data output.	
62	63	6	P42	F		General-purpose I/O ports.	
			SCK0			Serial clock I/O pin for UART channel 0. This function is enabled when UART channel 0 enables clock output.	
63	64	7	P43	F		General-purpose I/O ports.	
			SNIO*4			Trigger input pins for position detection of the waveform sequencer. When pins 0 are used for input operation, these pins are enabled as required and must not be used for any other I/P.	
64	1	8	P44	F			General-purpose I/O ports.

**Table 1.6-1 Pin Description (4/5)**

Pin no.			Pin name	I/O circuit	Pin status during reset	Function
QFP-M09*1	QFP-M06*2	SDIP*3				
1	2	9	P45	F	Port input	General-purpose I/O ports.
			SNI2*4			Trigger input pin for position detection of the Multi-pulse generator. When used for input operation, this pin is enabled as required and must not be used for any other I/P.
2	3	10	P46	F		General-purpose I/O ports.
			PPG2			Output pin for PPG channel 2. This function is enabled when PPG channel 2 output is enabled.
3 to 10	4 to 11	11 to 18	P50 to P57	I	Analog input	General-purpose I/O ports.
			AN0 to AN7			A/D converter analog input pins. This function is enabled when the analog input specification is enabled (ADER).
11	12	19	AVCC	J	Power input	Vcc power input pin for analog circuits.
12	13	20	AVR	K		Vref+ input pin for the A/D converter. This voltage must not exceed AVcc. Vref- is fixed to AVss.
13	14	21	AVSS	J		Vss power input pin for analog circuits.
14	15	22	P60	F	Port input	General-purpose I/O ports.
			SIN1			Serial data input pin for UART channel 1. While UART channel 1 is operating for input, the input of this pin is used as required and must not be used for any other input.
15	16	23	P61	F	Port input	General-purpose I/O ports.
			SOT1			Serial data output pin for UART channel 1. This function is enabled when UART channel 1 enables data output.
16	17	24	P62	F		General-purpose I/O port.
			SCK1			Serial clock I/O pin for UART channel 1. This function is enabled when UART channel 1 enables clock output.
17	18	25	P63	F		General-purpose I/O port.
			INT7			Can be used as interrupt request input channel 7. Input is enabled when "1" is set in EN7 in standby mode.

**Table 1.6-1 Pin Description (5/5)**

Pin no.			Pin name	I/O circuit	Pin status during reset	Function
QFP-M09*1	QFP-M06*2	SDIP*3				
18	19	26	MD0	L	Mode input	Input pin for operation mode specification. Connect this pin directly to Vcc or Vss.
20,21	21,22	28,29	MD1,MD2	L		Input pins for operation mode specification. Connect these pins directly to Vcc or Vss.
24,49	25,50	32,57	Vss	–	Power input	Power (0 V) input pin.
56	57	64	Vcc	–		Power (5 V) input pin.

\*1: FPT-64P-M09

\*2: FPT-64P-M06

\*3: DIP-64P-M01

\*4: Pin names are not applicable to MB90465 series

## 1.7 I/O Circuit Types

Table 1.7-1 summarize the I/O circuit types of MB90460/465 series

### ■ I/O Circuit Types

Table 1.7-1 I/O Circuit Type (1/3)

Classification	Type	Remarks
A		<p>Main clock(main clock crystal oscillator)</p> <ul style="list-style-type: none"> <li>At an ocillation feedback resistor of approximately <math>1\text{M}\Omega</math></li> </ul>
B		<ul style="list-style-type: none"> <li>Hysteresis input</li> <li>Resistor approximately <math>50\text{ k}\Omega</math></li> </ul>
C		<ul style="list-style-type: none"> <li>CMOS output</li> <li>Hysteresis input</li> <li>Selectable pull-up resistor approximately <math>50\text{ k}\Omega</math></li> <li><math>I_{OL} = 4\text{ mA}</math></li> </ul>
D		<ul style="list-style-type: none"> <li>CMOS output</li> <li>CMOS input</li> <li>Selectable pull-up resistor approximately <math>50\text{ k}\Omega</math></li> <li><math>I_{OL} = 12\text{ mA}</math></li> </ul>

**Table 1.7-1 I/O Circuit Type (2/3)**

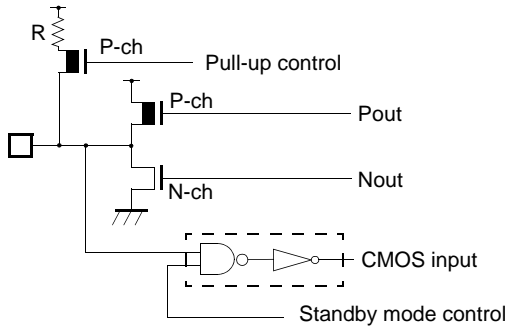
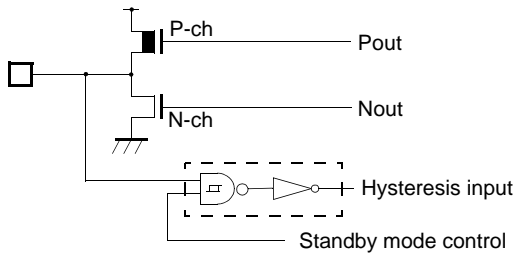
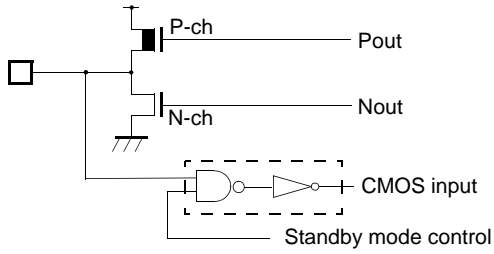
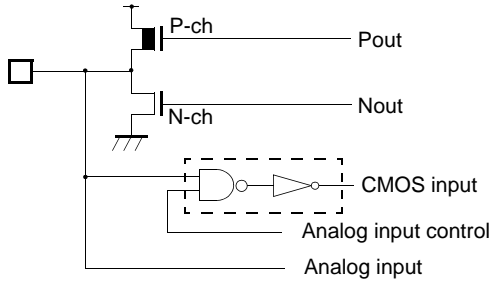
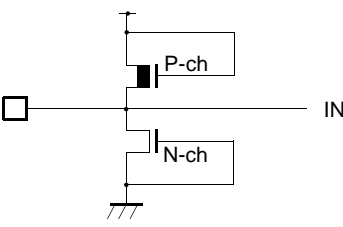
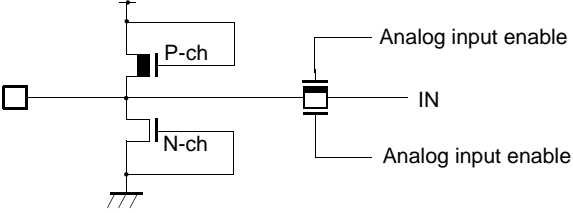
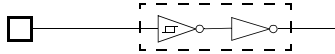
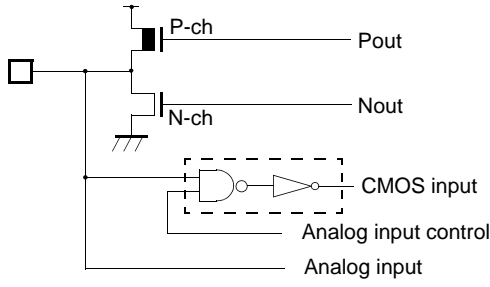
Classification	Type	Remarks
E		<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS input</li> <li>• Selectable pull-up resistor approximately 50 k<math>\Omega</math></li> <li>• <math>I_{OL} = 4</math> mA</li> </ul>
F		<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• Hysteresis input</li> <li>• <math>I_{OL} = 4</math> mA</li> </ul>
G		<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS input</li> <li>• <math>I_{OL} = 12</math> mA</li> </ul>
H		<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS input</li> <li>• <math>I_{OL} = 4</math> mA</li> </ul>
I		<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS input</li> <li>• Analog input</li> <li>• <math>I_{OL} = 4</math> mA</li> </ul>

Table 1.7-1 I/O Circuit Type (3/3)

Classification	Type	Remarks
J		<ul style="list-style-type: none"><li>Power supply input protection circuit</li></ul>
K		<ul style="list-style-type: none"><li>A/D converter reference voltage (AVR) input pin with protection circuit</li></ul>
L		<ul style="list-style-type: none"><li>Hysteresis input</li></ul>





# ***CHAPTER 2***

---

# ***NOTES ON HANDLING DEVICES***

**This chapter describes notes on Handling Devices.**

## **2.1 Notes on Handling Devices**

## 2.1 Notes on Handling Devices

---

When handling devices, pay special attention to the following eight items or procedures:

- **Strict observation of maximum rated voltage (latch-up prevention)**
  - **Stabilization of supply voltage**
  - **Power-on**
  - **Treatment of unused input pins**
  - **Treatment of A/D converter power pin**
  - **Notes on external clock**
  - **Caution on operation during PLL clock mode**
  - **Power supply pin**
  - **Analog power-on sequence of A/D converter**
- 

### ■ Notes on Handling Devices

#### ● Strict observation of maximum rated voltage

A latch-up may occur on a CMOS IC if a voltage higher than  $V_{cc}$  or lower than  $V_{ss}$  is applied to an input or output pin other than medium-to-high voltage pins. A latch-up may also occur if a voltage higher than the rating is applied between  $V_{cc}$  and  $V_{ss}$ . A latch-up causes a rapid increase in the power supply current, which can result in thermal damage to an element. Take utmost care that the maximum rated voltage is not exceeded.

When turning the power on or off to analog circuits, be sure that the analog supply voltages ( $AV_{cc}$  and  $AVR$ ) and analog input voltage do not exceed the digital supply voltage ( $V_{cc}$ ).

#### ● Stabilize of supply voltages

Even within the operation guarantee range of the  $V_{cc}$  supply voltage, a malfunction can be caused if the supply voltage undergoes a rapid change. For voltage stabilization guidelines, the  $V_{cc}$  ripple fluctuations (P-P value) at commercial frequencies (50 to 60 Hz) should be suppressed to "10%" or less of the reference  $V_{cc}$  value. During a momentary change such as when switching a supply voltage, voltage fluctuations should also be suppressed so that the "transient fluctuation rate" is 0.1 V/ms or less.

#### ● Power-on

To prevent a malfunction in the built-in voltage drop circuit, secure "50  $\mu$ s (between 0.2 V and 2.7 V)" or more for the voltage rise time during power-on.

#### ● Treatment of unused input pins

An unused input pin may cause a malfunction if it is left open. Every unused input pin should be pulled up or down.

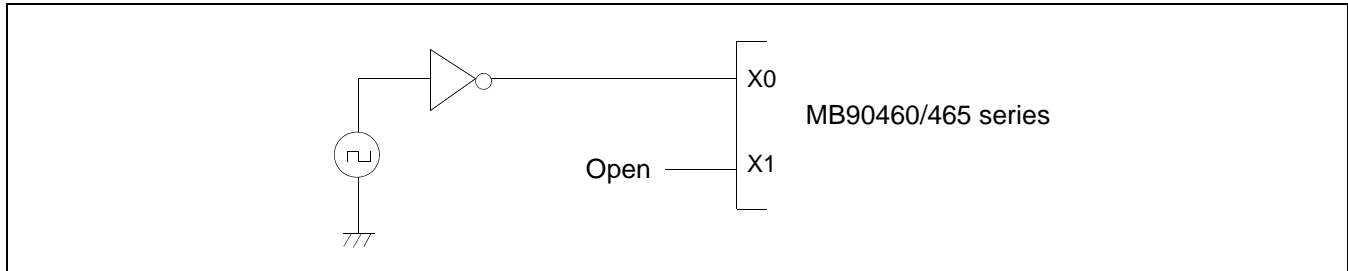
#### ● Treatment of A/D converter power pin

When the A/D converter is not used, connect the pins as follows:  $AV_{cc} = V_{cc}$ ,  $AV_{ss} = AVR = V_{ss}$ .

### ● Notes on external clock

When an external clock is used, the oscillation stabilization wait time is required at power-on reset or at cancellation of sub-clock mode or stop mode. As shown in Figure 2.1-1, when an external clock is used, connect only the X0 pin and leave the X1 pin open.

**Figure 2.1-1 Sample Application of External Clock**



### ● Caution on operation 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 operation if such failure occurs.

### ● Power supply pins

When a device has two or more Vcc or Vss pins, the pins that should have equal potential are connected within the device in order to prevent a latch-up or other malfunction. To reduce extraneous emission, to prevent a malfunction of the strobe signal due to an increase in the group level, and to maintain the local output current rating, connect all these power supply pins to an external power supply and ground them.

The current source should be connected to the Vcc and Vss pins of the device with minimum impedance. It is recommended that a bypass capacitor of about 0.1  $\mu\text{F}$  be connected near the terminals between Vcc and Vss.

### ● Analog power-on sequence of A/D converter

The power to the A/D converter (AVcc, AVR) and analog inputs (AN0 to AN7) must be turned on after the power to the digital circuits (Vcc) is turned on. When turning off the power, turn off the power to the digital circuits (Vcc) after turning off the power to the A/D converter and analog inputs. When the power is turned on or off, AVR should not exceed AVcc. Also, when a pin that is used for analog input is also used as an input port, the input voltage should not exceed AVcc. (The power to the analog circuits and the power to the digital circuits can be simultaneously turned on or off.)



# **CHAPTER 3**

---

## **CPU**

**This chapter describes memory space for the MB90460/465 series.**

- 3.1 CPU
- 3.2 Memory Space
- 3.3 Memory Maps
- 3.4 Addressing
- 3.5 Memory Location of Multi-byte Data
- 3.6 Registers
- 3.7 Dedicated Registers
- 3.8 General-purpose Registers
- 3.9 Prefix Codes

## 3.1 CPU

---

The F<sup>2</sup>MC-16LX CPU is a 16-bit CPU designed for use in applications, such as welfare and mobile equipment, which require high-speed real-time processing. The instruction set of the F<sup>2</sup>MC-16LX was designed for controllers so that it can perform various types of control at high speed and efficiency.

The F<sup>2</sup>MC-16LX CPU process not only 16-bit data but also 32-bit data using a built-in 32-bit accumulator. Memory space, which can be extended up to 16M bytes, can be accessed in either linear or bank access mode. The instruction set inherits the AT architecture of F<sup>2</sup>MC-8L, and has additional instructions supporting high-level languages. In addition, it has an extended addressing mode, enhanced multiply/divide instructions and reinforced bit manipulation instructions. The features of the F<sup>2</sup>MC-16LX CPU are shown below:

---

### ■ CPU

- Minimum instruction execution time: 62.5 ns (when the source oscillation is 4 MHz and the PLL clock is multiplied by 4)
  - Maximum memory address space: 16M bytes. Access in linear or bank addressing mode
  - Instruction set optimum for controller applications
    - Many data types (bit, byte, word and long word)
    - As many as 23 addressing modes
    - Enhanced high-precision arithmetic operation by a 32-bit accumulator
    - Enhanced signed multiply/divide instructions and RETI instruction function
  - Enhanced interrupt function
    - Eight programmable priority levels
  - Automatic transfer function independent of CPU
    - Extended intelligent I/O service using up to 16 channels
  - Instruction set supporting high-level language (C) and multitasking
    - System stack pointer, instruction set symmetry and barrel shift instructions
  - Increased execution speed: 4-byte instruction queue
- 

#### Note:

The MB90460/465 series runs only in single-chip mode so only internal ROM and RAM and internal peripheral address space can be accessed.

---

## 3.2 Memory Space

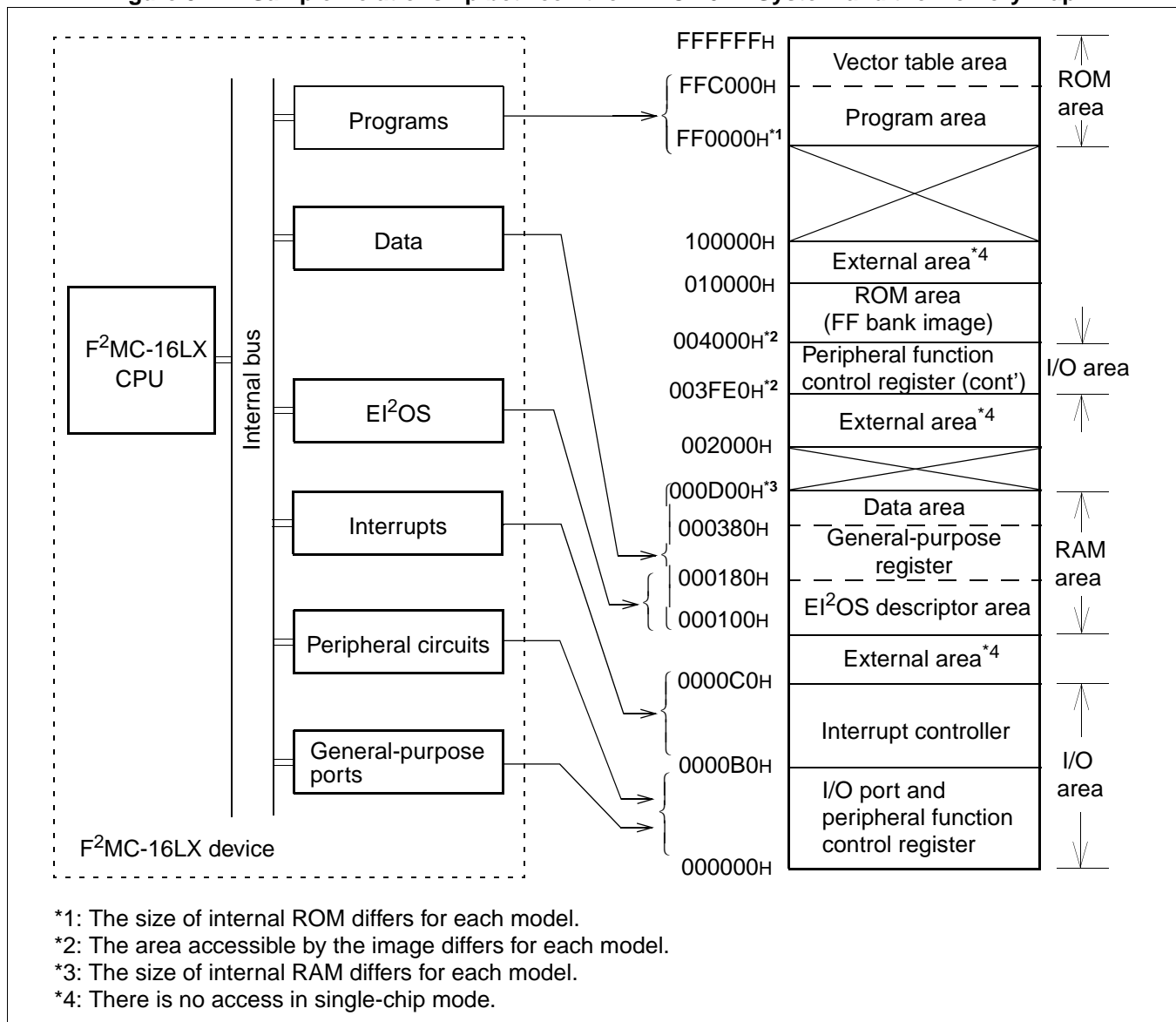
All I/O, programs and data are located in the 16-megabyte memory space of the F<sup>2</sup>MC-16LX. A part of the memory space is used for special purposes, such as extended intelligent I/O service (EI<sup>2</sup>OS) descriptors, general-purpose registers and vector tables.

### ■ Memory Space

All I/O, programs, and data are located in the 16-megabyte memory space of the F<sup>2</sup>MC-16LX CPU. The CPU is able to access each resource through an address indicated by the 24-bit address bus.

Figure 3.2-1 shows a sample relationship between the F<sup>2</sup>MC-16LX system and the memory map.

**Figure 3.2-1 Sample Relationship between the F<sup>2</sup>MC-16LX System and the 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 vector call instructions, interrupt vectors and reset 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 differs for each model.

## ■ RAM Area

- Data area (address: from 000100<sub>H</sub>)
  - The static RAM is built in as an internal data area.
  - The size of internal RAM differs for each 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: 000100<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) correspond to all peripheral functions that have an interrupt function. These registers set interrupt levels and control the extended intelligent I/O service (EI<sup>2</sup>OS).
- Peripheral function control register area (address: 000008<sub>H</sub> to 00000F<sub>H</sub>, 000019<sub>H</sub> to 0000AF<sub>H</sub>, 003FE0<sub>H</sub> to 003FFF<sub>H</sub>) This register controls the built-in peripheral functions, inputs and outputs data. Instruction using I/O addressing e.g. MOV A, io, is not supported for registers area 003FE0<sub>H</sub> to 003FFF<sub>H</sub>
- I/O port control register area (address: 000000<sub>H</sub> to 000006<sub>H</sub>, 000010<sub>H</sub> to 000017<sub>H</sub>) This register controls I/O ports, inputs and outputs data.

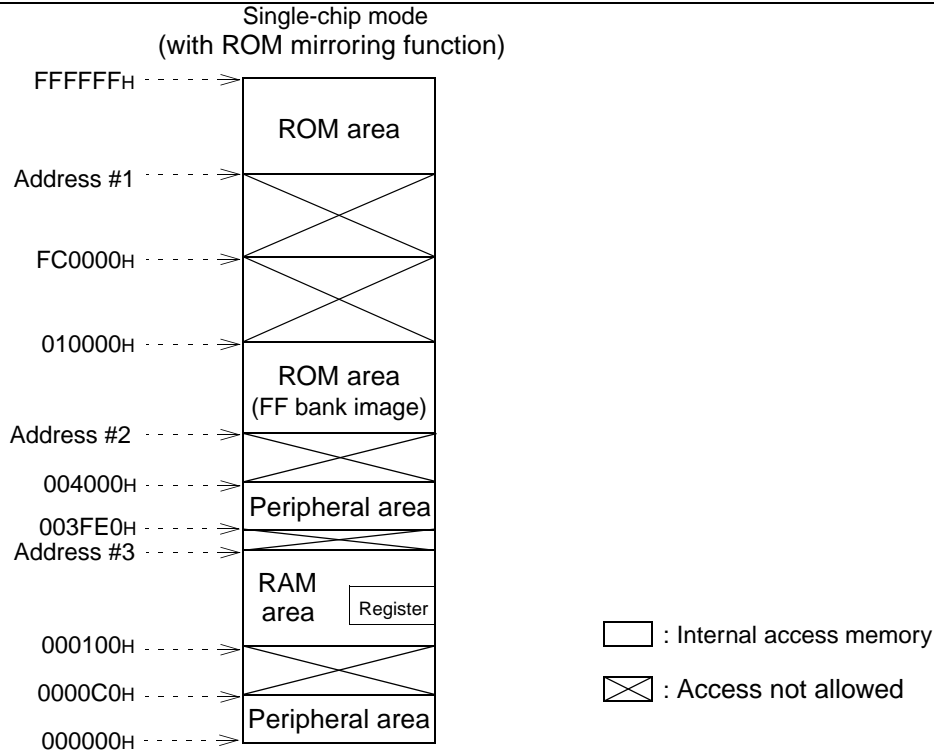
### 3.3 Memory Maps

This section shows the memory map for each MB90460/465 series model.

#### ■ Memory Maps

Figure 3.3-1 shows the memory maps for the MB90460/465 series.

**Figure 3.3-1 Memory Maps**



Model	Address #1	Address #2	Address #3
MB90462	FF0000 <sub>H</sub>	004000 <sub>H</sub>	000900 <sub>H</sub>
MB90467	FF0000 <sub>H</sub>	004000 <sub>H</sub>	000900 <sub>H</sub>
MB90F462	FF0000 <sub>H</sub>	004000 <sub>H</sub>	000900 <sub>H</sub>
MB90F462A	FF0000 <sub>H</sub>	004000 <sub>H</sub>	000900 <sub>H</sub>
MB90F463A	FE0000 <sub>H</sub>	004000 <sub>H</sub>	000900 <sub>H</sub>
MB90V460	FF0000 <sub>H</sub> *	004000 <sub>H</sub> *	002100 <sub>H</sub>

\*: The MB90V460 does not contain ROM. Assume that the development tool uses these area for its ROM decode areas.

### Notes:

- If single-chip mode (without ROM mirroring function) is selected, see "CHAPTER 22 ROM MIRRORING FUNCTION SELECTION MODULE".
  - ROM data in the FF bank can be seen as an image in the higher 00 bank to validate the small model C compiler. Because addresses of the 16 low order bits in the FF bank are the same, the table in ROM can be referenced without the "far" specification. For example, when 00C000<sub>H</sub> is accessed, the contents of ROM at FFC000<sub>H</sub> are actually accessed. The ROM area in the FF bank exceeds 48 Kbytes, and all areas cannot be seen as images in the 00 bank. Because ROM data from FF4000<sub>H</sub> to FFFFFFF<sub>H</sub> is seen as an image at 004000<sub>H</sub> to 00FFFF<sub>H</sub>, the ROM data table should be stored in the area from FF4000<sub>H</sub> to FFFFFFF<sub>H</sub>.
-

## 3.4 Addressing

The methods for generating addresses are linear addressing and bank addressing. In linear addressing, the complete 24-bit address is specified directly by an instruction. In bank addressing, the upper 8 bits of the address are specified by a bank register for the required purpose, and the lower 16 bits of the address are specified by the instruction.

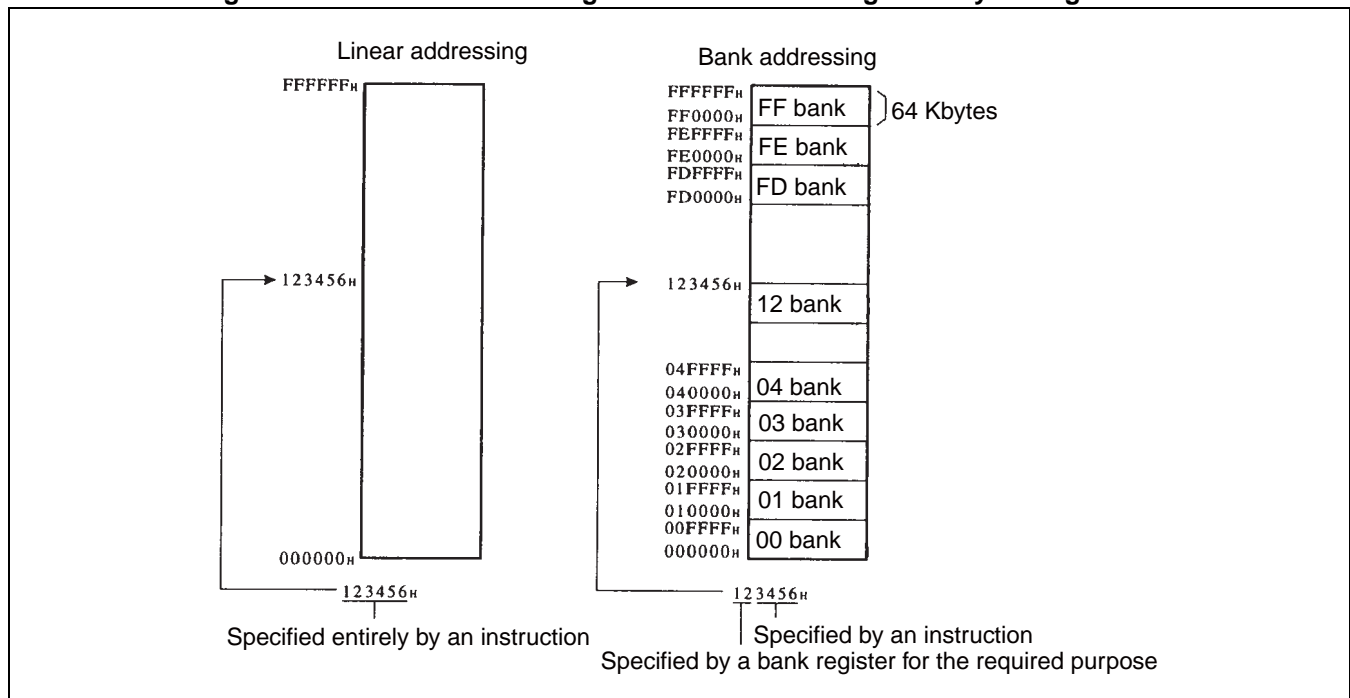
The F<sup>2</sup>MC-16LX family generally uses bank addressing.

### ■ Linear Addressing and Bank Addressing

In linear addressing, the 16-Mbyte space is accessed as consecutive address spaces. In bank addressing, the 16-Mbyte space is divided into and managed as 256 64-Kbyte banks.

Figure 3.4-1 is an overview of linear addressing and bank addressing memory management.

**Figure 3.4-1 Linear Addressing and Bank Addressing Memory Management**

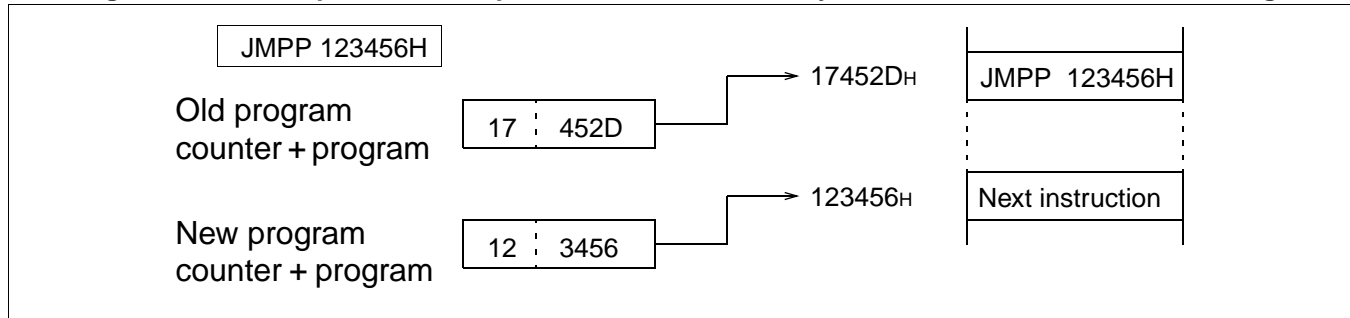


### 3.4.1 Address Specification by Linear Addressing

The two types of address specification by linear addressing are specification of a 24-bit address directly in the operand and specification of the lower 24 bits of a 32-bit general-purpose register.

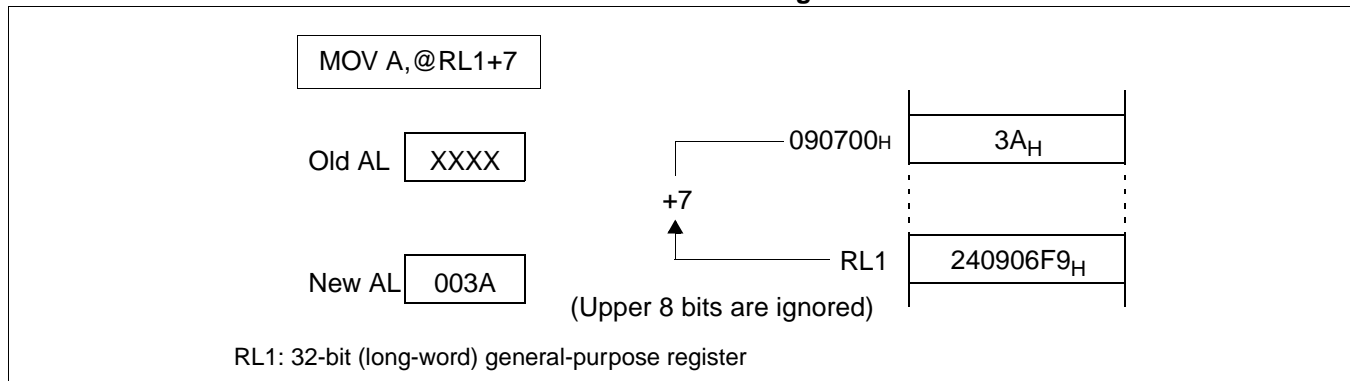
#### ■ Linear Addressing by 24-bit Operand Specification

Figure 3.4-2 Example of Direct Specification of a 24-bit Physical Address in Linear Addressing



#### ■ Addressing by Indirect Specification with a 32-bit

Figure 3.4-3 Example of Indirect Specification with a 32-bit General-purpose Register in Linear Addressing



## 3.4.2 Address Specification by Bank Addressing

In address specification by bank addressing, the 16-Mbyte memory space is divided into 256 64-Kbyte banks. A bank address that corresponds to each space is specified in the bank register to determine the upper 8 bits of the address. The lower 16 bits of the address are specified by the instruction.

The five types of bank registers classified by function are as follows:

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

### ■ Bank Registers and Access Space

Table 3.4-1 lists the access space and main function of each bank register.

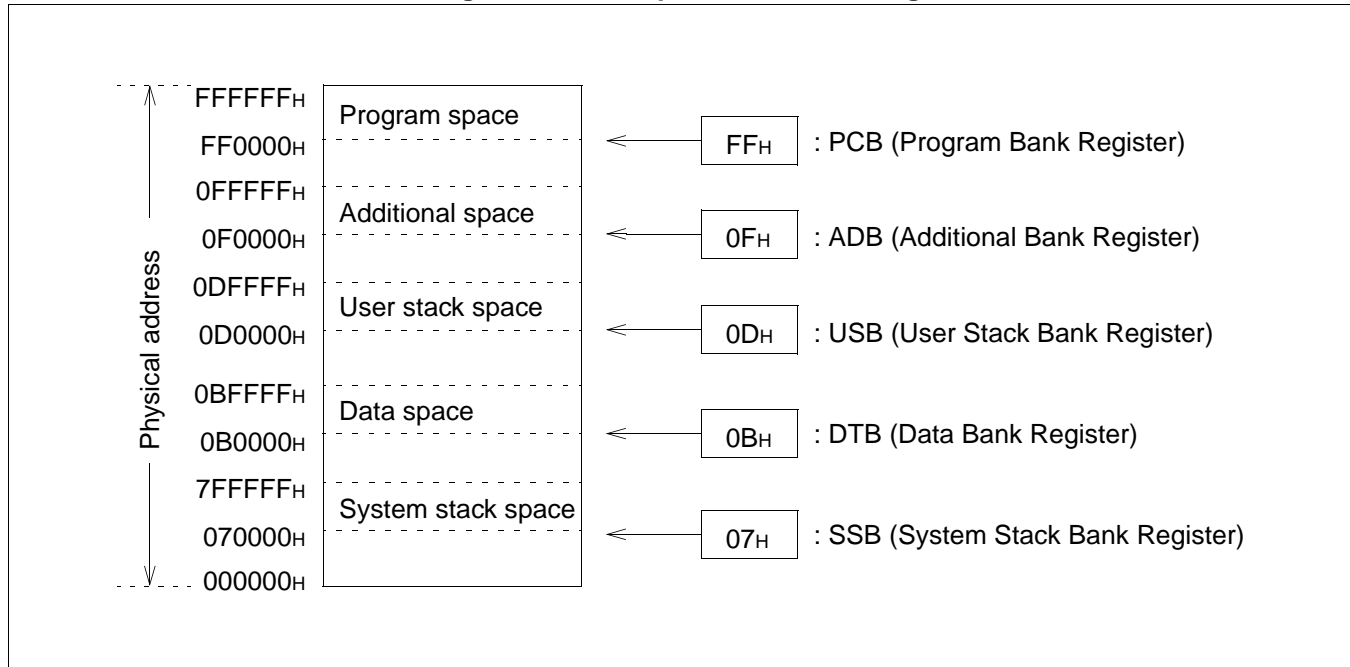
**Table 3.4-1 Access Space and Main Function of Each Bank Register**

Bank register name	Access space	Main function	Initial value after a reset
Program bank register (PCB)	Program (PC) space	Instruction codes, vector tables and immediate-value data are stored.	FF <sub>H</sub>
Data bank register (DTB)	Data (DT) space	Read/write data is stored. Internal or external peripheral control registers and data registers are accessed.	00 <sub>H</sub>
User stack bank register (USB)	Stack (SP) space	This area is used for stack accesses such as when PUSH/POP instructions and interrupt registers are saved. The SSB is used when the stack flag in the condition register (CCR: S) is "1". The USB is used when the stack flag in the condition register (CCR: S) is "0". *	00 <sub>H</sub>
System stack bank register (SSB) *1			00 <sub>H</sub>
Additional bank register (ADB)	Additional (AD) space	Data that overflows from the data (DT) space is stored.	00 <sub>H</sub>

\* : The SSB is always used as an interrupt stack.

Figure 3.4-4 shows the relationship between the memory space divisions and each register. See "3.7.9 Bank Registers (PCB, DTB, USB, SSB, ADB)", for details.

**Figure 3.4-4 Sample Bank Addressing**



## ■ Bank Addressing and Default Space

To improve instruction coding efficiency, each instruction has a defined default space for each addressing method, as shown in Table 3.4-2. To use a space other than the default space, specify a prefix code for a bank before the instruction. This enables the bank space that corresponds to the specified prefix code to be accessed. See "3.9 Prefix Codes", for details about prefix codes.

**Table 3.4-2 Addressing and Default Spaces**

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

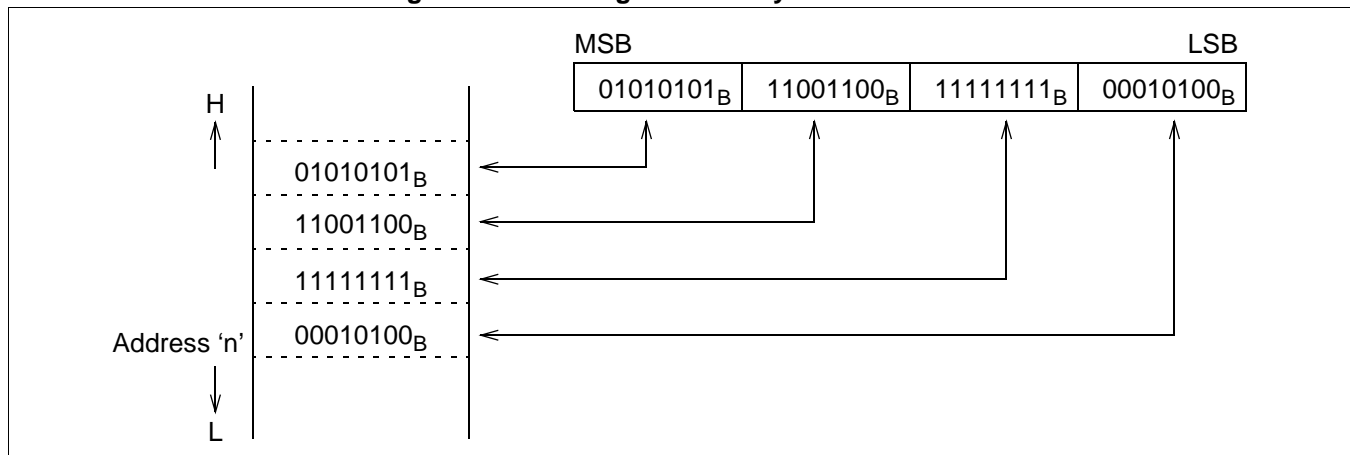
## 3.5 Memory Location of Multi-byte Data

Multi-byte data is written to memory sequentially from the lower address. If multi-byte data is 32-bit data, the lower 16 bits are transferred followed by the upper 16 bits. If a reset signal is input immediately after the low-order data is written, the high-order data may not be written.

### ■ Storage of Multi-byte Data in RAM

Figure 3.5-1 shows the data configuration of multi-byte data in memory. The lower 8 bits of the data is located at address  $n$ , and subsequent data is located at address  $n + 1$ , address  $n + 2$ , address  $n + 3$  and so on, in this sequence.

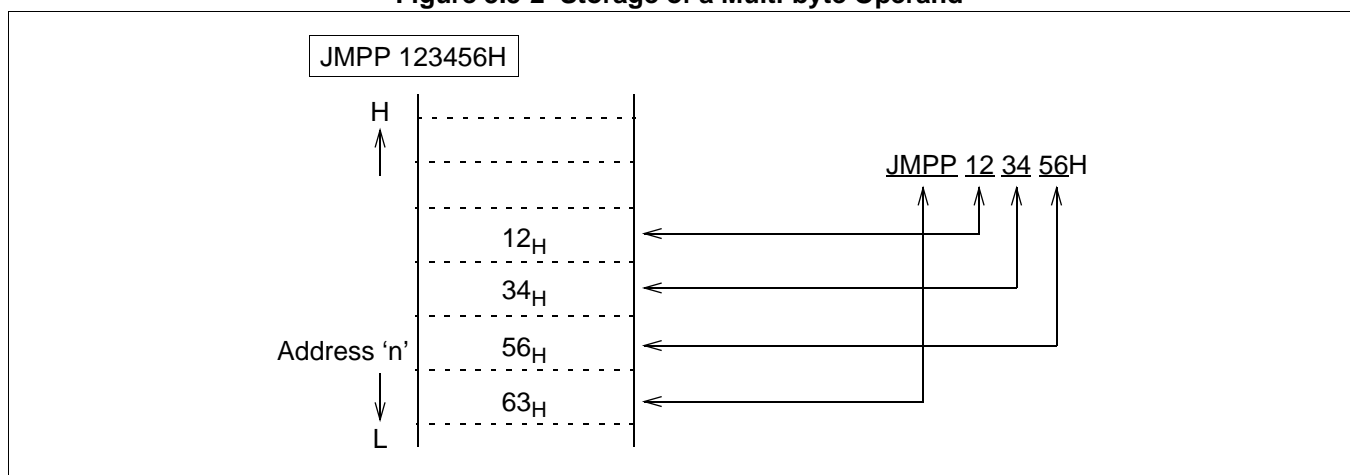
Figure 3.5-1 Storage of Multi-byte Data in RAM



### ■ Storage of Multi-byte Operand

Figure 3.5-2 shows the configuration of a multi-byte operand in memory.

Figure 3.5-2 Storage of a Multi-byte Operand

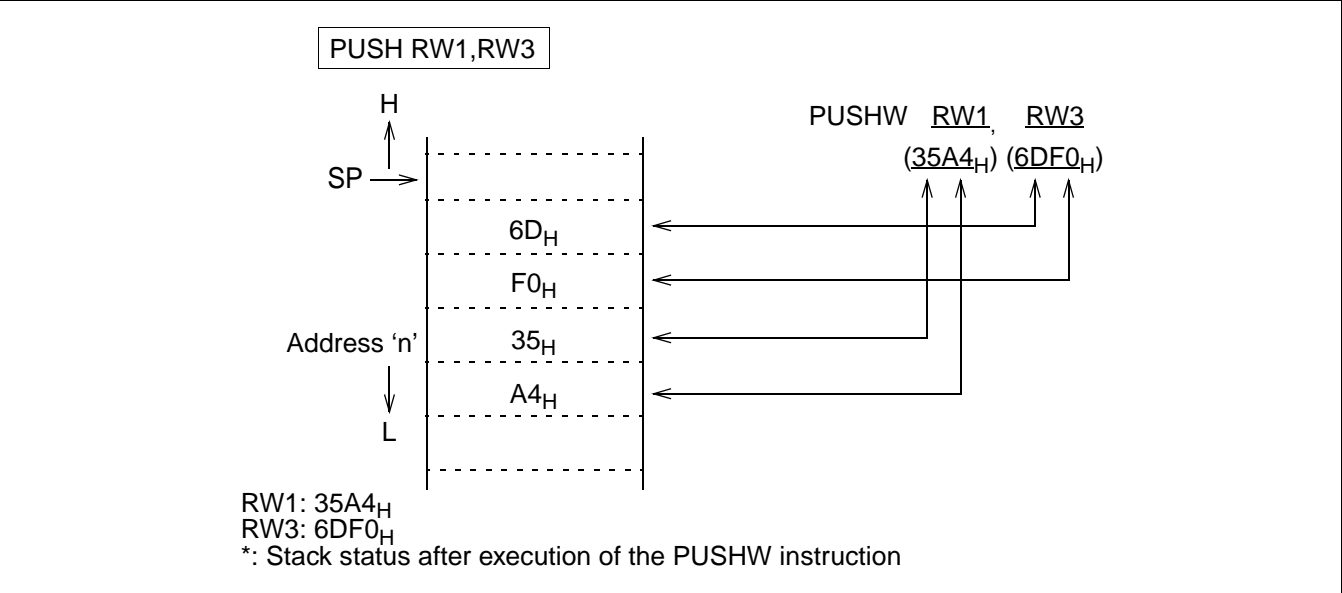




■ Storage of Multi-byte Data in a Stack

Figure 3.5-3 shows the configuration of multi-byte data in a stack.

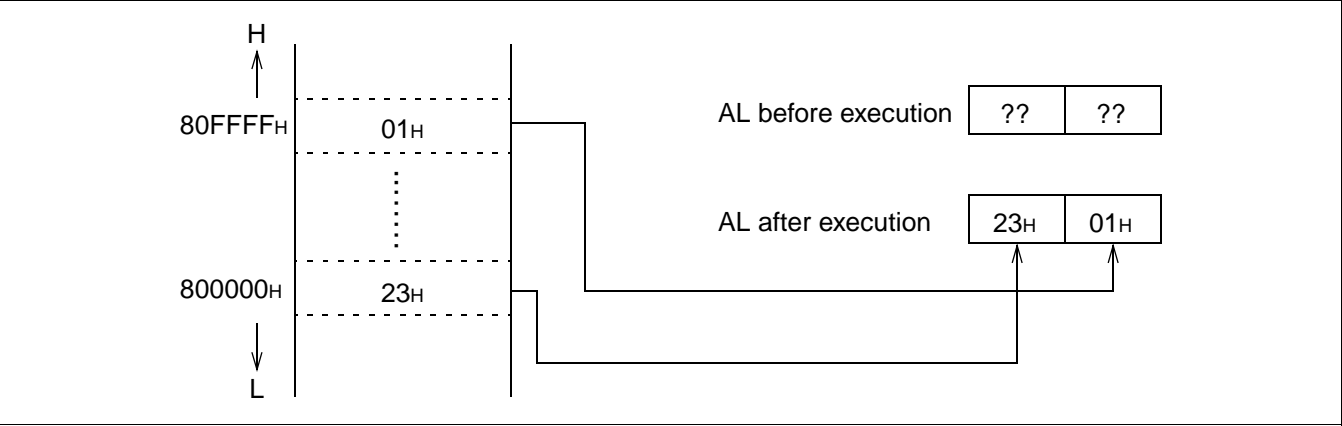
Figure 3.5-3 Storage of Multi-byte Data in a Stack



■ Multi-byte Data Access

Accessing is generally performed within a bank. For an instruction that accesses multi-byte data, the address following "FFFF<sub>H</sub>" is "0000<sub>H</sub>" in the same bank. Figure 3.5-4 shows an example of executing an instruction that accesses multi-byte data on a bank boundary.

Figure 3.5-4 Multi-byte Data Access on a Bank Boundary



## 3.6 Registers

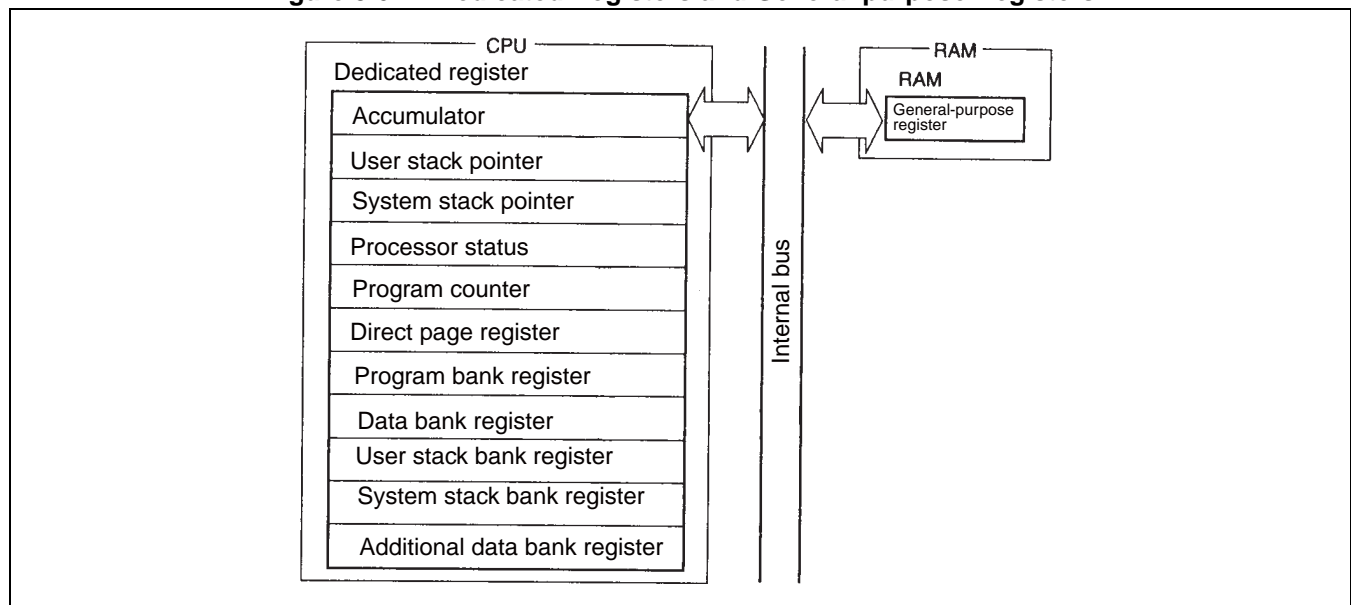
**F<sup>2</sup>MC-16LX registers are classified into internal dedicated CPU registers and built-in RAM general-purpose registers.**

### ■ Dedicated Registers and General-purpose Registers

Dedicated registers are dedicated hardware inside the CPU with limited use in the CPU architecture.

General-purpose registers exist together with RAM in the CPU address space. Just like dedicated registers, general-purpose registers can be accessed without addressing. Just like ordinary memory, the user can specify how the register is used. Figure 3.6-1 shows the location of the dedicated registers and general-purpose registers in the device.

**Figure 3.6-1 Dedicated Registers and General-purpose Registers**



### 3.7 Dedicated Registers

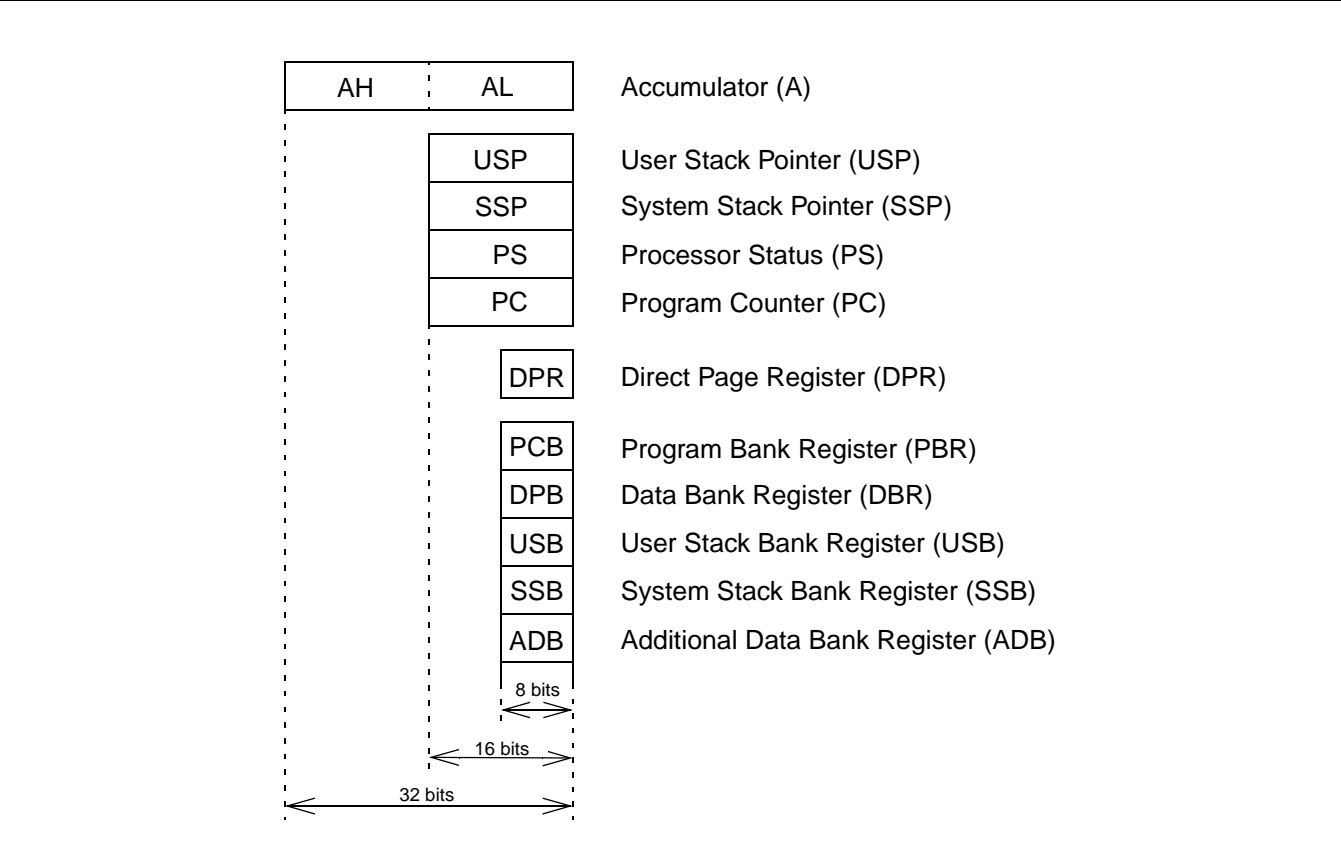
The following 11 registers are dedicated registers in the CPU.

- Accumulator (A)
- System stack pointer (SSP)
- Program counter (PC)
- Program bank register (PCB)
- User stack pointer (USP)
- User stack bank register (USB)
- Additional data bank register (ADB)
- Processor status (PS)
- Direct page register (DPR)
- Data bank register (DPP)
- System stack bank register (SSB)

■ Configuration of Dedicated Registers

Figure 3.7-1 shows the configuration of dedicated registers; Table 3.7-1 lists the initial values of the dedicated registers.

Figure 3.7-1 Configuration of Dedicated Registers



**Table 3.7-1 Initial Values of the Dedicated Registers**

Dedicated register	Initial value
Accumulator (A)	Undefined
User stack pointer (USP)	Undefined
System stack pointer (SSP)	Undefined
Processor status (PS)	<div style="text-align: center;"> <p>bit 15 ←→ 13 12 ←→ 8 7 ←→ 0</p> <p>PS    ILM    RP    CCR</p> <p>Default value ⇒    000    00000    -01xxxxx</p> </div>
Program counter (PC)	Value in reset vector (contents of FFFFDC <sub>H</sub> , FFFFDD <sub>H</sub> )
Direct page register (DPR)	01 <sub>H</sub>
Program bank register (PCB)	Value in reset vector (contents of FFFFDE <sub>H</sub> )
Data bank register (DTB)	00 <sub>H</sub>
User stack bank register (USB)	00 <sub>H</sub>
System stack bank register (SSB)	00 <sub>H</sub>
Additional data bank register (ADB)	00 <sub>H</sub>

-: Not used

x: Undefined

**Note:**

The above initial values are the initial values for the device. They are different from the ICE (emulator, etc.) values.

### 3.7.1 Accumulator (A)

The accumulator (A) consists of two 16-bit arithmetic operation registers (AH and AL). The accumulator is used to temporarily store the results of an arithmetic operation and data.

The A register can be used as a 32-bit, 16-bit or 8-bit register. Various arithmetic operations can be performed between memory and other registers or between the AH register and the AL register. The A register has a data retention function that automatically transfers pre-transfer data from the AL register to the AH register when data of word length or less is transferred to the AL register. (Data is not retained with some instructions.)

#### ■ Accumulator (A)

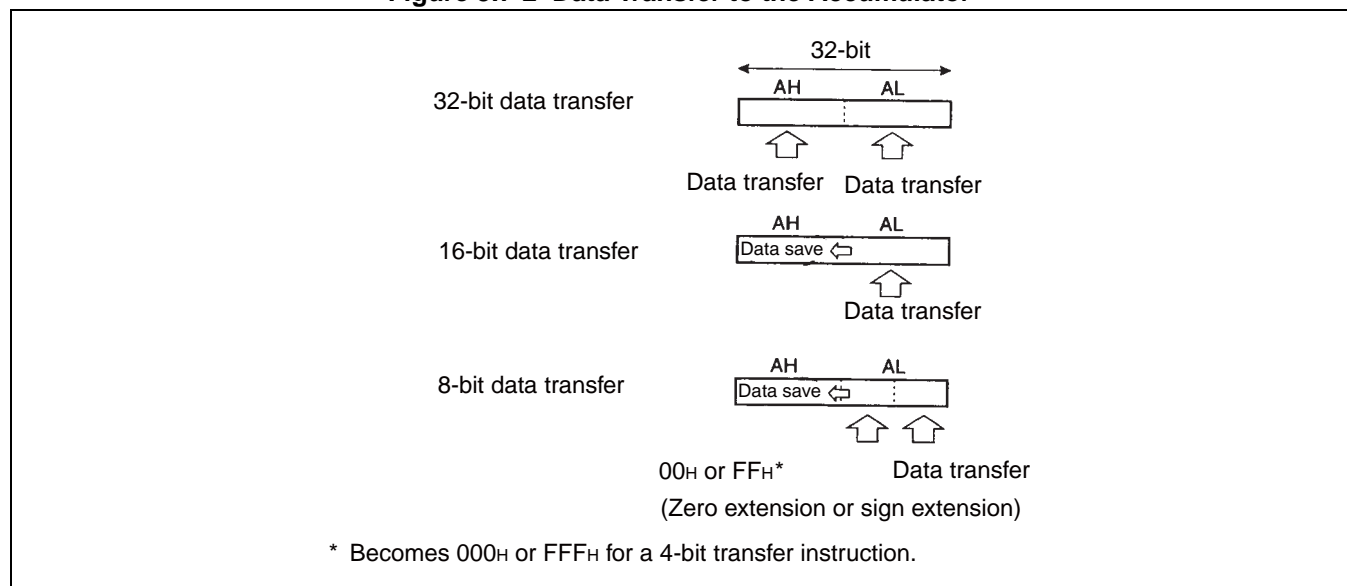
##### ● Data transfer to the accumulator

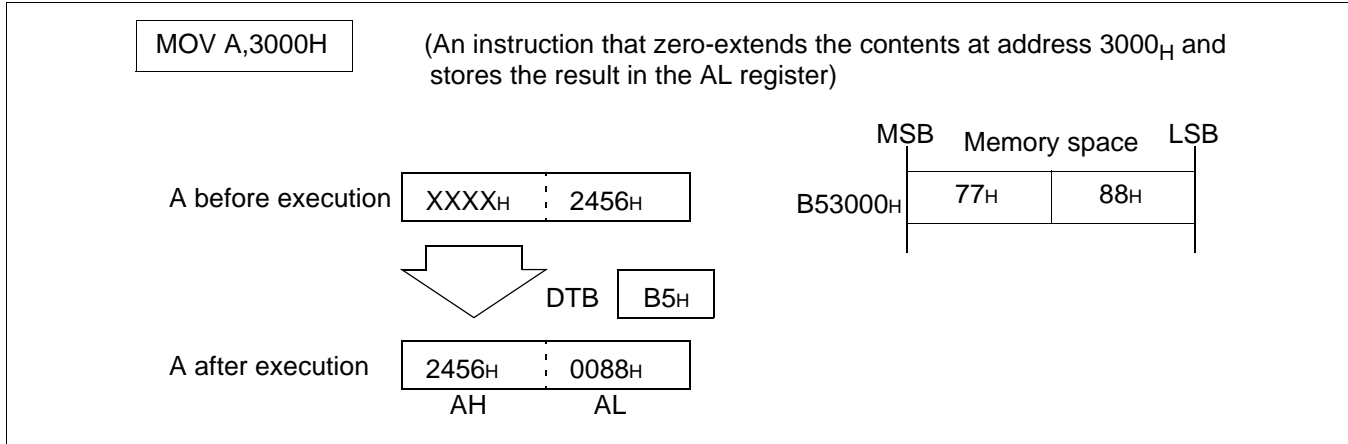
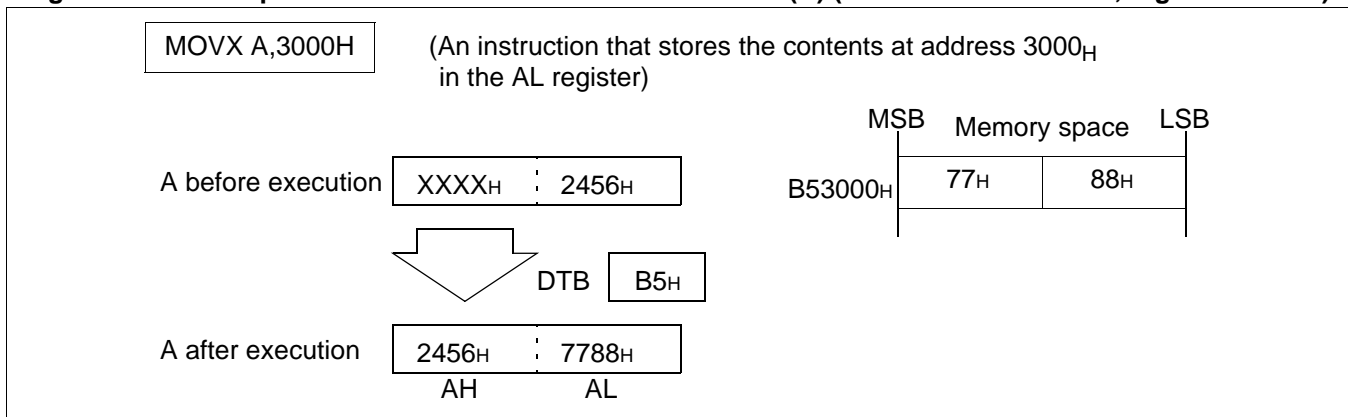
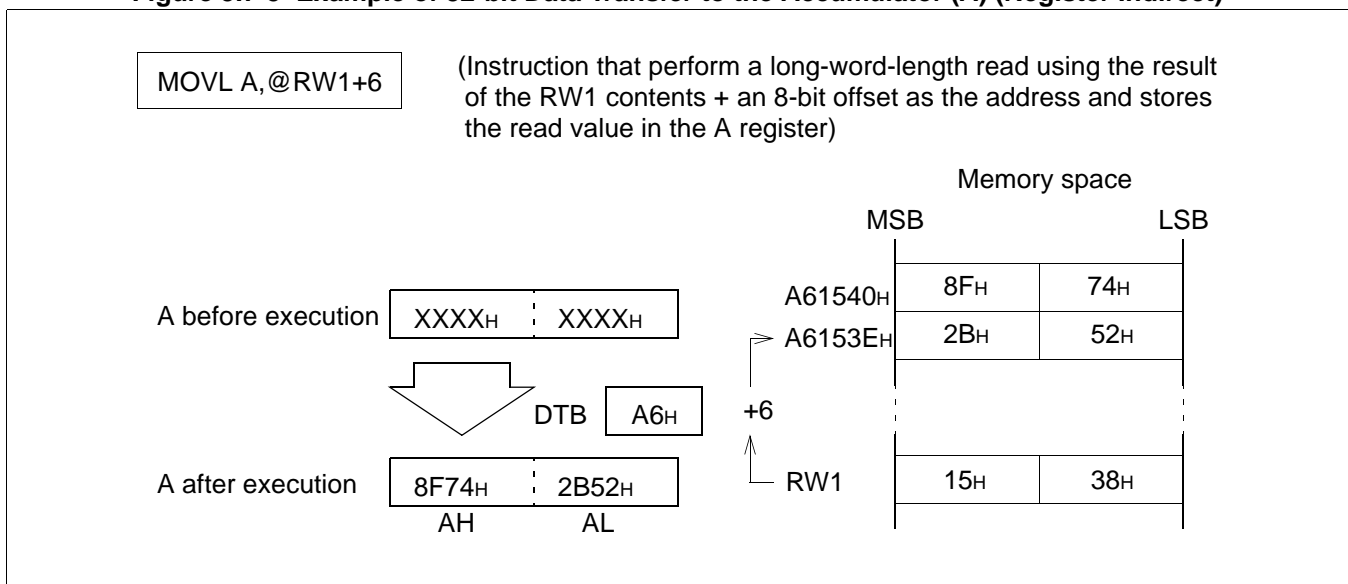
The accumulator can process 32-bit (long word), 16-bit (word) and 8-bit (byte) data. The 4-bit data transfer instruction (MOVN) is an exception. The explanation of 8-bit data also applies to 4-bit data.

- For 32-bit data processing, the AH register and AL register are combined.
- For 16-bit data and 8-bit data, only the AL register is used.
- When data of byte length or less is transferred to the AL register, data becomes 16 bits long by sign extension or zero extension, and is stored in the AL register. Data in the AL register can be handled as word-length or byte-length data.

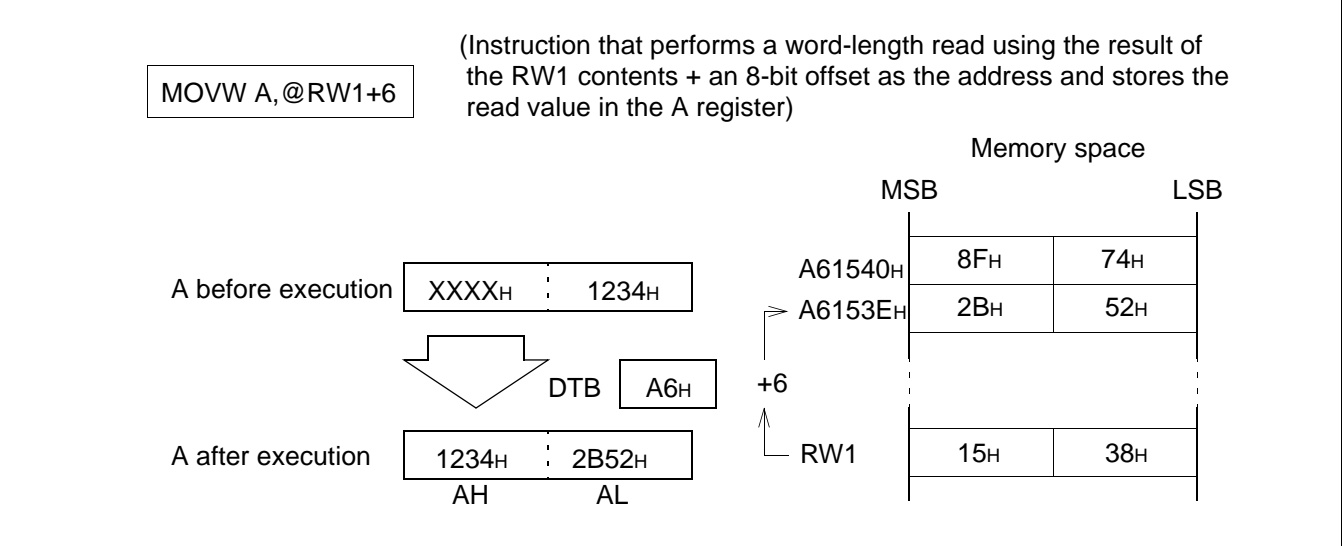
Figure 3.7-2 shows data transfer to the accumulator. Figure 3.7-3 to Figure 3.7-6 show specific transfer examples.

**Figure 3.7-2 Data Transfer to the Accumulator**



**Figure 3.7-3 Example of AL-AH Transfer in the Accumulator (A) (8-bit Immediate Value, Zero Extension)****Figure 3.7-4 Example of AL-AH Transfer in the Accumulator (A) (8-bit Immediate Value, Sign Extension)****Figure 3.7-5 Example of 32-bit Data Transfer to the Accumulator (A) (Register Indirect)**

**Figure 3.7-6 Example of AL-AH Transfer in the Accumulator (A) (16 Bits, Register Indirect)**



● Accumulator byte-processing arithmetic operation

When a byte-processing arithmetic operation instruction is executed for the AL register, the upper 8 bits of the AL register before the arithmetic operation is executed are ignored. The upper 8 bits of the arithmetic operation results are all zeros.

● Initial value of the accumulator

The initial value after a reset is undefined.

## 3.7.2 Stack Pointers (USP, SSP)

There are two types of stack pointers: a user stack pointer (USP) and a system stack pointer (SSP). Each stack pointer is a register that indicates the memory address of the location of the destination for saved data or a return address when PUSH instructions, POP instructions and subroutines are executed. The upper 8 bits of the stack address are specified by the user stack bank register (USB) or system stack bank register (SSB). When the S flag of the condition code register (CCR) is "0", the USP and USB registers are valid. When the S flag is "1", the SSP and SSB registers are valid.


### ■ Stack Selection

The F<sup>2</sup>MC-16LX uses two types of stack: a system stack and a user stack.

The stack address is determined, as shown in Table 3.7-2, by the S flag in the processor status (PS: CCR).

**Table 3.7-2 Stack Address Specification**

S flag	Stack address	
	Upper 8 bits	Lower 16 bits
0	User stack bank register (USB)	User stack pointer (USP)
1	System stack bank register (SSB)	System stack pointer (SSP)

 : Initial value

Because the S flag is initialized to "1" by a reset, the system stack is used as the default.

Ordinarily, the system stack is used for interrupt routine stack operations, and the user stack is used for all other types of stack operation. When separation of the stack space is not particularly necessary, only the system stack should be used.

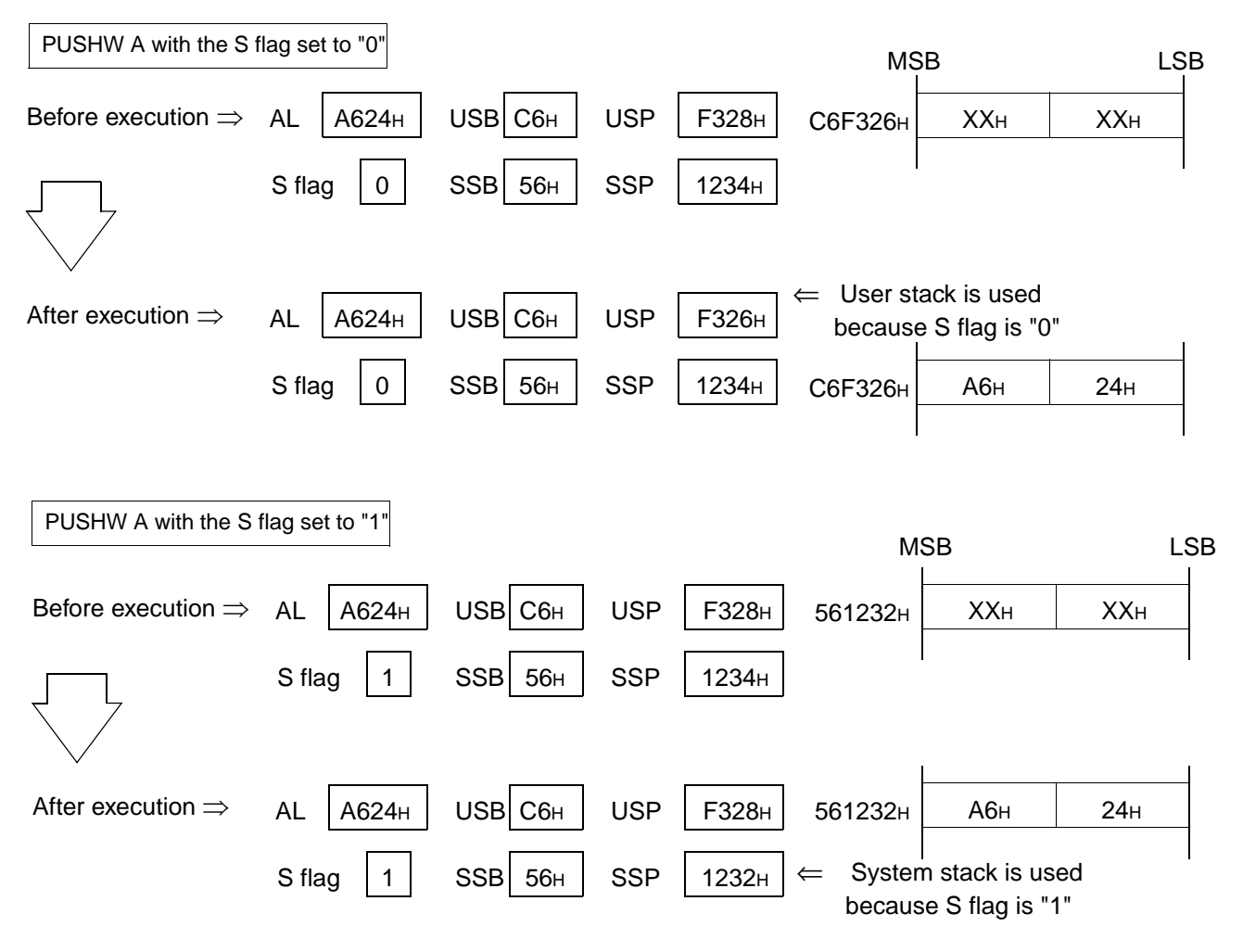
#### Note:

Since the S flag is set to "1" when an interrupt is accepted, the system stack is always used for interrupts.

Figure 3.7-7 shows an example of stack operation with the system stack.



Figure 3.7-7 Stack Operation Instruction and Dtask Pointer



Notes:

- To set a value for the stack pointer, generally use an even-numbered address. If an odd-numbered address is used, a word access is split into two parts, lowering efficiency.
- The initial values of the USP register and SSP register after a reset are undefined.

■ System Stack Pointer (SSP)

To use the system stack pointer (SSP), set the S flag in the condition code register (CCR) of the processor status (PS) to "1". The upper 8 bits of the address that will be used for the stack operation are indicated by the system stack bank register (SSB).

■ User Stack Pointer (USP)

To use the user stack pointer (USP), set the S flag in the condition code register (CCR) of the processor status (PS) to "0". The upper 8 bits of the address that will be used for the stack operation are indicated by the user stack bank register (USB).

### 3.7.3 Processor Status (PS)

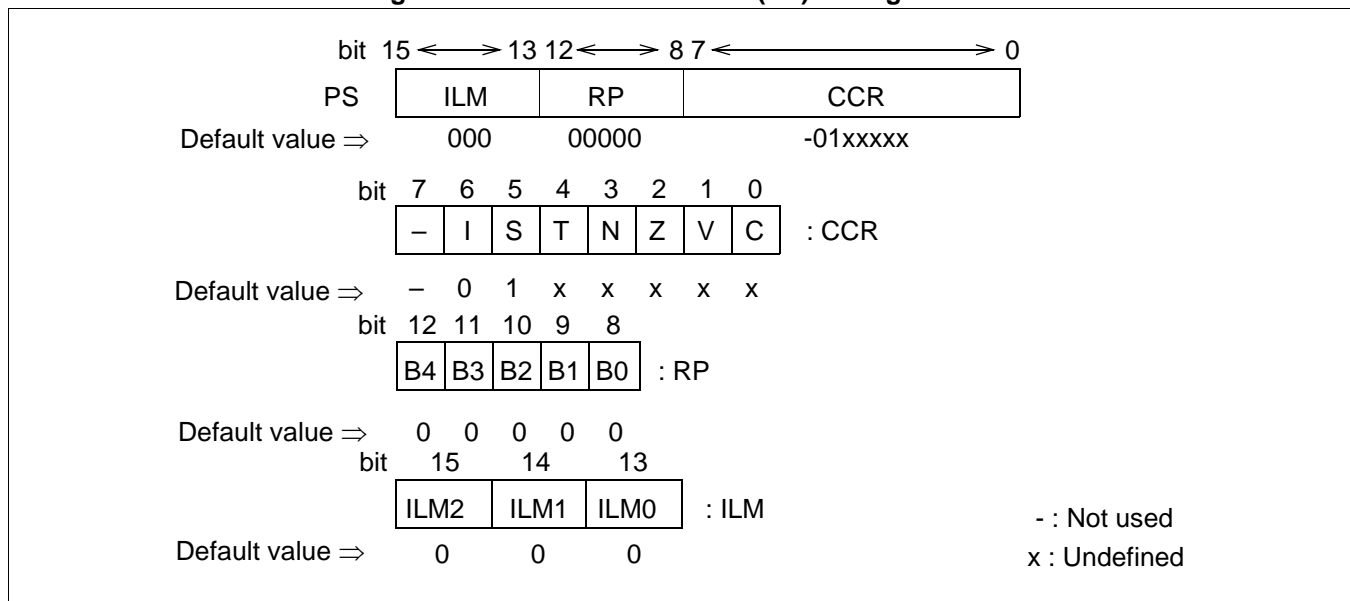
The processor status (PS) consists of CPU control bits and bits that indicate the CPU status. The PS register consists of the following three registers:

- Interrupt level mask register (ILM)
- Register bank pointer (RP)
- Condition code register (CCR)

#### ■ Processor Status (PS) Configuration

The processor status (PS) consists of CPU control bits and bits that indicate the CPU status. Figure 3.7-8 shows the configuration of the processor status (PS)

**Figure 3.7-8 Processor Status (PS) Configuration**



#### ● Interrupt level mask register (ILM)

This register indicates the level of the interrupt currently accepted by the CPU. The value is compared with the value of the interrupt level setting bits (ICR: IL0 to IL2) in the interrupt control register set for the peripheral resource interrupt request.

#### ● Register bank pointer (RP)

This pointer points to the first address of the memory block (register bank) used as the general-purpose register in the RAM area.

There are 32 banks for general-purpose registers. Values 0 to 31 are set in the RP to specify a bank.

#### ● Condition code register (CCR)

This register consists of flags that are set to "1" or reset to "0" by instruction execution results and by interrupts.

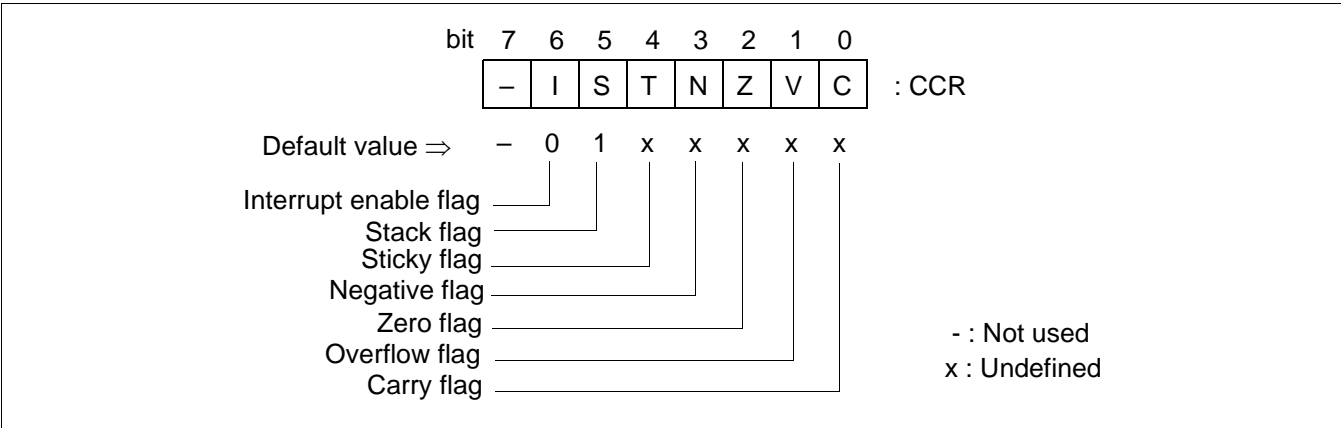
### 3.7.4 Condition Code Register (PS: CCR)

The condition code register (CCR) is an 8-bit register that consists of the bits that indicate the results of an arithmetic operation and the contents of transfer data and bits that control interrupt request acceptance.

#### ■ Condition Code Register (CCR) Configuration

Figure 3.7-9 shows the configuration of the CCR register. Refer to the programming manual for details about the status of the condition code register (CCR) during instruction execution.

Figure 3.7-9 Condition Code Register (CCR) Configuration



#### ● Interrupt enable flag (I)

In response to all interrupt requests other than software interrupts, when the I flag is "1", interrupts are enabled. When the I flag is "0", interrupts are disabled. Cleared by a reset.

#### ● Stack flag (S)

This flag indicates the pointer used for a stack operation. When the S flag is "0", the user stack pointer (USP) is valid. When the S flag is "1", the system stack pointer (SSP) is valid. Set when an interrupt is accepted or when a reset occurs.

#### ● Sticky bit flag (T)

Set to "1" when the data shifted out by the carry contains at least one 1 during execution of a logical right shift instruction or arithmetic right shift instruction. Otherwise, set to "0". Also set to "0" when the shift amount is zero.

#### ● Negative flag (N)

Set to "1" when the MSB is "1" as the result of an arithmetic calculation. Cleared to "0" when the MSB is "0".

#### ● Zero flag (Z)

Set to "1" when the result of an arithmetic calculation is all zeros. Otherwise, set to "0".

- Overflow flag (V)

Set to "1" if a signed numeric value overflows because of an arithmetic calculation. Cleared to "0" if no overflow occurs.

- Carry flag (C)

Set to "1" when there is an overflow from the MSB or an underflow from the LSB because of an arithmetic calculation. Cleared to "0" when there is no overflow or underflow because of an arithmetic calculation.

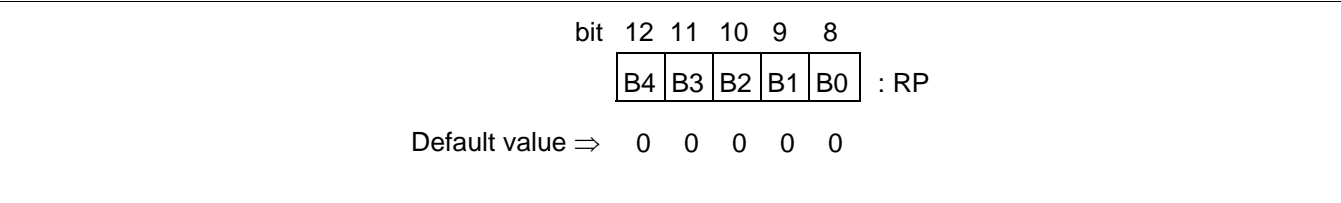
### 3.7.5 Register Bank Pointer (PS: RP)

The register bank pointer (RP) is a register that indicates the first address of the general-purpose register bank currently being used. The RP is used for real address conversion when general-purpose register addressing is used.

#### ■ Register Bank Pointer (RP)

Figure 3.7-10 shows the configuration of the register bank pointer (RP) register.

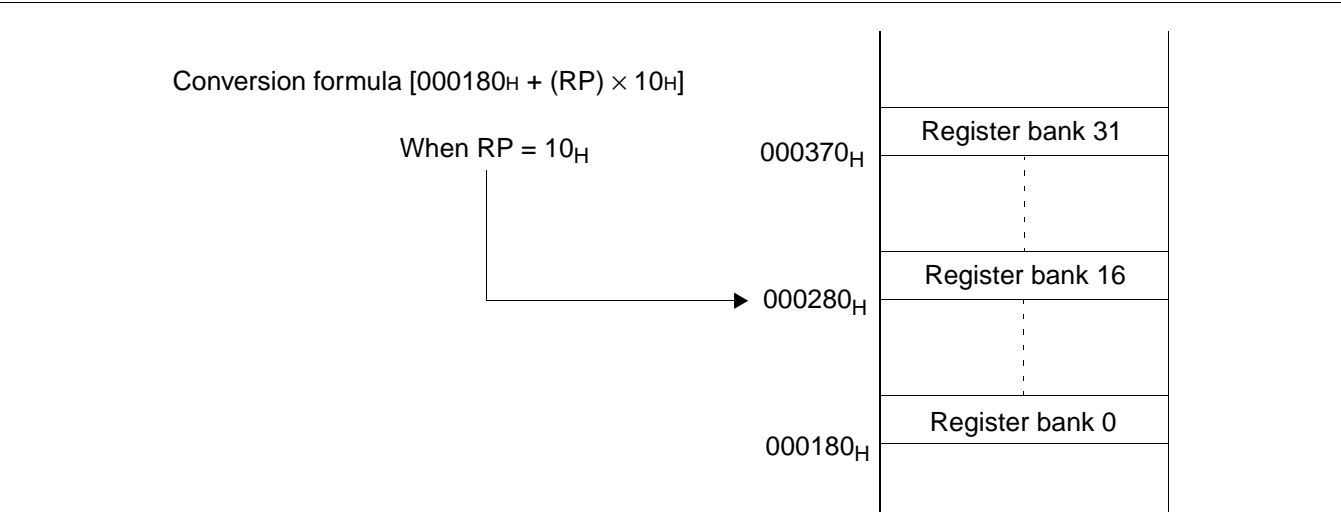
**Figure 3.7-10 Configuration of the Register Bank Pointer (RP)**



#### ■ General-purpose Register Area and Register Bank Pointer

The register bank pointer points to the relationship between the general-purpose register of the F<sup>2</sup>MC-16LX and the address in internal RAM where the general-purpose register exists. Figure 3.7-11 shows the conversion rules used for the relationship between the contents of the RP and the real address.

**Figure 3.7-11 Conversion Rules for Physical Address of General-purpose Register Area**



- Since the RP takes a value from 00<sub>H</sub> to 1F<sub>H</sub>, the first address of the register bank can be set in the range from 000180<sub>H</sub> to 00037F<sub>H</sub>.
- Although an assembler instruction can use an 8-bit immediate value transfer instruction for transfer to the RP, in actuality only the lower 5 bits of the data are used.
- The initial value of the RP register after a reset is 00<sub>H</sub>.

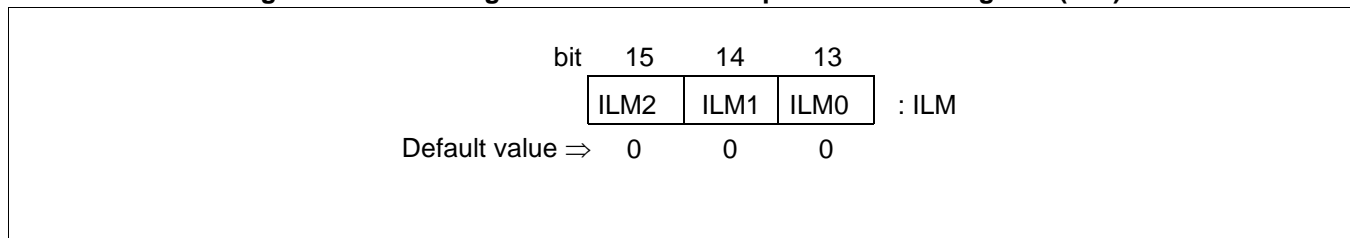
### 3.7.6 Interrupt Level Mask Register (PS: ILM)

The interrupt level mask register (ILM) is a 3-bit register that indicates the level of the interrupt currently accepted by the CPU.

#### ■ Interrupt Level Mask Register (ILM)

Figure 3.7-12 shows the configuration of the interrupt level mask register (ILM). See "CHAPTER 7 INTERRUPT", for details about interrupts.

**Figure 3.7-12 Configuration of the Interrupt Level Mask Register (ILM)**



The interrupt level mask register (ILM) indicates the level of the interrupt currently accepted by the CPU. The level is compared with the value of the IL0 to IL2 bits of the interrupt control register (ICR00 to ICR15) set according to the interrupt request from the peripheral function. If the interrupt enable flag has been set to enable (CCR: I = 1), the CPU processes the instruction only when the value (interrupt level) of the interrupt request is smaller than the value indicated by these bits.

- When an interrupt is accepted, the interrupt level value is set in the interrupt level mask register (ILM). Thereafter, interrupts with the same or lower level are not accepted.
- The interrupt level is set to the highest level, which is the interrupts disabled status, because the interrupt level mask register (ILM) is initialized to all 0's by a reset.
- Although an assembler instruction can use an 8-bit immediate value transfer instruction for transfer to the interrupt level mask register (ILM), only the lower 3 bits of the data are used.
- Interrupt level mask register (ILM) and interrupt level priority

**Table 3.7-3 Interrupt Level Mask Register (ILM) and Interrupt Level Priority**

ILM2	ILM1	ILM0	Interrupt level	Interrupt level priority
0	0	0	0	<div style="text-align: center;"> Highest (interrupts disabled)   ↑   ↓   Lowest </div>
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	
1	1	1	7	

### 3.7.7 Program Counter (PC)

---

**The program counter (PC) is a 16-bit counter that indicates the lower 16 bits of the memory address of the next instruction code to be executed by the CPU.**

---

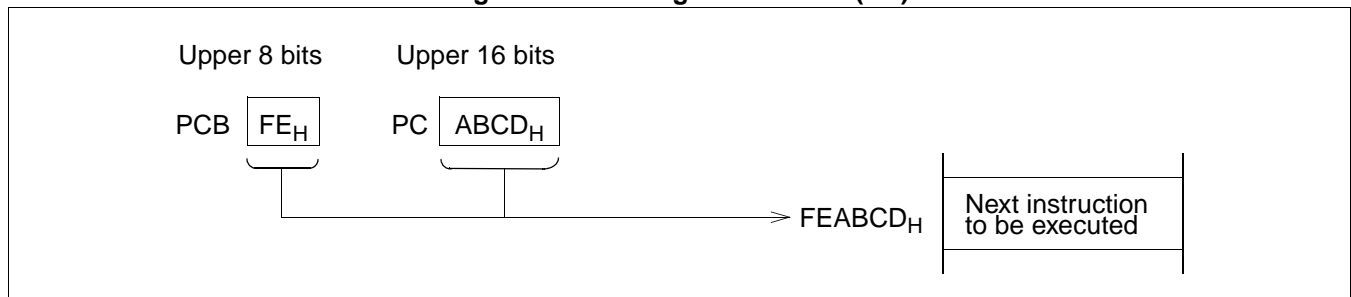
#### ■ Program Counter (PC)

The program bank register (PCB) specifies the upper 8 bits of the address where the next instruction code to be executed by the CPU is stored. The PC specifies the lower 16 bits. Before being used, the actual address is combined to become 24 bits, as shown in Figure 3.7-13.

The contents of the PC are updated by conditional branch instructions, subroutine call instructions, interrupts and resets.

The PC can be used as a bus pointer for reading operands.

**Figure 3.7-13 Program Counter (PC)**



Note:

The PC and PCB cannot be rewritten directly by a program (such as by MOV PC and #FF).

---

### 3.7.8 Direct Page Register (DPR)

**The direct page register (DPR) is an 8-bit register that specifies bits 8 to 15 (addr8 to addr15) of the operand address when a short direct addressing instruction is executed.**

## ■ Direct Page Register (DPR)

As shown in Figure 3.7-14, the DPR specifies bits 8 to 15 (addr8 to addr15) of the operand address when a short direct addressing instruction is executed. The DPR is 8-bits long. The DPR is initialized to 01<sub>H</sub> by a reset. The DPR can be read and written using an instruction.

**Figure 3.7-14 Physical Address Generation by the Direct Page Register (DPR)**

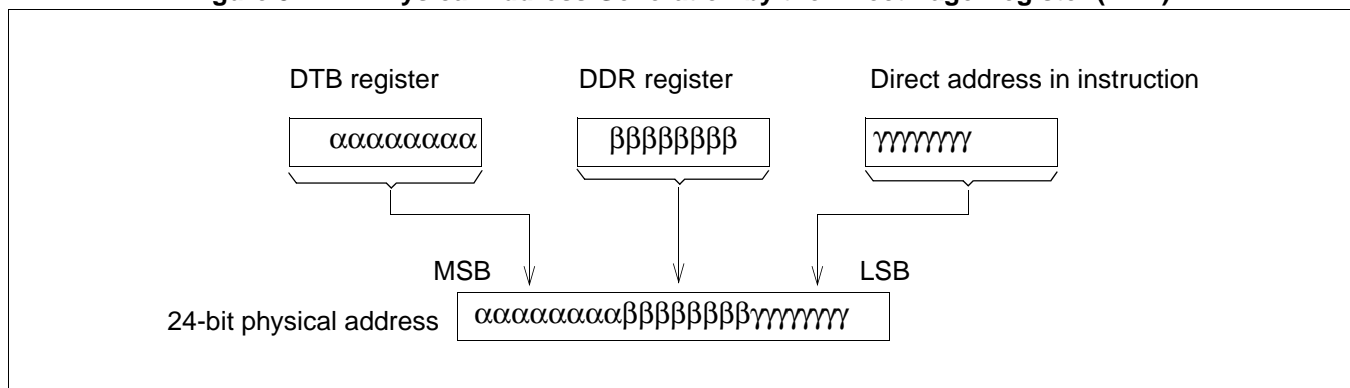
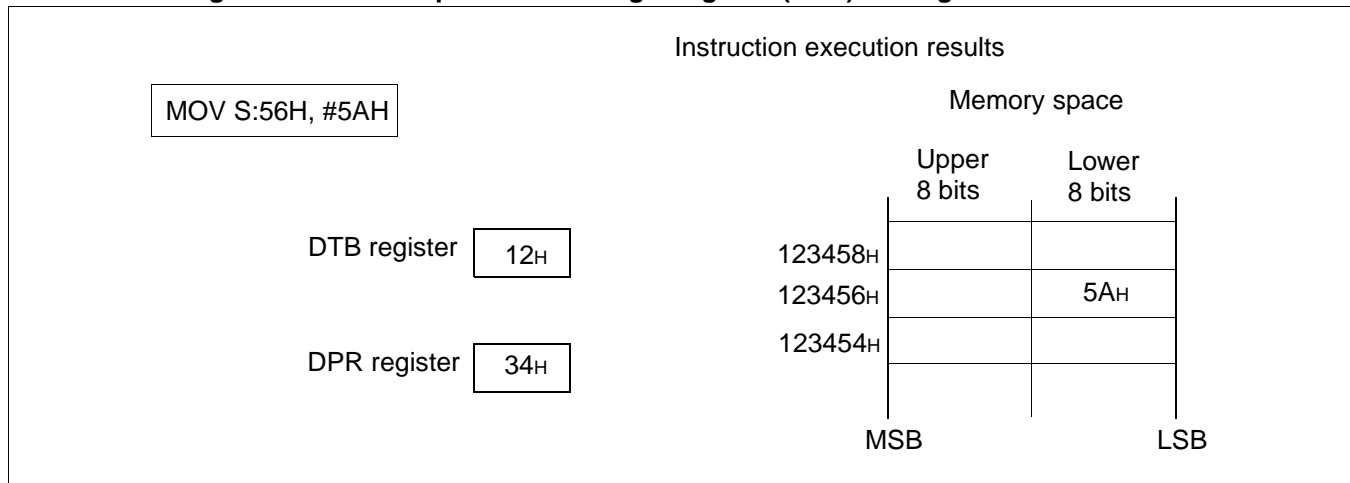


Figure 3.7-15 shows an example of direct page register (DPR) setting and data access.

### Figure 3.7-15 Example of Direct Page Register (DPR) Setting and Data Access





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

---

Bank registers specify the highest 8-bit address by bank addressing. The five bank registers are as follows:

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

The PCB, DTB, USB, SSB and ADB registers indicate the individual memory banks where the program space, data space, user stack space, system stack space and additional space are located.

---

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

##### ● Program bank register (PCB)

The PCB is a bank register that specifies the program (PC) space.

The PCB is rewritten when a software interrupt instruction is executed, when the JMPP, CALLP, RETP and RETI instructions that branch anywhere within the 16-megabyte space are executed, or when a hardware interrupt or exception occurs.

##### ● Data bank register (DTB)

The DTB is a bank register that specifies the data (DT) space.

##### ● User stack bank register (USB), system stack bank register (SSB)

The USB and SSB are bank registers that specify the stack (SP) space. Whether the USB or the SSB is used depends on the S flag value in the processor status (PS: CCR). See "3.7.2 Stack Pointers (USP, SSP)", for details.

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

The ADB is a bank register that specifies the additional (AD) space.

##### ● Bank setting and data access

All bank registers are byte length. The PCB is initialized to FF<sub>H</sub> by a reset. The other bank registers are initialized to 00<sub>H</sub> by a reset. The PCB can be read, but cannot be written to.

The other bank registers can be read and written to.

---

Note:

The MB90460/465 series supports up to the memory space contained in the device.  
See "3.4.2 Address Specification by Bank Addressing", for the operation of each register.

---

## 3.8 General-purpose Registers

The general-purpose registers are a memory block allocated in RAM at 000180<sub>H</sub> to 00037F<sub>H</sub> as banks, each of which consists of eight 16-bit segments.

The general-purpose registers can be used as general-purpose 8-bit registers (byte registers R0 to R7), 16-bit registers (word registers RW0 to RW7) or 32-bit registers (long-word registers RL0 to RL7).

General-purpose registers can access RAM with a short instruction at high speed. Since general-purpose registers are blocked into register banks, protection of register contents and division into function units can readily be performed. When a general-purpose register is used as a long-word register, it can be used as a linear pointer that directly accesses the entire space.

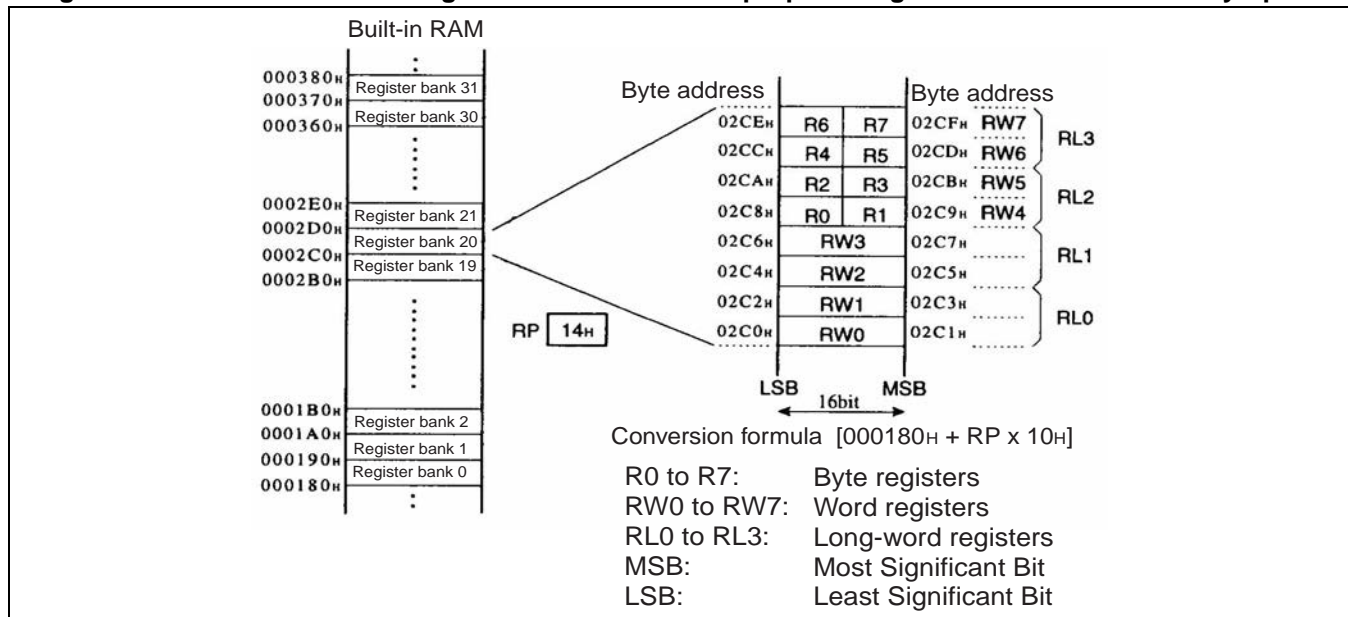
### ■ Configuration of a General-purpose Register

All general-purpose registers exist in RAM at 000180<sub>H</sub> to 00037F<sub>H</sub> and are configured as 32 banks. The register bank pointer (RP) specifies the bank that is to be used for a general-purpose register. The RP points to the bank currently being used.

The RP determines the first address of each bank with the following formula:

Address of first general-purpose register = 000180<sub>H</sub> + RP × 10<sub>H</sub> Figure 3.8-1 shows the location and configuration of the general-purpose register banks in the memory space.

**Figure 3.8-1 Location and Configuration of the General-purpose Register Banks in the Memory Space**



Note:

The register bank pointer (RP) is initialized to 00<sub>H</sub> after a reset.

## ■ Register Bank

A register bank can be used as general-purpose registers (byte registers R0 to R7, word registers RW0 to RW7, long-word registers RL0 to RL3) for various arithmetic operations and pointers. A long-word register can be used as a linear pointer that directly accesses the entire memory space.

The contents of the register bank, like ordinary RAM, are not initialized by a reset. The status before a reset is retained. At power-on, however, the contents are undefined. Table 3.8-1 lists the typical functions of general-purpose registers.

**Table 3.8-1 Typical Functions of General-purpose Registers**

Register name	Function
R0 to R7	Used as an operand in various instructions ( <b>Note</b> ) R0 is also used as a barrel shift counter and an instruction normalization counter
RW0 to RW7	Used as a pointer Used as an operand in various instructions ( <b>Note</b> ) RW0 is used also as a string instruction counter
RL0 to RL3	Used as a long pointer Used as an operand in various instructions

## 3.9 Prefix Codes

---

Prefix codes are placed before an instruction to partially change the operation of the instruction. The three types of prefix codes are as follows:

- Bank select prefix (PCB, DTB, ADB, SPB)
  - Common register bank prefix (CMR)
  - Flag change suppression prefix (NCC)
- 

### ■ Prefix Codes

#### ● Bank select prefix (PCB, DTB, ADB, SPB)

A bank select prefix is placed before an instruction to select the memory space to be accessed by the instruction regardless of the addressing method.

#### ● Common register bank prefix (CMR)

The common register bank prefix is placed before an instruction that accesses a register bank to change the register accessed by the instruction to the common bank (register bank selected when  $RP = 0$ ) at 000180<sub>H</sub> to 00018F<sub>H</sub> regardless of the current register bank pointer (RP) value.

#### ● Flag change suppression prefix (NCC)

The flag change suppression prefix code is placed before an instruction to suppress a flag change accompanying the execution of the instruction.

### 3.9.1 Bank Select Prefix (PCB, DTB, ADB, SPB)

A bank select prefix is placed before an instruction to select the memory space accessed by the instruction regardless of the addressing method.

#### ■ Bank Select Prefixes (PCB, DTB, ADB, SPB)

The memory space used for data access is defined for each addressing method. If a bank select prefix is placed before an instruction, the memory space accessed by the instruction can be selected regardless of the addressing method. Table 3.9-1 lists the bank select prefix codes and selected memory spaces.

**Table 3.9-1 Bank Select Prefix Codes and selected Memory Spaces**

Bank select prefix	Selected space
PCB	Program space
DTB	Data space
ADB	Additional space
SPB	When the value of the S flag in the condition code register (CCR) is "0" and the user stack space is "1", the system stack space is used.

If a bank select prefix is used, some instructions perform an unexpected operation.

Table 3.9-2 lists the instructions that are not affected by bank select prefix codes. Table 3.9-3 lists instructions that require caution when they are used.

**Table 3.9-2 Instructions not affected by Bank Select Prefix Codes**

Instruction type	Instruction		Effect of bank select prefix
String instruction	MOVS SCEQ FIL	MOVSW SCWEQ SFILSW	The bank register specified by the operand is used whether or not a prefix is used.
Stack operation	PUSHW	POPW	When the S flag is "0", the user stack bank (USB) is used whether or not there is a prefix. When the S flag is "1", the system stack bank (SSB) is used regardless of whether a prefix is used.
I/O access instruction	MOV A MOVW A, io MOV io, A MOV io, #imm8 MOVB A, io : bp SETB io : bp BBC io : bp, rel WBTC io, bp	MOVX A, io MOVW io, A MOVW io, #imm16 MOVB io : bp, A CLRB io : bp BBS io : bp, rel WBTS io : bp	The I/O space (000000 <sub>H</sub> to 0000FF <sub>H</sub> ) is accessed whether or not there is a prefix.
Interrupt return instruction	RETI		The system stack bank (SSB) is used whether or not a prefix is used.

**Table 3.9-3 Instructions whose use requires Caution when Bank Select Prefix Codes are used**

Instruction type	Instruction		Explanation
Flag change instruction	AND OR	CCR, #imm8 CCR, #imm8	The effect of the prefix extends to the next instruction.
ILM setting instruction	MOV	ILM, #imm8	The effect of the prefix extends to the next instruction.
PS return instruction	POPW	PS	Do not place a bank select prefix before the PS return instruction.

### 3.9.2 Common Register Bank Prefix (CMR)

The common register bank (CMR) prefix is placed before an instruction that accesses a register bank to change the register accessed by the instruction to the common bank (register bank selected when  $RP = 0$ ) at  $000180_H$  to  $00018F_H$  regardless of the current register bank pointer (RP) value.

#### ■ Common register bank prefix (CMR)

To facilitate data exchange between multiple tasks, a relatively simple means of accessing a fixed register bank regardless of the current register bank pointer (RP) value is necessary. This is the reason that the F<sup>2</sup>MC-16LX provides a common register bank for tasks, which is called the common bank. The common bank is located at address  $000180_H$  to  $00018F_H$ .

If the common register bank prefix (CMR) is placed before an instruction that accesses a register bank, registers accessed by the instruction can be changed to the common bank (register bank selected when  $RP = 0$ ) at  $000180_H$  to  $00018F_H$  regardless of the current register bank pointer (RP) value.

Note that caution is required when this prefix is used with the instructions listed in Table 3.9-4.

**Table 3.9-4 Instructions whose use requires Caution when the Common Register Bank Prefix (CMR) is used**

Instruction type	Instruction	Explanation
String instruction	MOVS                      MOVSW SCEQ                      SCWEQ FILS                      FILSW	Do not place the CMR prefix before the string instruction.
Flag change instruction	AND   CCR, #imm8    OR   CCR, #imm8	The effect of the prefix extends to the next instruction.
PS return instruction	POPW   PS	The effect of the prefix extends to the next instruction.
ILM setting instruction	MOV   ILM, #imm8	The effect of the prefix extends to the next instruction.

### 3.9.3 Flag Change Suppression Prefix (NCC)

The flag change suppression prefix (NCC) code is placed before an instruction to suppress a flag change accompanying the execution of the instruction.

#### ■ Flag Change Suppression Prefix (NCC)

The flag change suppression prefix (NCC) is used to suppress unnecessary flag changes. If a flag change suppression prefix code is placed before an instruction, a flag change accompanying the execution of the instruction is suppressed. Changes of the T, N, Z, V and C flags are suppressed.

Note that caution is required when this prefix is used with the instructions listed in Table 3.9-5.

**Table 3.9-5 Instructions whose use requires Caution when the Flag Change Suppression Prefix (NCC) is used**

Instruction type	Instruction	Explanation
String instruction	MOVS                  MOVSW SCEQ                  SCWEQ FILS                  FILSW	Do not place the NCC prefix before the string instruction.
Flag change instruction	AND CCR, #imm8   OR CCR, #imm8	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used. The effect of prefix extends to the next instruction.
PS return instruction	POPW   PS	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used. The effect of prefix extends to the next instruction.
ILM setting instruction	MOV                  ILM, #imm8	The effect of prefix extends to the next instruction.
Interrupt instruction Interrupt return instruction	INT    #vct8           INT9 INT    adder16        INTP    addr24 RETI	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used.
Context switch instruction	JCTX@A	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used.



### 3.9.4 Restrictions on Prefix Codes

The following three restrictions are imposed on the use of prefix codes:

- Interrupt/hold requests are not accepted during the execution of prefix codes and interrupt/hold suppression instructions.
- If a prefix code is placed before an interrupt/hold instruction, the effect of the prefix code is delayed.
- If consecutively placed prefix codes conflict, the last prefix code is valid.

#### ■ Prefix Codes and Interrupt/hold Suppression Instructions

Table 3.9-6 lists the interrupt/hold suppression instructions and prefix codes that have restrictions.

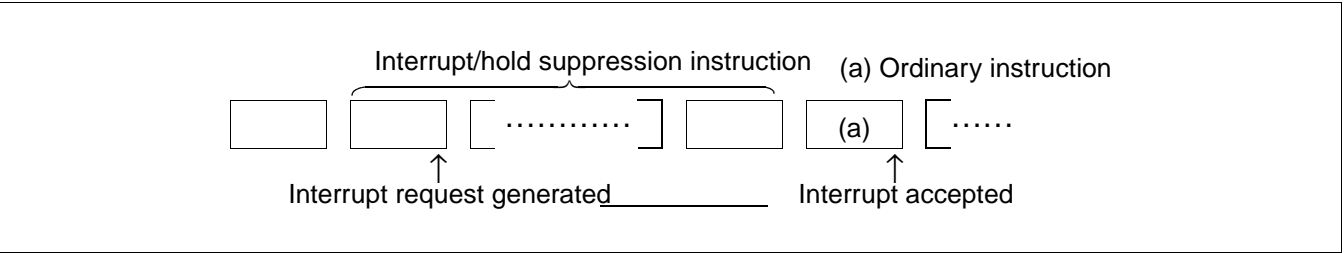
**Table 3.9-6 Prefix Codes and Interrupt/hold Suppression Instructions**

	Prefix codes	Interrupt/hold suppression instructions (instructions that delay the effect of prefix codes)
Instructions that do not accept interrupt and hold requests	PCB DTB ADB SPB CMR NCC	MOV ILM, #imm8 OR CCR, #imm8 AND CCR, #imm8 POPW PS

#### ● Interrupt/hold suppression

As shown in Figure 3.9-1, an interrupt or hold request generated during the execution of prefix codes and interrupt/hold instructions is not accepted. The interrupt/hold is not processed until the first instruction that is not governed by a prefix code or that is not an interrupt/hold suppression instruction is executed.

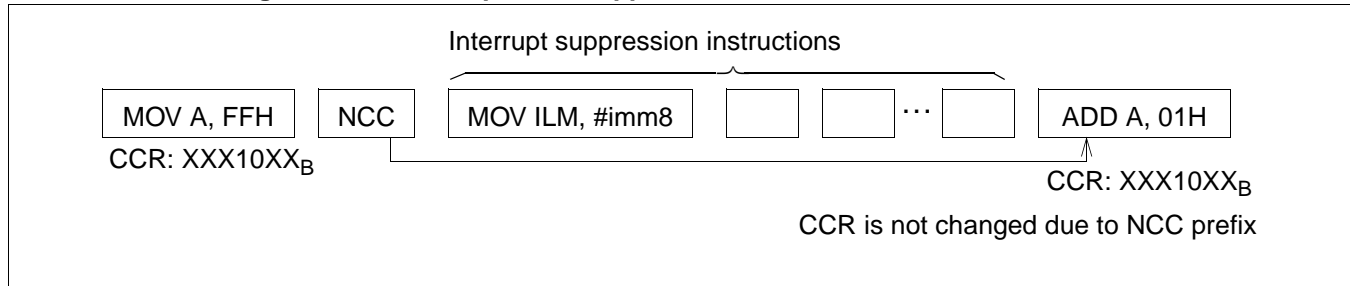
**Figure 3.9-1 Interrupt/hold Suppression**



● Delay of the effect of prefix codes

As shown in Figure 3.9-2, if a prefix code is placed before an interrupt/hold suppression instruction, the prefix code takes effect with the first instruction executed after the interrupt/hold suppression instruction.

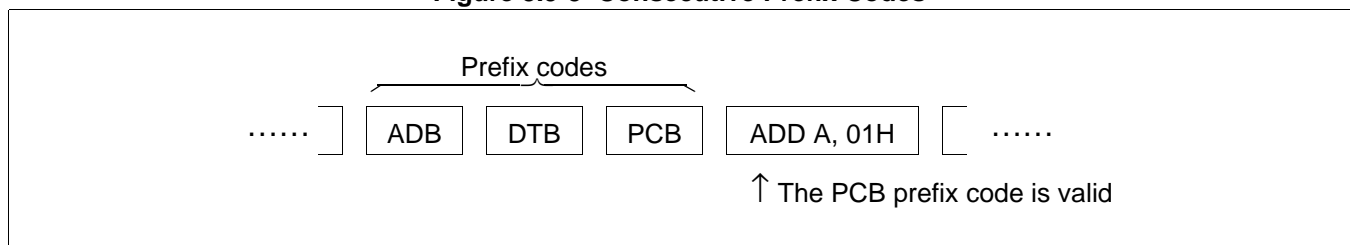
**Figure 3.9-2 Interrupt/hold Suppression Instructions and Prefix Codes**



■ Consecutive Prefix Codes

As shown in Figure 3.9-3, when consecutive conflicting prefix codes (PCB, ADB, DTB and SPB) are specified, the last prefix code is valid.

**Figure 3.9-3 Consecutive Prefix Codes**





# **CHAPTER 4**

---

# ***RESET***

**This chapter describes the reset for the MB90460/465 series microcontrollers.**

- 4.1 Reset
- 4.2 Reset Causes and Oscillation Stabilization Wait Intervals
- 4.3 External Reset Pin
- 4.4 Reset Operation
- 4.5 Reset Cause Bits
- 4.6 Status of Pins in a Reset

## 4.1 Reset

If a reset cause is generated, the CPU immediately stops the current execution process and waits for the reset to be cleared. When the reset is cleared, the CPU begins processing at the address indicated by the reset vector.

There are four causes of a reset:

Power-on reset

Watchdog timer overflow

External reset request via the RSTX pin

Software reset request

### ■ Reset Causes

Table 4.1-1 lists the reset causes.

**Table 4.1-1 Reset Causes**

Type of reset	Cause	Machine clock	Watchdog timer	Oscillation stabilization wait
External pin	Low level input to RSTX pin	Previous state retained	Previous state retained	No
Software	A “0” is written to the RST bit of the low power consumption mode control register (LPMCR)	Previous state retained	Previous state retained	No
Watchdog timer	Watchdog timer overflow	MCLK	Stop	Yes
Power-on	When the power is turned on	MCLK	Stop	Yes

MCLK: Main clock (oscillation clock frequency divided by 2)

#### ● External reset

An external reset is generated by the L level input to an external reset pin (RSTX pin). The minimum required period of the L level input to the RSTX pin is 16 machine cycles ( $16/\phi$ ). The oscillation stabilization wait interval is not required for external resets.

#### Reference:

For external reset requests via the RSTX pin, if the reset cause is generated during a write operation (during the execution of a transfer instruction such as MOV), the CPU waits for the reset to be cleared after the instruction is completed. The normal write operation is therefore completed even though a reset is input concurrently.

Note, however, that waiting for the reset to be cleared may start before the transfer of the contents of a counter specified by a string-processing instruction (such as MOVS) is completed.

### ● Software reset

A software reset is an internal reset of three machine cycles ( $3/\phi$ ) generated by writing "0" to the RST bit of the low power consumption mode control register (LPMCR). The oscillation stabilization wait interval is not required for software resets.

### ● Watchdog timer reset

A watchdog timer reset is generated by a watchdog timer overflow that occurs when a "0" is not written to the WTE bit of the watchdog timer control register (WDTC) within a given time after the watchdog timer is activated. The oscillation stabilization wait interval can be set by the clock selection register (CKSCR).

### ● Power-on reset

A power-on reset is generated when the power is turned on. The oscillation stabilization wait interval is fixed at  $2^{18}$  oscillation clock cycles ( $2^{18}/\text{HCLK}$ ). After the oscillation stabilization wait interval has elapsed, the reset is executed.

See also

- Definition of clocks
  - HCLK: Oscillation clock frequency
  - MCLK: Main clock frequency
  - $\phi$ : Machine clock (CPU operating clock) frequency
  - $1/\phi$ : Machine cycle (CPU operating clock cycle)

See "5.1 Clock", for details.

## 4.2 Reset Causes and Oscillation Stabilization Wait Intervals

The F<sup>2</sup>MC-16LX has four reset causes. The oscillation stabilization wait interval for a reset depends on the reset cause.

### ■ Reset Causes and Oscillation Stabilization Wait Intervals

Table 4.2-1 and Figure 4.2-1 summarize reset causes and oscillation stabilization wait intervals.

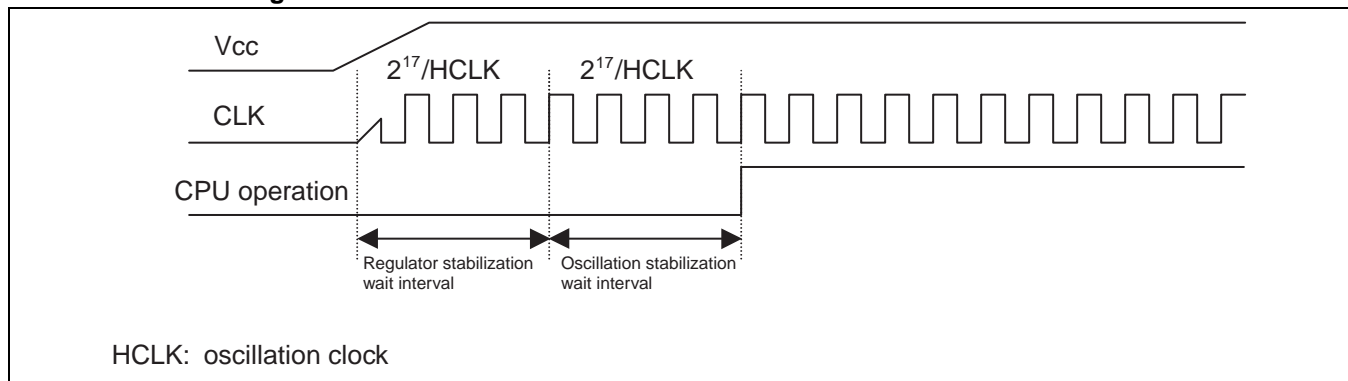
**Table 4.2-1 Reset Causes and Oscillation Stabilization Wait Intervals**

Reset cause	Oscillation stabilization wait interval The corresponding time interval for an oscillation clock frequency of 4 MHz is given in parentheses.
Power-on reset	$2^{18}/\text{HCLK}$ (approximately 65.54 ms)
Watchdog timer	$2^{17}/\text{HCLK}$ (approximately 32.77 ms)
External reset via the RSTX pin	None. However the WS1 & WS0 bits are initialized to “11”.
Software reset	None. However the WS1 & WS0 bits are initialized to “11”.

HCLK: Oscillation clock frequency, source oscillation.

Figure 4.2-1 shows the oscillation stabilization wait interval of the product at power-on reset.

**Figure 4.2-1 Oscillation Stabilization Wait Interval at Power-on Reset**



Note:

Ceramic and crystal oscillators generally require an oscillation stabilization wait interval of a few to several dozen milliseconds until they stabilize at their natural frequency. Be sure to set a proper oscillation stabilization wait interval for the specific oscillator used.

See "4.2 Reset Causes and Oscillation Stabilization Wait Intervals" for detail about Oscillation Stabilization Wait Interval.

### ■ Oscillation Stabilization Wait and Reset State

A reset operation in response to a power-on reset and other externally activated resets during stop mode and hardware standby mode is performed after the oscillation stabilization wait interval has elapsed. This time interval is generated by the time-base timer. If the external reset has not been cleared after the interval, the reset operation is performed after the external reset is cleared.

## 4.3 External Reset Pin

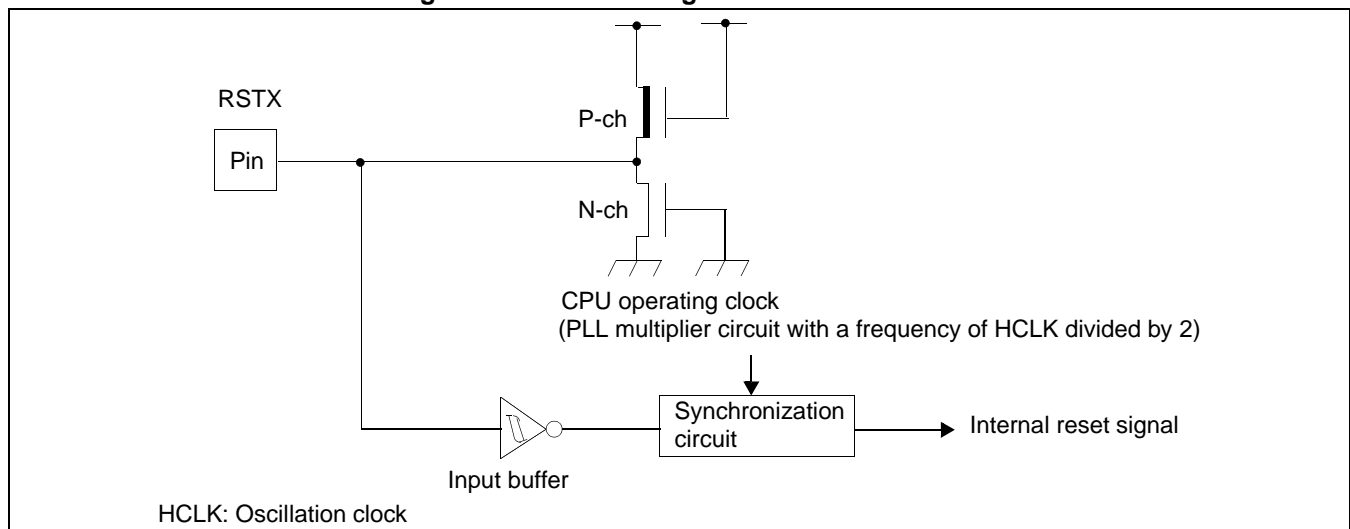
The external reset pin (RSTX pin) is a dedicated pin for inputting, with an L level signal, a reset and generating an internal reset by the L level input.

For MB90460/465 series microcontrollers, resets are generated in synchronization with the CPU operating clock. Asynchronous resets are generated only for the external terminals.

### ■ Block Diagrams of the External Reset Pin

- Block diagram of internal reset

Figure 4.3-1 Block Diagram of Internal Reset



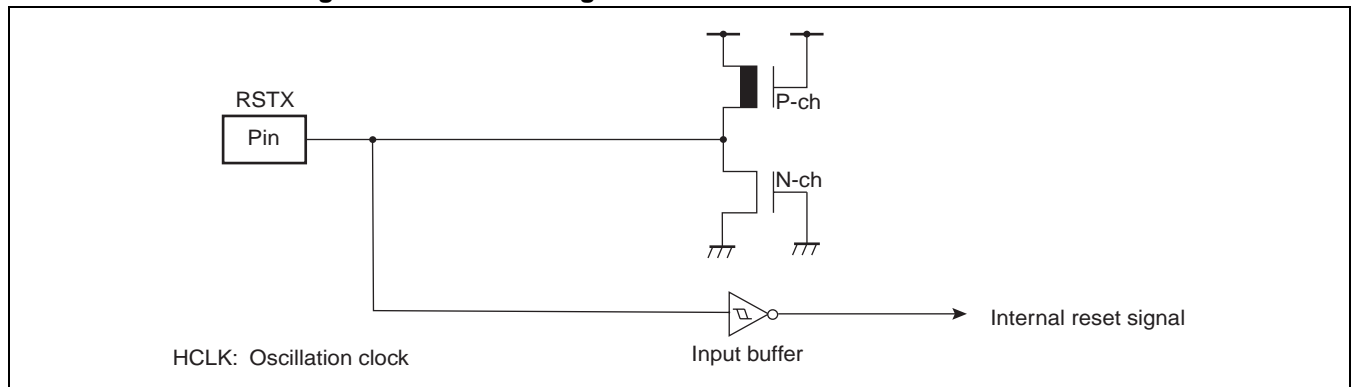
#### Note:

Inputs to the RSTX pin are accepted during cycles in which memory is not affected to prevent memory from being destroyed by a reset during a write operation.  
A clock is required to initialize the internal circuit. In particular, an operation with an external clock requires clock input together with reset input.



- Block diagram of internal reset for external pin

**Figure 4.3-2 Block Diagram of Internal Reset for External Pin**



## 4.4 Reset Operation

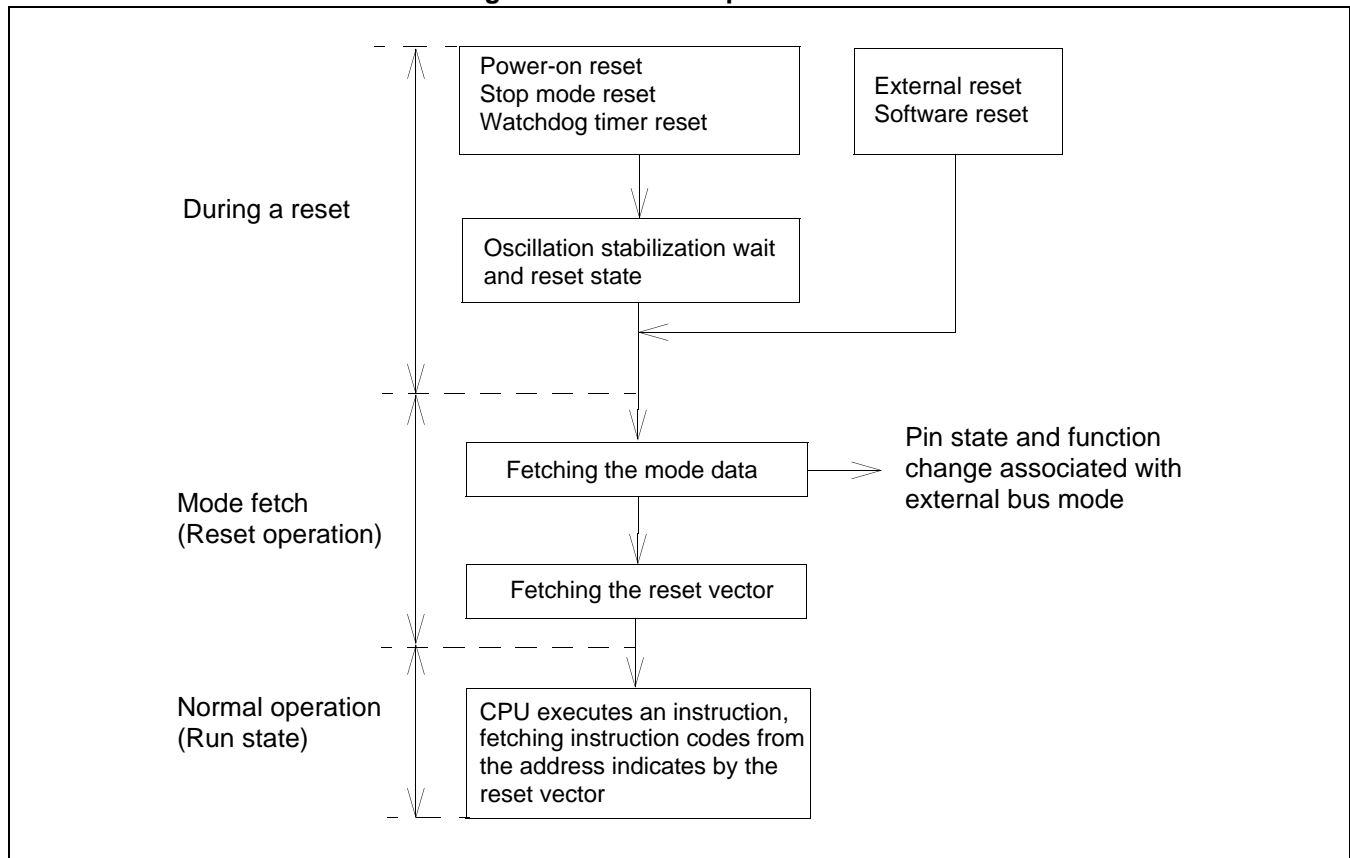
When a reset is cleared, the memory locations from which the mode data and the reset vector are read are selected according to the setting of the mode pins, and the mode setting data is fetched. Mode setting data determines the CPU operating mode and the execution start address after a reset operation ends.

For power-on or recovery from stop mode by a reset, the mode is fetched after the oscillation stabilization wait time has elapsed.

### ■ Overview of Reset Operation

Figure 4.4-1 shows the reset operation flow.

**Figure 4.4-1 Reset Operation Flow**



### ■ Mode Pins

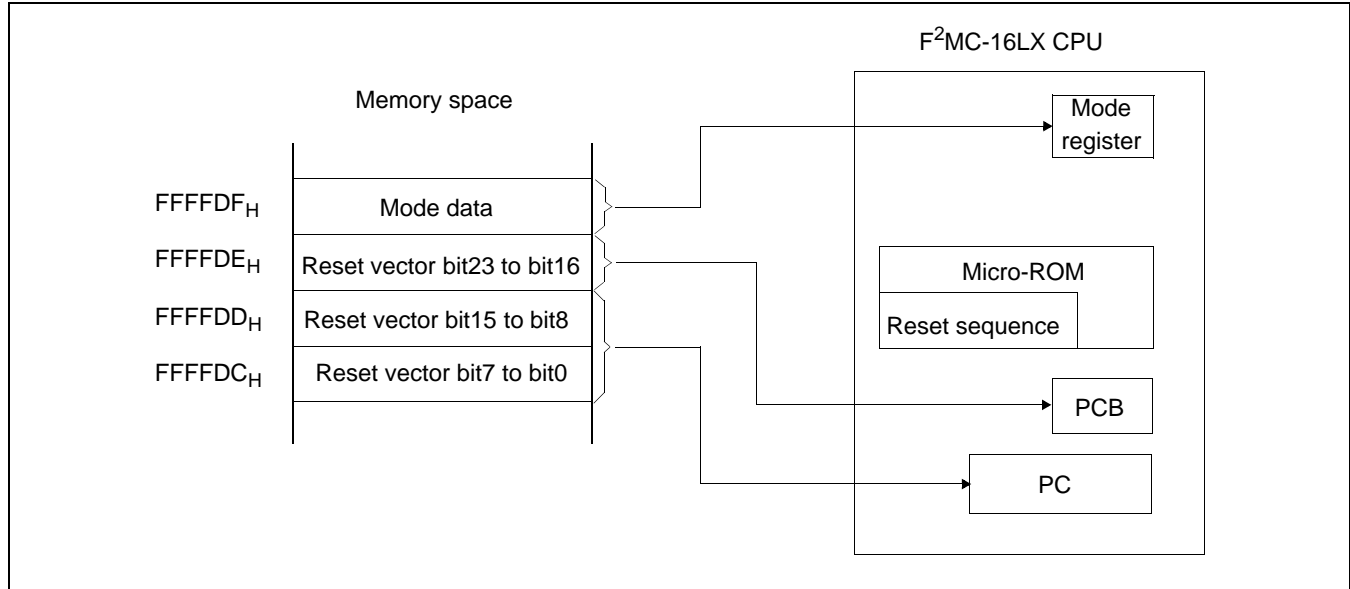
Setting the mode pins (MD0 to MD2) specifies how to fetch the reset vector and the mode data. Fetching the reset vector and the mode data is performed in the reset sequence. See "8.1 Mode Setting", for details about mode pins.

## ■ Mode Fetch

When the reset is cleared, the CPU transfers the reset vector and the mode data stored in the hardware memory to the appropriate registers in the CPU core. The reset vector and mode data are allocated to the four bytes from FFFFDC<sub>H</sub> to FFFFDF<sub>H</sub>. The CPU outputs these addresses to the bus immediately after the reset is cleared and fetches the reset vector and mode data. Using mode fetching, the CPU can begin processing at the address indicated by the reset vector.

Figure 4.4-2 shows the transfer of the reset vector and mode data.

**Figure 4.4-2 Transfer of Reset Vector and Mode Data**



### Reference:

Whether the reset vector and the mode data are read from internal ROM or from external memory is specified by the setting of the mode pins. If external vector mode is specified by the mode pin settings, the CPU will always read the reset vector and the mode data from external memory instead of from internal ROM. If single-chip mode and internal ROM external bus mode are used, setting the mode pins to specify internal vector mode is recommended.

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

Only the reset operation changes the contents of the mode register. The mode register setting is valid after a reset operation. See "8.1 Mode Setting", for details about mode data.

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

The execution start address after the reset operation ends is written as the reset vector. Execution starts at the address contained in the reset vector.

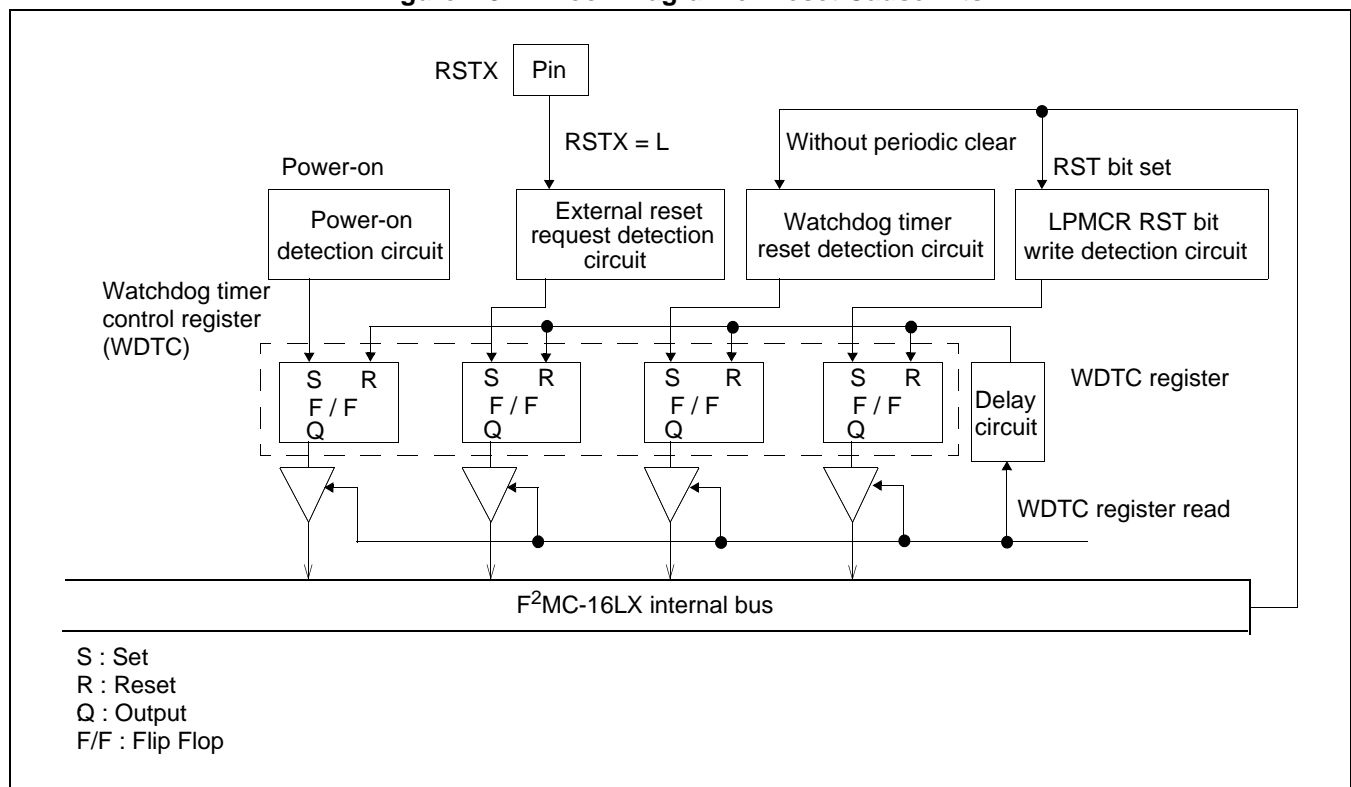
## 4.5 Reset Cause Bits

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

### ■ Reset Cause Bits

As shown in Figure 4.5-1, a flip-flop is associated with each reset cause. The contents of the flip-flops are obtained by reading the watchdog timer control register (WDTC). If it is necessary to identify the cause of a reset after the reset has been cleared, the value read from the WDTC should be processed by the software and a branch made to the appropriate program.

Figure 4.5-1 Block Diagram of Reset Cause Bits



## ■ Correspondence between Reset Cause Bits and Reset Causes

Figure 4.5-2 shows the configuration of the reset cause bits of the watchdog timer control register (WDTC).

Table 4.5-1 maps the correspondence between the reset cause bits and reset causes.

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

Watchdog timer control register								
Address : 0000A8H	bit 7	6	5	4	3	2	1	0
	PONR	-	WRST	ERST	SRST	WTE	WT1	WT0
Read/write ⇒	(R)	(-)	(R)	(R)	(R)	(W)	(W)	(W)
Default value ⇒	(X)	(-)	(X)	(X)	(X)	(1)	(1)	(1)

WDTC

**Table 4.5-1 Correspondence between Reset Cause Bits and Reset Causes**

Reset cause	PONR	WRST	ERST	SRST
Power-on reset	1	X	X	X
Watchdog timer overflow	*	1	*	*
External reset request via RSTX pin	*	*	1	*
Software reset request	*	*	*	1

\* : Previous state retained

X: Undefined

## ■ Notes about Reset Cause Bits

### ● Multiple reset causes generated at the same time

When multiple reset causes are generated at the same time, the corresponding reset cause bits of the watchdog timer control register (WDTC) are set to "1".

If, for example, an external reset request via the RSTX pin and the watchdog timer overflow occur at the same time, both the ERST bit and the WRST bit are set to "1".

### ● Power-on reset

For a power-on reset, the PONR bit is set to "1", but all other reset cause bits are undefined.

Consequently, program the software so that it will ignore all reset cause bits except the PONR bit if it is "1".

### ● Clearing the reset cause bits

The reset cause bits are cleared only when the watchdog timer control register (WDTC) is read. Any bit that corresponds to a reset cause that has already been generated once is not cleared even though another reset is generated (its setting of "1" is retained).

## 4.6 Status of Pins in a Reset

---

**This section describes the status of pins when a reset occurs.**

---

### ■ Status of Pins during a Reset

The status of pins during a reset depends on the settings of mode pins ( $MD2$  to  $MD0 = 011_B$ ).

- When internal vector mode has been set:

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

### ■ Status of Pins after Mode Data is read

The status of pins after mode data has been read depends on the mode data ( $M1$  and  $M0 = 00_B$ ).

- When single-chip mode has been selected ( $M1, M0 = 00_B$ ):

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

---

Note:

For those pins that change to high impedance when a reset cause is generated, take care that devices connected to them do not malfunction.

---

See Table 6.7-1 for information about the state of pins during a reset



# **CHAPTER 5**

---

# **CLOCK**

**This chapter describes the clock used by MB90460/465 series microcontrollers.**

5.1 Clock

5.2 Block Diagram of the Clock Generation Block

5.3 Clock Selection Register (CKSCR)

5.4 Clock Mode

5.5 Oscillation Stabilization Wait Interval

5.6 Connection of an Oscillator or an External Clock to the Microcontroller



## 5.1 Clock

---

**The clock generation block controls the operation of the internal clock that controls operation of the CPU and peripheral functions. This internal clock is called the machine clock. One internal clock cycle is regarded as one machine cycle.**

**Other clocks include a clock generated by source oscillation, called an oscillation clock, and a clock generated by the internal PLL oscillation, called a PLL clock.**

---

### ■ Clock

The clock generation block contains the oscillation circuit that generates the oscillation clock. An external oscillator is attached to this circuit. The oscillation clock can also be supplied by inputting an external clock to the clock generation block.

The clock generation block also contains the PLL clock multiplier circuit, which generates four clocks that are multiples of the oscillation clock.

The clock generation block controls the oscillation stabilization wait interval and PLL clock multiplication as well as controls internal clock operation by changing the clock with a clock selector.

#### ● Oscillation clock (HCLK)

The oscillation clock is generated either from an external oscillator attached to the oscillation circuit or by input of an external clock.

#### ● Main clock (MCLK)

The main clock, which is the oscillation clock divided by 2, supplies the clock input to the time-base timer and the clock selector.

#### ● PLL clock (PCLK)

The PLL clock is obtained by multiplying the oscillation clock with the internal PLL clock multiplier circuit (PLL oscillation circuit). Selection can be made from among four different PLL clocks.

#### ● Machine clock ( $\phi$ )

The machine clock controls the operation of the CPU and peripheral functions. One clock cycle is regarded as one machine cycle ( $1/\phi$ ). An operating machine clock can be selected from among the main clock that is generated from the source clock frequency divided by 2 and the four clocks that are multiples of the source clock frequency.

---

#### Note:

Although an oscillation clock of 3 MHz to 32 MHz can be generated when the operating voltage is 5 V, the maximum operating frequency for the CPU and peripheral functions is 16 MHz. If a frequency multiplier rate exceeding the operating frequency is specified, devices will not operate correctly.

If, for example, a source oscillation of 16 MHz is generated, only a multiplier of 1 can be specified.

---

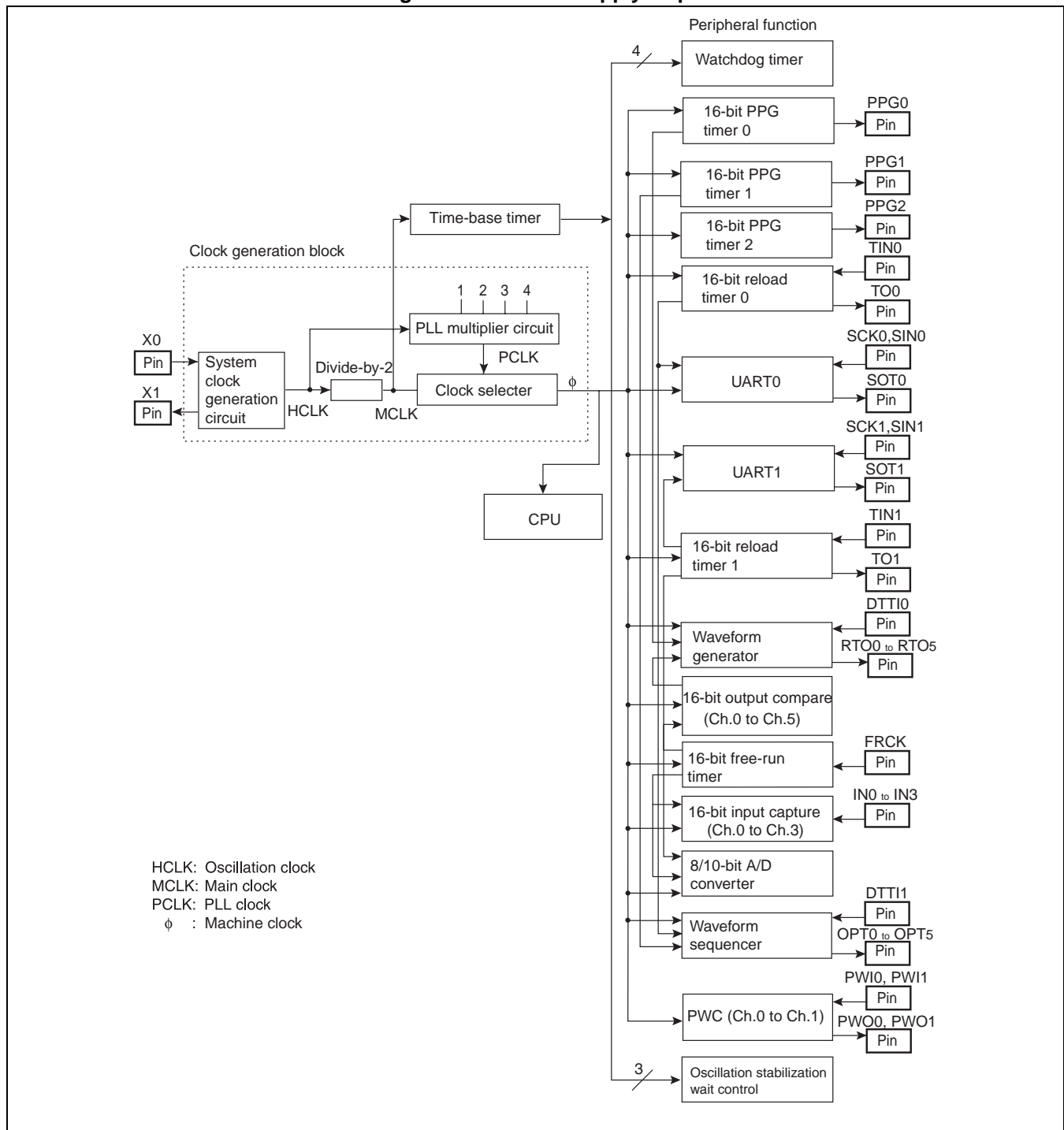
A PLL oscillation of 3 to 16 MHz is possible, but this range depends on the operating voltage and multiplier. See "Data Sheet", for details.

## ■ Clock Supply Map

Since the machine clock generated in the clock generation block is supplied as the clock that controls operation of the CPU and peripheral functions, the operation of the CPU and peripheral functions is affected by switching of the main clock and the PLL clock (clock mode) and a change in the PLL clock multiplier.

Since some peripheral functions receive frequency-divided output from the time-base timer, a peripheral unit can select the clock best suited for its operation. Figure 5.1-1 shows the clock supply map.

**Figure 5.1-1 Clock Supply Map**



## 5.2 Block Diagram of the Clock Generation Block

The clock generation block consists of five blocks:

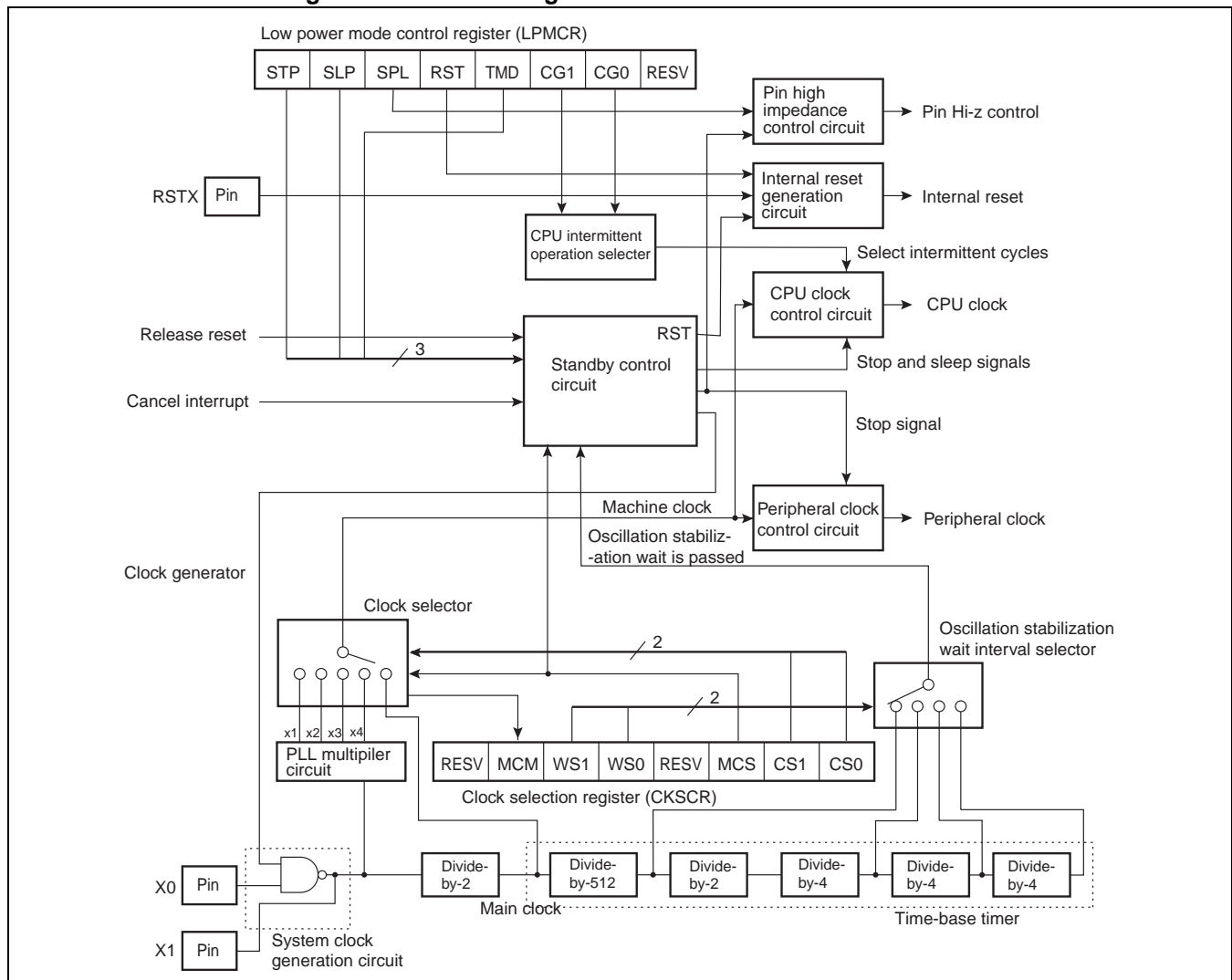
- System clock generation circuit
- PLL multiplier circuit
- Clock selector
- Clock selection register (CKSCR)
- Oscillation stabilization wait interval selector

### ■ Block Diagram of the Clock Generation Block

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

Figure 5.2-1 also includes the standby control circuit and time-base timer circuit.

**Figure 5.2-1 Block Diagram of the Clock Generation Block**



- System clock generation circuit

The system clock generation circuit generates an oscillation clock (HCLK) from an external oscillator attached to it. Alternatively, an external clock can be input to this circuit.

- PLL multiplier circuit

The PLL multiplier circuit multiplies the oscillation clock through PLL oscillation and supplies a clock that is a multiple of the frequency to the CPU clock selector.

- Clock selector

From among the main clock and four different PLL clocks, the clock selector selects the clock that is supplied to the CPU and peripheral clock control circuits.

- Clock selection register (CKSCR)

The clock selection register is used to set switching between the oscillation clock and a PLL clock, selection of an oscillation stabilization wait interval, and selection of a PLL clock multiplier.

- Oscillation stabilization wait interval selector

This selector selects an oscillation stabilization wait interval for the oscillation clock when stop mode is released or when a watchdog timer reset occurs. Selection is made from among three kinds time-base timer output. In all other cases, an oscillation stabilization wait interval is not selected.

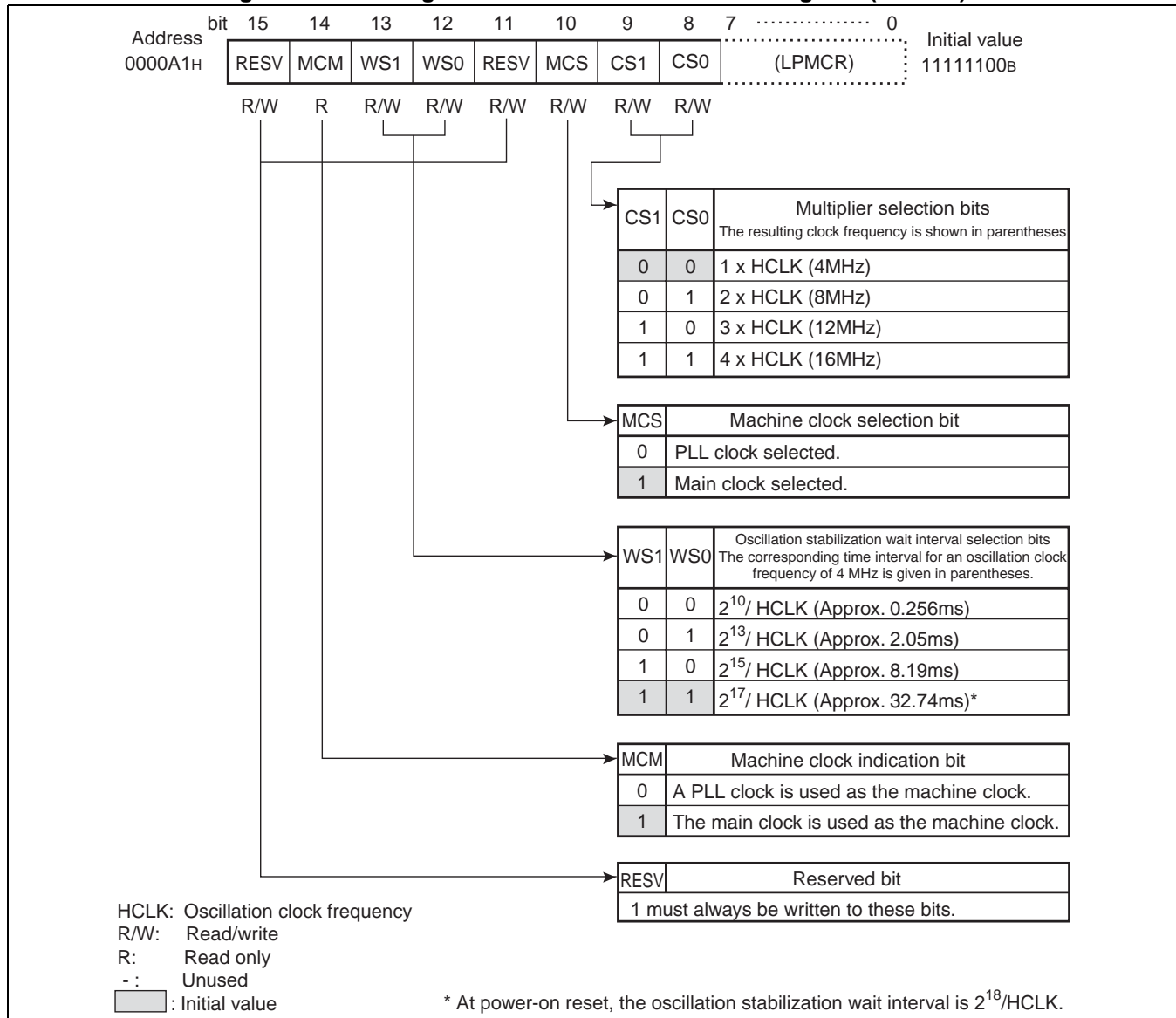
## 5.3 Clock Selection Register (CKSCR)

The clock selection register (CKSCR) is used to set switching between the main clock and a PLL clock, selection of an oscillation stabilization wait interval, and selection of a PLL clock multiplier.

### ■ Configuration of the Clock Selection Register (CKSCR)

Figure 5.3-1 shows the configuration of the clock selection register (CKSCR). Table 5.3-1 describes the function of each bit of the clock selection register (CKSCR).

**Figure 5.3-1 Configuration of the Clock Selection Register (CKSCR)**



**Note:** If the machine clock selection bit is not set, the main clock is used as the machine clock.

**Table 5.3-1 Function Description of Each Bit of the Clock Selection Register (CKSCR)**

Bit name		Function
bit15, bit11	RESV: Reserved bit	(Note) "1" must always be written to these bits.
bit14	MCM: Machine clock indication bit	<ul style="list-style-type: none"> <li>This bit indicates whether the main clock or a PLL clock has been selected as the machine clock.</li> <li>When this bit is set to "0", a PLL clock has been selected. When it is set to "1", the main clock has been selected.</li> <li>If MCS = 0 and MCM = 1, the PLL clock oscillation stabilization wait period is in effect.</li> <li>Writing has no effect on the operation.</li> </ul>
bit13, bit12	WS1, WS0: Oscillation stabilization wait interval selection bits	<ul style="list-style-type: none"> <li>These bits select an oscillation stabilization wait interval of the oscillation clock after stop mode has been released.</li> <li>These bits are initialized to 11<sub>B</sub> by all reset causes.</li> </ul> (Note) The oscillation stabilization wait interval must be set to a value appropriate for the oscillator used. See "4.2 Reset Causes and Oscillation Stabilization Wait Intervals". (Reference) The oscillation stabilization period for all PLL clocks is fixed at $2^{14}/\text{HCLK}$ .
bit10	MCS: Machine clock selection bit	<ul style="list-style-type: none"> <li>This bit specifies whether the main clock or a PLL clock is selected as the machine clock.</li> <li>When this bit is "0", a PLL clock is selected. When this bit is "1", the main clock is selected.</li> <li>If this bit has been set to "1" and "0" is written to it, the oscillation stabilization wait interval for the PLL clock starts. As a result, the time-base timer is automatically cleared, and the TBOF bit of the time-base timer control register (TBTC) is also cleared.</li> <li>For PLL clocks, the oscillation stabilization period is fixed at <math>2^{14}/\text{HCLK}</math> (the oscillation stabilization wait interval is approx. 2 ms for an oscillation clock frequency of 4 MHz).</li> <li>When the main clock has been selected, the operating clock frequency is the frequency of the oscillation clock divided by 2 (e.g., the operating clock is 2 MHz when the oscillation clock frequency is 4 MHz).</li> <li>This bit is initialized to "1" by power-on or watchdog reset.</li> </ul> (Note) When the MCS bit is "1", write "0" to it only when the time-base timer interrupt is masked by the TBIE bit of the time-base timer control register (TBTC) or the interrupt level register (ILM). For 8 machine cycles after "1" is written to the MCS bit, writing "0" to it may be disabled. Write to the bit after 8 machine cycles have passed.
bit9, bit8	CS1, CS0: Multiplier selection bits	<ul style="list-style-type: none"> <li>These bits select a PLL clock multiplier.</li> <li>Selection can be made from among four different multipliers.</li> <li>These bits are initialized to 00<sub>B</sub> by all reset causes.</li> </ul> (Note) When the MCS bit is "0", writing to these bits is not allowed. Write to the CS1 and CS0 bits only after setting the MCS bit to "1" (main clock mode).

HCLK: Oscillation clock frequency

## 5.4 Clock Mode

---

**Two clock modes are provided: main clock mode and PLL clock mode.**

---

### ■ Main Clock Mode and PLL Clock Mode

- Main clock mode

In main clock mode, the main clock, whose frequency is the oscillation clock divided by 2, is used as the operating clock for the CPU and peripheral resources, and the PLL clocks are disabled.

- PLL clock mode

In PLL clock mode, a PLL clock is used as the operating clock for the CPU and peripheral resources. A PLL clock multiplier is selected with the clock selection register (CKSCR: CS1 and CS0).

### ■ Clock Mode Transition

Switching between main clock mode and PLL clock mode is done by writing to the MCS bit of the clock selection register (CKSCR).

- Switching from main clock mode to PLL clock mode

When the MCS bit of CKSCR is "1" and "0" is written to it, the switch from the main clock to a PLL clock occurs after the PLL clock oscillation stabilization wait period ( $2^{14}/HCLK$ ).

- Switching from PLL clock mode to main clock mode

When the MCS bit of CKSCR is "0" and "1" is written to it, the switch from the PLL clock to the main clock occurs when the edges of the PLL clock and the main clock coincide (after 1 to 8 PLL clocks).

---

Note:

Even though the MCS bit of CKSCR is rewritten, machine clock switching does not occur immediately. When operating a resource that depends on the machine clock, make sure that machine clock switching has been done by referring to the MCM bit of CKSCR before operating the resource. If the mode is switched to another clock mode or low-power-consumption mode before completion of switching, the mode may not be switched.

---

### ■ Selection of a PLL Clock Multiplier

Writing a value from "00<sub>B</sub>" to "11<sub>B</sub>" to the CS1 and CS0 bits of CKSCR selects one to the four PLL clock multipliers.

### ■ Selection of a PLL Clock Multiplier

Writing a value from "00<sub>B</sub>" to "11<sub>B</sub>" to the CS1 and CS0 bits of CKSCR selects one to the four PLL clock multipliers.

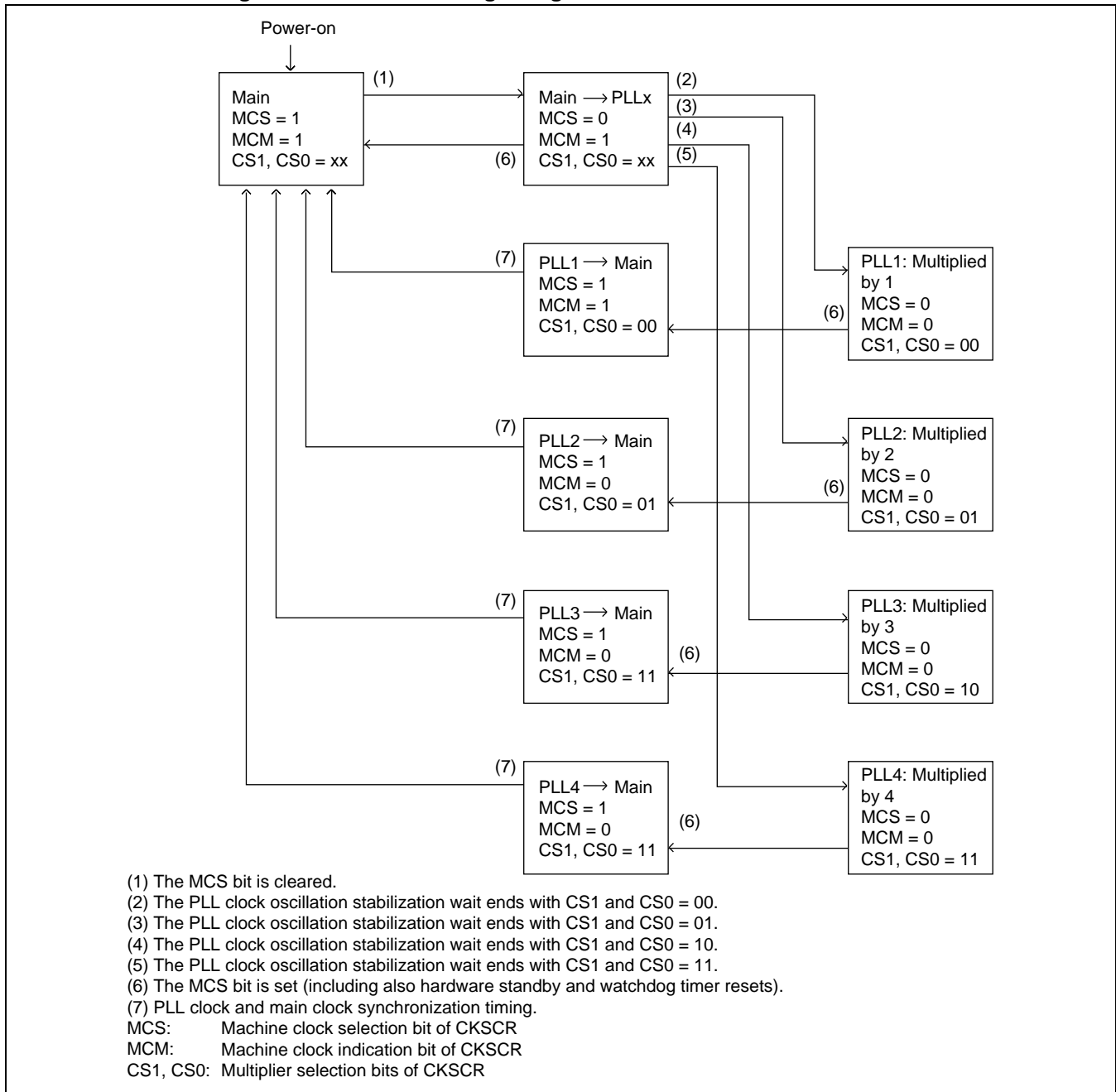
### ■ Machine Clock

The machine clock may be either a PLL clock output from the PLL multiplier circuit or the clock that is the source oscillation frequency divided by 2. This machine clock is supplied to the CPU and peripheral functions.

Either the main clock or a PLL clock can be selected by writing to the MCS bit of CKSCR.

Figure 5.4-1 shows the status change caused by the machine clock switching.

**Figure 5.4-1 Status Change Diagram for Machine Clock Selection**



Note:

The initial value for the machine clock setting is main clock (MCS of CKSCR = 1).



## 5.5 Oscillation Stabilization Wait Interval

When the power is turned on, when stop mode is released, or when a watchdog timer reset occurs, the oscillation clock starts, oscillation is unstable initially. Therefore, an oscillation stabilization wait interval is required. When the switch from the main clock to a PLL clock occurs, an oscillation stabilization wait interval is also required when PLL oscillation starts.

### ■ Oscillation Stabilization Wait Interval

Ceramic and crystal oscillators generally require an oscillation stabilization wait interval of a few to several dozen milliseconds until they stabilize at their natural frequency when oscillation starts.

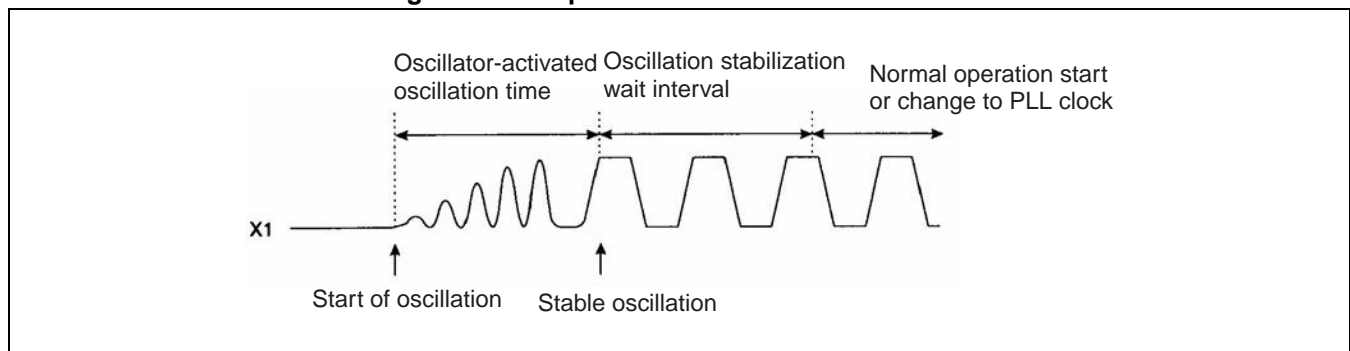
For this reason, CPU operation is not allowed as soon as oscillation starts and is allowed only after full stabilization of oscillation. After the oscillation stabilization wait interval has elapsed, the clock is supplied to the CPU.

Because the oscillation stabilization time depends on the type of the oscillator (crystal, ceramic, etc.), the proper oscillation stabilization wait interval for the oscillator used must be selected. An oscillation stabilization wait interval is selected by setting the clock selection register (CKSCR).

In a switch from the main clock to a PLL clock, the CPU continues to operate on the main clock during the oscillation stabilization wait interval. After this interval, the operating clock switches to the PLL clock.

Figure 5.5-1 shows the operation after oscillation starts.

**Figure 5.5-1 Operation when Oscillation Starts**



## 5.6 Connection of an Oscillator or an External Clock to the Microcontroller

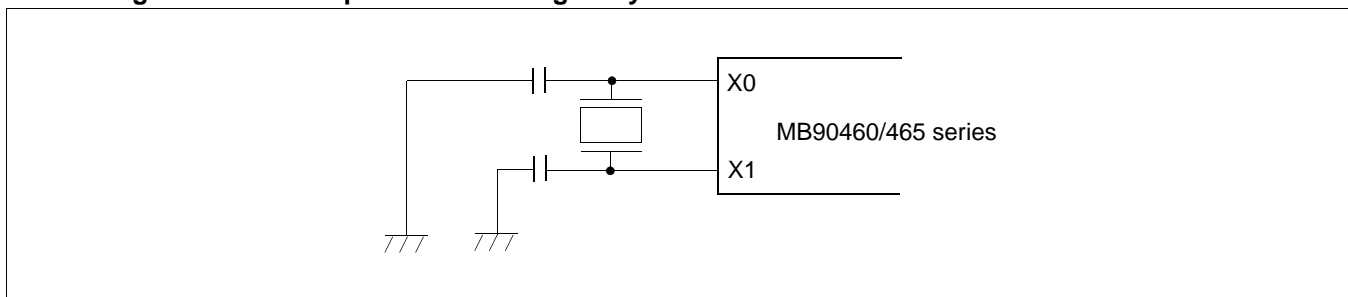
The F<sup>2</sup>MC-16LX microcontroller contains a system clock generation circuit. Connecting an external oscillator to this circuit generates the system clock. Alternatively, an externally generated clock can be input to the microcontroller.

### ■ Connection of an Oscillator or an External Clock to the Microcontroller

- Example of connecting a crystal or ceramic oscillator to the microcontroller

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

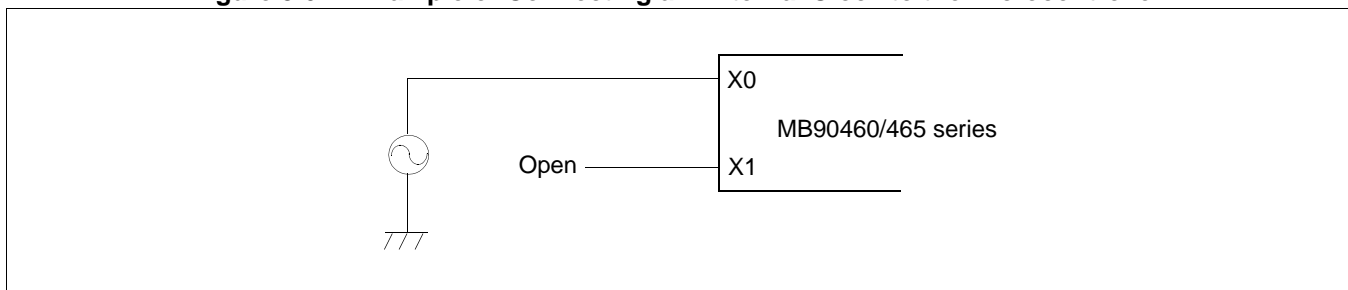
**Figure 5.6-1 Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller**



- Example of connecting an external clock to the microcontroller

As shown in Figure 5.6-2 , connect an external clock to pin X0. Pin X1 must be open.

**Figure 5.6-2 Example of Connecting an External Clock to the Microcontroller**





# ***CHAPTER 6***

---

# ***LOW POWER CONSUMPTION MODE***

**This chapter describes the low power consumption mode of MB90460/465 series microcontrollers.**

- 6.1 Low Power Consumption Mode
- 6.2 Block Diagram of the 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 Change Diagram
- 6.7 State of Pins in Standby Mode and during Reset
- 6.8 Usage Notes on Low Power Consumption Mode

## 6.1 Low Power Consumption Mode

F<sup>2</sup>MC-16LX microcontrollers have the following CPU operating modes, any of which can be used depending on the operating clock selection and clock operation control:

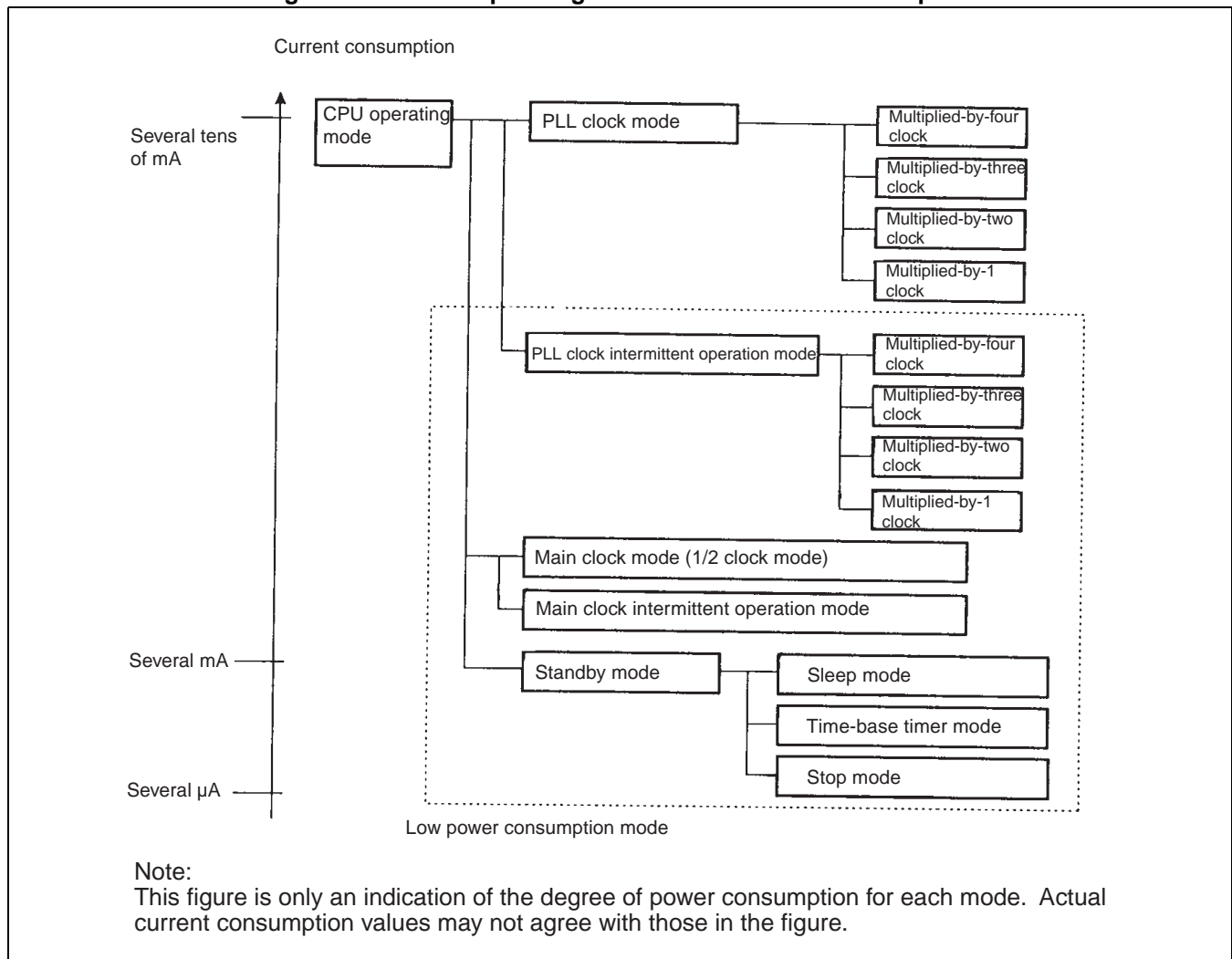
- Clock mode (PLL clock mode and main clock mode)
- CPU intermittent operation mode (PLL clock intermittent operation mode and main clock intermittent operation mode)
- Standby mode (sleep, time-base timer and stop modes)

All modes other than PLL clock mode are low power consumption mode.

### ■ CPU Operating Modes and Current Consumption

Figure 6.1-1 shows the relation between the CPU operating modes and current consumption

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



## ■ Clock Mode

### ● PLL clock mode

A PLL clock that is a multiple of the oscillation clock (HCLK) frequency is used to operate the CPU and peripheral functions.

### ● Main clock mode

The main clock, with a frequency one-half that of the oscillation clock (HCLK), is used to operate the CPU and peripheral functions. In main clock mode, the PLL multiplier circuit is inactive.

---

Reference:

See "5.1 Clock", for details about clock mode.

---

## ■ CPU Intermittent Operation Mode

CPU intermittent operation mode causes the CPU to operate intermittently, while high-speed clock pulses are supplied to peripheral functions, reducing power consumption. In CPU intermittent operation mode, intermittent clock pulses are only applied to the CPU when it is accessing a register, internal memory, a peripheral function, or an external unit.

## ■ Standby Mode

In standby mode, the low power consumption control circuit stops supplying the clock to the CPU (sleep mode) or the CPU and peripheral functions (time-base timer mode), or stops the oscillation clock itself (stop mode), reducing power consumption.

### ● PLL sleep mode

PLL sleep mode is activated to stop the CPU operating clock when the microcontroller enters PLL clock mode; other components continue to operate on the PLL clock.

### ● Main sleep mode

Main sleep mode is activated to stop the CPU operating clock when the microcontroller enters main clock mode; other components continue to operate on the main clock.

### ● PLL time-base timer mode

PLL time-base timer mode causes microcontroller operation, with the exception of the oscillation clock, PLL clock and time-base timer, to stop. All functions other than the time-base timer are deactivated.

### ● Main time-base timer mode

Main time-base timer mode causes microcontroller operation, with the exception of the oscillation clock, main clock and the time-base timer, to stop. All functions other than the time-base timer are deactivated.

### ● Stop mode

Stop mode causes the source oscillation to stop. All functions are deactivated.

---

Note:

Because stop mode turns the oscillation clock off, this mode saves most power while data is being retained.

If the mode is switched to another clock mode or low-power-consumption mode before completion of switching, the mode may not be switched.

---

## 6.2 Block Diagram of the Low Power Consumption Control Circuit

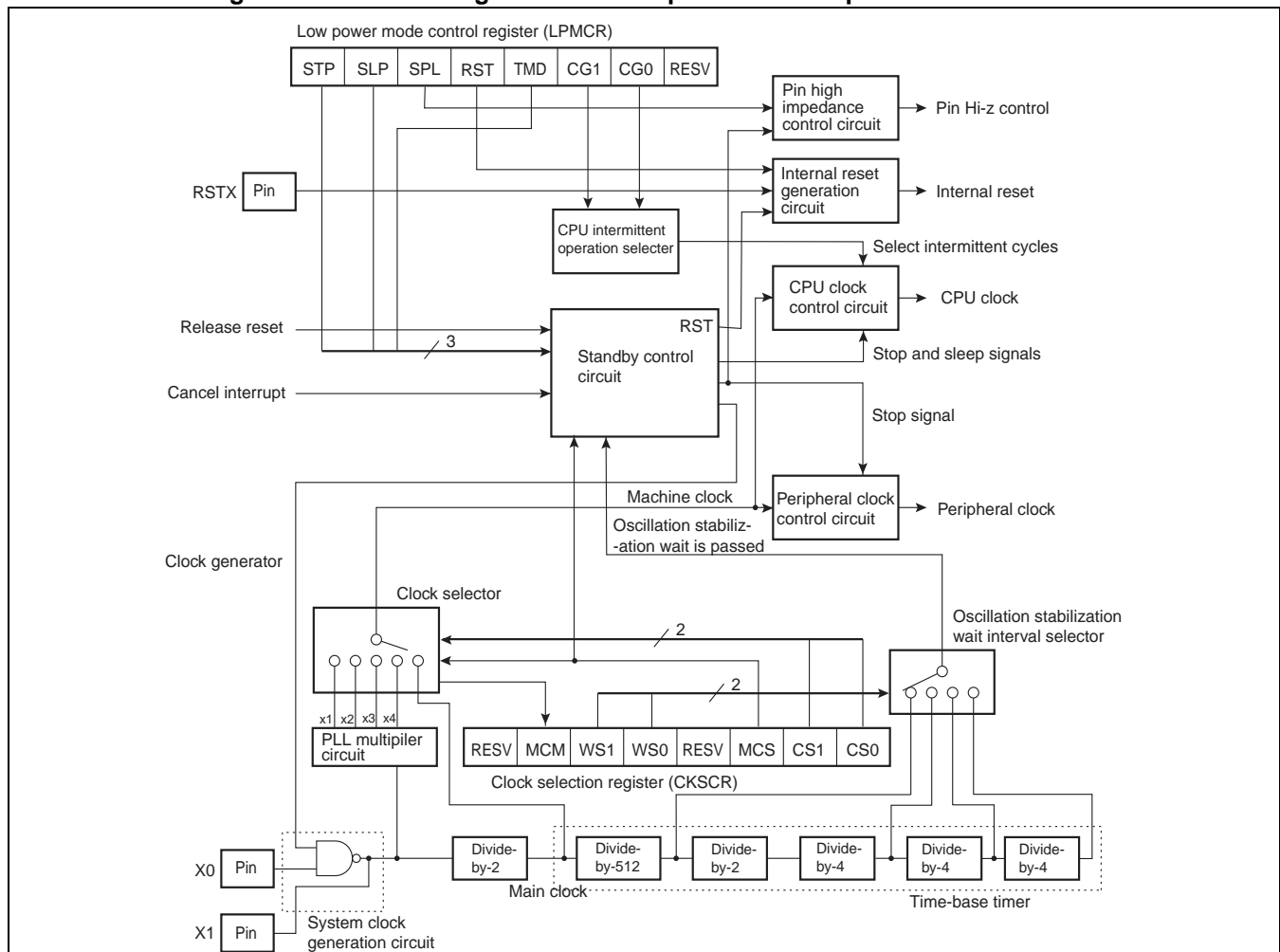
The low power consumption control circuit consists of the following seven blocks:

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

### ■ Block Diagram of the 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 the low power consumption control circuit**



- CPU intermittent operation selector

This selector selects the number of clock pulses the CPU is to be halted during CPU intermittent operation mode.

- Standby control circuit

The standby control circuit controls the CPU clock control circuit and the peripheral clock control circuit, and turns the low power consumption mode on and off.

- CPU clock control circuit

This circuit controls the clocks supplied to the CPU. This circuit controls the clocks supplied to peripheral functions for the peripheral clock control.

- Peripheral clock control circuit

This circuit controls the clocks supplied to peripheral functions.

- Pin high-impedance control circuit

This circuit makes the external pins high-impedance when the microcontroller enters time-base timer mode and stop mode.

For the pins with the pull-up option, this circuit disconnects the pull-up resistor when the microcontroller enters stop mode.

- Internal reset generation circuit

This circuit generates an internal reset signal.

- Low power consumption mode control register (LPMCR)

This register is used to switch to and release standby mode and to set the CPU intermittent operation function.



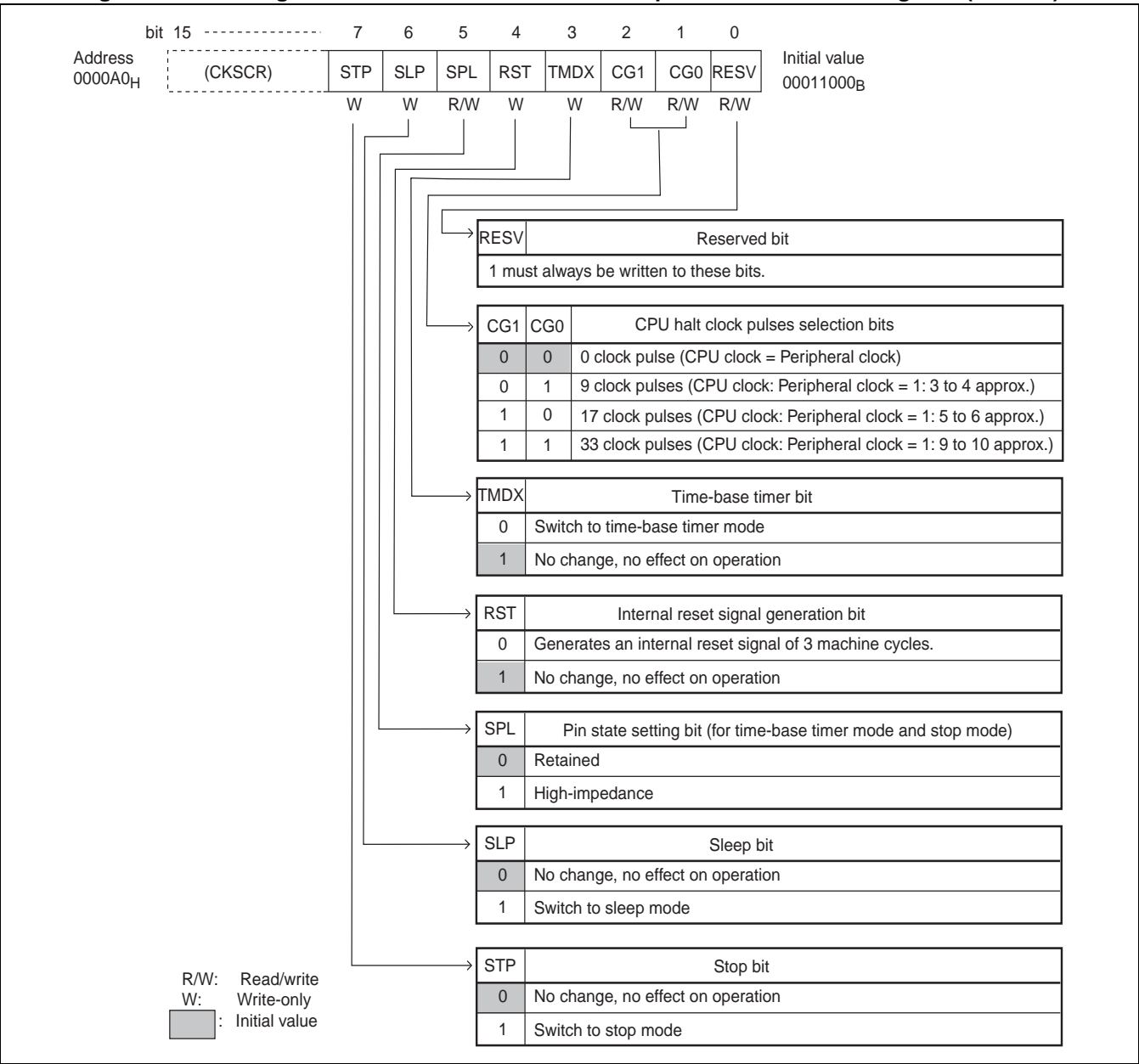
# 6.3 Low Power Consumption Mode Control Register (LPMCR)

The low power consumption mode control register (LPMCR) switches to or releases low power consumption mode. It is also used to set the number of CPU clock pulses the CPU is to be halted during CPU intermittent 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 the Low Power Consumption Mode Control Register (LPMCR)**



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

Bit name		Function
bit7	STP: Stop bit	<ul style="list-style-type: none"> <li>• This bit indicates switching to stop mode.</li> <li>• When "1" is written to this bit, a switch to stop mode.</li> <li>• Writing "0" to this bit has no effect on operation.</li> <li>• This bit is cleared to "0" by a reset or by release of stop state.</li> <li>• The read value of this bit is always "0".</li> </ul>
bit6	SLP: Sleep bit	<ul style="list-style-type: none"> <li>• This bit indicates switching to sleep mode.</li> <li>• When "1" is written to this bit, the mode switches to sleep mode.</li> <li>• Writing "0" to this bit has no effect on operation.</li> <li>• This bit is cleared to "0" by a reset or by release of sleep mode.</li> <li>• The read value of this bit is always "0".</li> </ul>
bit5	SPL: Pin state setting bit (for time-base timer mode and stop mode)	<ul style="list-style-type: none"> <li>• This bit is enabled while either time-base timer mode or stop mode is in effect.</li> <li>• When this bit is "0", the level of the external pins is retained.</li> <li>• When this bit is "1", the status of the external pins changes to high- impedance.</li> <li>• This bit is initialized to "0" by a reset.</li> </ul>
bit4	RST: Internal reset signal generation bit	<ul style="list-style-type: none"> <li>• When "0" is written to this bit, an internal reset signal of 3 machine cycles is generated.</li> <li>• Writing "1" to this bit has no effect on operation.</li> <li>• The read value of this bit is always "1".</li> </ul>
bit3	TMDX: Time-base timer bit	<ul style="list-style-type: none"> <li>• This bit indicates switching to time-base timer mode.</li> <li>• When "0" is written to this bit, the mode switches to time-base timer mode.</li> <li>• Writing "1" to this bit has no effect on operation.</li> <li>• This bit is set to "1" by a reset or by release of time-base timer mode.</li> <li>• The read value of this bit is always "1".</li> </ul>
bit2, bit1	CG1, CG0: CPU halt clock pulses selection bits	<ul style="list-style-type: none"> <li>• These bits set the number of CPU halt clock pulses for the CPU intermittent operation function.</li> <li>• The clock supplied to the CPU is stopped after the execution of every instruction for the specified number of clock pulses.</li> <li>• Selection can be made from among four different clock pulses.</li> <li>• These bits are initialized to 00<sub>B</sub> by a power-on or watchdog timer reset. Other resets do not initialize these bits.</li> </ul>
bit0	RESV: Reserved bit	<p>(Note)</p> <p>"1" must always be written to this bit.</p>

**Note:**

If "1" is written to the STP bit, SLP bit and "0" is written to TMDX bit at the same time, switching to stop mode takes the highest priority, then time-base timer mode and sleep mode has the lowest priority.

## ■ Access to the Low Power Consumption Mode Control Register

Switching to low power consumption mode (including stop mode and sleep mode) is performed by writing to the low power consumption mode control register. Only the instructions listed in Table 6.3-2 should be used for this purpose. If other instructions are used for switching to low power consumption mode, operation cannot be assured. To control functions not listed in Table 6.3-2, any instruction can be used.

When word-length is used for writing to the low power consumption mode control register, even addresses must be used. Writing with odd addresses to switch to low power consumption mode may cause a malfunction.

**Table 6.3-2 Instructions to be used for switching to Low Power Consumption Mode**

MOV io, #imm8 MOV io, A MOV @RLi+disp8, A MOVW io, #imm16 MOVW io, A MOVW @RLi+disp8, A	MOV dir, #imm8 MOV dir, A  MOVW dir, #imm16 MOVW dir, A	MOV eam, #imm8 MOV addr, 16A  MOVW eam, #imm16 MOVW addr16, A	MOV eam, Ri MOV eam, A  MOVW eam, RWi MOVW eam, A
SETB io:bp CLRB io:bp	SETB dir:bp CLRB dir:bp	SETB addr16:bp CLRB addr16:bp	

## 6.4 CPU Intermittent Operation Mode

**CPU intermittent operation mode is used for intermittent operation of the CPU while external buses and peripheral functions continue to operate at high speed. Its purpose is to reduce power consumption.**

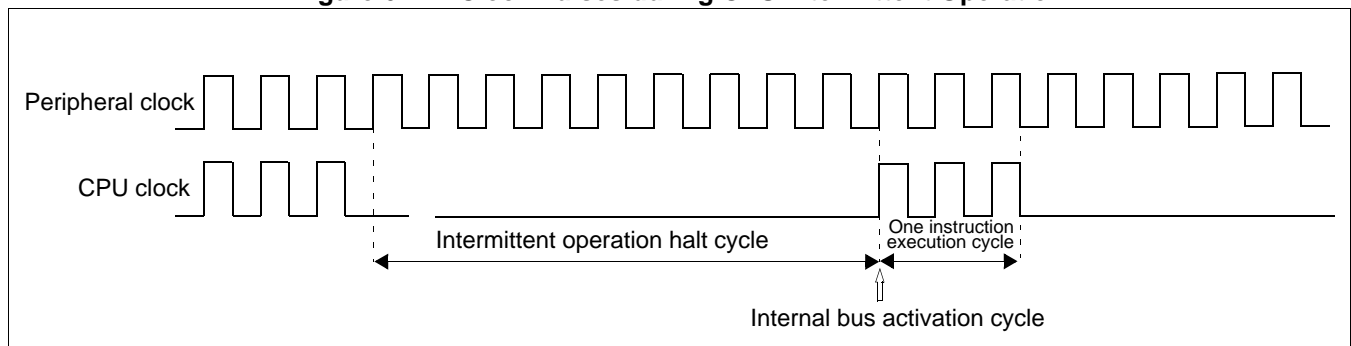
### ■ CPU Intermittent Operation Mode

CPU intermittent operation mode halts the supply of the clock to the CPU for a certain period. The halt occurs after the execution of every instruction that accesses a register, internal memory (ROM and RAM), I/O, peripheral functions and the external bus. Internal bus cycle activation is therefore delayed. While a steady rate of peripheral clock pulses are supplied to the peripheral functions, the rate of CPU execution is reduced, enabling processing with low power consumption.

- The CG1 and CG0 bits of the low power consumption mode control register (LPMCR) are used to select the number of clock pulses per halt cycle of the clock supplied to the CPU.
- External bus operation uses the same clock as that used for peripheral functions.
- Instruction execution time in CPU intermittent mode can be calculated. A correction value should be obtained by multiplying the number of times instructions that access a register, internal memory, internal peripheral functions, and the external bus are executed by the number of clock pulses per halt cycle. Add this correction value to the normal execution time.

Figure 6.4-1 shows the operating clock pulses during CPU intermittent operation mode.

**Figure 6.4-1 Clock Pulses during CPU Intermittent Operation**



## 6.5 Standby Mode

Standby mode includes the sleep (PLL sleep and main sleep), time-base timer and stop modes.

### ■ Operating Status during Standby Mode

Table 6.5-1 summarizes the operating statuses during standby mode.

**Table 6.5-1 Operation Statuses during Standby Mode**

Standby mode		Condition for switch	Oscillation	Clock	CPU	Peripheral	Pin	Release event			
Sleep mode	PLL sleep mode	MCS = 0 SLP = 1	Active	Active	In-active	Active	Active	Reset or Interrupt			
	Main sleep mode	MCS = 1 SLP = 1									
Time-base timer mode	PLL time-base timer mode (SPL = 0)	MCS = 0 TMDX = 0				Inactive	Inactive		Inactive	Inactive *	Hold
	PLL time-base timer mode (SPL = 1)										Hi-Z
	Main time-base timer mode (SPL = 0)	MCS = 1 STP = 1									Hold
	Main time-base timer mode (SPL = 1)										Hi-Z
Stop mode	Main/PLL stop mode (SPL = 0)	MCS = x STP = 1	Inactive	Inactive	Inactive	Inactive	Hold				
	Main/PLL stop mode (SPL = 1)						Hi-Z				

\*: Only the time-base timer is active.

SPL: Pin state setting bit of low power consumption mode control register (LPMCR)

SLP: Sleep bit of LPMCR

STP: Stop bit of LPMCR

TMDX: Time-base timer bit of LPMCR

MCS: Machine clock selection bit of clock selection register (CKSCR)

Hi-Z: High-impedance

## 6.5.1 Sleep Mode

---

**Sleep mode causes the CPU operating clock to stop while other components continue to operate.**

**When the low power consumption mode control register (LPMCR) indicates a switch to sleep mode, a switch to PLL sleep mode occurs if PLL clock mode has been set. Alternatively, a switch to main sleep mode occurs if main clock mode has been set.**

---

### ■ Switching to Sleep Mode

Writing "1" to the SLP and TMDX bit of LPMCR and 0 to the STP bit of LPMCR triggers a switch to sleep mode.

At this time, if the MCS bit of the clock selection register (CKSCR) is "0", the microcontroller enters PLL sleep mode. If the MCS bit of CKSCR is "1", the microcontroller enters main sleep mode.

---

#### Note:

Since the STP/TMDX bit setting overrides the SLP bit setting when "1" is written to the SLP, STP and "0" to TMDX bit at the same time, the mode switches to stop/time-base timer mode.

---

#### ● Data retention function

In sleep mode, the contents of dedicated registers, such as accumulators and internal RAM, are retained.

#### ● Operation during an interrupt request

Writing "1" to the SLP bit of LPMCR during an interrupt request does not trigger a switch to sleep mode. If the CPU does not accept the interrupt, the CPU executes the next instruction. If the CPU accepts the interrupt, CPU operation immediately branches to the interrupt processing routine.

#### ● Status of pins

During sleep mode, all pins retain the state they had immediately before the switch to sleep mode. The only exceptions are the pins used for bus input/output or bus control.

### ■ Release of Sleep Mode

The low power consumption control circuit releases sleep mode. Releasing is caused by the input of a reset or by an interrupt.

#### ● Return to normal mode by a reset

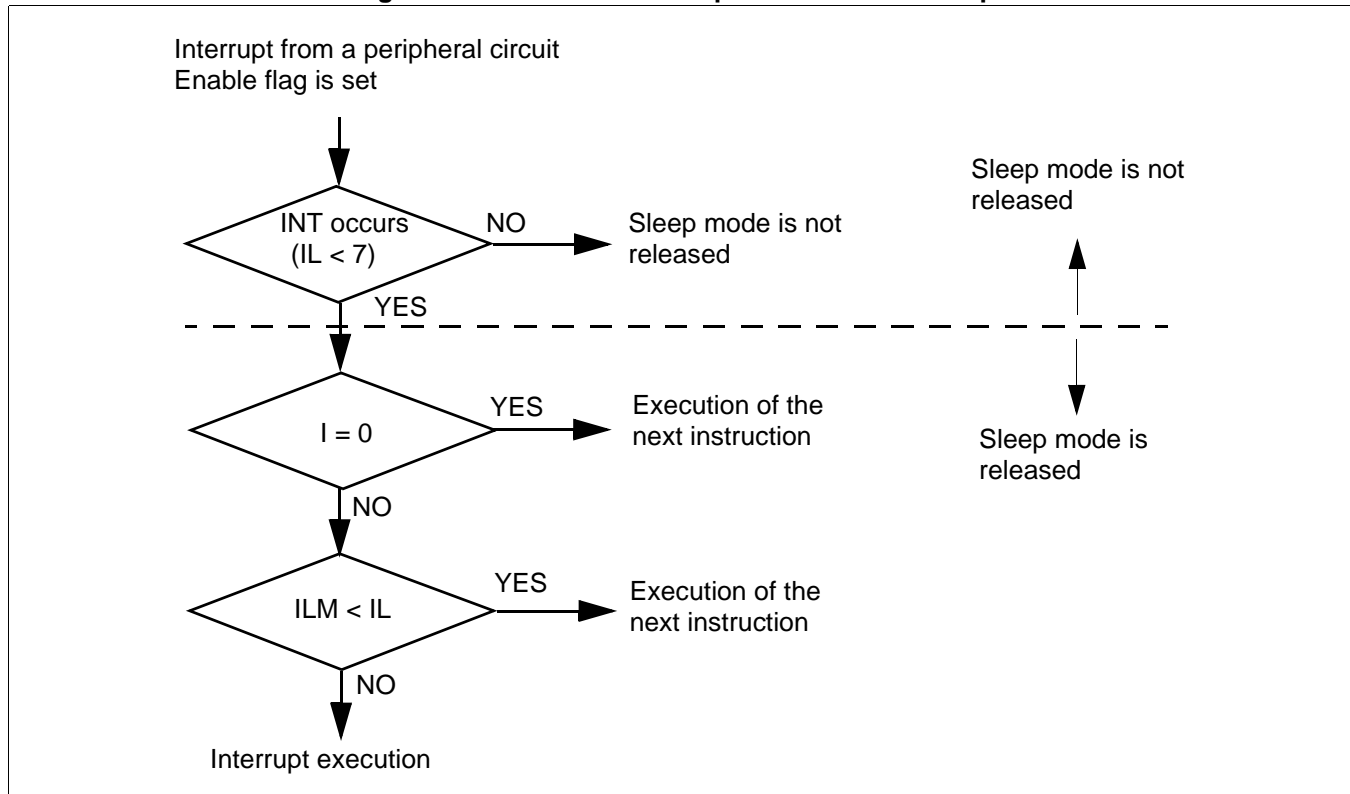
When sleep mode is released by a reset, the microcontroller is placed in the reset state on release from sleep mode.

● Return to normal mode by an interrupt

If an interrupt request higher than level 7 is issued from a peripheral circuit during sleep mode, sleep mode is released. After release, the CPU handles the interrupt as it would any other interrupt. The CPU executes processing according to the settings of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR). If that interrupt is accepted, the CPU executes interrupt processing. If the interrupt is not accepted, the CPU resumes execution with the instruction that follows the instruction in which switching to sleep mode was specified.

Figure 6.5-1 shows the release of sleep mode for an interrupt.

**Figure 6.5-1 Release of Sleep Mode for an Interrupt**

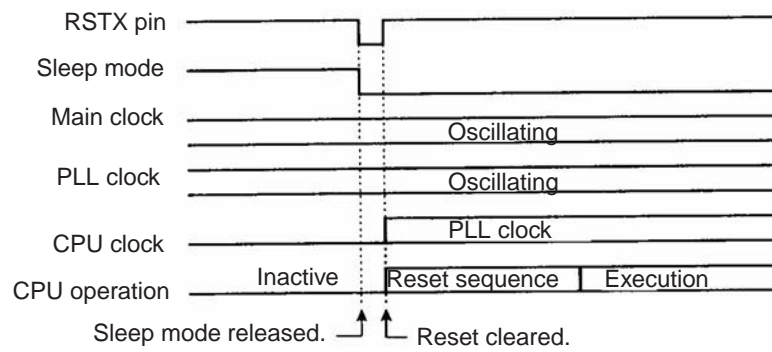


Note:

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to sleep mode was specified. The CPU then proceeds to interrupt processing.

● Return to normal mode from PLL sleep mode by an external reset

During PLL sleep mode, the main clock and the PLL clock generate clock pulses. Since an external reset does not initialize the MCS bit in the clock selection register (CKSCR) to "1", PLL clock mode remains selected (MCS of CKSCR = 0). On return from PLL sleep mode by an external reset, the CPU starts operation using the PLL clock immediately after PLL sleep mode is released as shown in Figure 6.5-2.

**Figure 6.5-2 Release of PLL Sleep Mode (by External Reset)**



## 6.5.2 Time-base Timer Mode

---

**Time-base timer mode causes the microcontroller operation to stop with the exception of the source oscillation and the time-base timer. All functions other than time-base timer are deactivated.**

---

### ■ Switching to Time-base Timer Mode

Writing "0" to the TMDX and STP bit of LPMCR triggers a switch to time-base timer mode.

At this time, if the MCS bit of the clock selection register (CKSCR) is "0", the microcontroller enters PLL time-base timer mode. If the MCS bit of CKSCR is "1", the microcontroller enters main time-base timer mode.

---

Note:

Since the STP bit setting overrides the TMDX bit setting when "0" is written to the TMDX and STP bits at the same time, the mode switches to stop mode.

---

#### ● Data retention function

In time-base timer mode, the contents of dedicated registers, such as accumulators and internal RAM, are retained.

#### ● Operation during an interrupt request

Writing "0" to the TMDX bit of LPMCR during an interrupt request does not trigger switching to time-base timer mode.

#### ● Status of pins

Selection of whether the external pins retain the state they had immediately before switching to time-base timer mode or go to high-impedance with switching to this mode can be controlled by the SPL bit of LPMCR.

### ■ Release of Time-base Timer Mode

The low power consumption control circuit releases time-base timer mode. Release is caused by input of a reset or an interrupt. If time-base timer mode is released by a reset, the microcontroller is placed in the reset state after its release from time-base timer mode.

#### ● Return to normal mode by a reset

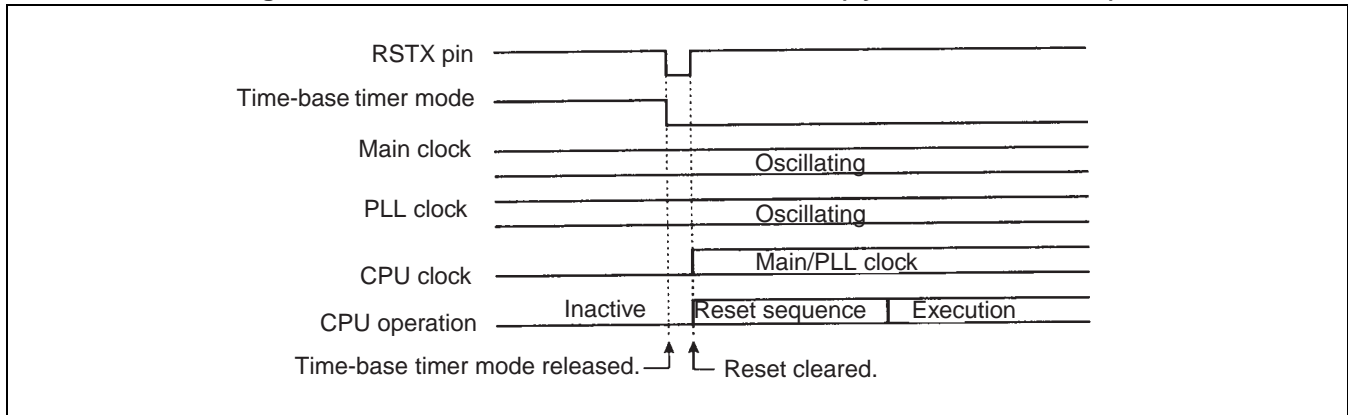
If time-base timer mode is released by a reset, the microcontroller is placed in the reset state after release from time-base timer mode.

#### ● Return to normal mode by an external reset

Since an external reset does not initialize the MCS bit of the clock selection register (CKSCR) to "1", PLL clock mode remains selected (MCS of CKSCR = 0) or main clock mode remains selected (MCS of CKSCR = 1). On return from time-base timer mode by an external reset, the CPU starts operation using PLL/Main clock immediately after time-base timer mode is released.

Figure 6.5-3 shows the operation for return to normal mode from time-base timer mode triggered by an external reset.

**Figure 6.5-3 Release of Time-base Timer Mode (by an External Reset)**



#### ● Return to normal mode by an interrupt

If an interrupt request higher than level 7 is issued from a peripheral circuit in time-base timer mode (when IL2, IL1 and IL0 of the interrupt control register (ICR) are set to a value other than "111<sub>B</sub>"), the low power consumption control circuit releases time-base timer mode. After the release, the CPU handles the interrupt as it would any other interrupt. The CPU executes processing according to the settings of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR). If the interrupt is accepted, the CPU executes interrupt processing. If the interrupt is not accepted, the CPU resumes execution with the instruction that follows the instruction in which switching to time-base timer mode was specified.

#### Note:

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to time-base timer mode was specified. The CPU then proceeds to interrupt processing.

### 6.5.3 Stop Mode

---

**Stop mode causes the source oscillation to stop and deactivates all functions. It therefore saves the most power saving while data is being retained.**

---

#### ■ Switching to Stop Mode

Writing "1" to the STP bit of LPMCR triggers a switch to stop mode.

At this time, if the MCS bit of the clock selection register (CKSCR) is "0", the microcontroller enters PLL stop mode. If the MCS bit of CKSCR is "1", the microcontroller enters main stop mode.

##### ● Data retention function

In stop mode, the contents of dedicated registers, such as accumulators and internal RAM, are retained.

##### ● Operation during an interrupt

Writing "1" to the STP bit of LPMCR during an interrupt request does not trigger switching to stop mode.

##### ● Pin state setting

Selection of whether the external pins retain the state they had immediately before switching to stop mode or go to high-impedance with switching to stop mode can be controlled by the SPL bit of LPMCR.

#### ■ Release of Stop Mode

The low power consumption control circuit releases stop mode. The release is caused by input of a reset or by an interrupt.

Because the oscillation of the operating clock is halted before return to normal mode from stop mode, the low power consumption control circuit puts the microcontroller into the oscillation stabilization wait state, then releases stop mode.

##### ● Return to normal mode by a reset

When stop mode is released by a reset cause, the microcontroller is placed in the oscillation stabilization wait and reset state after release from stop mode. The reset sequence proceeds after the oscillation stabilization wait interval has elapsed.

##### ● Return to normal mode by a interrupt

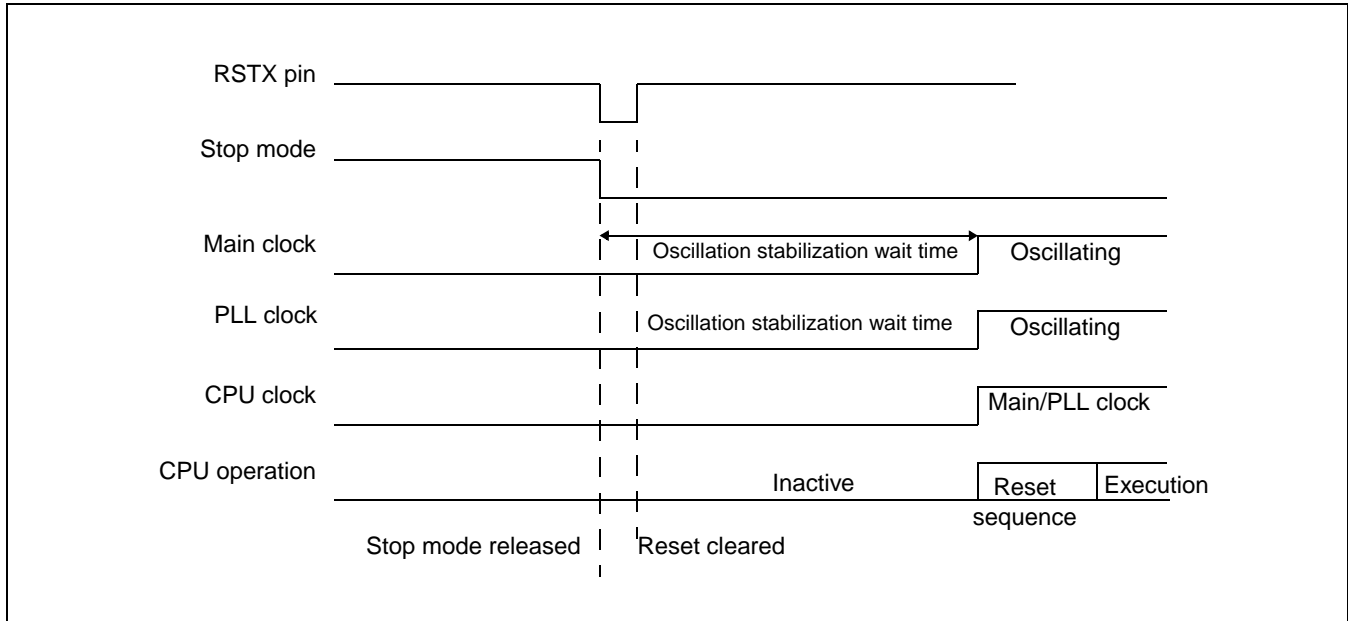
If an interrupt request higher than level 7 is issued from a peripheral circuit during stop mode (when IL2, IL1 and IL0 of the interrupt control register (ICR) are set to a value other than "111<sub>B</sub>"), the low power consumption control circuit releases stop mode. After release, the CPU handles the interrupt as it would any other interrupts. However, the CPU starts after the main clock oscillation stabilization wait interval specified by the WS1 and WS0 bits of the clock selection register (CKSCR) has elapsed. The CPU executes processing according to the settings of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR). If the interrupt is accepted, the CPU executes interrupt processing. If the interrupt is not accepted, the CPU resumes the execution with the instruction that follows the instruction in which switching to stop mode was specified.

**Note:**

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to stop mode was specified. The CPU then proceeds to interrupt processing.

Figure 6.5-4 shows the operation of return to normal mode from stop mode.

**Figure 6.5-4 Release of Main Stop Mode (by External Reset)**

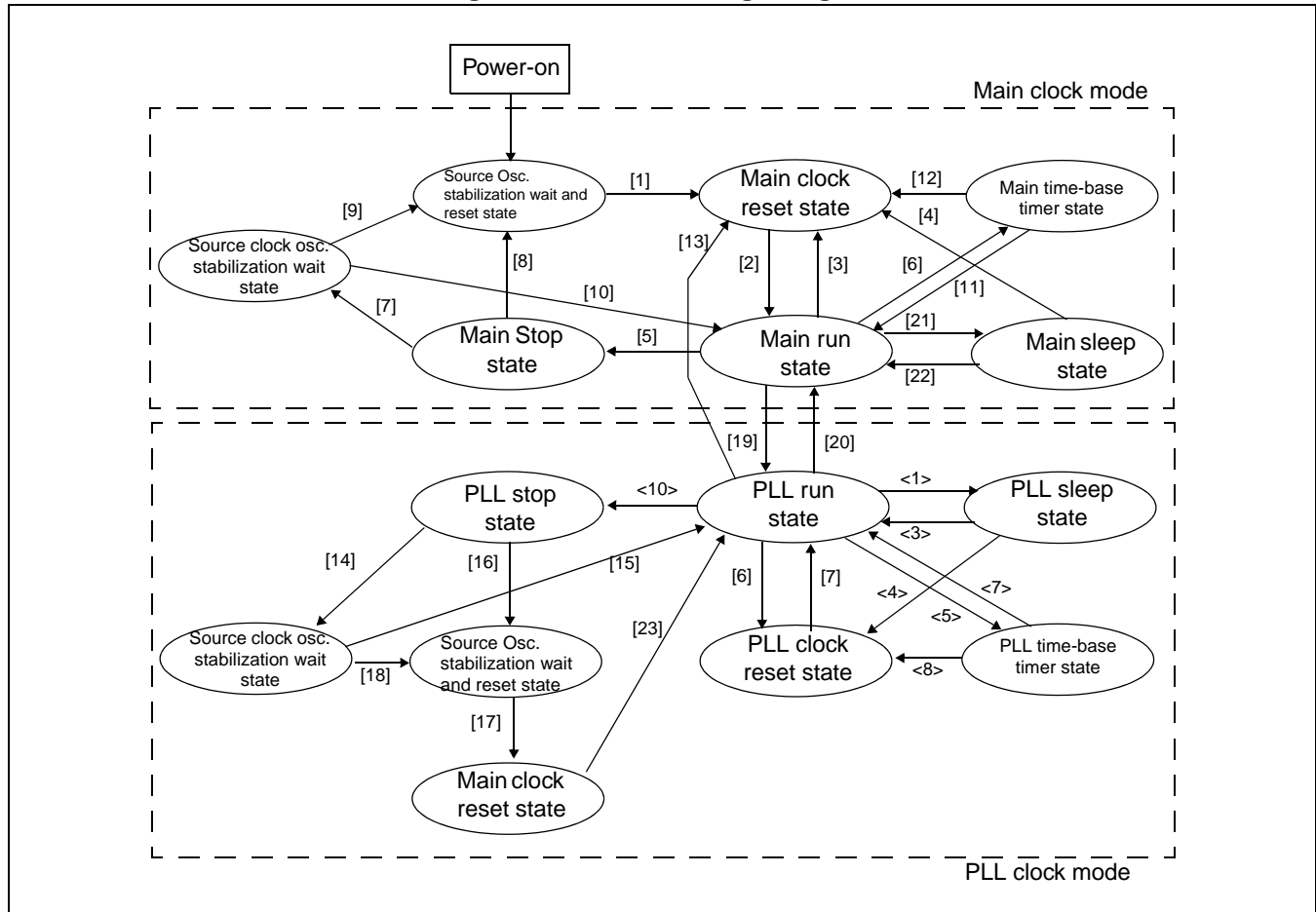


## 6.6 State Change Diagram

Figure 6.6-1 shows the state change diagram of F<sup>2</sup>MC-16LX operation and gives change conditions.

### ■ State Change Diagram

Figure 6.6-1 State Change Diagram



Note:

When the clock mode is switched, do not switch to low power consumption mode and other clock mode before this switching is completed. Confirm the completion of clock mode switching by referring to the MCM and SCM bits of the clock selection register (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.

## ■ Low Power Consumption Mode Operating States

Table 6.6-1 lists the operating states of low power consumption mode.

**Table 6.6-1 Low Power Consumption Mode Operating States**

Low power consumption mode	Condition for transition	Oscillation	Clock	CPU	Peripheral	Pin	Release event
Main sleep	MCS = 1 SLP = 1	Active	Active	Inactive	Active	Active	Reset or interrupt
PLL sleep	MCS = 0 SLP = 1	Active	Active	Inactive	Active	Active	Reset or interrupt
Main/PLL time-base timer (SPL = 0)	MCS = x TMDX = 0	Active	Active	Inactive	Inactive	Hold	Reset or interrupt
Main/PLL time-base timer (SPL = 1)	MCS = x TMDX = 0	Active	Active	Inactive	Inactive	Hi-Z	Reset or interrupt
Main/PLL stop (SPL = 0)	MCS = x STP = 1	Inactive	Inactive	Inactive	Inactive	Hold	Reset or interrupt
Main/PLL stop (SPL = 1)	MCS = x STP = 1	Inactive	Inactive	Inactive	Inactive	Hi-Z	Reset or interrupt

### ● Clock mode switching and release (excluding standby mode)

Table 6.6-2 lists clock mode switching and release.

**Table 6.6-2 Clock Mode Wwitching and Release**

Transition	Conditions
After power-on, transition to the main run state	[1] Source clock oscillation stabilization wait interval ends. (Time-base timer output) [2] Reset input has been cleared.
Reset during main run state	[3] External reset, software reset, or watchdog timer reset
Transition from main run state to PLL run state	[19] MCS = 0 (After PLL clock oscillation stabilization wait, switch to PLL clock) *
Return to main run state from PLL run state	[20] MCS = 1 (PLL clock deactivated)
Reset during PLL run state	[6] External reset or software reset ([7] After reset, return to PLL run state) [13] Watch dog reset ([2] After reset, return to main run state)

\*:The microcontroller operates using the main clock during the PLL clock oscillation stabilization wait state.

● Switching to and release of standby mode

Table 6.6-3 lists switching to and release of standby mode.

**Table 6.6-3 Switching to and Release of Standby Mode**

Transition	Conditions
Transition to main sleep mode	[21] SLP = 1, MCS = 1 (Transition from main run state) [2] SLP = 1, MCS = 1 (Transition from PLL run state)
Release of main sleep mode	[22] Interrupt input [4] External reset
Transition to main stop mode	[5] STP = 1, MCS = 1 (Transition from main run state)
Transition to PLL stop mode	<10> STP = 1, MCS = 0 (Transition from PLL run state)
Release of main stop mode	[7] Interrupt input ([10] indicates return to main run state after oscillation stabilization wait) [8] External reset ([9] indicates external reset during oscillation stabilization wait state)
Release of PLL stop mode	[14] Interrupt input ([15] indicates return to PLL run state after oscillation stabilization wait) [16] External reset ([18] indicates external reset during oscillation stabilization wait state)
Transition to PLL sleep mode	<1> SLP = 1, MCS = 0 (Transition from PLL run state) <2> SLP = 1, MCS = 0 (Transition from main run state, switch to PLL clock after PLL clock oscillation stabilization wait) *
Release of PLL sleep mode	<3> Interrupt input <4> External reset
Transition to main time-base timer mode	[6] STP = 1, MCS = 1 (Transition from main run state)
Transition to PLL time-base timer mode	<5> STP = 1, MCS = 0 (Transition from PLL run state)
Release of main time-base timer mode	[11] Interrupt input [12] External reset ([2] After reset, return to main run state)
Release of PLL time-base timer mode	<7> Interrupt input <8> External reset ([7] After reset, return to PLL run state)

\*The microcontroller operates using the main clock during the PLL clock oscillation stabilization wait state.

## 6.7 State of Pins in Standby Mode and during Reset

The state of pins in standby mode and during reset are summarized below for each memory access mode.

### ■ Software Pull-up Resistor

For pins with a pull-up resistor selected by software, the pull-up resistor is disconnected during L level output.

### ■ State of Pins in Single-chip Mode

Table 6.7-1 lists the state of pins in single-chip mode.

**Table 6.7-1 State of Pins in Single-chip Mode**

Pin name	Standby mode		Reset	
	Sleep	Stop		
		SPL = 0		SPL = 1
P00 to P07 P17 P20 to P27 P30 to P37 P40 P42 to P47 P50 to P57 P60 to P63	The preceding state is retained*2	The preceding state is retained*2	Input shut off*3 / output Hi-Z	Input disabled / output Hi-Z
P10 to P16 P63		Input enabled*1		

\*1: "Input enabled" means that the input function is enabled when corresponding external interrupt pin is enable. Select either the pull-up or the pull-down option. Alternatively, an external input is required. Pins used as output ports are the same as other ports.

\*2: "The preceding state is retained" means that the state of the pin output existing immediately before switching to this mode is retained. Note that input is disabled if the preceding state was input.

- "State of the pin output is retained" means that the pin retains the value output from an operating internal peripheral unit or the value output from the port if the pin is used as a port.
- "Input disabled" means that the input to the pin is not accepted because the internal circuit is inactive, although operation of the input gate adjacent to the pin is enabled.

\*3: In the "Input shut off" state, input is masked and the "L" level is transmitted internally. "Output Hi-Z" means that the pin state is high-impedance because driving of the pin driving transistor is disabled.



## 6.8 Usage Notes on Low Power Consumption Mode

---

**Note the following six items to use low power consumption mode:**

- **Switching to standby mode and interrupts**
  - **Release of standby mode by an interrupt**
  - **Setting of standby mode**
  - **Release of stop mode**
  - **Release of time-base timer mode**
  - **Oscillation stabilization wait time**
- 

### ■ Notes on Standby Mode

#### ● Switching to standby mode and interrupts

During an interrupt request to the CPU from a peripheral function, the CPU ignores the STP and SLP bits of the low power consumption mode control register (LPMCR) even though 1 has been written to these bits. Thus, switching to any standby mode is disabled (even after processing the interrupt is completed, there is no switch to standby mode). If the interrupt level is higher than 7, this action does not depend on whether the interrupt request is accepted by the CPU.

However, during execution of interrupt processing by the CPU, if the interrupt request flag for the interrupt is cleared and no other interrupt requests have been issued, switching to standby mode can be done.

#### ● Release of standby mode caused by an interrupt

If an interrupt request higher than level 7 is issued from a peripheral function during the sleep, time-base timer, or stop modes, the standby mode is released. This action does not depend on whether the CPU accepts that interrupt.

After the release of standby mode, normal interrupt processing is performed. The CPU branches to the interrupt handling routine provided that the priority of the interrupt request indicated by the interrupt level setting bits (IL2, IL1 and IL0 of ICR) is higher than the interrupt level mask register (ILM); and the interrupt enable flag (I) of the condition code register (CCR) is set to "1" (enabled). If the interrupt is not accepted, the CPU starts the execution with the instruction that follows the instruction in which switching to standby mode was specified.

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to standby mode was specified. The CPU then proceeds to interrupt processing. Depending on the condition when switching to standby mode was performed, however, the CPU may proceed to interrupt processing before executing the next instruction.

If the CPU should not branch to the interrupt processing routine immediately on return to normal mode from standby mode, action must be taken to disable interrupts before standby mode is set.

#### ● Setting of standby mode

When "1" is written to the STP bit and SLP bit of LPMCR at the same time, switching to standby mode is performed. If the MCS bit of the clock selection register (CKSCR) is "0", switching to time-base timer mode is performed; if this bit is "1", switching to stop mode is performed.

## ■ Release of Stop Mode

To use an external interrupt for releasing stop mode, use an input that has been set as an interrupt input cause before the system enters stop mode. As an input cause, H level, L level, rising edge or falling edge can be selected.

## ■ Release of Time-base Timer Mode

When time-base timer mode is released, the microcontroller is placed in the PLL clock oscillation stabilization wait state. If the PLL clock is not used, change the MCS bit of the clock selection register (CKSCR) to "1" with the instruction that is to be executed immediately after a reset or on return from an interrupt.

If an external interrupt is used to release time-base timer mode, the input cause can be selected as H level, L level, rising edge or falling edge.

## ■ Oscillation Stabilization Wait Interval

### ● Source clock oscillation stabilization wait interval

Because the oscillator for source oscillation is halted in stop mode, an oscillation stabilization wait interval is required. A time period selected by the WS1 and WS0 bits of CKSCR is used as the oscillation stabilization wait interval.

### ● PLL clock oscillation stabilization wait interval

The CPU may be working with the main clock and the PLL clock may be stopped. If the microcontroller will enter a mode in which the CPU and peripheral functions work with the PLL clock, the PLL clock initially enters the oscillation stabilization wait state. In this state, the CPU still operates using the main clock.

The PLL clock oscillation stabilization wait interval is fixed at  $2^{14}/\text{HCLK}$  (HCLK: oscillation clock frequency).

However, this interval may range from  $2^{14}/\text{HCLK}$  to  $2 \times 2^{14}/\text{HCLK}$  depending on the status of the time-base timer, if the time-base timer is not cleared before the PLL clock oscillation stabilization wait state is entered. (For example, return to the PLL run state from time-base timer mode occurs because of an external reset.)

## ■ Switching the Clock Mode

When the clock mode is switched, do not switch to low power consumption mode and other clock mode before this switching is completed. Confirm the completion of clock mode switching by referring to the MCM and SCM bits of the clock selection register (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.



# **CHAPTER 7**

---

# **INTERRUPT**

**This chapter explains the interrupt and extended intelligent I/O service (EI<sup>2</sup>OS) in the MB90460/465 series.**

- 7.1 Interrupt
- 7.2 Interrupt Causes and Interrupt Vectors
- 7.3 Interrupt Control Registers and Peripheral Functions
- 7.4 Hardware Interrupt
- 7.5 Software Interrupt
- 7.6 Interrupt of Extended Intelligent I/O Service (EI<sup>2</sup>OS)
- 7.7 Exception Processing Interrupt
- 7.8 Stack Operations for Interrupt Processing
- 7.9 Sample Programs for Interrupt Processing

## 7.1 Interrupt

---

This chapter explains the interrupt and extended intelligent I/O service (EI<sup>2</sup>OS) in the MB90460/465 series.

- Hardware interrupt
  - Software interrupt
  - Interrupt from extended intelligent I/O service (EI<sup>2</sup>OS)
  - Exception processing
- 

### ■ Interrupt Types and Functions

#### ● Hardware interrupt

A hardware interrupt transfers control to a user-defined interrupt processing program in response to an interrupt request from a peripheral function.

#### ● Software interrupt

A software interrupt transfers control to a user-defined interrupt processing program triggered by the execution of a dedicated software interrupt instruction (such as the INT instruction).

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

The EI<sup>2</sup>OS function automatically transfers data between a peripheral function and memory. Data transfer, which has ordinarily been executed by an interrupt processing program, can be handled like a direct memory access (DMA). When the specified number of data transfers has been terminated, the interrupt processing program is automatically executed.

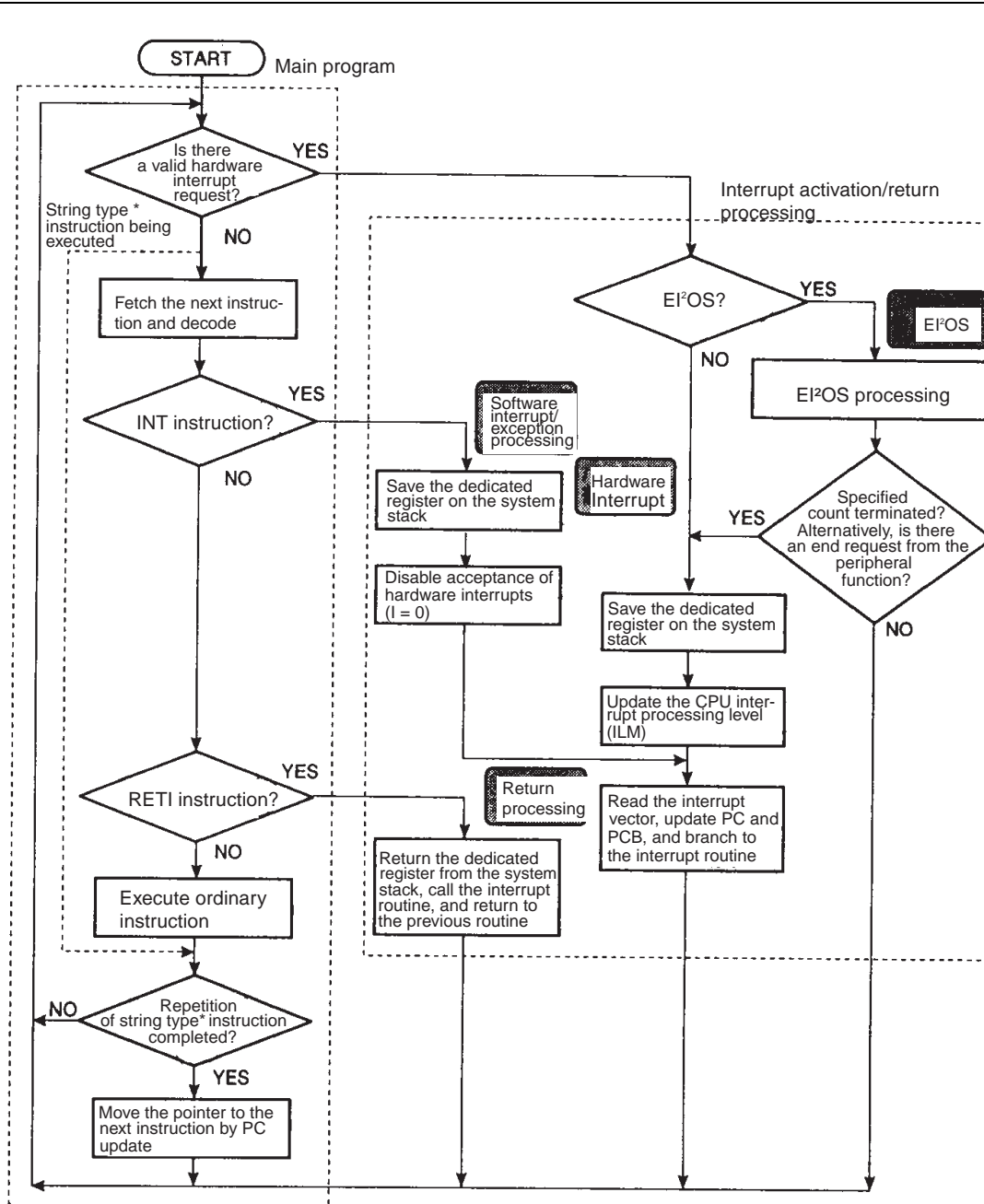
An instruction from EI<sup>2</sup>OS is a type of hardware interrupt.

- Exception processingException processing is basically the same as an interrupt. When an exception event (execution of an undefined instruction) is detected on the instruction boundary, ordinary processing is interrupted and exception processing is performed. This is equivalent to software interrupt instruction INT10.

## ■ Interrupt Operation

Figure 7.1-1 shows the activation and return processing for the four types of interrupt functions.

**Figure 7.1-1 Overall Flow of Interrupt Operation**



\*: When a string type instruction is being executed, the interrupt is evaluated in each step.

## 7.2 Interrupt Causes and Interrupt Vectors

The F<sup>2</sup>MC-16LX has functions for handling 256 types of interrupt causes. The 256 interrupt vector tables are allocated to the memory at the highest addresses. These interrupt vectors are shared by all interrupts.

Software interrupt can use all these interrupt vectors (INT0 to INT256). Software interrupt shares same interrupt vectors with the hardware interrupt and exception processing interrupt. Hardware interrupt uses a fixed interrupt vector and interrupt control register (ICR) for each peripheral function.

### ■ Interrupt Vectors

Interrupt vector tables referenced during interrupt processing are allocated to the highest addresses in the memory area (FFFC00<sub>H</sub> to FFFFFFF<sub>H</sub>). Interrupt vectors share the same area with EI<sup>2</sup>OS, exception processing, hardware and software interrupt.

Table 7.2-1 shows the assignment of interrupt numbers and interrupt vectors.

**Table 7.2-1 Interrupt Vectors**

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode data	Interrupt no.	Hardware interrupt
INT0	FFFFFC <sub>H</sub>	FFFFFD <sub>H</sub>	FFFFFE <sub>H</sub>	Not used	#0	None
:	:	:	:	:	:	:
INT7	FFFFE0 <sub>H</sub>	FFFFE1 <sub>H</sub>	FFFFE2 <sub>H</sub>	Not used	#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>	Not used	#9	None
INT10	FFFFD4 <sub>H</sub>	FFFFD5 <sub>H</sub>	FFFFD6 <sub>H</sub>	Not used	#10	<Exception processing>
INT11	FFFFD0 <sub>H</sub>	FFFFD1 <sub>H</sub>	FFFFD2 <sub>H</sub>	Not used	#11	Hardware interrupt #0
INT12	FFFFCC <sub>H</sub>	FFFFCD <sub>H</sub>	FFFFCE <sub>H</sub>	Not used	#12	Hardware interrupt #1
INT13	FFFFC8 <sub>H</sub>	FFFFC9 <sub>H</sub>	FFFFCA <sub>H</sub>	Not used	#13	Hardware interrupt #2
INT14	FFFFC4 <sub>H</sub>	FFFFC5 <sub>H</sub>	FFFFC6 <sub>H</sub>	Not used	#14	Hardware interrupt #3
:	:	:	:	:	:	:
INT254	FFFC04 <sub>H</sub>	FFFC05 <sub>H</sub>	FFFC06 <sub>H</sub>	Not used	#254	None
INT255	FFFC00 <sub>H</sub>	FFFC01 <sub>H</sub>	FFFC02 <sub>H</sub>	Not used	#255	None

(Reference)

Unused interrupt vectors should be set as the exception processing address.

## ■ Interrupt Causes and Interrupt Vectors/interrupt Control Registers

Table 7.2-2 shows the relationship among interrupt causes (excluding software interrupt), interrupt vectors, and interrupt control registers.

**Table 7.2-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers**

Interrupt cause	EI <sup>2</sup> OS support	Interrupt vector			Interrupt control register		Priority <sup>*2</sup>
		Number		Address	ICR	Address	
Reset	X	#08	08 <sub>H</sub>	FFFFDC <sub>H</sub>	-	-	High
INT9 instruction	X	#09	09 <sub>H</sub>	FFFFD8 <sub>H</sub>	-	-	<div>↑</div> <div>↓</div>
Exception processing	X	#10	0A <sub>H</sub>	FFFFD4 <sub>H</sub>	-	-	
A/D converter conversion termination	O	#11	0B <sub>H</sub>	FFFFD0 <sub>H</sub>	ICR00	0000B0 <sub>H</sub> <sup>*1</sup>	
Output compare channel 0 match	O	#12	0C <sub>H</sub>	FFFFCC <sub>H</sub>			
End of measurement by PWC0 timer / PWC0 timer over-flow*	O	#13	0D <sub>H</sub>	FFFFC8 <sub>H</sub>	ICR01	0000B1 <sub>H</sub> <sup>*1</sup>	
16-bit PPG timer 0	O	#14	0E <sub>H</sub>	FFFFC4 <sub>H</sub>			
Output compare channel 1 match	O	#15	0F <sub>H</sub>	FFFFC0 <sub>H</sub>	ICR02	0000B2 <sub>H</sub> <sup>*1</sup>	
16-bit PPG timer 1*	O	#16	10 <sub>H</sub>	FFFFBC <sub>H</sub>			
Output compare channel 2 match	O	#17	11 <sub>H</sub>	FFFFB8 <sub>H</sub>	ICR03	0000B3 <sub>H</sub> <sup>*1</sup>	
16-bit reload timer 1 underflow	O	#18	12 <sub>H</sub>	FFFFB4 <sub>H</sub>			
Output compare channel 3 match	O	#19	13 <sub>H</sub>	FFFFB0 <sub>H</sub>	ICR04	0000B4 <sub>H</sub> <sup>*1</sup>	
DTP/ext. interrupt channels 0/1 detection	O	#20	14 <sub>H</sub>	FFFFAC <sub>H</sub>			
DTTI0	Δ						
Output compare channel 4 match	O	#21	15 <sub>H</sub>	FFFFA8 <sub>H</sub>	ICR05	0000B5 <sub>H</sub> <sup>*1</sup>	
DTP/ext. interrupt channels 2/3 detection	O	#22	16 <sub>H</sub>	FFFFA4 <sub>H</sub>			
DTTI1*	Δ						
Output compare channel 5 match	O	#23	17 <sub>H</sub>	FFFFA0 <sub>H</sub>	ICR06	0000B6 <sub>H</sub> <sup>*1</sup>	
End of measurement by PWC1 timer / PWC1 timer over-flow	O	#24	18 <sub>H</sub>	FFFF9C <sub>H</sub>			
DTP/ext. interrupt channels 4/5 detection	O	#25	19 <sub>H</sub>	FFFF98 <sub>H</sub>	ICR07	0000B7 <sub>H</sub> <sup>*1</sup>	
Waveform sequencer timer compare match / write timing*	O	#26	1A <sub>H</sub>	FFFF94 <sub>H</sub>			
DTP/ext. interrupt channels 6/7 detection	O	#27	1B <sub>H</sub>	FFFF90 <sub>H</sub>	ICR08	0000B8 <sub>H</sub> <sup>*1</sup>	
Waveform sequencer position detect / compare interrupt*	O	#28	1C <sub>H</sub>	FFFF8C <sub>H</sub>			
Waveform generator 16-bit timer 0/1/2 underflow	Δ	#29	1D <sub>H</sub>	FFFF88 <sub>H</sub>	ICR09	0000B9 <sub>H</sub> <sup>*1</sup>	
16-bit reload timer 0 underflow	O	#30	1E <sub>H</sub>	FFFF84 <sub>H</sub>			
16-bit free-run timer zero detect	Δ	#31	1F <sub>H</sub>	FFFF80 <sub>H</sub>	ICR10	0000BA <sub>H</sub> <sup>*1</sup>	
16-bit PPG timer 2	O	#32	20 <sub>H</sub>	FFFF7C <sub>H</sub>			
Input capture channels 0/1	O	#33	21 <sub>H</sub>	FFFF78 <sub>H</sub>	ICR11	0000BB <sub>H</sub> <sup>*1</sup>	
16-bit free-run timer compare clear	Δ	#34	22 <sub>H</sub>	FFFF74 <sub>H</sub>			
Input capture channels 2/3	O	#35	23 <sub>H</sub>	FFFF70 <sub>H</sub>	ICR12	0000BC <sub>H</sub> <sup>*1</sup>	
Time-base timer	Δ	#36	24 <sub>H</sub>	FFFF6C <sub>H</sub>			
UART1 receive	⊙	#37	25 <sub>H</sub>	FFFF68 <sub>H</sub>	ICR13	0000BD <sub>H</sub> <sup>*1</sup>	
UART1 send	Δ	#38	26 <sub>H</sub>	FFFF64 <sub>H</sub>			
UART0 receive	⊙	#39	27 <sub>H</sub>	FFFF60 <sub>H</sub>	ICR14	0000BE <sub>H</sub> <sup>*1</sup>	
UART0 send	Δ	#40	28 <sub>H</sub>	FFFF5C <sub>H</sub>			
Flash memory status	Δ	#41	29 <sub>H</sub>	FFFF58 <sub>H</sub>	ICR15	0000BF <sub>H</sub> <sup>*1</sup>	
Delayed interrupt generator module	Δ	#42	2A <sub>H</sub>	FFFF54 <sub>H</sub>			
							Low



O: Can be used and interrupt request flag is cleared by EI<sup>2</sup>OS interrupt clear signal.

X: Cannot be used.

⊙: Can be used and support the EI<sup>2</sup>OS stop request.

Δ: Usable when an interrupt cause that shares the ICR is not used.

\*1:- For peripheral functions that share the ICR register, the interrupt level will be the same.

- If the extended intelligent I/O service is to be used with a peripheral function that shares the ICR register with another peripheral function, the service can be started by either of the function. And if EI<sup>2</sup>OS clear is supported, both interrupt request flags for the two interrupt causes are cleared by EI<sup>2</sup>OS interrupt clear signal. It is recommended to mask either of the interrupt request during the use of EI<sup>2</sup>OS.

- EI<sup>2</sup>OS service cannot be started multiple times simultaneously. Interrupt other than the operating interrupt is masked during EI<sup>2</sup>OS operation. It is recommended to mask either of the interrupt requests during the use of EI<sup>2</sup>OS.

\*2: This priority is applied when interrupts of the same level occur simultaneously.

\*: In MB90465 series, these resources are not present, and therefore the interrupts are not available.

## 7.3 Interrupt Control Registers and Peripheral Functions

Interrupt control registers (ICR00 to ICR15) are located inside the interrupt controller. The interrupt control registers correspond to all peripheral functions that have the interrupt function. These registers control interrupts and the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt Control Registers

Table 7.3-1 lists the interrupt control registers and corresponding peripheral functions.

**Table 7.3-1 Interrupt Control Registers**

Address	Register	Abbreviation	Corresponding peripheral function
0000B0 <sub>H</sub>	Interrupt control register 00	ICR00	A/D converter, output compare 0
0000B1 <sub>H</sub>	Interrupt control register 01	ICR01	PWC 0*, 16-bit PPG timer 0
0000B2 <sub>H</sub>	Interrupt control register 02	ICR02	Output compare 1, 16-bit PPG timer 1*
0000B3 <sub>H</sub>	Interrupt control register 03	ICR03	Output compare 2, 16-bit reload timer 1
0000B4 <sub>H</sub>	Interrupt control register 04	ICR04	Output compare 3, DTP/external interrupt 0/1, DTTI0
0000B5 <sub>H</sub>	Interrupt control register 05	ICR05	Output compare 4, DTP/external interrupt 2/3, DTTI1*
0000B6 <sub>H</sub>	Interrupt control register 06	ICR06	Output compare 5, PWC timer 1
0000B7 <sub>H</sub>	Interrupt control register 07	ICR07	DTP/external interrupt 4/5, waveform sequencer*
0000B8 <sub>H</sub>	Interrupt control register 08	ICR08	DTP/external interrupt 6/7, waveform sequencer*
0000B9 <sub>H</sub>	Interrupt control register 09	ICR09	Waveform generator, 16-bit reload timer 0
0000BA <sub>H</sub>	Interrupt control register 10	ICR10	16-bit free-run timer zero detect, 16-bit PPG timer 2
0000BB <sub>H</sub>	Interrupt control register 11	ICR11	Input capture 0/1, 16-bit free-run timer compare clear
0000BC <sub>H</sub>	Interrupt control register 12	ICR12	Input capture 2/3, time-base timer
0000BD <sub>H</sub>	Interrupt control register 13	ICR13	UART1
0000BE <sub>H</sub>	Interrupt control register 14	ICR14	UART0
0000BF <sub>H</sub>	Interrupt control register 15	ICR15	Flash memory, delayed interrupt generator module

\*: In MB90465 series, these resources are not present, therefore, interrupt is not available.

## ■ Interrupt Control Register Functions

All interrupt control registers (ICR) do the following

- Set the interrupt level of the corresponding peripheral function
- Select ordinary interrupt or the extended intelligent I/O service as interrupt of the corresponding peripheral function
- Select an extended intelligent I/O service (EI<sup>2</sup>OS) channel
- Display the status of the extended intelligent I/O service (EI<sup>2</sup>OS)

Some of the functions of the interrupt control registers (ICR) differ during writing and reading, as shown in Figure 7.3-1 and Figure 7.3-2.

---

Note:

Do not use a read-modify-write instruction to access the interrupt control registers (ICR), since operation will not be correct.

---

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

Interrupt control registers correspond to all peripheral functions that have the interrupt function. The interrupt control registers control the processing when an interrupt request occurs. The functions of these registers partially differ at writing and reading.

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

Figure 7.3-1 Interrupt Control Registers (ICR00 to ICR15) during writing

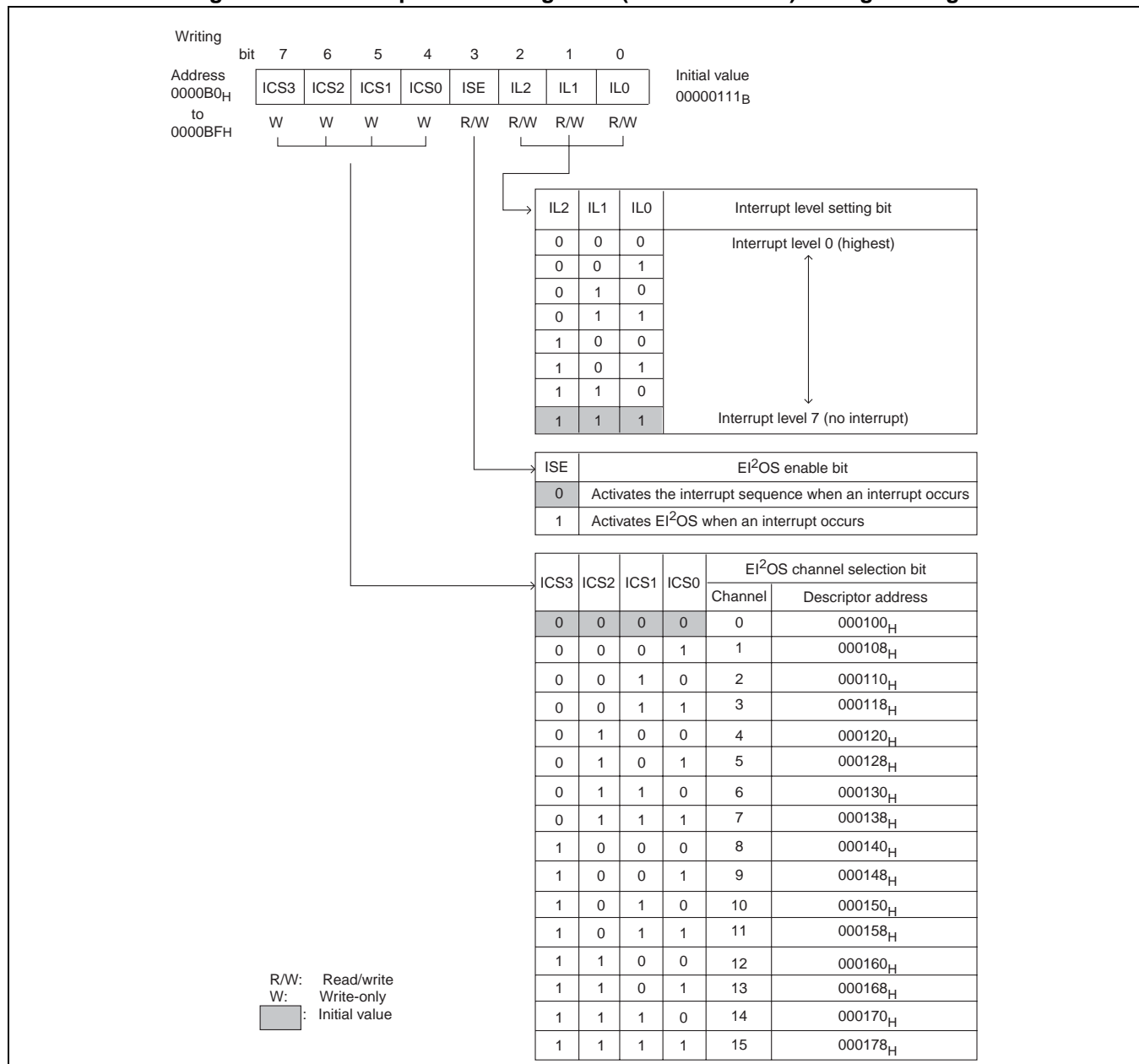
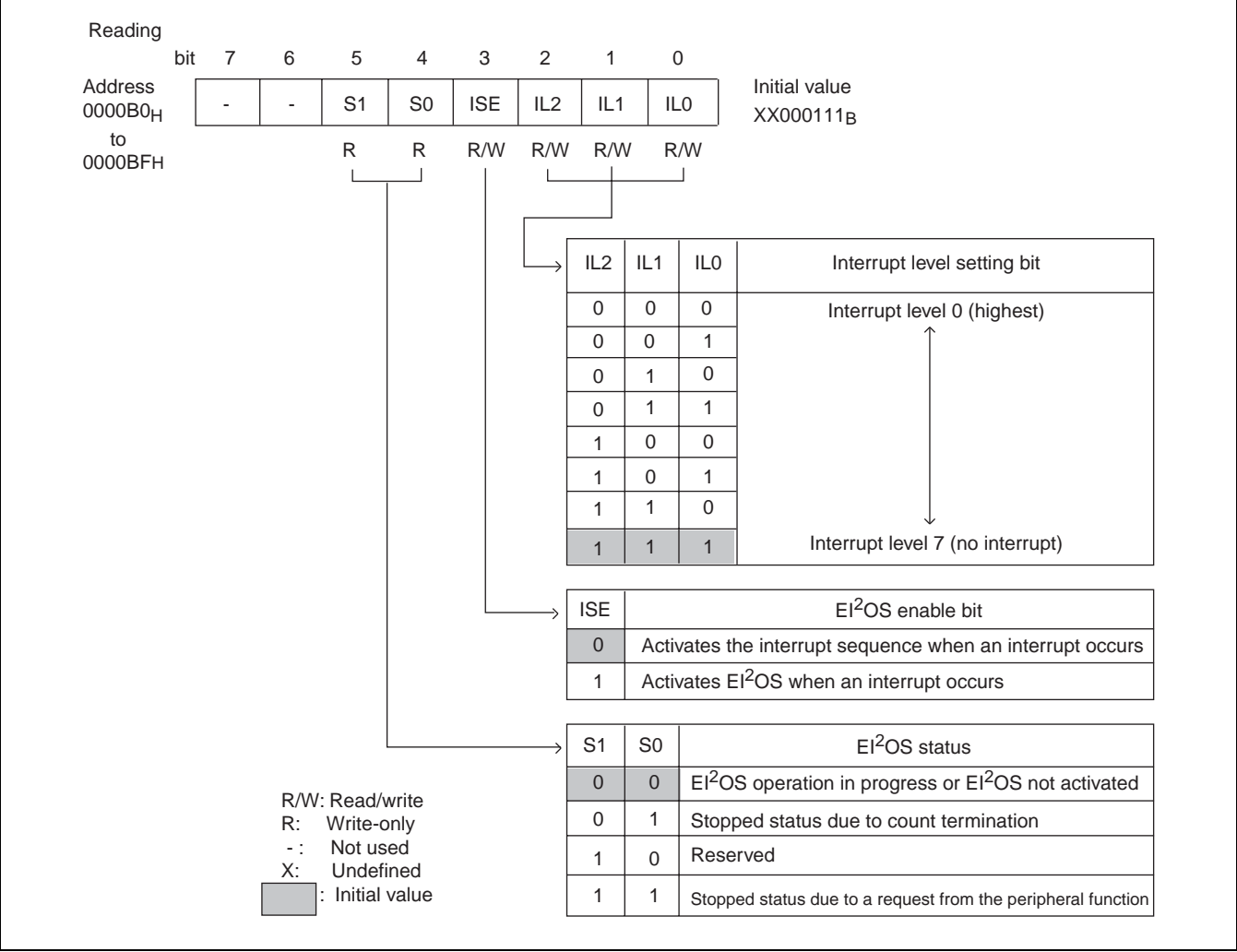


Figure 7.3-2 Interrupt Control Registers (ICR00 to ICR15) during reading



## 7.3.2 Interrupt Control Register Functions

The interrupt control registers (ICR00 to ICR15) consist of the following four functional bits:

- Interrupt level setting bits (IL2 to IL0)
- Extended intelligent I/O service (EI<sup>2</sup>OS) enable bit (ISE)
- Extended intelligent I/O service (EI<sup>2</sup>OS) channel selection bits (ICS3 to ICS0)
- Extended intelligent I/O service (EI<sup>2</sup>OS) status (S1 and S0)

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

Figure 7.3-3 shows the configuration of the interrupt control register (ICR) bits.

**Figure 7.3-3 Configuration of Interrupt Control Registers (ICR)**

Writing to interrupt control register (ICR)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B0H	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	00000111 <sub>B</sub>
to									
0000BFH	W	W	W	W	W	W	W	W	
Reading of interrupt control register (ICR)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B0H	—	—	S1	S0	ISE	IL2	IL1	IL0	XX000111 <sub>B</sub>
to									
0000BFH	—	—	R	R	R	R	R	R	

R: Read-only  
W: Write-only  
- : Not used

References:

- The ICS3 to ICS0 bits are valid only when the extended intelligent I/O service (EI<sup>2</sup>OS) has been activated. To activate EI<sup>2</sup>OS, set the ISE bit to "1". To not activate EI<sup>2</sup>OS, set the ISE bit to "0". When EI<sup>2</sup>OS is not activated, setting ICS3 to ICS0 is optional.
- ICS1 and ICS0 are valid only for writing. S1 and S0 are valid only for reading.

### ■ Interrupt Control Register Functions

#### ● Interrupt level setting bits (IL2 to IL0)

These bits set the interrupt level of the corresponding peripheral function. These bits are initialized to level 7 (no interrupt) by a reset.

Table 7.3-2 shows the correspondence between the interrupt level setting bits and interrupt levels.

**Table 7.3-2 Correspondence between the Interrupt Level Setting Bits and Interrupt Levels**

IL2	IL1	IL0	Interrupt level
0	0	0	0 (highest priority)
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	6 (lowest priority)
1	1	1	7 (no interrupt)

● Extended intelligent I/O service (EI<sup>2</sup>OS) enable bit (ISE)

If this bit is "1" when an interrupt request is generated, EI<sup>2</sup>OS is activated. If this bit is "0" at when an interrupt request is generated, the interrupt sequence is activated. When the EI<sup>2</sup>OS termination condition is met (when the S1 and S0 bits are not 00<sub>B</sub>), the ISE bit is cleared. If the corresponding peripheral function does not have the EI<sup>2</sup>OS function, the ISE bit must be set to "0" by software. The ISE bit is initialized to "0" by a reset.

● Extended intelligent I/O service (EI<sup>2</sup>OS) channel selection bits (ICS3 to ICS0)

These write-only bits specify the EI<sup>2</sup>OS channel. The EI<sup>2</sup>OS descriptor address is determined based on the value set here. The ICS bit is initialized to 0000<sub>B</sub> by a reset.

Table 7.3-3 shows the correspondence between the EI<sup>2</sup>OS channel selection bits and descriptor addresses.

**Table 7.3-3 Correspondence between the EI<sup>2</sup>OS Channel Selection Bits and Eescriptor Addresses (1 / 2)**

ICS3	ICS2	ICS1	ICS0	Selected channel	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>

**Table 7.3-3 Correspondence between the EI<sup>2</sup>OS Channel Selection Bits and Eescriptor Addresses (2 / 2)**

ICS3	ICS2	ICS1	ICS0	Selected channel	Descriptor address
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>

● Extended intelligent I/O service (EI<sup>2</sup>OS) status bits (S1, S0)

These are read-only bits. When this value is checked at EI<sup>2</sup>OS termination, the operating status and termination status can be distinguished. These bits are initialized to 00<sub>B</sub> by a reset.

Table 7.3-4 shows the relationship between the S0 and S1 bits and the EI<sup>2</sup>OS status.

**Table 7.3-4 Relationship between EI<sup>2</sup>OS Status Bits and the EI<sup>2</sup>OS Status**

S1	S0	EI <sup>2</sup> OS status
0	0	EI <sup>2</sup> OS operation in progress or EI <sup>2</sup> OS not activated
0	1	Stopped status due to count termination
1	0	Reserved
1	1	Stopped status due to a request from the peripheral function



## 7.4 Hardware Interrupt

---

**The hardware interrupt function temporarily interrupts the program being executed by the CPU and transfers control to a user-defined interrupt processing program in response to an interrupt signal from a peripheral function.**

**The extended intelligent I/O service (EI<sup>2</sup>OS) and external interrupt are executed as a type of hardware interrupt.**

---

### ■ Hardware Interrupt

#### ● Hardware interrupt function

The hardware interrupt function compares the interrupt level of the interrupt request signal output by a peripheral function with the interrupt level mask register (ILM) in the CPU processor status (PS). The function then references the contents of the I flag in the processor status (PS) through the hardware and decides if the interrupt can be accepted.

When the hardware interrupt is accepted, the CPU internal registers are automatically saved on the system stack. The currently requested interrupt level is stored in the interrupt level mask register (ILM), and the function branches to the corresponding interrupt vector.

#### ● Multiple interrupts

Multiple hardware interrupts can be activated.

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

EI<sup>2</sup>OS is an automatic transfer function between memory and I/O. When the specified transfer count has been completed, a hardware interrupt is activated. Multiple EI<sup>2</sup>OS activation does not occur. During EI<sup>2</sup>OS processing, all other interrupt requests and EI<sup>2</sup>OS requests are held.

#### ● External interrupt

An external interrupt (including wake-up interrupt) is accepted from a peripheral function (interrupt request detection circuit) as a hardware interrupt.

#### ● Interrupt vector

Interrupt vector tables referenced during interrupt processing are allocated to memory at FFFC00<sub>H</sub> to FFFFFFF<sub>H</sub>. These tables are shared by software interrupts.

See "7.2 Interrupt Causes and Interrupt Vectors", for more information about the allocation of interrupt numbers and interrupt vectors.

## ■ Hardware Interrupt Structure

Table 7.4-1 lists four mechanisms used for hardware interrupt. These four mechanisms must be included in the program before hardware interrupt can be used.

**Table 7.4-1 Mechanisms used for Hardware Interrupt**

	Mechanism	Function
Peripheral function	Interrupt enable bit, interrupt request bit	Controls interrupt requests from a peripheral function
Interrupt controller	Interrupt control register (ICR)	Sets the interrupt level and controls EI <sup>2</sup> OS
CPU	Interrupt enable flag (I)	Identifies the interrupt enable status
	Interrupt level mask register (ILM)	Compares the request interrupt level and current interrupt level
	Microcode	Executes the interrupt processing routine
FFFC00 <sub>H</sub> to FFFFFF <sub>H</sub> in memory	Interrupt vector table	Stores the branch destination address for interrupt processing

These four mechanisms must be included in the program before hardware interrupt can be used.

## ■ Hardware Interrupt Suppression

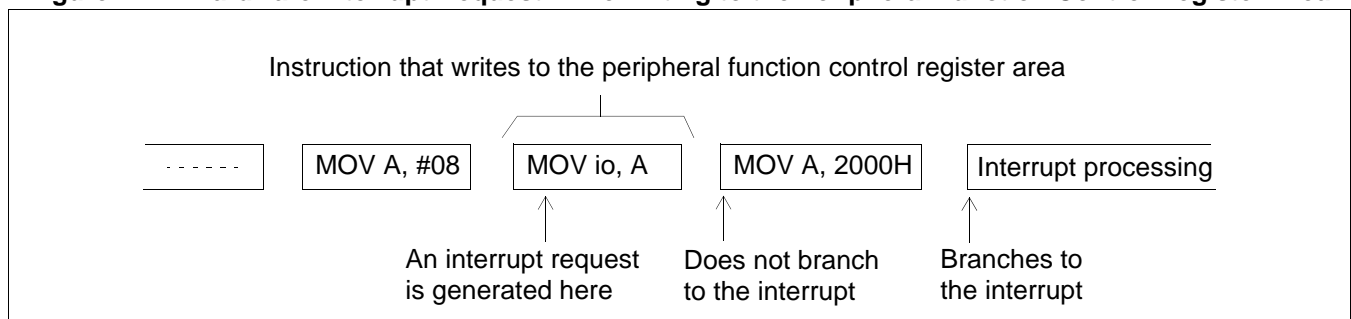
Acceptance of hardware interrupt requests is suppressed under the following conditions.

- Hardware interrupt suppression during writing to the peripheral function control register area

When data is being written to the peripheral function control register area, hardware interrupt requests are not accepted. This prevents the CPU from making operational mistakes. The mistakes may be caused if an interrupt request is generated during data is written to the interrupt control registers for a resource. The peripheral function control register area is not the I/O addressing area at 000000<sub>H</sub> to 0000FF<sub>H</sub>, but the area allocated to the control register of the peripheral function control register and data register.

Figure 7.4-1 shows hardware interrupt operation during writing to the built-in resource area.

**Figure 7.4-1 Hardware Interrupt Request while writing to the Peripheral Function Control Register Area**



● Hardware interrupt suppression by interrupt suppression instruction

The ten types of hardware interrupt suppression instructions listed in Table 7.4-2 ignore interrupt requests without detecting whether a hardware interrupt request exists.

**Table 7.4-2 Hardware Interrupt Suppression Instruction**

	Prefix code	Interrupt/hold suppression instructions (instructions that delay the effect of the prefix code)
Instructions that do not accept interrupt and hold requests	PCB DTB ADB SPB CMR NCC	MOV ILM, #imm8 OR CCR, #imm8 AND CCR, #imm8 POPW PS

Even if a valid hardware interrupt request is generated during execution of one of these instructions, the interrupt is not processed until the first time an instruction of a different type is executed.

● Hardware interrupt suppression during execution of software interrupt

When a software interrupt is activated, the I flag is cleared to "0". In this state, other interrupt requests cannot be accepted.

## 7.4.1 Operation of Hardware Interrupt

---

**This section explains hardware interrupt operation from generation of a hardware interrupt request to the completion of interrupt processing.**

---

### ■ Hardware Interrupt Activation

#### ● Peripheral function operation (generation of an interrupt request)

A peripheral function that has a hardware interrupt request function also has an interrupt request flag that indicates the presence of interrupt requests and an interrupt enable flag that determines whether CPU interrupt requests are enabled or disabled. The interrupt request flag is set when an event specific to the peripheral function occurs.

#### ● Interrupt controller operation (interrupt request control)

The interrupt controller compares the interrupt levels (IL) of interrupt requests received at the same time. The interrupt controller selects the request with the highest level (with the smallest IL value) and posts it to the CPU. When multiple requests have the same level, the request with the smallest interrupt number has the highest priority.

#### ● CPU operation (interrupt request acceptance and interrupt processing)

The CPU compares the received interrupt level (ICR: IL2 to IL0) and the interrupt level mask register (ILM). If  $IL < ILM$  and interrupts are enabled (PS: CCR: I = 1), the CPU activates the interrupt processing microcode after the instruction currently being executed terminates.

At the beginning of the interrupt processing microcode, the CPU references the ISE bit in the interrupt control register (ICR). If  $ISE = 0$ , the CPU continues the execution of interrupt processing. (If  $ISE = 1$ , EI<sup>2</sup>OS is activated.)

Interrupt processing saves the contents of the dedicated registers (12 bytes including A, DPR, ADB, DTB, PCB, PC and PS) on the system stack (the system stack space indicated by the SSB and SSP).

The CPU then loads data into the interrupt vector program counters (PCB and PC), updates the ILM, and sets the stack flag (S) (sets CCR: S = 1 and activates the system stack).

### ■ Returning from a Hardware Interrupt

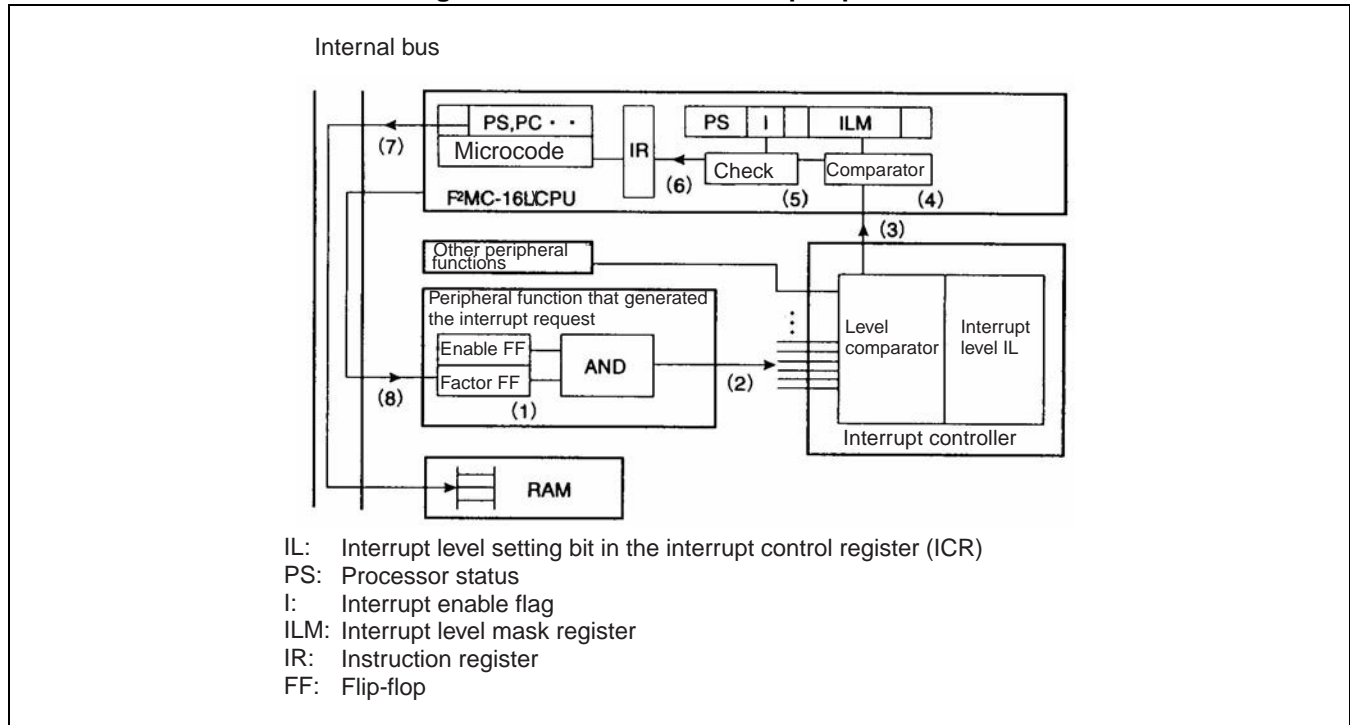
In an interrupt processing program, when the interrupt request flag of the peripheral function that generated the interrupt cause is cleared and the RETI instruction is executed, 12-byte data saved on the system stack is restored to the dedicated registers and the processing that was being executed before branching for the interrupt is resumed.

When the interrupt request flag is cleared, interrupt requests output by the peripheral function to the interrupt controller are automatically canceled.

## ■ Hardware Interrupt Operation

Figure 7.4-2 shows hardware interrupt operation from generation of a hardware interrupt to the completion of interrupt processing.

**Figure 7.4-2 Hardware Interrupt Operation**



- (1) An interrupt cause is generated within the peripheral function.
- (2) The interrupt enable bit of the peripheral function is referenced. If the interrupt is enabled, the interrupt request is output from the peripheral function to the interrupt controller.
- (3) The interrupt controller that receives the interrupt request determines the priority of simultaneous interrupt requests, then transfers the interrupt level (IL) that matches the corresponding interrupt request to the CPU.
- (4) The CPU compares the interrupt level (IL) requested by the interrupt controller with the interrupt level mask register (ILM).
- (5) If the comparison indicates a higher priority than the current interrupt processing level, the CPU checks the contents of the I flag in the condition code register (CCR).
- (6) If in the check in (5) the I flag is interrupt enabled ( $I = 1$ ), the CPU waits until the execution of the instruction currently being executed terminates. At termination, the CPU sets the requested level (IL) in the ILM.
- (7) Registers are saved, and processing branches to the interrupt processing routine.
- (8) The interrupt cause that was generated in (1) is cleared by software in the interrupt processing routine. Execution of the RETI instruction terminates the interrupt processing.

## 7.4.2 Processing for Interrupt Operation

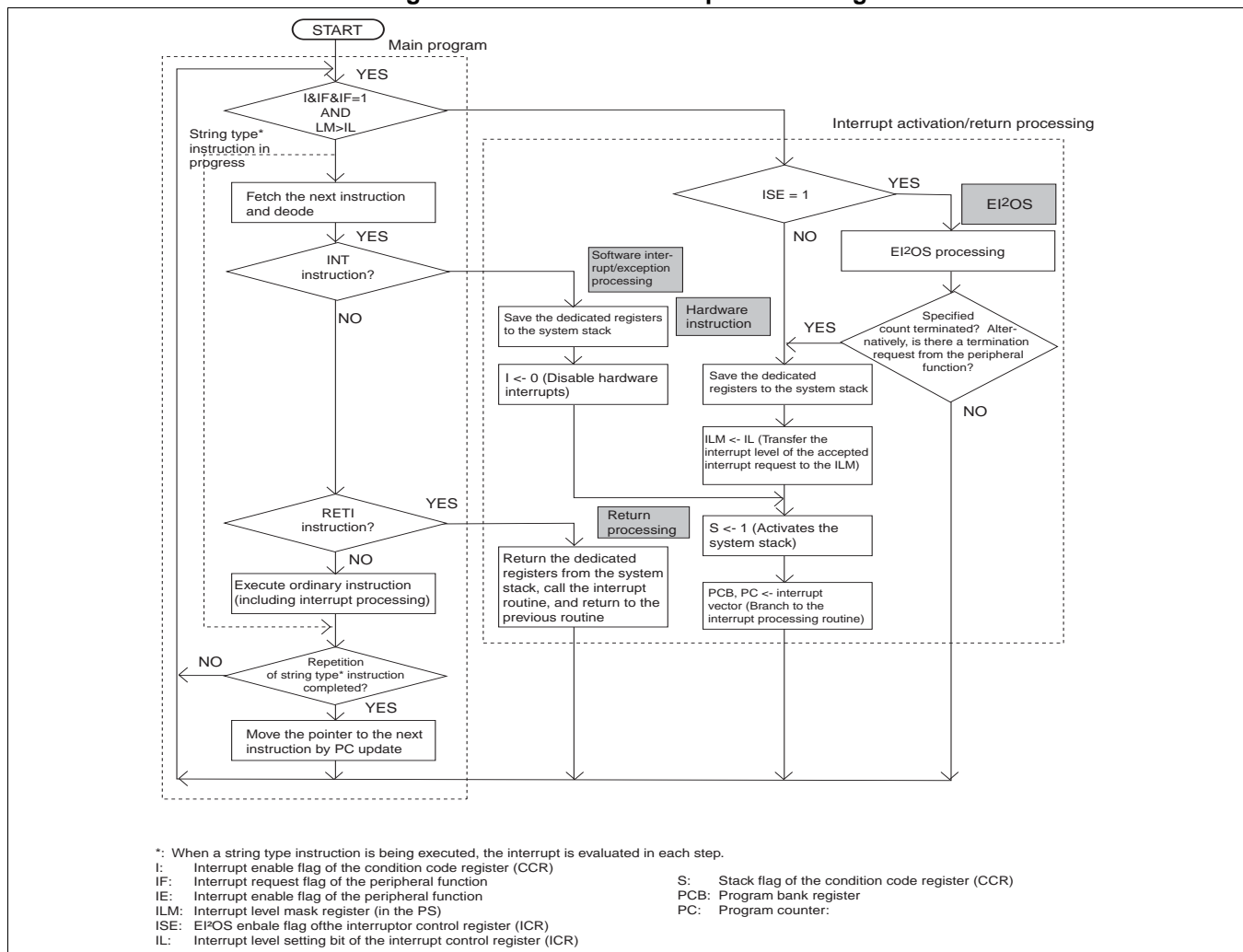
When an interrupt request is generated by the peripheral function, the interrupt controller transmits the interrupt level to the CPU. If the CPU is able to accept interrupt, the interrupt controller temporarily interrupts the instruction currently being executed. The interrupt controller then executes the interrupt processing routine or activates the extended intelligent I/O service (EI<sup>2</sup>OS).

If a software interrupt is generated by the INT instruction, the interrupt processing routine is executed regardless of the CPU status. In this case, hardware interrupt is not allowed.

### ■ Processing for Interrupt Operation

Figure 7.4-3 shows the flow of processing for interrupt operation.

Figure 7.4-3 Flow of Interrupt Processing



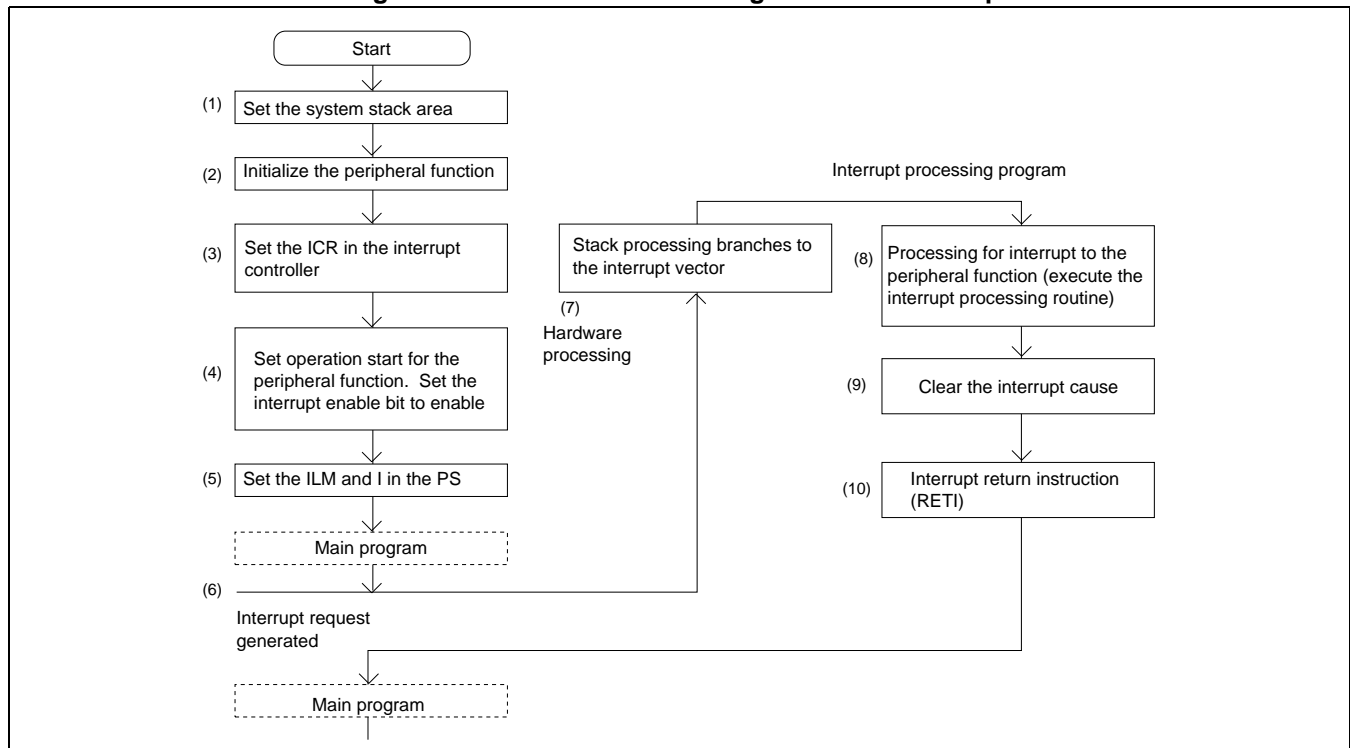
### 7.4.3 Procedure for using Hardware Interrupt

Before hardware interrupt can be used, the system stack area, peripheral function, and interrupt control register (ICR) must be set.

#### ■ Procedure for using Hardware Interrupt

Figure 7.4-4 shows an example of the procedure for using hardware interrupt.

**Figure 7.4-4 Procedure for using Hardware Interrupt**



- (1) Set the system stack area.
- (2) Initialize a peripheral function that can generate interrupt requests.
- (3) Set the interrupt control register (ICR) in the interrupt controller.
- (4) Set the peripheral function to the operation start status, and set the interrupt enable bit to enable.
- (5) Set the interrupt level mask register (ILM) and interrupt enable flag (I) to interrupt acceptable.
- (6) An interrupt generated in the peripheral function causes a hardware interrupt request.
- (7) The interrupt processing hardware saves the registers and branches to the interrupt processing program.
- (8) The interrupt processing program processes the peripheral function in response to the generated interrupt.
- (9) Clear the peripheral function interrupt request.
- (10) Execute the interrupt return instruction, and return to the program before branching.

## 7.4.4 Multiple Interrupts

---

**Multiple hardware interrupts can be implemented by setting different interrupt levels in the interrupt level setting bits (IL0, IL1, IL2) of the interrupt control register (ICR) in response to multiple interrupt requests from peripheral functions. Use of multiple interrupts, however, is not possible with the extended intelligent I/O service.**

---

### ■ Multiple Interrupts

#### ● Operation of multiple interrupts

During execution of an interrupt processing routine, if an interrupt request with a higher-priority interrupt level is generated, the current interrupt processing is interrupted and the interrupt request with the higher-priority interrupt level is accepted. When the interrupt request with the higher-priority interrupt level terminates, the CPU returns to the previous interrupt processing.

0 to 7 can be set as the interrupt level. If level 7 is set, the CPU does not accept interrupt requests.

During execution of interrupt processing, if an interrupt request with the same or lower-priority interrupt level is generated, the new interrupt request is held until the current interrupt terminates unless the I flag or ILM is changed.

Other multiple interrupts to be activated during an interrupt can be temporarily disabled by setting the I flag in the condition code register (CCR) in the interrupt processing routine to interrupts not allowed (CCR: I = 0) or the interrupt level mask register (ILM) to interrupts not allowed (ILM = 000<sub>B</sub>).

#### Note:

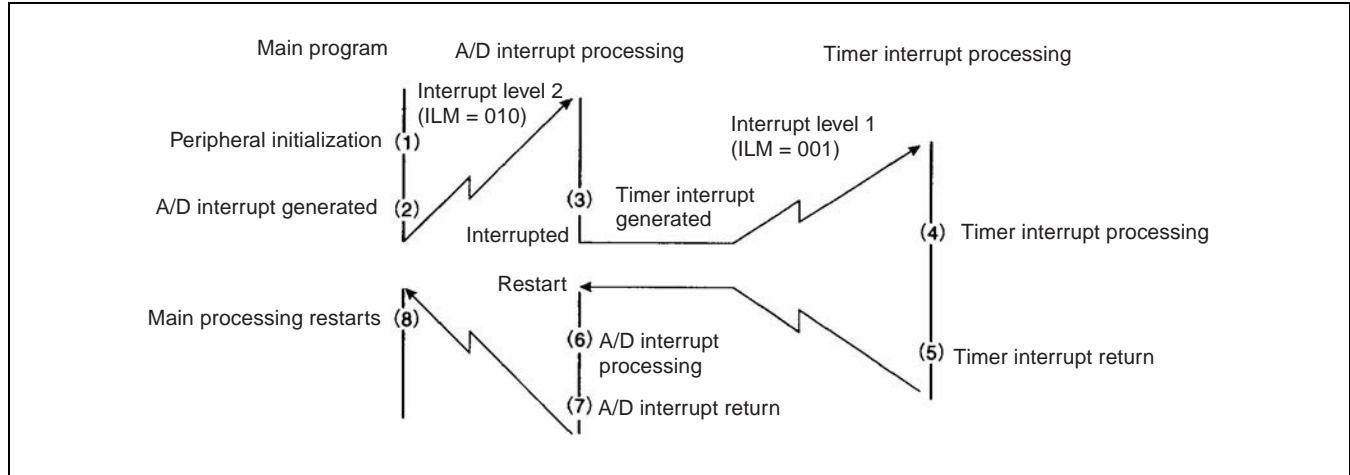
The extended intelligent I/O service (EI<sup>2</sup>OS) cannot be used for the activation of multiple interrupts. During processing of the extended intelligent I/O service (EI<sup>2</sup>OS), all other interrupt requests and extended intelligent I/O service requests are held.

---

#### ● Example of multiple interrupts

This example of multiple interrupt processing assumes that a timer interrupt is given a higher priority than an A/D converter interrupt. In this example, the A/D converter interrupt level is set to "2", and the timer interrupt level is set to "1". If a timer interrupt is generated during processing of the A/D converter interrupt, the processing shown in Figure 7.4-5 is performed.



**Figure 7.4-5 Example of Multiple Interrupts****(1) A/D interrupt generated**

When the A/D converter interrupt processing starts, the interrupt level mask register (ILM) automatically has the same value (2 in the example) as the A/D converter interrupt level (ICR: IL2 to IL0).

If a level-1 or level-0 interrupt request is generated, this interrupt processing has priority.

**(2) Interrupt processing terminated**

When the interrupt processing terminates and the return instruction (RETI) is executed, the values of the dedicated registers (A, DPR, ADB, DTB, PCB, PC and PS) are returned from the stack, and the interrupt level mask register (ILM) has the value that it had before the interrupt.

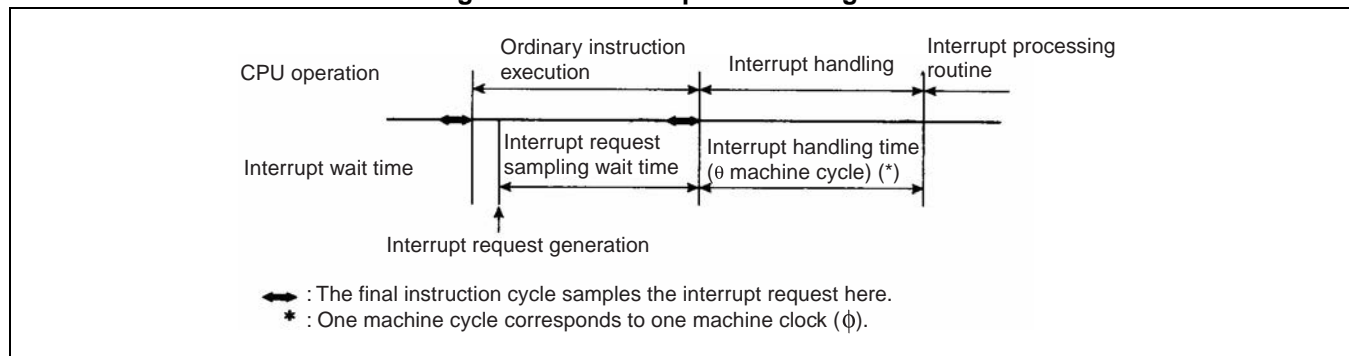
## 7.4.5 Hardware Interrupt Processing Time

From the generation of a hardware interrupt request to the execution of an interrupt processing routine, the time for the instruction currently being executed to terminate and the time required to handle an interrupt are necessary.

### ■ Hardware Interrupt Processing Time

From the generation of a hardware interrupt request to the acceptance of the interrupt and to the execution of an interrupt processing routine, the time to wait for sampling for an interrupt request and the time required to handle an interrupt (time to prepare for interrupt processing) are necessary. Figure 7.4-6 shows the interrupt processing time.

**Figure 7.4-6 Interrupt Processing Time**



#### ● Interrupt request sampling wait time

The interrupt request sampling wait time is the time from the generation of an interrupt request to the termination of the instruction currently being executed.

Whether an interrupt request has been generated is determined by sampling the instruction for an interrupt request in the final cycle of the instruction. Consequently, the CPU cannot identify an interrupt request during execution of each instruction creating a delay.

The interrupt request sampling wait time is the maximum when an interrupt request is generated as soon as the POPW RW0, ... RW7 instruction (45 machine cycles), which takes the longest to execute, starts.

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

The CPU saves dedicated registers to the system stack and fetches interrupt vectors after it receives an interrupt request. The required handling time for this processing is  $f$  machine cycles. The interrupt handling time is calculated with the following formula:

When an interrupt is activated:  $\theta = 24 + 6 + Z$  machine cycles

When control is returned from an interrupt:  $\theta = 11 + 6 + Z$  machine cycles (RETI instruction)

The interrupt handling time is different for each address pointed to by the stack pointer.

Table 7.4-3 shows the interpolation values ( $Z$ ) for the interrupt handling time.

**Table 7.4-3 Interpolation Values ( $Z$ ) for the Interrupt Handling Time**

Address pointed to by the stack pointer	Interpolation value ( $Z$ )
External 8-bit	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

---

Reference:

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

---

## 7.5 Software Interrupt

---

**When the software interrupt instruction (INT instruction) is executed, the software interrupt function transfers control from the program being executed by the CPU to the user-defined interrupt processing program. Hardware interrupt is disabled during execution of a software interrupt.**

---

### ■ Software Interrupt Activation

#### ● Software interrupt activation

The INT instruction is used to activate a software interrupt. There is no interrupt request flag or enable flag for software interrupt requests. When the INT instruction is executed, an interrupt request is always generated.

#### ● Hardware interrupt suppression

Since the INT instruction does not have interrupt levels, the interrupt level mask register (ILM) is not updated. During the execution of the INT instruction, the I flag of the condition code register (CCR) is set to "0", and hardware interrupts are masked.

To enable hardware interrupts during software interrupt processing, set the I flag to "1" in the software interrupt processing routine.

#### ● Software interrupt operation

When the CPU fetches the INT instruction, the software interrupt processing microcode is activated. This microcode saves the internal CPU registers on the system stack, masks hardware interrupts (CCR: I = 0), and branches to the corresponding interrupt vector.

See "7.2 Interrupt Causes and Interrupt Vectors", in Chapter 7 for more information about the allocation of interrupt numbers and interrupt vectors.

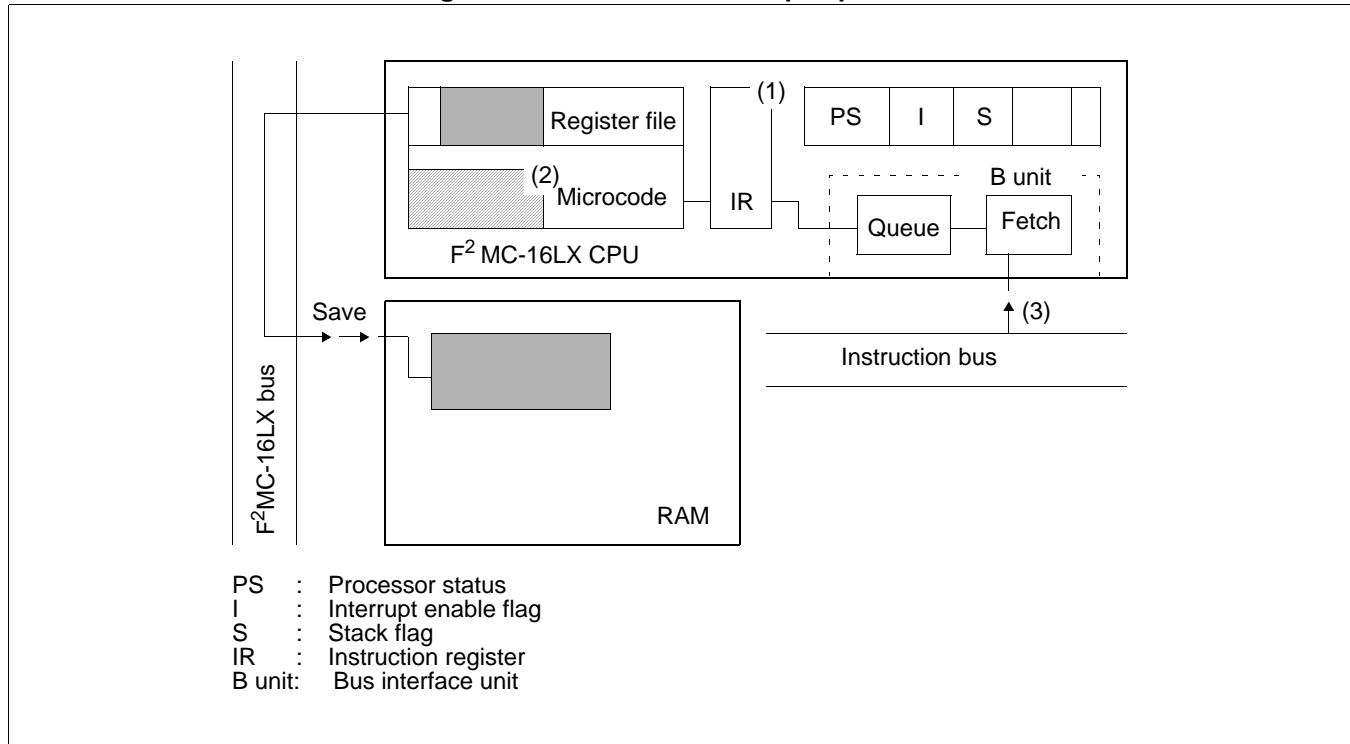
### ■ Returning from a Software Interrupt

In the interrupt processing program, when the interrupt return instruction (RETI instruction) is executed, the 12-byte data saved to the system stack is restored to the dedicated registers and the processing that was being executed before branching for the interrupt is resumed.

## ■ Software Interrupt Operation

Figure 7.5-1 shows software interrupt operation from the generation of a software interrupt to the completion of interrupt processing.

**Figure 7.5-1 Software Interrupt Operation**



- (1) A software interrupt instruction is executed.
- (2) The dedicated registers are saved according to the microcode that corresponds to the software interrupt instruction, and other necessary processing is performed. Branch processing is then executed.
- (3) The RETI instruction in the user interrupt processing routine terminates the interrupt processing.

### Note:

When the program bank register (PCB) is  $FF_H$ , the vector area of the CALLV instruction overlaps the INT #vct8 instruction table. When creating the software, be careful of the duplicated address of the CALLV instruction and INT #vct8 instruction.

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

---

**The extended intelligent I/O service (EI<sup>2</sup>OS) automatically transfers data between a peripheral function (I/O) and memory. When the data transfer terminates, a hardware interrupt is generated.**

---

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

The extended intelligent I/O service is a type of hardware interrupt. It automatically transfers data between a peripheral function (I/O) and a memory. Traditionally, data transfer with a peripheral function (I/O) has been performed by the interrupt processing program. EI<sup>2</sup>OS performs this data transfer in the same way as direct memory access (DMA). At termination, EI<sup>2</sup>OS sets the termination condition and automatically branches to the interrupt processing routine. The user creates programs only for EI<sup>2</sup>OS activation and termination. Data transfer programs in between are not required.

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

Compared to data transfer performed by the interrupt processing routine, EI<sup>2</sup>OS has the following advantages.

- Coding a transfer program is not necessary, reducing program size.
- Because transfer can be stopped depending on the peripheral function (I/O) status, unnecessary data transfer can be eliminated.
- Incrementing or no update can be selected for the buffer address.
- Incrementing or no update can be selected for the I/O register address.

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

When data transfer by EI<sup>2</sup>OS terminates, a termination condition is set in the S1 and S0 bits in the interrupt control register (ICR). Processing then automatically branches to the interrupt processing routine.

The EI<sup>2</sup>OS termination factor can be determined by checking the EI<sup>2</sup>OS status (ICR: S1, S0) with the interrupt processing program.

Interrupt numbers and interrupt vectors are permanently set for each peripheral. See "7.2 Interrupt Causes and Interrupt Vectors", in Chapter 7 for more information.

#### ● Interrupt control register (ICR)

This register, which is located in the interrupt controller, activates EI<sup>2</sup>OS, specifies the EI<sup>2</sup>OS channel, and displays the EI<sup>2</sup>OS termination status.

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

This descriptor, which is located in RAM at 000100<sub>H</sub> to 00017F<sub>H</sub>, is an eight-byte data that retains the transfer mode, I/O address, transfer count, and buffer address. The descriptor handles 16 channels. The channel is specified by the interrupt control register (ICR).

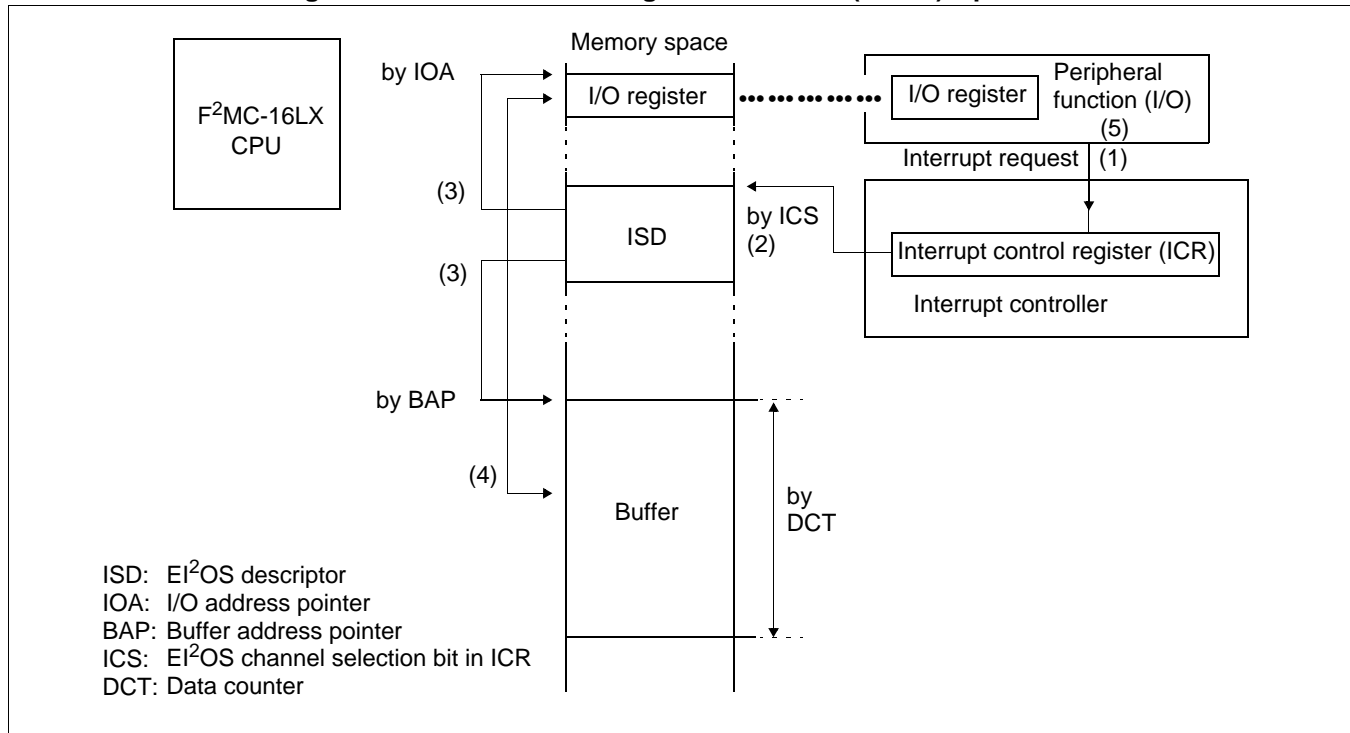
Note:

When the extended intelligent I/O service (EI<sup>2</sup>OS) is operating, execution of the CPU program stops.

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

Figure 7.6-1 shows EI<sup>2</sup>OS operation.

**Figure 7.6-1 Extended Intelligent I/O Service (EI<sup>2</sup>OS) Operation**



- (1) I/O requests transfer.
- (2) The interrupt controller selects the descriptor.
- (3) The transfer source and transfer destination are read from the descriptor.
- (4) Transfer is performed between I/O and memory.
- (5) The interrupt cause is automatically cleared.

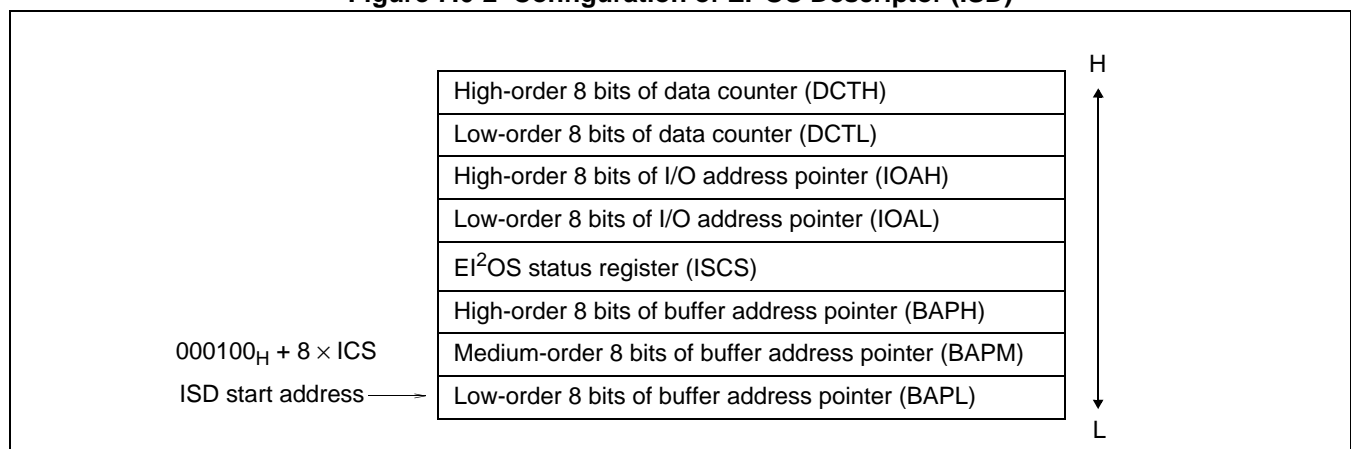
## 7.6.1 Extended Intelligent I/O Service (EI<sup>2</sup>OS) Descriptor (ISD)

The extended intelligent I/O service (EI<sup>2</sup>OS) descriptor (ISD) resides in internal RAM at 000100<sub>H</sub> to 00017F<sub>H</sub>. The ISD consists of 8 bytes x 16 channels.

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

The ISD consists of 8 bytes x 16 channels. Each ISD has the structure shown in Figure 7.6-2. Table 7.6-1 shows the correspondence between channel numbers and ISD addresses.

**Figure 7.6-2 Configuration of EI<sup>2</sup>OS Descriptor (ISD)**



**Table 7.6-1 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>

Registers of the extended intelligent I/O service (EI<sup>2</sup>OS) descriptor (ISD)



## 7.6.2 Registers of EI<sup>2</sup>OS Descriptor (ISD)

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

**Note:** that the initial value of each register is undefined after a reset.

### ■ Data Counter (DCT)

The DCT is a 16-bit register that serves as a counter for the data transfer count. After each data transfer, the counter is decremented by "1". When the counter reaches zero, EI<sup>2</sup>OS terminates.

Figure 7.6-3 shows the configuration of the DCT.

**Figure 7.6-3 Configuration of DCT**

	bit	15	14	13	12	11	10	9	8	
Upper byte of data counter		B15	B14	B13	B12	B11	B10	B09	B08	DCTH
Initial value ⇨		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
	bit	7	6	5	4	3	2	1	0	
Lower byte of data counter		B07	B06	B05	B04	B03	B02	B01	B00	DCTL
Initial value ⇨		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

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

The IOA is a 16-bit register that indicates the lower address (A15 to A00) of the I/O register used to transfer data to and from the buffer. The upper address (A23 to A16) is all zeros. Any I/O from 000000<sub>H</sub> to 00FFFF<sub>H</sub> can be specified by address. Figure 7.6-4 shows the configuration of the IOA.

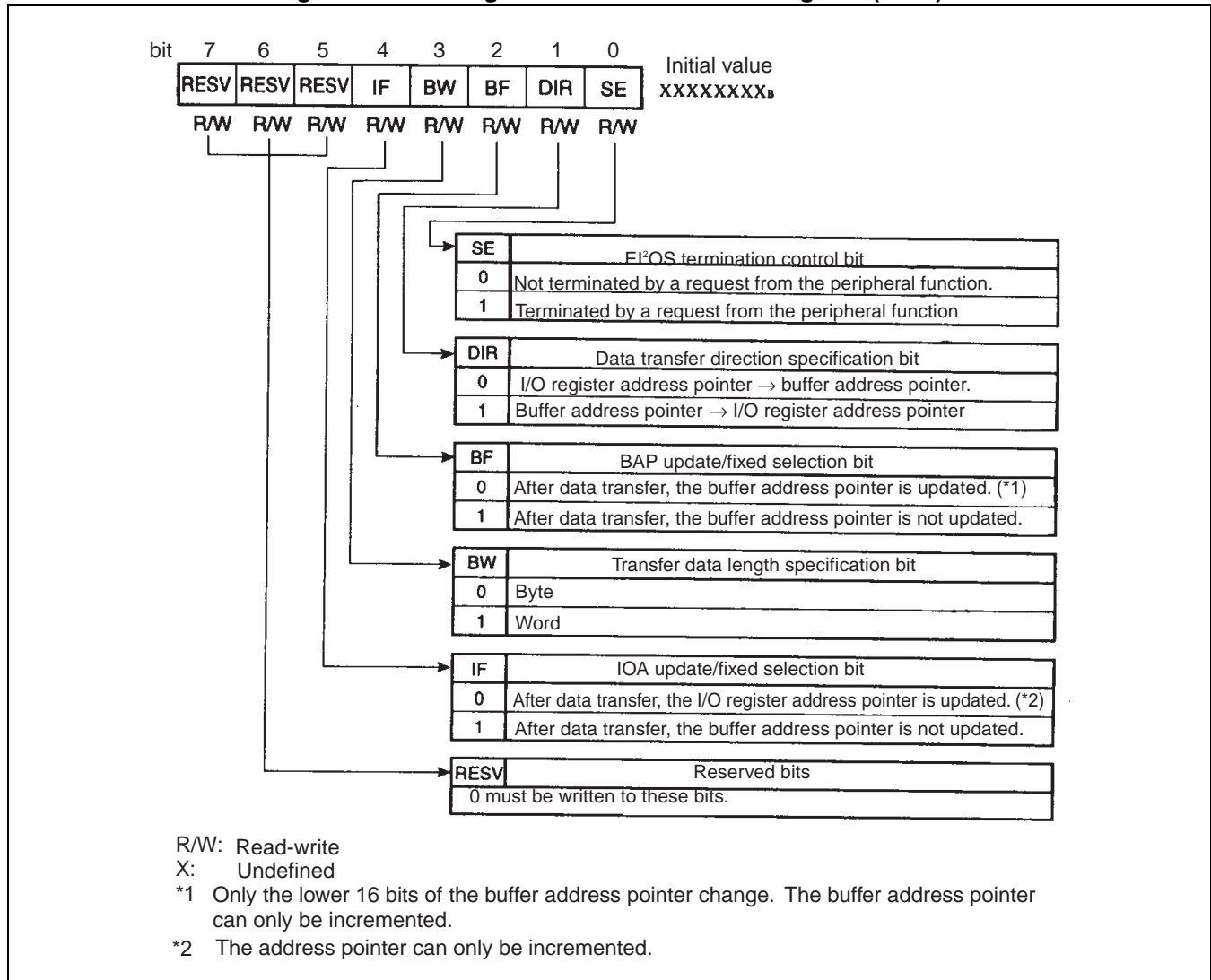
**Figure 7.6-4 Configuration of I/O Register Address Pointer (IOA)**

	bit	15	14	13	12	11	10	9	8	
Upper address pointer		A15	A14	A13	A12	A11	A10	A09	A08	IOAH
Initial value ⇨		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
	bit	7	6	5	4	3	2	1	0	
Lower address pointer		A07	A06	A05	A04	A03	A02	A01	A00	IOAL
Initial value ⇨		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

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

The ISCS is an 8-bit register. The ISCS indicates the update/fixed for the buffer address pointer and I/O register address pointer, transfer data format (byte or word), and transfer direction. Figure 7.6-5 shows the configuration of the ISCS.

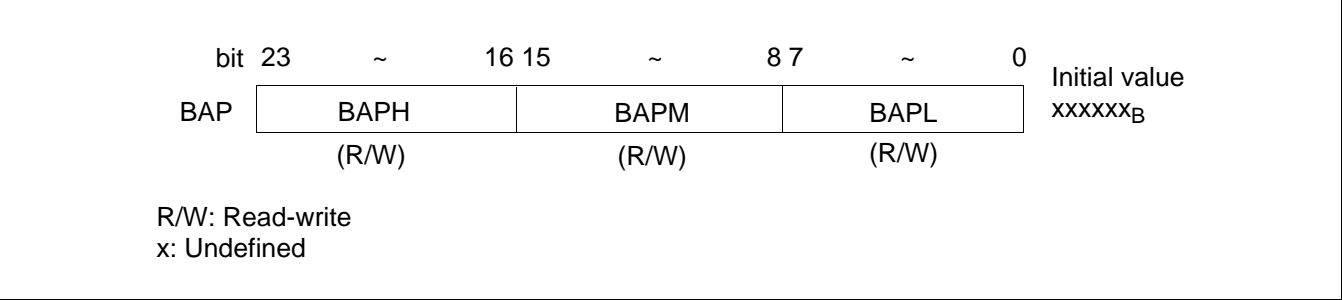
Figure 7.6-5 Configuration of EI<sup>2</sup>OS Status Register (ISCS)



## Buffer Address Pointer (BAP)

The BAP is a 24-bit register that retains the address used by EI<sup>2</sup>OS for the next transfer. Since one independent BAP exists for each EI<sup>2</sup>OS channel, each EI<sup>2</sup>OS channel can transfer data between any address in the 16-megabyte space and the I/O. If the BF bit (BAP update/fixed selection bit in the EI<sup>2</sup>OS status register) in the EI<sup>2</sup>OS status register (ISCS) is set to "update yes", only the lower 16 bits (BAPM, BAPL) of the BAP change; the upper 8 bits (BAPH) do not change. Figure 7.6-6 shows the configuration of the BAP.

**Figure 7.6-6 Configuration of Buffer Address Pointer (BAP)**



References:

- The area that can be specified by the I/O address pointer (IOA) extends from 000000<sub>H</sub> to 00FFFF<sub>H</sub>.
- The area that can be specified with the buffer address pointer (BAP) extends from 000000<sub>H</sub> to FFFFFFF<sub>H</sub>.
- The maximum transfer count that can be specified by the data counter (DCT) is 65,536 (64 Kbytes).

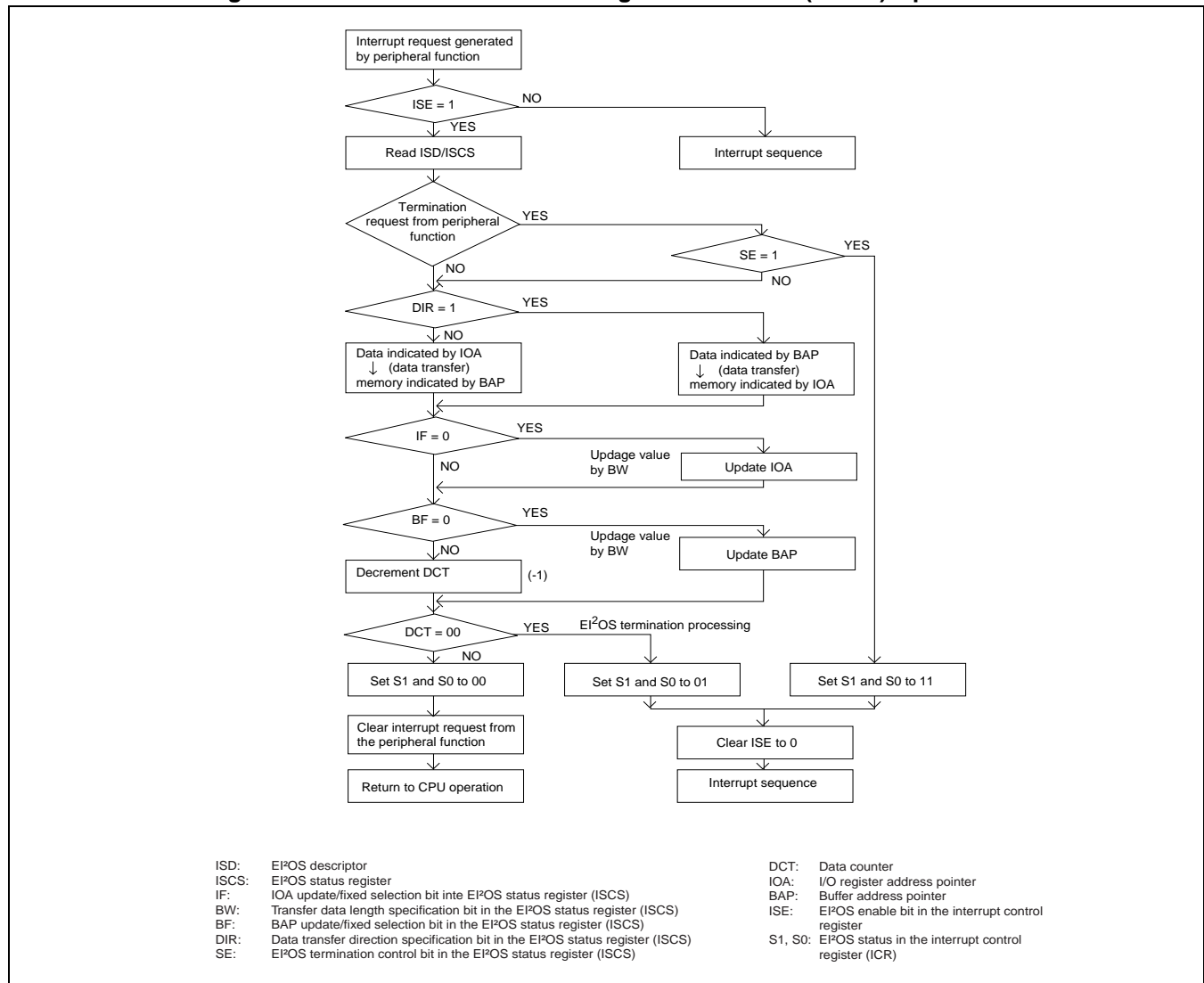
### 7.6.3 Operation of the Extended Intelligent I/O Service (EI<sup>2</sup>OS)

If an interrupt request is generated by a peripheral function, EI<sup>2</sup>OS activation is set in the corresponding interrupt control register (ICR) that the CPU uses EI<sup>2</sup>OS to transfer data. When the specified data transfer count terminates, the hardware interrupt is automatically processed.

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

Figure 7.6-7 shows the flow of EI<sup>2</sup>OS operation based on the internal microcode of the CPU.

Figure 7.6-7 Flow of Extended Intelligent I/O Service (EI<sup>2</sup>OS) Operation



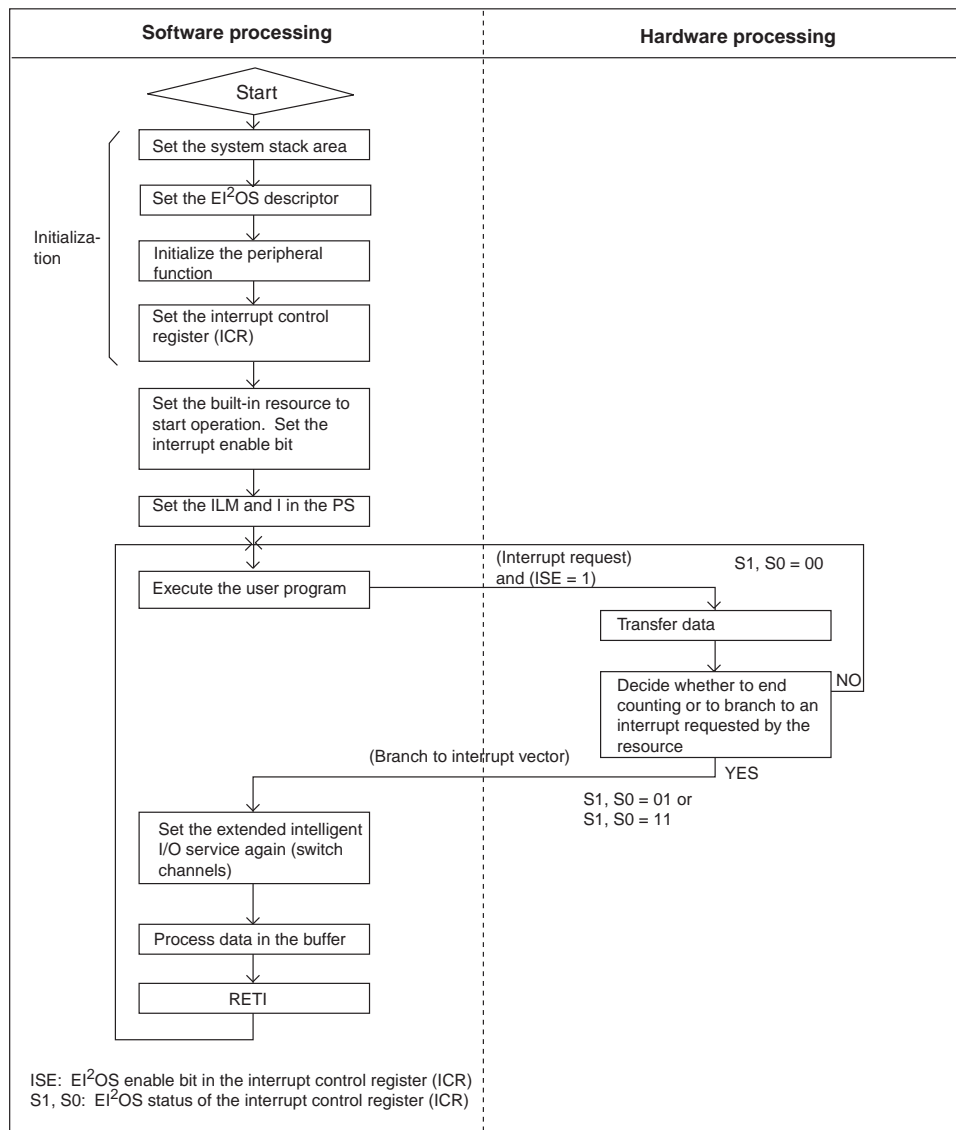
## 7.6.4 Procedure for using the Extended Intelligent I/O Service (EI<sup>2</sup>OS)

Before the extended intelligent I/O service (EI<sup>2</sup>OS) can be used, the system stack area, extended intelligent I/O service (EI<sup>2</sup>OS) descriptor, interrupt function, and interrupt control register (ICR) must be set.

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

Figure 7.6-8 shows the EI<sup>2</sup>OS software and hardware processing.

Figure 7.6-8 Procedure for using the Extended Intelligent I/O Service (EI<sup>2</sup>OS)



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

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

- EI<sup>2</sup>OS status register (ISCS) setting
- Address (area) pointed to by the I/O register address pointer (IOA)
- Address (area) pointed to by the buffer address pointer (BAP)
- External data bus length for external access
- Transfer data length

Because the hardware interrupt is activated when data transfer by EI<sup>2</sup>OS terminates, the interrupt handling time is added.

### ■ Processing Time (one transfer time) of the Extended Intelligent I/O Service (EI<sup>2</sup>OS)

- When data transfer continues

The EI<sup>2</sup>OS processing time for data transfer continuation is shown in Table 7.6-2 based on the EI<sup>2</sup>OS status register (ISCS) setting.

**Table 7.6-2 Extended Intelligent I/O Service Execution Time**

EI <sup>2</sup> OS termination control bit (SE) setting		Terminates due to termination request from the peripheral		Ignores termination request from the peripheral	
IOA update/fixed selection bit (IF) setting		Fixed	Update	Fixed	Update
BAP address update/fixed selection bit (BF) setting	Fixed	32	34	33	35
	Update	34	36	35	37

Unit: Machine cycle (One machine cycle corresponds to one clock cycle of the machine clock,  $\phi$ ).

As shown in Table 7.6-3, interpolation is necessary depending on the EI<sup>2</sup>OS execution condition.

**Table 7.6-3 Data Transfer Interpolation Value for EI<sup>2</sup>OS Execution 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 using the 8-bit word transfer

Even: Even-numbered address word transfer

Odd: Odd-numbered address word transfer

● When the data counter (DCT) count terminates (final data transfer)

Because the hardware interrupt is activated when data transfer by EI<sup>2</sup>OS terminates, the interrupt handling time is added. The EI<sup>2</sup>OS processing time when counting terminates is calculated with the following formula:

$$\text{EI}^2\text{OS processing time when counting terminates} = \text{EI}^2\text{OS processing time when data is transferred} + (21 + 6 \times Z) \text{ Machine cycles}$$

Interrupt handling time

The interrupt handling time is different for each address pointed to by the stack pointer. Table 7.6-4 shows the interpolation value (Z) for the interrupt handling time.

**Table 7.6-4 Interpolation Value (Z) for the Interrupt Handling Time**

Address pointed to by the stack pointer	Interpolation value (Z)
External 8-bit	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

● For termination by a termination request from the peripheral function (I/O)

When data transfer by EI<sup>2</sup>OS is terminated before completion due to a termination request from the peripheral function (I/O) (ICR: S1, S0 = 11<sub>B</sub>), the data transfer is not performed and a hardware interrupt is activated. The EI<sup>2</sup>OS processing time is calculated with the following formula. Z in the formula indicates the interpolation value for the interrupt handling time (Table 7.6-4).

$$\text{EI}^2\text{OS processing time for termination before completion} = 36 + 6 \times Z \text{ Machine cycle}$$

---

Reference:

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

---

## 7.7 Exception Processing Interrupt

---

In the F<sup>2</sup>MC-16LX, the execution of an undefined instruction results in exception processing.

Exception processing is basically the same as an interrupt. When the generation of an exception processing is detected on the instruction boundary, ordinary processing is interrupted and exception processing is executed.

Generally, exception processing occurs as the result of an unexpected operation. Exception processing should be used only to activate recovery software required for debugging or an emergency.

---

### ■ Exception Processing

#### ● Exception processing operation

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 executed.

#### ● The following processing is executed before exception processing branches to the interrupt routine:

- The A, DPR, ADB, DTB, PCB, PC and PS registers are saved to the system stack.
- The I flag of the condition code register (CCR) is cleared to "0", and hardware interrupts are masked.
- The S flag of the condition code register (CCR) is set to "1", and the system stack is activated.

The program counter (PC) value saved to the stack is the exact address where the undefined instruction is stored. For 2-byte or longer instruction codes, the code identified as undefined is stored at this address. When the exception factor type must be determined within the exception processing routine, use this PC value.

#### ● Return from exception processing

When the RETI instruction returns control from exception processing, exception processing restarts because the PC is pointing to the undefined instruction. Provide a solution such as resetting the software.



## 7.8 Stack Operations for Interrupt Processing

Once an interrupt is accepted, the contents of the dedicated registers are automatically saved to the system stack before a branch to interrupt processing. When the interrupt processing terminates, the previous processing is automatically restored from the stack.

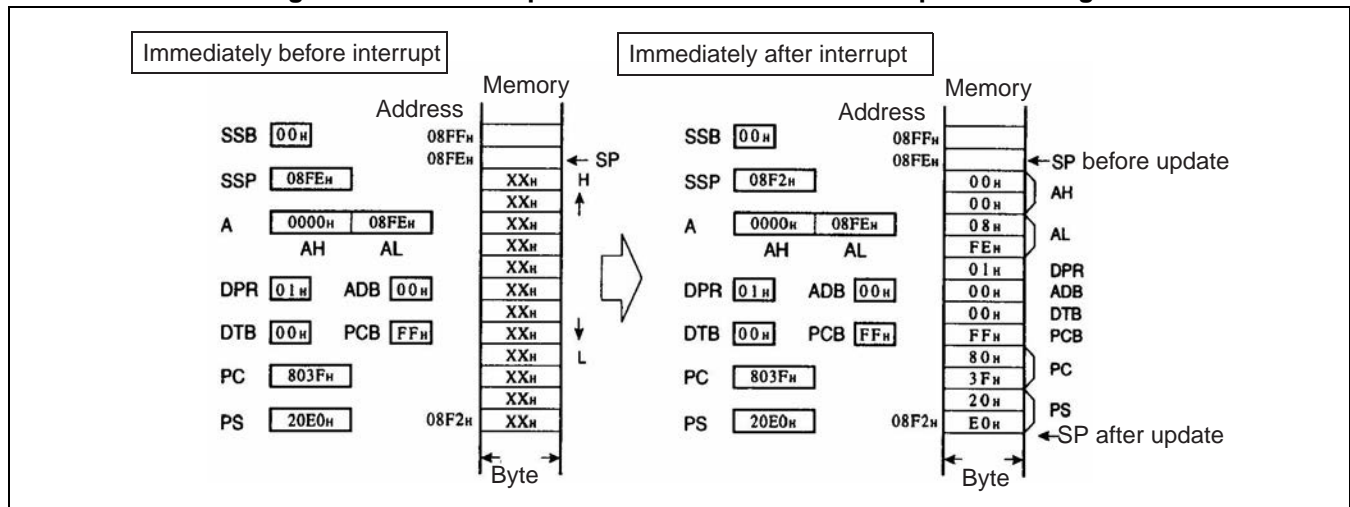
### ■ Stack Operations at the Start of Interrupt Processing

Once an interrupt is accepted, the CPU automatically saves the contents of the current dedicated registers to the system stack in the order given below:

- Accumulator (A)
- Direct page register (DPR)
- Additional data bank register (ADB)
- Data bank register (DTB)
- Program bank register (PCB)
- Program counter (PC)
- Processor status (PS)

Figure 7.8-1 shows the stack operations at the start of interrupt processing.

**Figure 7.8-1 Stack Operations at the Start of Interrupt Processing**



### ■ Stack Operations on Return from Interrupt Processing

When the interrupt return instruction (RETI) is executed at the termination of interrupt processing, the PS, PC, PCB, DTB, ADB, DPR and A values are returned from the stack in reverse order from the order they were placed on the stack. The dedicated registers are restored to the status they had immediately before the start of interrupt processing.

## ■ Stack Area

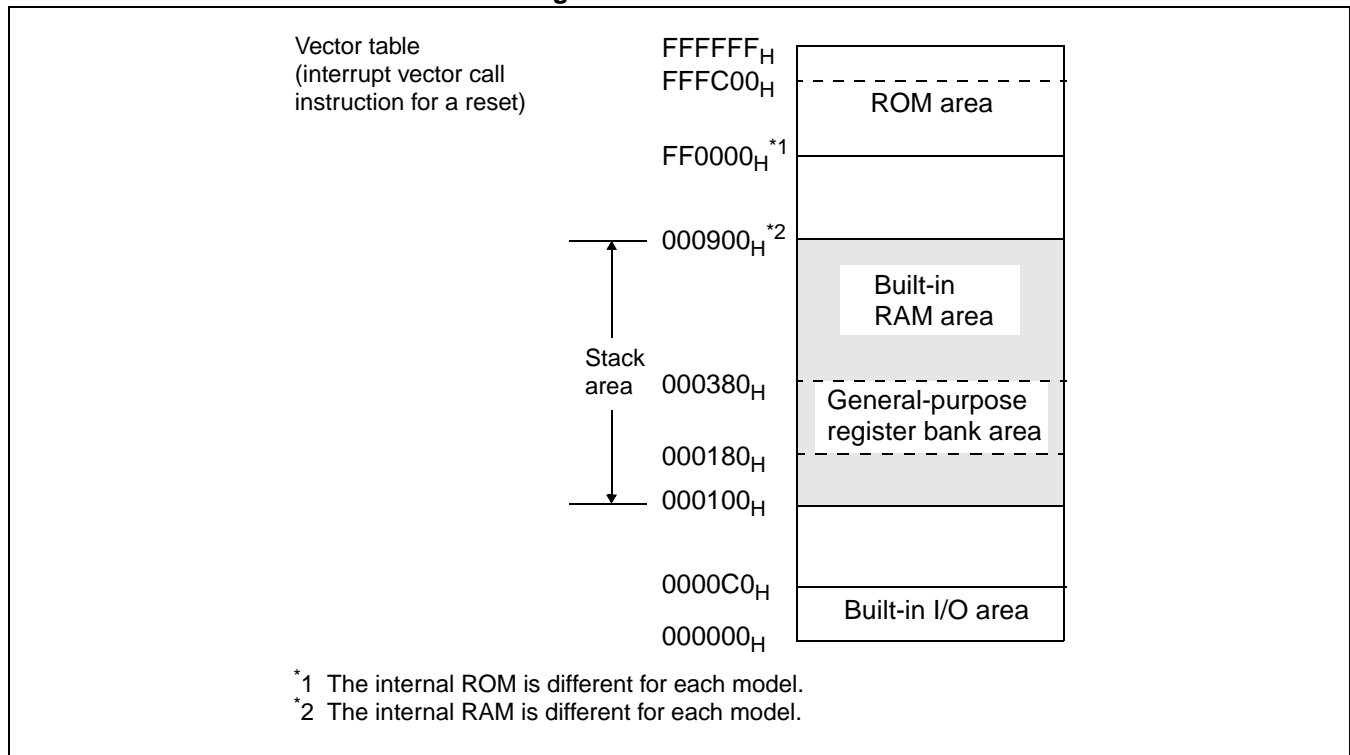
### ● Stack area allocation

The stack area is used for saving and restoring the program counter (PC) when the subroutine call instruction (CALL) and vector call instruction (CALLV) are executed in addition to interrupt processing. The stack area is used for temporary saving and restoring of registers by the PUSHW and POPW instructions.

The stack area is allocated together with the data area in RAM.

Figure 7.8-2 shows the stack area.

**Figure 7.8-2 Stack Area**



Notes:

- Generally set an even-numbered address in the stack pointers (SSP and USP).
- Allocate the system stack area, user stack area, and data area so that they do not overlap.

### ● System stack and user stack

The system stack area is used for interrupt processing. When an interrupt occurs, the user stack area being used is forcibly switched to the system stack. The system stack area must be set correctly even in a system that mainly uses the user stack area.

If division of the stack space is not particularly necessary, use only the system stack.

## 7.9 Sample Programs for Interrupt Processing

---

**This section contains sample programs for interrupt processing.**

---

### ■ Sample Programs for Interrupt Processing

#### ● Processing specifications

The following is a sample program for an interrupt that uses external interrupt 0 (INT0).

#### ● Sample coding

```

DDR1      EQU          000011H          ;Port 1 direction register
ENIR      EQU          030H             ;Interrupt/DTP enable register
EIR       EQU          031H             ;Interrupt/DTP flag
ELVR      EQU          032H             ;Request level setting register
ICR04     EQU          0B4H             ;Interrupt control register
STACK     SSEG
          RW            100             ;Stack
STACK_T   RW            1
STACK     ENDS
;-----Main program -----
CODE      CSEG
START:
          MOV           RP,#0            ;General-purpose registers use the first bank
          MOV           ILM, #07H        ;Sets ILM in PS to level 7
          MOV           A, #!STACK_T     ;Sets system stack
          MOV           SSB, A
          MOVW          A, #STACK_T      ;Sets stack pointer, then
          MOVW          SP, A            ;Sets SSP because S flag = 1
          MOV           DDR1, #00000000B ;Sets P10/INT0 pin to input
          OR            CCR, #40H        ;Sets I flag of CCR in PS, enables interrupts
          MOV           I:ICR04, #00H    ;Sets interrupt level to "0" (highest priority)
          MOV           I:ELVR, #00000001B ;Requests that INT0 be made level H
          MOV           I:EIRR, #00H     ;Clears INT0 interrupt cause
          MOV           I:EIRR, #01H     ;Enables INT0 input
          :
LOOP:     NOP                          ;Dummy loop
          NOP
          NOP

```

```

                                NOP
                                BRA                LOP                ;Unconditional jump
;-----Interrupt program -----
ED_INT1:
                                MOV                I:EIRR, #00H        ;Acceptance of new INT0 not allowed
                                NOP
                                NOP
                                NOP
                                NOP
                                NOP
                                NOP
                                RETI                                ;Return from interrupt
CODE                            ENDS
;-----Vector setting-----
VECT                            CSEG                ABS=0FFH
                                ORG                0FFACH                ;Sets vector for interrupt #20 (14H)
                                DSLED_INT1
                                ORG                0FFDCH                ;Sets reset vector
                                DSL                START
                                DB                00H                ;Sets single-chip mode
VECT                            ENDS
                                END                START

```

## ■ Processing Specifications of Sample Program for Extended Intelligent I/O Service (EI<sup>2</sup>OS)

1. This program detects the H level signal input to the INT0 pin and activates the extended intelligent I/O service (EI<sup>2</sup>OS).
2. When the H level is input to the INT0 pin, EI<sup>2</sup>OS is activated. Data is transferred from port 0 to the memory at the 3000<sub>H</sub> address.
3. The number of transfer data bytes is 100 bytes. After 100 bytes are transferred, an interrupt is generated because EI<sup>2</sup>OS transfer has terminated.

### ● Sample coding

```

DDR1        EQU                000011H        ;Port 1-direction register
ENIR        EQU                000030H        ;Interrupt/DTP enable register
EIRR        EQU                000031H        ;Interrupt/DTP factor register
ELVR        EQU                000032H        ;Request level setting register
ICR04       EQU                0000B4H        ;Interrupt control register
BAPL        EQU                000100H        ;Lower buffer address pointer
BAPM        EQU                000101H        ;Middle buffer address pointer

```

## CHAPTER 7 INTERRUPT

BAPH	EQU	000102H	;Upper buffer address pointer
ISCS	EQU	000103H	;EI <sup>2</sup> OS status
IOAL	EQU	000104H	;Lower I/O address pointer
IOAH	EQU	000105H	;Upper I/O address pointer
DCTL	EQU	000106H	;Low-order data counter
DCTH	EQU	000107H	;High-order data counter
ER0	EQU	EIRR:0	;Definition of external interrupt request flag bit
STACK	SSEG;Stack		
RW100			
STACK_T	RW1		
STAC	KENDS		
-----Main program-----			
CODE	CSEG		
START:			
	AND	CCR, #0BFH	;Clears the I flag of the CCR in the PS and prohibits interrupts
	MOV	RP, #00	;Sets the register bank pointer
	MOV	A, #STACK_T	;Sets the system stack
	MOV	SSB, A	
	MOVW	A, #STACK_T	;Sets the stack pointer, then
	MOVW	SP, A	;Sets SSP because the S flag = 1
	MOV	I:DDR1, #00000000B	; Sets the P10/INT0 pin to input
	MOV	BAPL, #00H	;Sets the buffer address (003000H)
	MOV	BAPM, #30H	
	MOV	BAPH, #00H	
	MOV	ISCS, #00010001B	;No I/O address update, byte transfer, buffer address updated I/O -> buffer transfer, terminated by the peripheral function
	MOV	IOAL, #00H	;Sets the transfer source address (port 0: 000000H)
	MOV	IOAH, #00H	
	MOV	DCTL, #64H	;Sets the number of transfer bytes (100 bytes)
	MOV	DCTH, #00H	
	MOV	I:ICR04, #00001000B	;EI <sup>2</sup> OS channel 0, EI <sup>2</sup> OS enable, interrupt level 0 (highest priority)
	MOV	I:ELVR, #00000001B	;Requests that INT0 be made H level
	MOV	I:EIRR, #00H	;Clears the INT0 interrupt cause

```

        MOV        I:ENIR, #01H        ;Enables INT0 interrupts
        MOV        ILM, #07H           ;Sets the ILM in the PS to level 7
        OR         CCR, #40H           ;Sets the I flag of the CCR in the PS
                                        and enables interrupts
:
LOOP    BRA        LOOP                ;Infinite loop
;-----Interrupt program-----
WARI    CLRB        ER0                ;Clears interrupt/DTP request flag
        :
        User processing                ;Checks EI2OS termination factor,
        :                             ;processes data in buffer, sets EI2OS
                                        again
        RETI
CODE    ENDS
;-----Vector processing-----
VECT    CSEG        ABS=0FFH
        ORG        0FFACH              ; Sets vector for interrupt #20 (14H)
        DSL        WARI
        ORG        0FFDCH              ;Sets reset vector
        DSL        START
        DB         00H                ;Sets single-chip mode
VECT    ENDS
        END        START

```



# ***CHAPTER 8***

---

# ***MODE SETTING***

**This chapter describes the operating modes and memory access modes supported by the MB90460/465 series.**

- 8.1 Mode Setting
- 8.2 Mode Pins (MD2 to MD0)
- 8.3 Mode Data



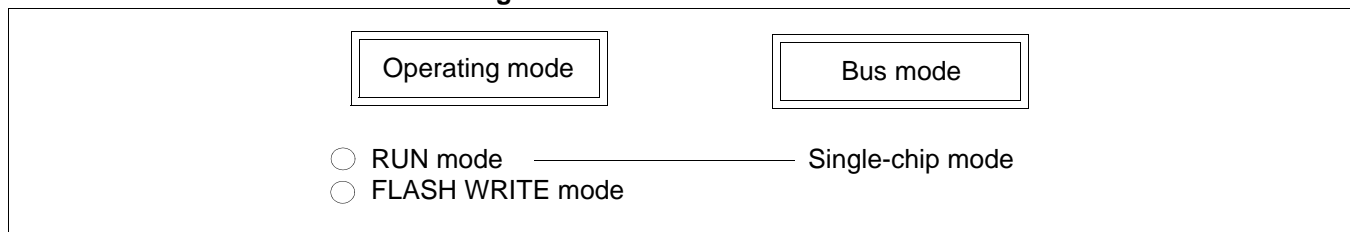
## 8.1 Mode Setting

The F<sup>2</sup>MC-16LX supports the modes for access method, and access areas. A mode is determined based on the settings by the mode pin at a reset as well as the mode data fetched.

### ■ Mode Setting

The F<sup>2</sup>MC-16LX supports the modes for access method, and access areas, classified as shown in Figure 8.1-1 in this module.

Figure 8.1-1 Mode Classification



### ■ Operating Modes

The operating modes control the operating state of the device, and are specified by the mode setting pin (MDx) and Mx bit contents in mode data.

Note:

Because the MB90460/465 series is only used in single-chip mode, set MD2, MD1, MD0 to 011<sub>B</sub> and set M1, M0 to 00<sub>B</sub>.

### ■ Bus Mode

The bus mode controls the operation of internal ROM and external access functions, and is specified by the mode setting pin (MDx) and Mx bit contents in mode data. The mode setting pin (MDx) specifies bus mode when reset vector and mode data are read. The Mx bit in mode data specifies bus mode during normal operation.

### ■ RUN Mode

The RUN mode means CPU operating mode. The RUN mode includes main clock mode, PLL clock mode, and various low power consumption modes. See "CHAPTER 6 LOW POWER CONSUMPTION MODE", for details.

Note:

Because the MB90460/465 series is only used in single-chip mode, set MD2, MD1, MD0 to 011<sub>B</sub> and set M1, M0 to 00<sub>B</sub>.

## 8.2 Mode Pins (MD2 to MD0)

Three external pins, MD2 to MD0, are supported as the mode pins. These are used to specify how the reset vector and mode data are fetched.

### ■ Mode Pins (MD2 to MD0)

The mode pins are used to select the data bus (external or internal) used for reading the reset vector and to specify the bus width when the external data bus is selected.

For a built-in FLASH version, the mode pins are also used to specify FLASH programming mode, which is used to write programs and other data to internal FLASH.

Table 8.2-1 shows the mode pin settings.

**Table 8.2-1 Mode Pin Settings**

MD2	MD1	MD0	Mode name	Reset vector access area	External data bus width	Remarks
0	0	0	Setting not allowed			
0	0	1				
0	1	0				
0	1	1	Internal vector mode	Internal	Mode data	The reset sequence and subsequent sequences are controlled by mode data.
1	0	0	Setting not allowed			
1	0	1				
1	1	0	FLASH serial write mode	-	-	-
1	1	1	FLASH memory mode	-	-	Mode when the parallel writer is used.

MD2 to MD0: Connect the pins to Vss for 0 and to Vcc for 1.

\*: The flash serial write mode cannot be executed by just setting the mode pins. Other terminal also need to be set. For details, see "CHAPTER 24 EXAMPLE OF F<sup>2</sup>MC-16LX MB90F462/F462A/F463A CONNECTION FOR SERIAL WRITING".

Note:

Because the MB90460/465 series is only used in single-chip mode, set MD2, MD1, MD0 to 011<sub>B</sub> and set M1, M0 to 00<sub>B</sub>.

### 8.3 Mode Data

The mode data is at memory location FFFDF<sub>H</sub>, and is used to specify the operation after a reset sequence. The mode data is automatically fetched to the CPU.

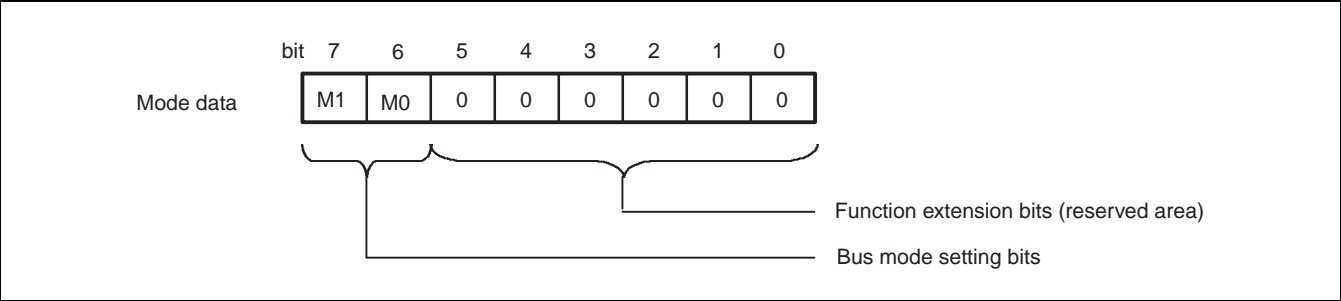
■ Mode Data

During a reset sequence, the mode data at address FFFDF<sub>H</sub> is fetched to the mode register in the CPU. The CPU uses the mode data to set the memory access mode.

The contents of the mode register can only be changed during the reset sequence. The settings in the register take effect after the reset sequence.

Figure 8.3-1 shows the mode data configuration.

Figure 8.3-1 Mode Data Configuration



■ Bus Mode Setting Bits

The bus mode setting bits specify operating mode after a reset sequence. Table 8.3-1 lists the relationship between the bits and functions.

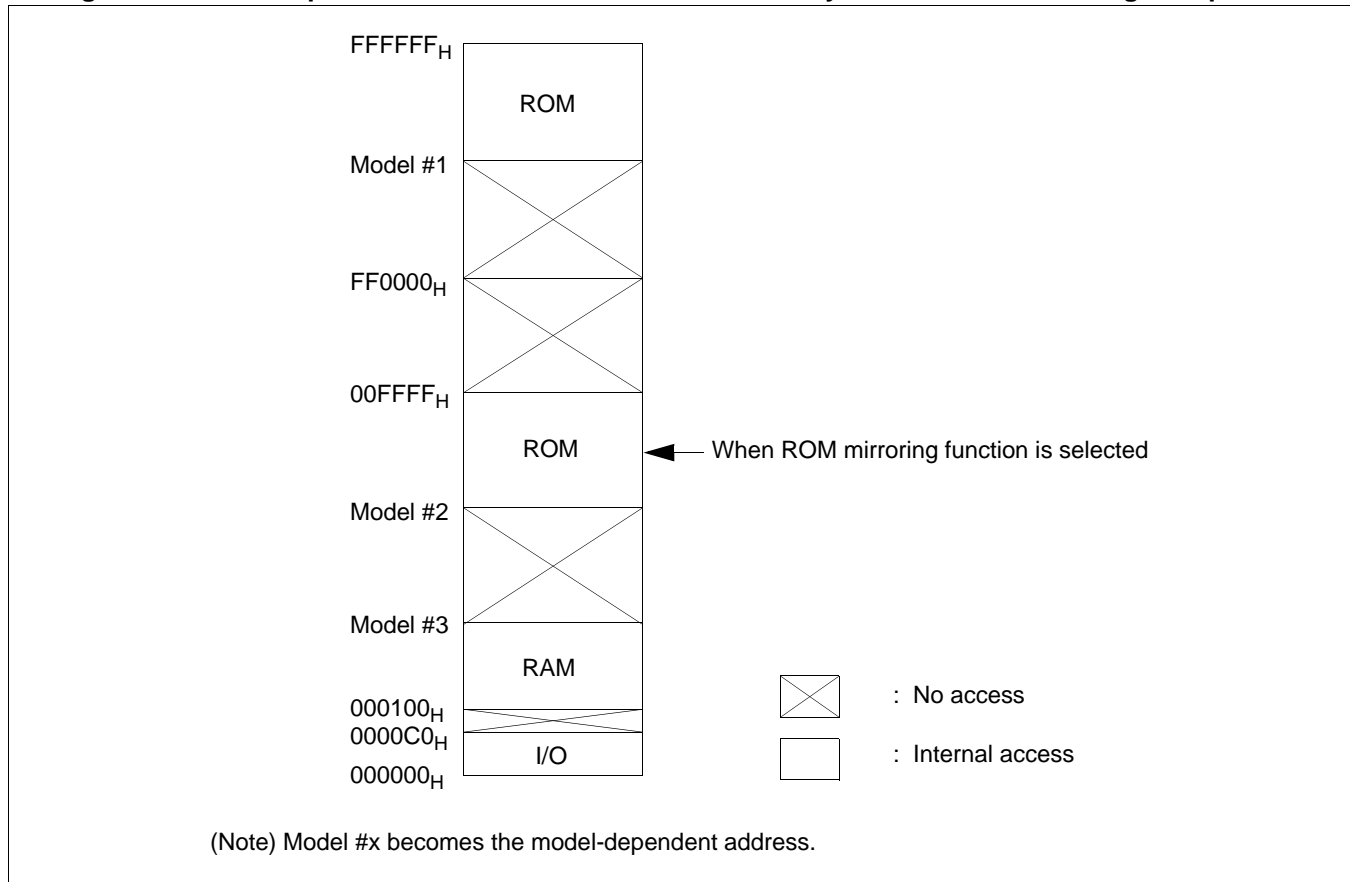
Table 8.3-1 Bus Mode Setting Bits and Functions

M1	M0	Function
0	0	Single-chip mode
0	1	(Setting not allowed)
1	0	
1	1	

Note: Because the MB90460/465 series is only used in single-chip mode, set MD2, MD1, MD0 to 011<sub>B</sub> and set M1, M0 to 00<sub>B</sub>.

Figure 8.3-2 shows the correspondence between access areas and physical addresses in single-chip mode.

**Figure 8.3-2 Correspondence between Access Areas and Physical Addresses in Single-chip Mode**



## ■ Relationship between Mode Pins and Mode Data

Table 8.3-2 lists the relationship between mode pins and mode data.

**Table 8.3-2 Relationship between Mode Pins and Mode Data**

Mode	MD2	MD1	MD0	M1	M0
Single-chip mode	0	1	1	0	0

Note:

The MB90460/465 series is only used in single-chip mode.



# **CHAPTER 9**

---

## ***I/O PORT***

**This chapter describes the functions and operation of the I/O port.**

- 9.1 Overview of I/O Port
- 9.2 Registers of I/O Port
- 9.3 Port 0
- 9.4 Port 1
- 9.5 Port 2
- 9.6 Port 3
- 9.7 Port 4
- 9.8 Port 5
- 9.9 Port 6
- 9.10 Sample I/O Port Program

## 9.1 Overview of I/O Port

An I/O port can be used as a general-purpose I/O port (parallel I/O port). The MB90460/465 series has 7 ports (51 lines). The ports are also used for resource I/O pins (peripheral function I/O pins).

### ■ I/O Port Functions

Each I/O port outputs data from the CPU to the I/O pins or inputs signals from the I/O pins to the CPU as directed by the port data register (PDR). Each I/O port can also designate the direction of a data flow (input or output) at the I/O pins in bit units using the port data direction register (DDR). The function of each port and the resources using it are described below:

- Port 0: General-purpose I/O port/resource (Waveform sequencer\*/PWC0\*)
- Port1: General-purpose I/O port/resource (External interrupt/Waveform sequencer\*/Waveform generator/16-bit reload timer 0)
- Port 2: General-purpose I/O port/resource (16-bit reload timer/PWC1/Input capture)
- Port 3: General-purpose I/O port/resource (Waveform generator/PPG1\*/PPG0)
- Port 4: General-purpose I/O port/resource (UART0/Waveform sequencer\*/PPG2)
- Port 5: General-purpose I/O port/resource (A/D converter)
- Port6: General-purpose I/O port/resource (UART1/External interrupt)

Table 9.1-1 summarizes the functions of individual port.

**Table 9.1-1 Functions of Individual Port**

Port	Pin	Input form	Output form	Function	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Port 0	P00/OPT0 to P07/PW00	CMOS	CMOS pull-up resistor selectable	General I/O port	—	—	—	—	—	—	—	—	P07	P06	P05	P04	P03	P02	P01	P00
				Resource	—	—	—	—	—	—	—	—	PW00*	PW10*	OPT5*	OPT4*	OPT3*	OPT2*	OPT1*	OPT0*
Port 1	P10/INT0/DTT0 to P17/FRCK	CMOS (hysteresis)		General I/O port	P17	P16	P15	P14	P13	P12	P11	P10	—	—	—	—	—	—	—	—
				Resource	FRCK	INT6/TO0	INT5/TIN0	INT4	INT3	INT2/DTT1*	INT1	INT0/DTT0	—	—	—	—	—	—	—	—
Port 2	P20/TIN1 to P27/IN3			General I/O port	—	—	—	—	—	—	—	—	P27	P26	P25	P24	P23	P22	P21	P20
				Resource	—	—	—	—	—	—	—	—	IN3	IN2	IN1	IN0	PW01	PW11	TO1	TIN1
Port 3	P30/RT00 to P37/PPG0	CMOS	CMOS	General I/O port	P37	P36	P35	P34	P33	P32	P31	P30	—	—	—	—	—	—	—	—
				Resource	PPG0	PPG1*	RT05	RT04	RT03	RT02	RT01	RT00	—	—	—	—	—	—	—	—
Port 4	P40/SIN to P46/PPG2	CMOS (hysteresis)		General I/O port	—	—	—	—	—	—	—	—	—	P46	P45	P44	P43	P42	P41	P40
				Resource	—	—	—	—	—	—	—	—	—	PPG2	SN12*	SN11*	SN10*	SCK0	SOT0	SIN0
Port 5	P50/AN0 to P57/AN7	Analog/CMOS	CMOS	General I/O port	P57	P56	P55	P54	P53	P52	P51	P50	—	—	—	—	—	—	—	—
				Analog input	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	—	—	—	—	—	—	—	—
Port 6	P60/SIN1 to P63/INT7	CMOS (hysteresis)	CMOS	General I/O port	—	—	—	—	—	—	—	—	—	—	—	—	P63	P62	P61	P60
				Resource	—	—	—	—	—	—	—	—	—	—	—	—	INT7	SCK1	SOT1	SIN1

\*: Pin names not applicable to MB90465 series

---

**Note:**

Port 5 is also used as analog input pins. To use the port as a general-purpose port, be sure to reset the corresponding bit of the analog data input enable register (ADER) to "0". Resetting the CPU sets the ADER register bits to "1".

---



## 9.2 Registers of I/O Port

This section provides a list of the registers related to the I/O port settings.

### ■ Registers for I/O Ports

Table 9.2-1 is a list of the registers corresponding to individual port.

**Table 9.2-1 Registers and Corresponding port**

Register	Read/Write	Address	Initial value
Port 0 data register (PDR0)	R/W	000000 <sub>H</sub>	XXXXXXXX <sub>B</sub>
Port 1 data register (PDR1)	R/W	000001 <sub>H</sub>	XXXXXXXX <sub>B</sub>
Port 2 data register (PDR2)	R/W	000002 <sub>H</sub>	XXXXXXXX <sub>B</sub>
Port 3 data register (PDR3)	R/W	000003 <sub>H</sub>	XXXXXXXX <sub>B</sub>
Port 4 data register (PDR4)	R/W	000004 <sub>H</sub>	-XXXXXXXX <sub>B</sub>
Port 5 data register (PDR5)	R/W	000005 <sub>H</sub>	XXXXXXXX <sub>B</sub>
Port 6 data register (PDR6)	R/W	000006 <sub>H</sub>	----XXXX <sub>B</sub>
Port 0 data direction register (DDR0)	R/W	000010 <sub>H</sub>	00000000 <sub>B</sub>
Port 1 data direction register (DDR1)	R/W	000011 <sub>H</sub>	00000000 <sub>B</sub>
Port 2 data direction register (DDR2)	R/W	000012 <sub>H</sub>	00000000 <sub>B</sub>
Port 3 data direction register (DDR3)	R/W	000013 <sub>H</sub>	00000000 <sub>B</sub>
Port 4 data direction register (DDR4)	R/W	000014 <sub>H</sub>	-0000000 <sub>B</sub>
Port 5 data direction register (DDR5)	R/W	000015 <sub>H</sub>	00000000 <sub>B</sub>
Port 6 data direction register (DDR6)	R/W	000016 <sub>H</sub>	----0000 <sub>B</sub>
Analog data input enable register (ADER)	R/W	000017 <sub>H</sub>	11111111 <sub>B</sub>
Port 0 pull-up resistor setting register (RDR0)	R/W	00001C <sub>H</sub>	00000000 <sub>B</sub>
Port 1 pull-up resistor setting register (RDR1)	R/W	00001D <sub>H</sub>	00000000 <sub>B</sub>

R/W: Read/write enabled

R: Read-only

X: Undefined

–: Not used

Note:

For port (other than Port 0,1,2 and 3) that is multiplexed with resource, use Read-modify-write instruction (such as an instruction that sets bits) may accidentally write unexpected value to the DDR and PDR register when resource is enabled.

## 9.3 Port 0

**Port 0 is a general-purpose I/O port. It can also be used for resource I/O. The port pins can be switched in units of bits between the I/O port and resource. This section focuses on the general I/O port function. This section also provides the configuration of port 0, lists of pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 0 Configuration

Port 0 consists of the following:

- General-purpose I/O pins/waveform sequencer output/PWC0 input, output (P00/OPT0 to P07/PWO0) (waveform sequencer output and PWC0 not present in MB90465 series)
- Port 0 data register (PDR0)
- Port 0 data direction register (DDR0)
- Port 0 pull-up resistor setting register (RDR0)

### ■ Port 0 Pins

The port 0 I/O pins are also used as resource I/O pins. Therefore, the pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins. Table 9.3-1 lists the port 0 pins.

**Table 9.3-1 Port 0 Pins**

Port	Pin	Port function (single-chip mode)		Resource function		I/O form		Circuit type
						Input	Output	
Port 0	P00/OPT0*	P00	General-purpose I/O	OPT0*	Waveform sequencer output	CMOS	CMOS	D
	P01/OPT1*	P01		OPT1*	Waveform sequencer output			
	P02/OPT2*	P02		OPT2*	Waveform sequencer output			
	P03/OPT3*	P03		OPT3*	Waveform sequencer output			
	P04/OPT4*	P04		OPT4*	Waveform sequencer output			
	P05/OPT5*	P05		OPT5*	Waveform sequencer output			
	P06/PWI0*	P06		PWI0*	PWC0 input			E
	P07/PWO0*	P07		PWO0*	PWC0 output			

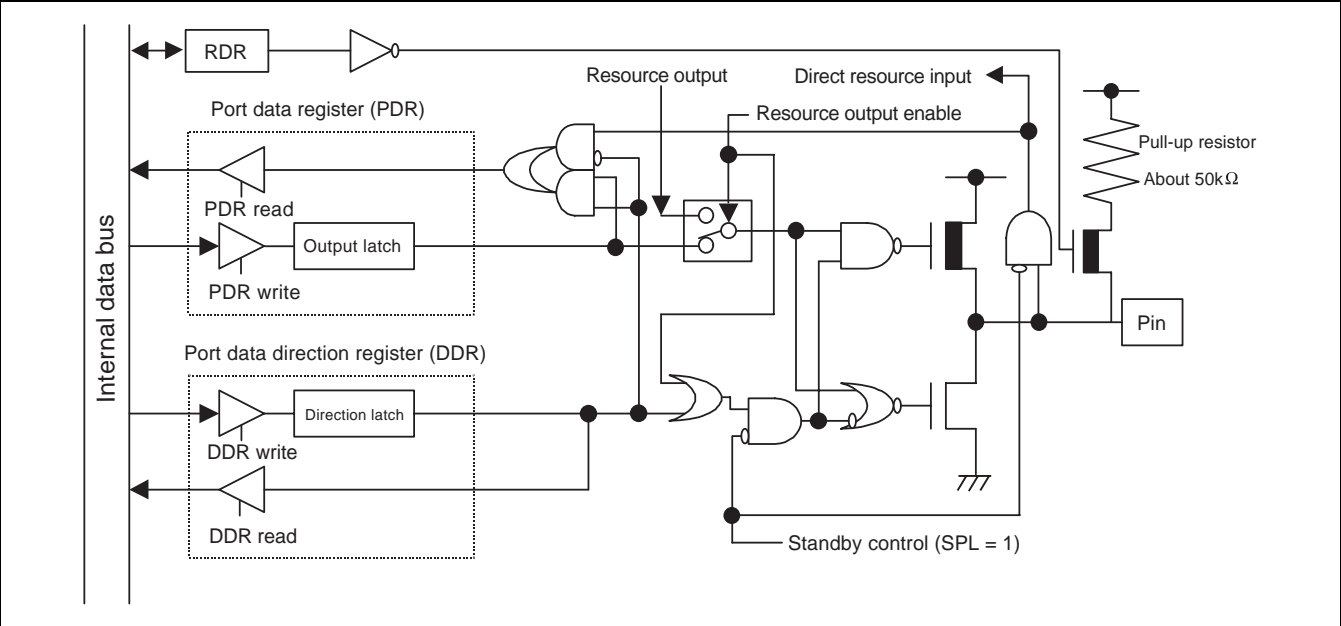
\*: Pin names not applicable to MB90465 series

See "1.7 I/O Circuit Types", for information on the circuit types.

■ Block Diagram of Port 0 Pins

Figure 9.3-1 is a block diagram of the port 0 pins.

Figure 9.3-1 Block Diagram of port 0 Pins



When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the DDR0 register.

■ Port 0 Registers

Port 0 registers are PDR0, DDR0 and RDR0. The bits making up each register correspond to the port 0 pins on a one-to-one basis. Table 9.3-2 lists the port 0 pins and their corresponding register bits.

Table 9.3-2 Port 0 Pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 0	PDR0, DDR0, RDR0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	P07	P06	P05	P04	P03	P02	P01	P00

See "1.7 I/O Circuit Types", for information on the circuit types.

### 9.3.1 Port 0 Registers (PDR0, DDR0 and RDR0)

---

**This section describes the port 0 registers.**

---

#### ■ Functions of Port 0 Registers

- Port 0 data register (PDR0)

The PDR0 register indicates the state of each pin of port 0.

- Port 0 data direction register (DDR0)

The DDR0 register specifies the direction of a data flow (input or output) at each pin (bit) of port 0. When a DDR0 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

- Port 0 pull-up resistor setting register (RDR0)

The RDR0 register specifies the selection of a pull-up resistor at each pin (bit) of port 0. When a RDR0 register bit is "1", a pull-up resistor is selected for the corresponding port (pin). When the bit is "0", the pull-up resistor is deselected.

---

**Notes:**

- When a resource having output pins is used, the port functions as resource output pins regardless of the value in the DDR0 register as long as the resource output enable bit corresponding to the pins is set.
  - To use a resource having input pins, reset the port direction register bit corresponding to each resource input pin to "0" to place the port in input mode.
-

Table 9.3-3 lists the functions of the port 0 registers.

**Table 9.3-3 Port 0 Register Functions**

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 0 data register (PDR0)	0	The pin is at the low level.	The output latch is loaded with 0. When the pin functions as an output port, the pin is set to the low level.	R/W	000000 <sub>H</sub>	XXXXXXXX <sub>B</sub>
	1	The pin is at the high level.	The output latch is loaded with 1. When the pin functions as an output port, the pin is set to the high level.			
Port 0 data direction register (DDR0)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000010 <sub>H</sub>	00000000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			
Port 0 pull-up resistor setting register (RDR0)	0	The setting latch is "0".	The pull-up resistor is cut and the port is placed in the Hi-Z state in input mode.	R/W	00001C <sub>H</sub>	00000000 <sub>B</sub>
	1	The setting latch is "1".	The pull-up resistor is selected and the port is held at the high level in input mode.			

R/W: Read/write enabled

X : Undefined

## 9.3.2 Operation of Port 0

---

**This section describes the operation of port 0.**

---

### ■ Operation of Port 0

#### ● Port operation in output mode

- Setting a bit of the DDR0 register to "1" places the corresponding port pin in output mode.
  - Data written to the PDR0 register in output mode is held in the output latch of the PDR and output to the port pins as is.
  - The PDR0 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).
- 

#### Note:

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR0 register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- However, when the RDR0 register is set to "1" to select a pull-up resistor, the pins are held at the high level.
- Data written to the PDR0 register in input mode is held in the output latch of the PDR but is not output to the port pins.
- The PDR0 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for resource output

- The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR0 register bit is "0", the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

#### ● Port operation for resource input

- When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, reset the DDR0 register to "0" to place the port in input mode.

● Port operation after a reset

- When the CPU is reset, the DDR0 and RDR registers are initialized to "0". As a result, the output buffer is turned off (I/O mode changes to input), the pull-up resistor is cut, and the pins are placed in a high impedance state.
- The PDR0 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR0 register after the output data is set in the PDR0 register.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR0 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Note also that when a pull-up resistor is selected, the port pins are held at the high level and not placed in a high-impedance state even when the SPL bit is set to "1". Table 9.3-4 lists the states of the port 0 pins.

**Table 9.3-4 States of Port 0 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 1)
P00/OPT0 to P07/PWO0	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-Z	Input shut down/held at H level

SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z: High impedance

## 9.4 Port 1

**Port 1 is a general-purpose I/O port. It can also be used for resource I/O. The port pins can be switched in units of bits between the I/O port and resource. This section focuses on the general I/O port function. The section provides the configuration of port 1, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 1 Configuration

Port 1 consists of the following:

- General-purpose I/O pins/external interrupt input pins (P10/INT0/DTT0 to P17/FRCK)
- Port 1 data register (PDR1)
- Port 1 data direction register (DDR1)
- Port 1 pull-up resistor setting register (RDR1)

### ■ Port 1 Pins

The port 1 pins are also used as resource input pins. The pins cannot be used as output port pins when they are used as resource input pins. Table 9.4-1 lists the port 1 pins.

**Table 9.4-1 Port 1 Pins**

Port	Pin	Port function		Resource function		I/O form		Circuit type
						Input	Output	
Port 1	P10/INT0/ DTTI0	P10	General- purpose I/O	INT0/ DTTI0	External interrupt input/waveform generator input	CMOS (hysteresis)	CMOS	C
	P11/INT1	P11		INT1	External interrupt input			
	P12/INT2/ DTTI1*	P12		INT2/ DTTI1*	External interrupt input/waveform sequencer input			
	P13/INT3	P13		INT3	External interrupt input			
	P14/INT4	P14		INT4				
	P15/INT5/ TIN0	P15		INT5/ TIN0	External interrupt input/reload timer input			
	P16/INT6/ TO0	P16		INT6/ TO0	External interrupt input/reload timer output			
	P17/FRCK	P17		FRCK	Free-run timer clock input			

\*: Pin names not applicable to MB90465 series

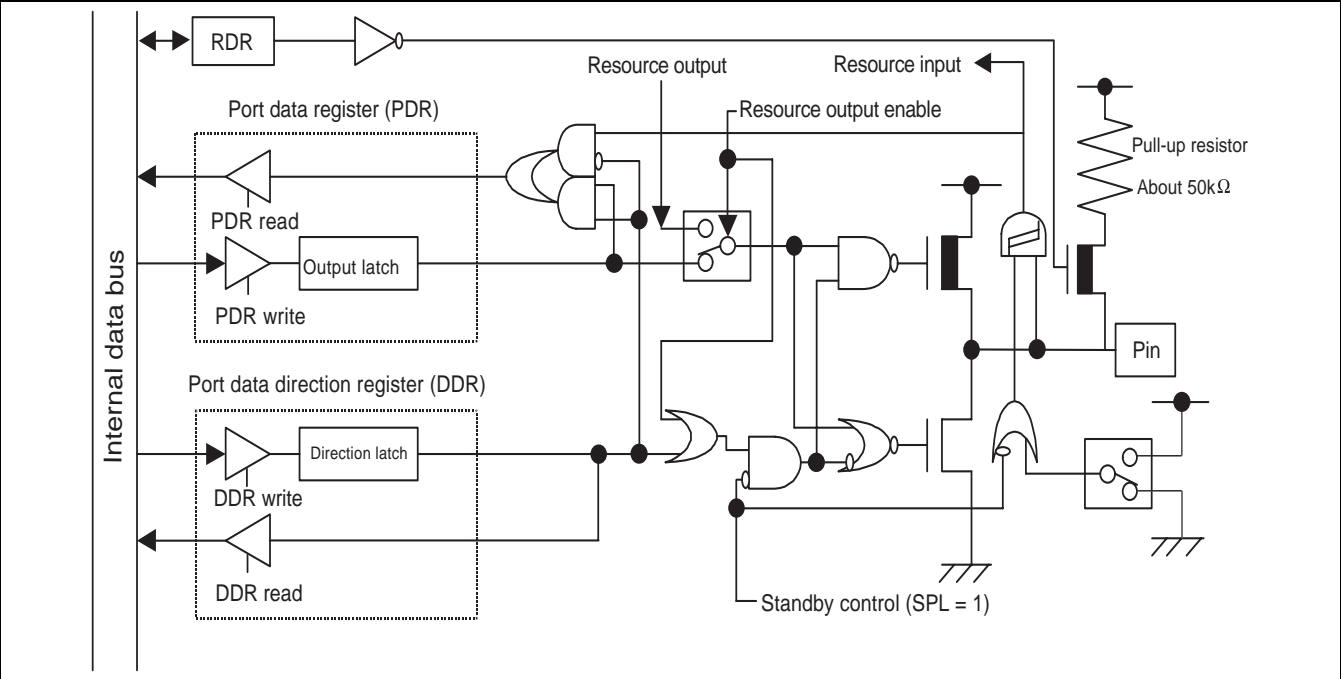
See "1.7 I/O Circuit Types", for information on the circuit types.



■ Block Diagram of Port 1 Pins

Figure 9.4-1 is a block diagram of port 1 pins.

Figure 9.4-1 Block Diagram of Port 1 Pins



■ Port 1 Registers

Port 1 registers are PDR1, DDR1 and RDR1. The bits making up each register correspond to the port 1 pins on a one-to-one basis. Table 9.4-2 lists the port 1 pins and their corresponding register bits.

Table 9.4-2 Port 1 Pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 1	PDR1, DDR1, RDR1	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	Corresponding pin	P17	P16	P15	P14	P13	P12	P11	P10

## 9.4.1 Port 1 Registers (PDR1, DDR1 and RDR1)

This section describes the port 1 registers.

### ■ Functions of Port 1 Registers

#### ● Port 1 data register (PDR1)

The PDR1 register indicates the state of each pin of port 1.

#### ● Port 1 data direction register (DDR1)

The DDR1 register specifies the direction of a data flow (input or output) at each pin (bit) of port 1. When a DDR1 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

#### ● Port 1 pull-up resistor setting register (RDR1)

The RDR1 register specifies the selection of a pull-up resistor at each pin (bit) of port 1. When a RDR1 register bit is "1", a pull-up resistor is selected for the corresponding port (pin). When the bit is "0", the pull-up resistor is deselected.

Notes:

- When a resource having output pins is used, the port functions as resource output pins regardless of the value in the DDR1 register as long as the resource output enable bit corresponding to the pins is set.
- To use a resource having input pins, reset the port direction register bit corresponding to each resource input pin to "0" to place the port in input mode.

Table 9.4-3 lists the functions of the port 1 registers.

**Table 9.4-3 Port 1 Register Functions**

Register	Data	During reading	During writing	Read/Write	Address	Initial value
Port 1 data register (PDR1)	0	The pin is at the low level.	The output latch is loaded with 0. When the pin functions as an output port, the pin is set to the low level.	R/W	000001 <sub>H</sub>	XXXXXXXX <sub>B</sub>
	1	The pin is at the high level.	The output latch is loaded with 1. When the pin functions as an output port, the pin is set to the high level.			
Port 1 data direction register (DDR1)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000011 <sub>H</sub>	00000000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			
Port 1 pull-up resistor setting register (RDR1)	0	The setting latch is "0".	The pull-up resistor is cut and the port is placed in the Hi-Z state in input mode.	R/W	00001D <sub>H</sub>	00000000 <sub>B</sub>
	1	The setting latch is "1".	The pull-up resistor is selected and the port is held at the high level in input mode.			

R/W: Read/write enabled

X : Undefined

## 9.4.2 Operation of Port 1

---

**This section describes the operation of port 1.**

---

### ■ Operation of Port 1

#### ● Port operation in output mode

- Setting a bit of the DDR1 register to "1" places the corresponding port pin in output mode.
- Data written to the PDR1 register in output mode is held in the output latch of the PDR and output to the port pins.
- The PDR1 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

---

**Note:**

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR1 register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- However, when the RDR1 register is set to "1" to select a pull-up resistor, the pins are held at the high level.
- Data written to the PDR1 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR1 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for resource output

- The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR1 register bit is "0", the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

#### ● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, reset the DDR1 register to "0" to place the port in input mode.

● Port operation after a reset

- When the CPU is reset, the DDR1 and RDR registers are initialized to "0". As a result, the output buffer is turned off (I/O mode changes to input), the pull-up resistor is cut, and the pins are placed in a high impedance state.
- The PDR1 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR1 register after the output data is set in the PDR1 register.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR1 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit. Note also that when a pull-up resistor is selected, the port pins are held at the high level and not placed in a high-impedance state even when the SPL bit is set to "1". Table 9.4-4 lists the states of the port 1 pins.

**Table 9.4-4 States of Port 1 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 1)
P10/INT0/ DTTI0 to P17/ FRCK	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input enabled*/output in Hi-Z	Input shut down/held at H level

SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z : High impedance

\* : Only when P10/INT0 to P16/INT6 is configured as external interrupt pins, otherwise input shutdown

## 9.5 Port 2

**Port 2 is a general-purpose I/O port. It can also be used for resource I/O. The port pins can be switched in units of bits between the I/O port and the resource. This section focuses on the general I/O port function. This section also provides the configuration of port 2, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 2 Configuration

Port 2 consists of the following:

- General-purpose I/O pins/resource I/O pins (P20/TIN1 to P27/IN3)
- Port 2 data register (PDR2)
- Port 2 data direction register (DDR2)

### ■ Port 2 Pins

The port 2 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins. Table 9.5-1 lists the port 2 pins.

**Table 9.5-1 Port 2 Pins**

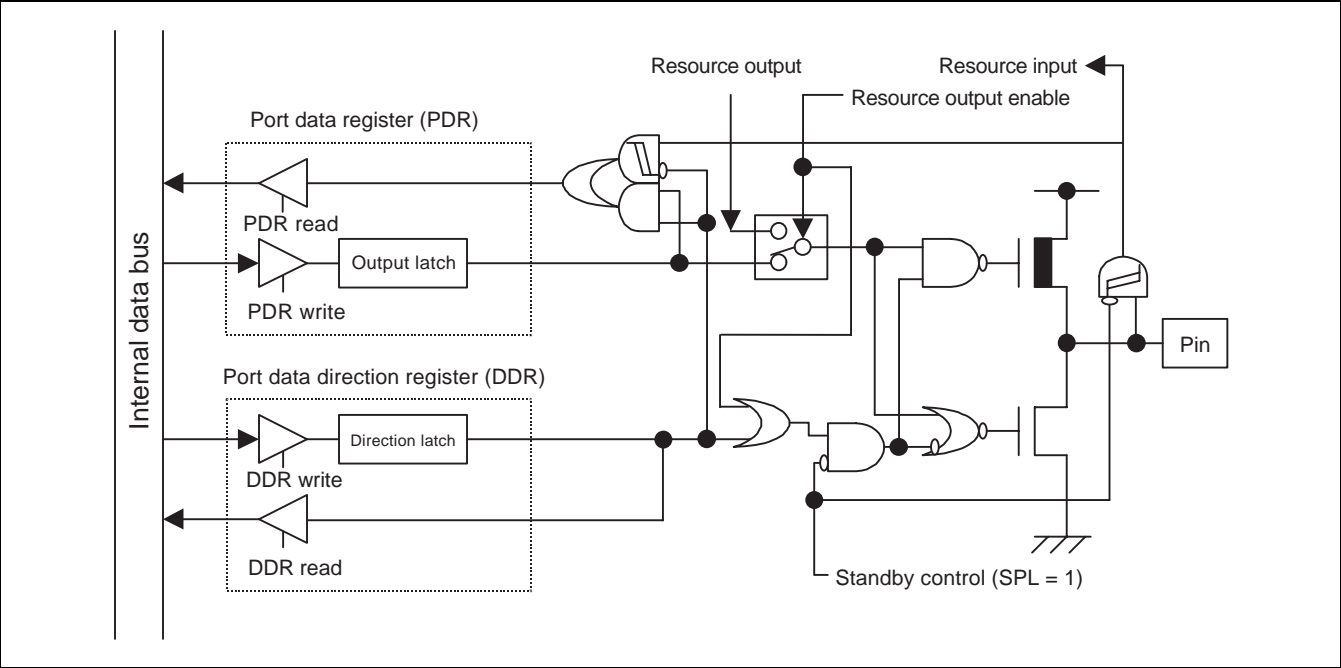
Port	Pin	Port function	Resource function	I/O form		Circuit type
				Input	Output	
Port 2	P20/TIN1	P20	TIN1	CMOS (hysteresis)	CMOS	F
	P21/TO1	P21	TO1			
	P22/PWI1	P22	PWI1			
	P23/PWO1	P23	PWO1			
	P24/IN0	P24	IN0			
	P25/IN1	P25	IN1			
	P26/IN2	P26	IN2			
	P27/IN3	P27	IN3			

See "1.7 I/O Circuit Types", for information on the circuit types.

■ Block Diagram of Port 2 Pins

Figure 9.5-1 is a block diagram of port 2 pins.

Figure 9.5-1 Block Diagram of Port 2 Pins



When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the DDR2 register.

■ Port 2 Registers

Port 2 registers are PDR2 and DDR2. The bits making up each register correspond to the port 2 pins on a one-to-one basis. Table 9.5-2 lists the port 2 pins and their corresponding register bits.

Table 9.5-2 Port 2 Pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 2	PDR2, DDR2	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	P27	P26	P25	P24	P23	P22	P21	P20

## 9.5.1 Port 2 Registers (PDR2 and DDR2)

This section describes the port 2 registers.

### ■ Functions of Port 2 Registers

#### ● Port 2 data register (PDR2)

The PDR2 register indicates the state of each pin of port 2.

#### ● Port 2 data direction register (DDR2)

The DDR2 register specifies the direction of a data flow (input or output) at each pin (bit) of port 2. When a DDR2 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

#### References:

- When a resource having output pins is used, the port functions as resource output pins regardless of the value in the DDR2 register as long as the resource output enable bit corresponding to the pins is set.
- To use a resource having input pins, reset the DDR2 register bit corresponding to each resource input pin to "0" to place the port in input mode.

Table 9.5-3 lists the functions of the port 2 registers.

**Table 9.5-3 Port 2 Register Functions**

Register	Data	During reading	During writing	Read/Write	Address	Initial value
Port 2 data register (PDR2)	0	The pin is at the low level.	The output latch is loaded with 0. When the pin functions as an output port, the pin is set to the low level.	R/W	000002 <sub>H</sub>	XXXXXXXX <sub>B</sub>
	1	The pin is at the high level.	The output latch is loaded with 1. When the pin functions as an output port, the pin is set to the high level.			
Port 2 data direction register (DDR2)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000012 <sub>H</sub>	00000000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			

R/W: Read/write enabled

X : Undefined

## 9.5.2 Operation of Port 2

---

**This section describes the operation of port 2.**

---

### ■ Operation of Port 2

#### ● Port operation in output mode

- Setting a bit of the DDR2 register to "1" places the corresponding port pin in output mode.
- Data written to the PDR2 register in output mode is held in the output latch of the PDR and output to the port pins.
- The PDR2 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

---

**Note:**

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR2 register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR2 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR2 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for resource output

The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR2 register bit is "0", the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

#### ● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, reset the DDR2 register to "0" to place the port in input mode.



● Port operation after a reset

- When the CPU is reset, the DDR2 register is initialized to "0". As a result, the output buffer is turned off (I/O mode changes to input), and the pins are placed in a high impedance state.
- The PDR2 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR2 register after the output data is set in the PDR2 register.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR2 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit. Table 9.5-4 lists the states of the port 2 pins.

**Table 9.5-4 States of Port 2 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P20/TIN1 to P27/IN3	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-Z

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z: High impedance

## 9.6 Port 3

**Port 3 is a general-purpose I/O port. It can also be used for resource output. The port pins can be switched in units of bits between the I/O port and resource. This section focuses on the general I/O port function. It provides the configuration of port 3, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 3 Configuration

Port 3 consists of the following:

- General-purpose I/O pins/resource I/O pins (P30/RTO0 to P37/PPG0)
- Port 3 data register (PDR3)
- Port 3 data direction register (DDR3)

### ■ Port 3 Pins

The port 3 I/O pins are also used as resource output pins. Therefore, the pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins. Table 9.6-1 lists the port 3 pins.

**Table 9.6-1 Port 3 Pins**

Port	Pin	Port function (single-chip mode)	Resource function	I/O form		Circuit type
				Input	Output	
Port 3	P30/RTO0*	P30	General-purpose I/O	RTO0*	Waveform generator output 0	G
	P31/RTO1*	P31		RTO1*	Waveform generator output 1	
	P32/RTO2*	P32		RTO2*	Waveform generator output 2	
	P33/RTO3*	P33		RTO3*	Waveform generator output 3	
	P34/RTO4*	P34		RTO4*	Waveform generator output 4	
	P35/RTO5*	P35		RTO5*	Waveform generator output 5	
	P36/PPG1*	P36		PPG1*	PPG1 output	H
	P37/PPG0	P37		PPG0	PPG0 output	

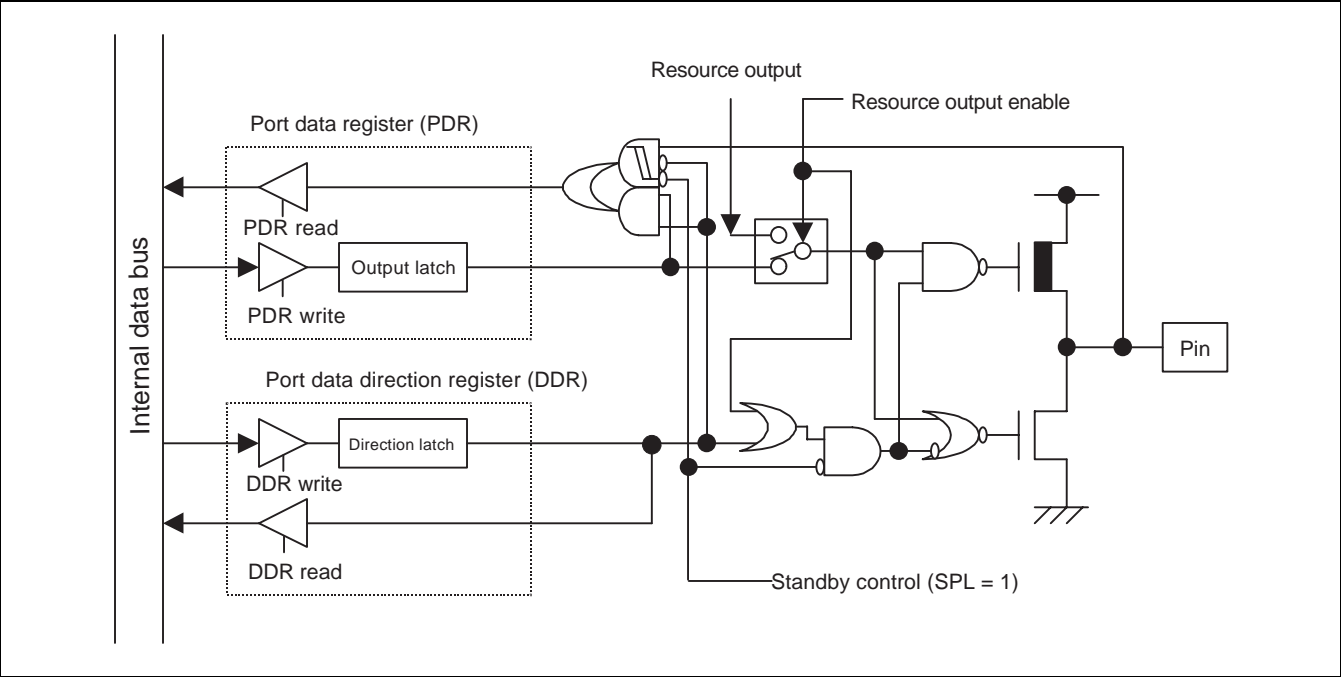
\*: Pin names not applicable to MB90465 series

See "1.7 I/O Circuit Types", for information on the circuit types.

■ Block Diagram of Port 3 Pins

Figure 9.6-1 is a block diagram of port 3 pins.

Figure 9.6-1 Block Diagram of Port 3 Pins



When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the DDR3 register.

■ Port 3 Registers

Port 3 registers are PDR3 and DDR3. The bits making up each register correspond to the port 3 pins on a one-to-one basis. Table 9.6-2 lists the port 3 pins and their corresponding register bits.

Table 9.6-2 Port 3 pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 3	PDR3, DDR3	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	Corresponding pin	P37	P36	P35	P34	P33	P32	P31	P30

## 9.6.1 Port 3 Registers (PDR3 and DDR3)

This section describes the port 3 registers.

### ■ Functions of Port 3 Registers

#### ● Port 3 data register (PDR3)

The PDR3 register indicates the state of each pin of port 3.

#### ● Port 3 data direction register (DDR3)

The DDR3 register specifies the direction of a data flow (input or output) at each pin (bit) of port 3. When a DDR3 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

Note:

When a resource having output pins is used, the port functions as resource output pins regardless of the value in the DDR3 register as long as the resource output enable bit corresponding to the pins is set.

Table 9.6-3 lists the functions of the port 3 registers.

**Table 9.6-3 Port 3 Register Functions**

Register	Data	During reading	During writing	Read/Write	Address	Initial value
Port 3 data register (PDR3)	0	The pin is at the low level.	The output latch is loaded with "0". When the pin functions as an output port, the pin is set to the low level.	R/W	000003 <sub>H</sub>	XXXXXXXX <sub>B</sub>
	1	The pin is at the high level.	The output latch is loaded with "1". When the pin functions as an output port, the pin is set to the high level.			
Port 3 data direction register (DDR3)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000013 <sub>H</sub>	00000000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			

R/W: Read/write enabled

X : Undefined

## 9.6.2 Operation of Port 3

---

**This section describes the operation of port 3.**

---

### ■ Operation of Port 3

#### ● Port operation in output mode

- Setting a bit of the DDR3 register to "1" places the corresponding port pin in output mode.
- Data written to the PDR3 register in output mode is held in the output latch of the PDR and output to the port pins.
- The PDR3 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

---

Note:

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR3 register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR3 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR3 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for resource output

The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR3 register bit is "0", the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

#### ● Port operation after a reset

- When the CPU is reset, the DDR3 register is initialized to "0". As a result, the output buffer is turned off (I/O mode changes to input), and the pins are placed in a high impedance state.
- The PDR3 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR3 register after the output data is set in the PDR3 register.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR3 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 9.6-4 lists the states of the port 3 pins.

**Table 9.6-4 States of Port 3 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P30/RTO0 to P37/PPG0	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-Z

SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z: High impedance

## 9.7 Port 4

**Port 4 is a general-purpose I/O port. It can also be used for resource I/O. The port pins can be switched in units of bits between the I/O port and resource. This section focuses on the general I/O port function. It provides the configuration of port 4, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 4 Configuration

Port 4 consists of the following:

- General-purpose I/O pins/resource I/O pins (P40/SIN0 to P46/PPG2)
- Port 4 data register (PDR4)
- Port 4 data direction register (DDR4)

### ■ Port 4 Pins

The port 4 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins. Table 9.7-1 lists the port 4 pins.

**Table 9.7-1 Port 4 Pins**

Port	Pin	Port function (single-chip mode)	Resource function	I/O form		Circuit type
				Input	Output	
Port 4	P40/SIN0	P40	SIN0	UART 0 data input	CMOS (hysteresis)	CMOS
	P41/SOT0	P41	SOT0	UART 0 data output		
	P42/SCK0	P42	SCK0	UART 0 serial clock I/O		
	P43/SNI0*	P43	SNI0*	Waveform sequencer input 0		
	P44/SNI1*	P44	SNI1*	Waveform sequencer input 1		
	P45/SNI2*	P45	SNI2*	Waveform sequencer input 2		
	P46/PPG2	P46	PPG2	PPG2 output		

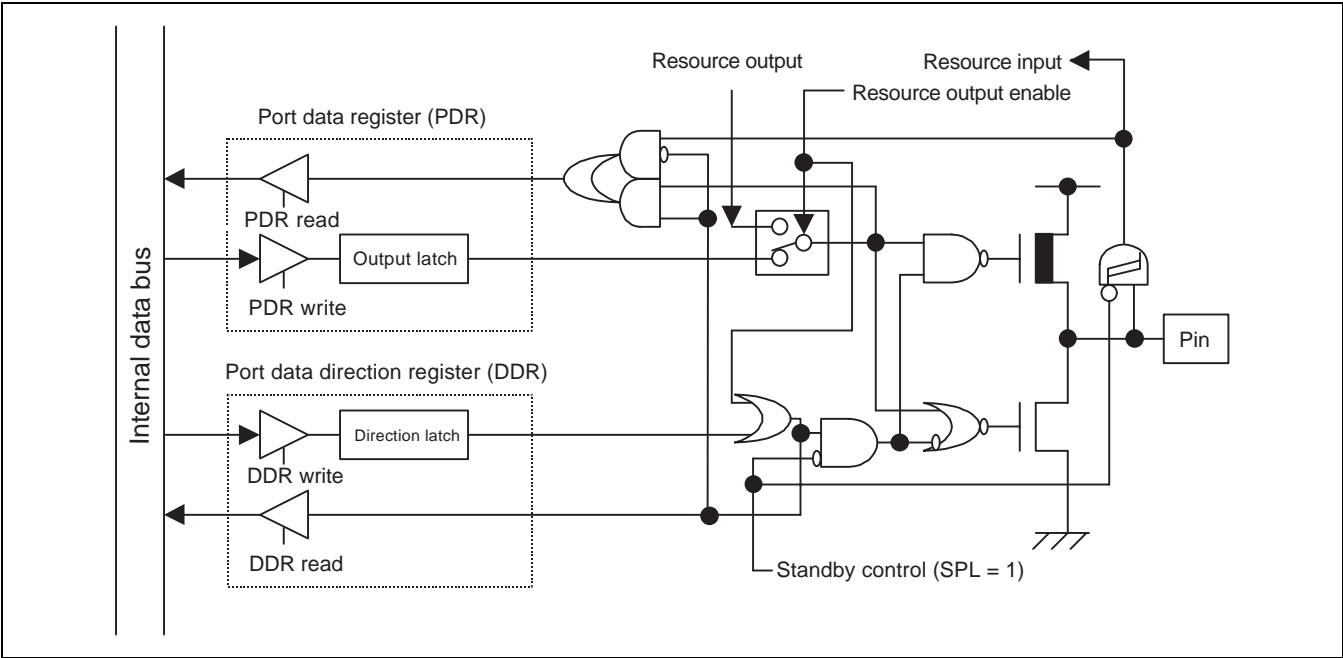
\*: Pin names not applicable to MB90465 series

See "1.7 I/O Circuit Types", for information on the circuit types.

■ Block Diagram of Port 4 Pins

Figure 9.7-1 is a block diagram of port 4 pins.

Figure 9.7-1 Block Diagram of Port 4 Pins



When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the DDR4 register.

■ Port 4 Registers

Port 4 registers are PDR4 and DDR4. The bits making up each register correspond to the port 4 pins on a one-to-one basis. Table 9.7-2 lists the port 4 pins and their corresponding register bits.

Table 9.7-2 Port 4 Pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 4	PDR4, DDR4	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	—	P46	P45	P44	P43	P42	P41	P40



## 9.7.1 Port 4 Registers (PDR4 and DDR4)

This section describes the port 4 registers.

### ■ Functions of Port 4 Registers

#### ● Port 4 data register (PDR4)

The PDR4 register indicates the state of each pin of port 4.

#### ● Port 4 data direction register (DDR4)

The DDR4 register specifies the direction of a data flow (input or output) at each pin (bit) of port 4. When a DDR4 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

#### References:

- When a resource having output pins is used, the port functions as resource output pins regardless of the value in the DDR4 register as long as the resource output enable bit corresponding to the pins is set.
- To use a resource having input pins, reset the DDR4 register bit corresponding to each resource input pin to "0" to place the port in input mode.

Table 9.7-3 lists the functions of the port 4 registers.

**Table 9.7-3 Port 4 Register Functions**

Register	Data	During reading	During writing	Read/Write	Address	Initial value
Port 4 data register (PDR4)	0	The pin is at the low level.	Setting an DDR5 bit to "0" enables the pin for a high-impedance. Setting an DDR5 bit to "1" enables the pin functions as an output port, and the pin is set to the low level.	R/W	000004 <sub>H</sub>	-XXXXXXX <sub>B</sub>
	1	The pin is at the high level.	Setting an DDR5 bit to "0" enables the pin for a high-impedance. Setting an DDR5 bit to "1" enables the pin functions as an output port, and the pin is set to the high level.			
Port 4 data direction register (DDR4)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000014 <sub>H</sub>	-0000000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			

R/W: Read/write enabled

X : Undefined

- : Empty bit

## 9.7.2 Operation of Port 4

---

**This section describes the operation of port 4.**

---

### ■ Operation of Port 4

#### ● Port operation in output mode

- Setting a bit of the DDR4 register to "1" places the corresponding port pin in output mode.
  - Data written to the PDR4 register in output mode is held in the output latch of the PDR and output to the port pins.
  - The PDR4 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).
- 

Note:

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR4 register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR4 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR4 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for resource output

The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR4 register bit is "0", the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

#### ● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, reset the DDR4 register to "0" to place the port in input mode.

● Port operation after a reset

- When the CPU is reset, the DDR4 register is initialized to "0". As a result, the output buffer is turned off (I/O mode changes to input), and the pins are placed in a high impedance state.
- The PDR4 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR4 register after the output data is set in the PDR4 register.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR4 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit. Table 9.7-4 lists the states of the port 4 pins.

**Table 9.7-4 States of Port 4 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P40/SIN0 to P46/PPG2	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-Z

SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z: High impedance

## 9.8 Port 5

**Port 5 is a general-purpose I/O port. It can also be used for A/D converter analog input. The port pins can be switched in units of bits between the I/O port and analog input. This section focuses on the general I/O port function. It provides the configuration of port 5, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 5 Configuration

Port 5 consists of the following:

- General-purpose I/O pins/analog input pins (P50/AN0 to P57/AN7)
- Port 5 data register (PDR5)
- Port 5 data direction register (DDR5)
- Analog input enable register (ADER)

### ■ Port 5 Pins

The port 5 I/O pins are also used as analog input pins. The pins cannot be used as general-purpose I/O port pins when they are used for analog input. Similarly, the port 5 I/O pins cannot be used for analog input when they are used as a general-purpose I/O port. Table 9.8-1 lists the port 5 pins.

**Table 9.8-1 Port 5 Pins**

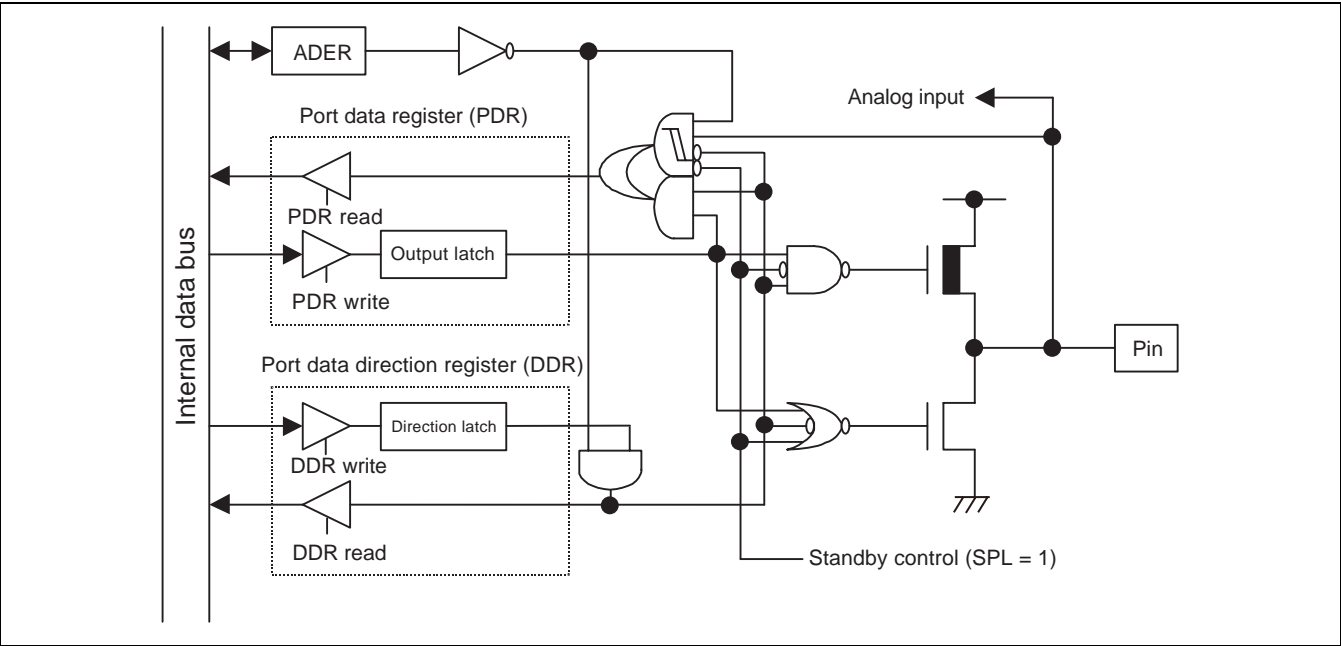
Port	Pin	Port function (single-chip mode)		Resource function		I/O form		Circuit type
						Input	Output	
Port 5	P50/AN0	P50	General- purpose I/O	AN0	Analog input 0	Analog/ CMOS	CMOS	I
	P51/AN1	P51		AN1	Analog input 1			
	P52/AN2	P52		AN2	Analog input 2			
	P53/AN3	P53		AN3	Analog input 3			
	P54/AN4	P54		AN4	Analog input 4			
	P55/AN5	P55		AN5	Analog input 5			
	P56/AN6	P56		AN6	Analog input 6			
	P57/AN7	P57		AN7	Analog input 7			

See "1.7 I/O Circuit Types", for information on the circuit types.

■ Block Diagram of Port 5 Pins

Figure 9.8-1 is a block diagram of port 5 pins.

Figure 9.8-1 Block Diagram of Port 5 Pins



For a pin used as an input port, reset the corresponding DDR5 register bit to "0", and also reset the corresponding ADER register bit to "0".

For an pin used as an analog input pin, reset the corresponding DDR5 register bit to "0" and set the corresponding ADER register bit to "1". In this case, the value read from the PDR5 register is "0".

■ Port 5 Registers

Port 5 registers are PDR5, DDR5 and ADER. The bits making up each register correspond to the port 5 pins on a one-to-one basis. Table 9.8-2 lists the port 5 pins and their corresponding register bits.

Table 9.8-2 Port 5 Pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 5	PDR5, DDR5, ADER	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	Corresponding pin	P57	P56	P55	P54	P53	P52	P51	P50

## 9.8.1 Port 5 Registers (PDR5, DDR5 and ADER)

This section describes the port 5 registers.

### ■ Functions of Port 5 Registers

#### ● Port 5 data register (PDR5)

The PDR5 register indicates the state of each pin of port 5.

#### ● Port 5 data direction register (DDR5)

The DDR5 register specifies the direction of a data flow (input or output) at each pin (bit) of port 5. When a DDR5 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

#### ● Analog input enable register (ADER)

Each bit of the ADER register specifies whether the corresponding port 5 pin is to be used as a general-purpose I/O port or an analog input pin. Setting an ADER bit to "1" enables the corresponding pin for analog input. Setting the bit to "0" enables the pin for general-purpose I/O.

Note:

If a signal at an intermediate level is input in port I/O mode, input leak current flows. Therefore, for a pin used for analog input, be sure to set the corresponding ADER bit to "1" for analog input.

Reference:

When the CPU is reset, the DDR5 register is reset to "0" and the ADER register is set to "1" for analog input.

Table 9.8-3 lists the functions of the port 5 registers.

**Table 9.8-3 Port 5 Register Functions**

Register	Data	During reading	During writing	Read/Write	Address	Initial value
Port 5 data register (PDR5)	0	The pin is at the low level.		R/W	000005 <sub>H</sub>	XXXXXXX <sub>B</sub>
	1	The pin is at the high level.				
Port 5 data direction register (DDR5)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000015 <sub>H</sub>	00000000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			
Analog input enable register (ADER)	0	Port I/O mode		R/W	000017 <sub>H</sub>	11111111 <sub>B</sub>
	1	Analog input mode				

R/W: Read/write enabled

X : Undefined

## 9.8.2 Operation of Port 5

---

**This section describes the operation of port 5.**

---

### ■ Operation of Port 5

#### ● Port operation in output mode

- Setting a bit of the DDR5 register to "1" places the corresponding port pin in output mode.
  - Data written to the PDR5 register in output mode is held in the output latch of the PDR and output to the port pins.
  - The PDR5 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).
- 

Note:

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR5 and ADER register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR5 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR5 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for analog input

To use a port pin for analog input, write "1" to the corresponding ADER bit. Doing so disables the port from operating as a general-purpose port pin and enables it to function as an analog input pin. When PDR5 is accessed in read mode in this situation, a value of "0" is read.

#### ● Port operation after a reset

When the CPU is reset, the DDR5 register is initialized to "0" and the ADER register is initialized to "1" to place the port in analog input mode. To use the port as a general-purpose port, write "0" to the ADER register in advance to place the port in port I/O mode.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 9.8-4 lists the states of the port 5 pins.

**Table 9.8-4 States of Port 5 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P50/AN0 to P57/AN7	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-Z

SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z: High impedance



## 9.9 Port 6

**Port 6 is a general-purpose I/O port. It can also be used for resource input and output. The port pins can be switched in units of bits between the I/O port and resource. This section focuses on the general I/O port function. It provides the configuration of port 6, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.**

### ■ Port 6 Configuration

Port 6 consists of the following:

- General-purpose I/O pins/resource I/O pins (P60/SIN1 to P63/INT7)
- Port 6 data register (PDR6)
- Port 6 data direction register (DDR6)

### ■ Port 6 Pins

The port 6 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins. Table 9.9-1 lists the port 6 pins.

**Table 9.9-1 Port 6 Pins**

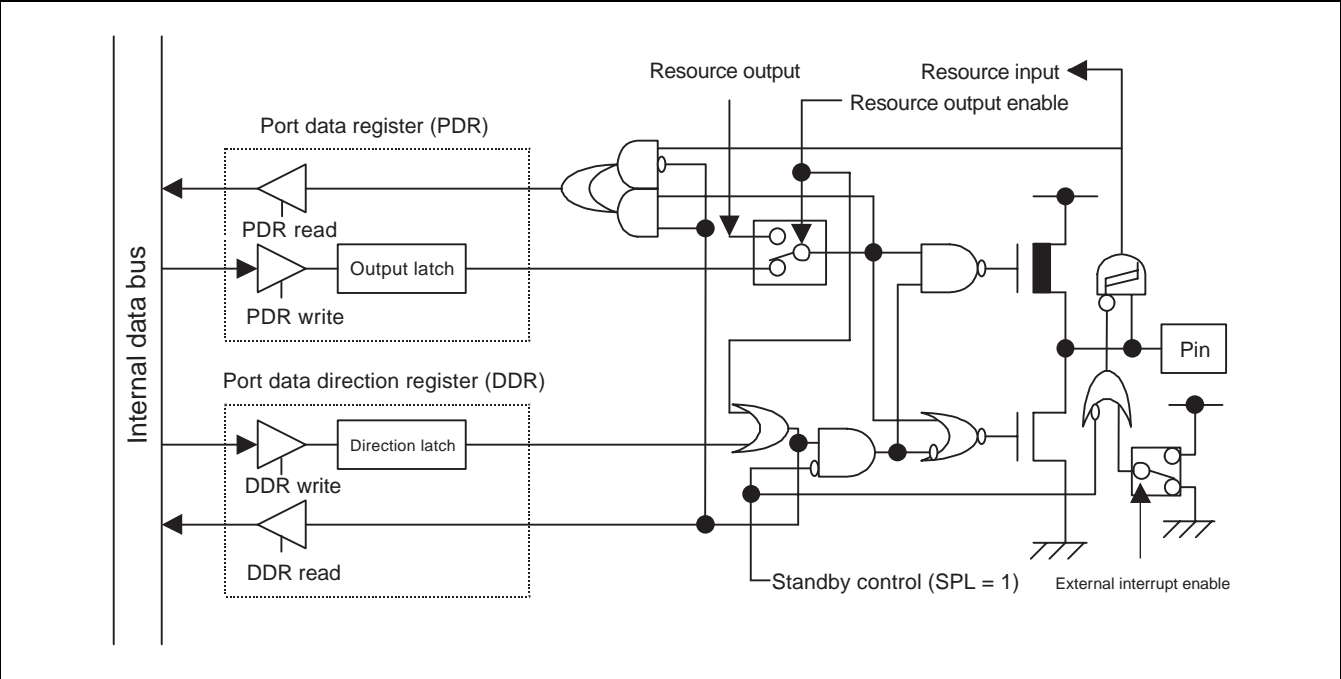
Port	Pin	Port function (single-chip mode)		Resource function		I/O form		Circuit type
						Input	Output	
Port 6	P60/SIN1	P60	General-purpose I/O	SIN1	UART1 data input	CMOS (hysteresis)	CMOS	F
	P61/SOT1	P61		SOT1	UART1 data output			
	P62/SCK1	P62		SCK1	UART1 serial clock I/O			
	P63/INT7	P63		INT7	External interrupt input			

See "1.7 I/O Circuit Types", for information on the circuit types.

■ Block Diagram of Port 6 Pins

Figure 9.9-1 is a block diagram of port 6 pins.

Figure 9.9-1 Block Diagram of Port 6 Pins



■ Port 6 Registers

Port 6 registers are PDR6 and DDR6. The bits making up each register correspond to the port 6 pins on a one-to-one basis. Table 9.9-2 lists the port 6 pins and their corresponding register bits.

Table 9.9-2 Port 6 Pins and their Corresponding Register Bits

Port	Register bits and corresponding port pins								
Port 6	PDR6, DDR6	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	—	—	—	—	P63	P62	P61	P60

## 9.9.1 Port 6 Registers (PDR6 and DDR6)

This section describes the port 6 registers.

### ■ Functions of Port 6 Registers

#### ● Port 6 data register (PDR6)

The PDR6 register indicates the state of each pin of port 6.

#### ● Port 6 data direction register (DDR6)

The DDR6 register specifies the direction of a data flow (input or output) at each pin (bit) of port 6. When a DDR6 register bit is "1", the corresponding port (pin) is set as an output port. When the bit is "0", the port (pin) is set as an input port.

Notes:

- When a resource having output pins is used, the port functions as resource output pins regardless of the value in the DDR6 register as long as the resource output enable bit corresponding to the pins is set.
- To use a resource having input pins, reset the DDR6 register bit corresponding to each resource input pin to "0" to place the port in input mode. Table 9.9-3 lists the functions of the port 6 registers.

**Table 9.9-3 Port 6 Register Functions**

Register	Data	During reading	During writing	Read/Write	Address	Initial value
Port 6 data register (PDR6)	0	The pin is at the low level.	The output latch is loaded with "0". When the pin functions as an output port, the pin is set to the low level.	R/W	000006 <sub>H</sub>	----XXXX <sub>B</sub>
	1	The pin is at the high level.	The output latch is loaded with "1". When the pin functions as an output port, the pin is set to the high level.			
Port 6 data direction register (DDR6)	0	The direction latch is "0".	The output buffer is turned off to place the port in input mode.	R/W	000016 <sub>H</sub>	----0000 <sub>B</sub>
	1	The direction latch is "1".	The output buffer is turned on to place the port in output mode.			

R/W: Read/write enabled

X : Undefined

- : Empty bit

## 9.9.2 Operation of Port 6

---

**This section describes the operation of port 6.**

---

### ■ Operation of Port 6

#### ● Port operation in output mode

- Setting a bit of the DDR6 register to "1" places the corresponding port pin in output mode.
- Data written to the PDR6 register in output mode is held in the output latch of the PDR and output to the port pins.
- The PDR6 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

---

**Note:**

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

---

#### ● Port operation in input mode

- Resetting a bit of the DDR6 register to "0" places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR6 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR6 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

#### ● Port operation for resource output

The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR6 register bit is "0", the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

#### ● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, reset the DDR6 register to "0" to place the port in input mode.

● Port operation after a reset

- When the CPU is reset, the DDR6 register is initialized to "0". As a result, the output buffer is turned off (I/O mode changes to input), and the pins are placed in a high impedance state.
- The PDR6 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR6 register after output data is set in the PDR6 register.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption mode control register (LPMCR) is already "1" when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR6 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 9.9-4 lists the states of the port 6 pins.

**Table 9.9-4 States of Port 6 Pins**

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P60/SIN1 to P63/INT7	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-Z

SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)

Hi-Z: High impedance

## 9.10 Sample I/O Port Program

This section provides a sample program using I/O port pins.

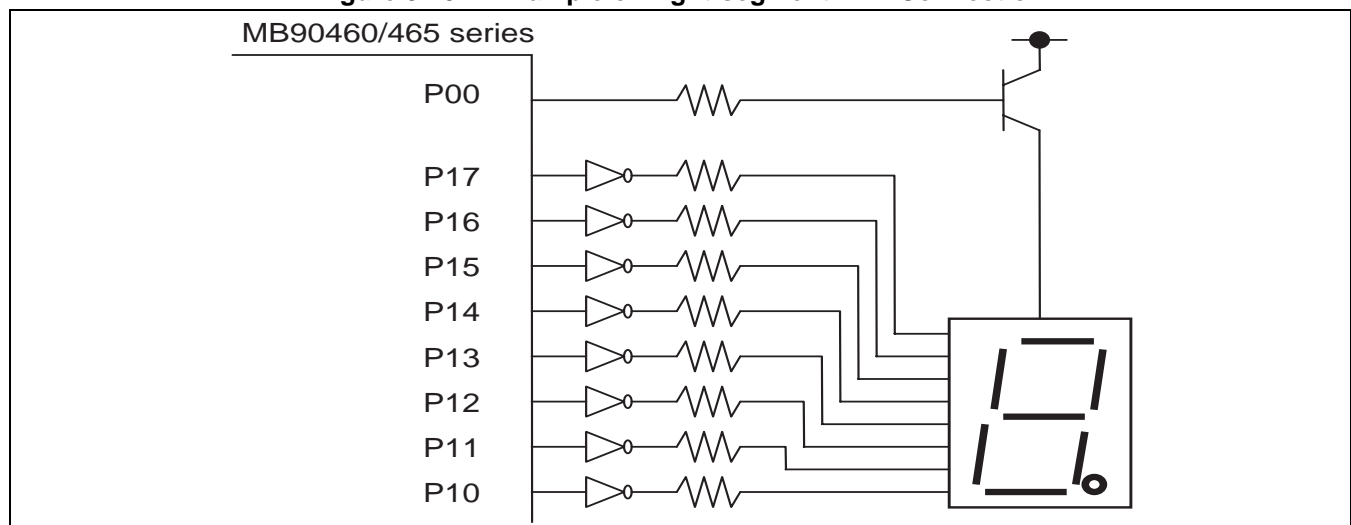
### ■ Sample I/O Port Program

#### ● Processing specifications

- Ports 0 and 1 are used to turn on all segments of a seven-segment (eight-segment if the decimal point is included) LED.
- Pin P00 corresponds to the anode common pin of the LED and pins P10 to P17 correspond to the segment pins.

Figure 9.10-1 is an example of connecting the eight-segment LED to the MB90460/465 series ports.

**Figure 9.10-1 Example of Eight-segment LED Connection**



#### ● Coding example

```

PDR      EQU      000000H
PDR1     EQU      000001H
DDR0     EQU      000010H
DDR1     EQU      000011H
;-----Main program-----
CODE                      CSEG
START:

                                ; Initialization
                                MOV     I:PDR0, #00000000B ; Puts P00 at a low level (#XXXXXXX0B)
                                MOV     I:DDR0, #11111111B ; Puts all port 0 bits in output mode
                                MOV     I:PDR1, #11111111B ; Sets all port 1 bits to "1"
                                MOV     I:DDR1, #11111111B ; Puts all port 1 bits in output mode

CODE                      ENDS
;-----
                                END          START

```



# **CHAPTER 10**

---

## ***TIME-BASE TIMER***

**This chapter describes the functions and operation of the time-base timer.**

- 10.1 Overview of the Time-base Timer
- 10.2 Configuration of the Time-base Timer
- 10.3 Time-base Timer Control Register (TBTC)
- 10.4 Time-base Timer Interrupts
- 10.5 Operation of the Time-base Timer
- 10.6 Usage Notes on the Time-base Timer
- 10.7 Sample Program for the Time-base Timer Program



## 10.1 Overview of the Time-base Timer

The time-base timer is an 18-bit free-run counter (time-base counter) that counts up in synchronization with the internal count clock (one-half of the source oscillation). The timer has an interval timer function that can select four intervals.

The time-base timer also has functions for timer output of the oscillation stabilization time and for supplying the clocks for the watchdog timer.

### ■ Interval Timer Function

The interval timer function repeatedly generates an interrupt request at a given interval.

- An interrupt request is generated when the interval timer bit for the time-base counter overflows.
- The interval timer bit (interval) can be selected from four types. Table 10.1-1 lists the intervals for the time-base timer.

**Table 10.1-1 Intervals for the Time-base Timer**

Internal count clock cycle	Interval cycle
$2 / \text{HCLK}$ (0.5 $\mu\text{s}$ )	$2^{12} / \text{HCLK}$ (Approx. 1.0 ms)
	$2^{14} / \text{HCLK}$ (Approx. 4.1 ms)
	$2^{16} / \text{HCLK}$ (Approx. 16.4 ms)
	$2^{19} / \text{HCLK}$ (Approx. 131.1 ms)

HCLK: Oscillation clock

Values in parentheses are for a 4 MHz oscillation clock.

## ■ Clock Supply Function

The clock supply function supplies clocks to the oscillation stabilization time timer and to some peripheral functions.

Table 10.1-2 lists the cycle times of clocks supplied from the time-base timer to each peripheral.

**Table 10.1-2 Clock Cycle Time supplied from the Time-base Timer**

Clock destination	Clock cycle time	Remarks
Oscillation stabilization time	$2^{13} / \text{HCLK}$ (Approx. 2.0 ms)	Oscillation stabilization time for ceramic vibrator
	$2^{15} / \text{HCLK}$ (Approx. 8.2 ms)	Oscillation stabilization time for crystal vibrator
	$2^{18} / \text{HCLK}$ (Approx. 65.4 ms)	
Watchdog timer	$2^{12} / \text{HCLK}$ (Approx. 1.0 ms)	Count-up clock for watchdog timer
	$2^{14} / \text{HCLK}$ (Approx. 4.1 ms)	
	$2^{16} / \text{HCLK}$ (Approx. 16.4 ms)	
	$2^{19} / \text{HCLK}$ (Approx. 131.1 ms)	

HCLK: Oscillation clock

Values in parentheses occurs during operation of the 4 MHz oscillation clock.

### Reference:

The oscillation stabilization time is the yardstick because the oscillation cycle time is unstable as soon as oscillation starts.

## 10.2 Configuration of the Time-base Timer

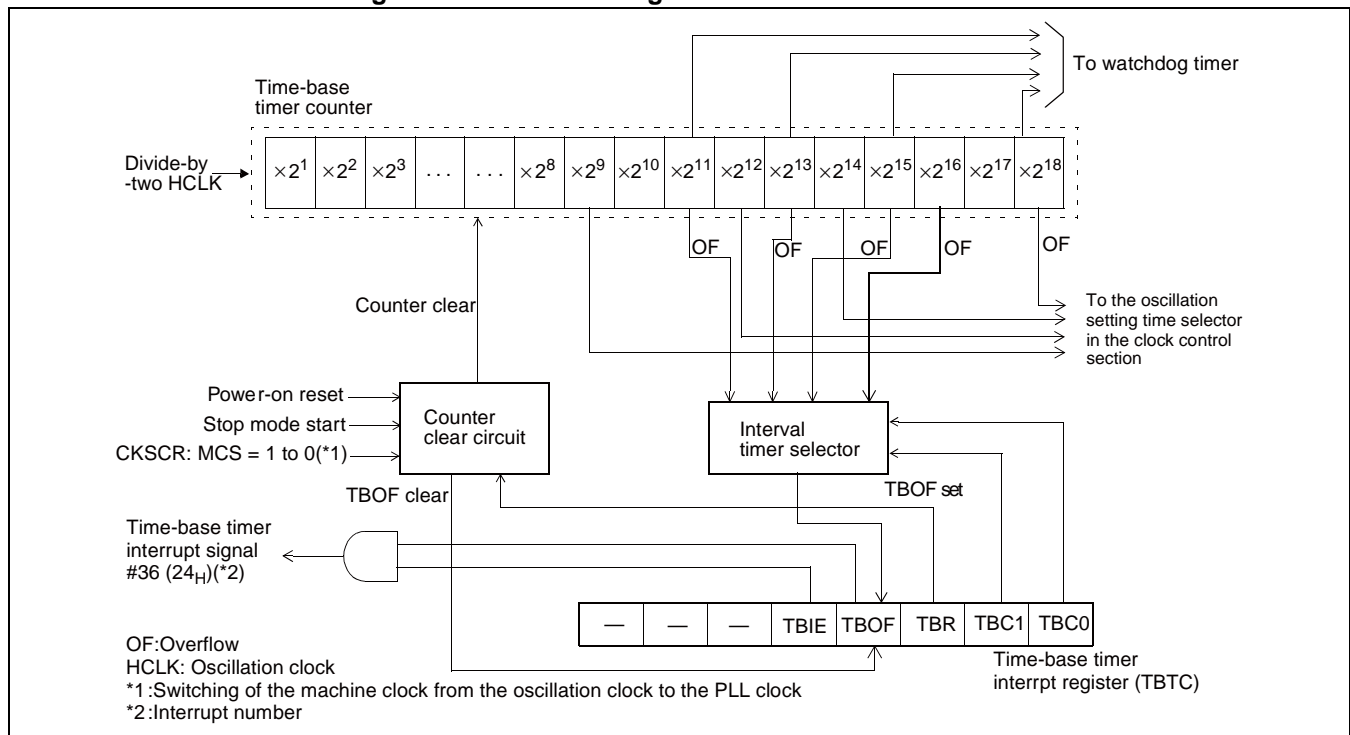
**The time-base timer consists of the following four blocks:**

- **Time-base timer counter**
- **Counter clear circuit**
- **Interval timer selector**
- **Time-base timer control register (TBTC)**

### ■ Block Diagram of the Time-base Timer

Figure 10.2-1 shows the block diagram of the time-base timer.

**Figure 10.2-1 Block Diagram of the Time-base Timer**



- Time-base timer counter

This 18-bit up counter uses the divide-by-two clock of the oscillation clock (HCLK) as the count clock.

- Counter clear circuit

Used to clear the counter by writing "0" to the TBTC:TBR bit, by a power-on reset or by transition to stop mode (LPMCR: STP = 1).

- Interval timer selector

Selects one of four outputs of the time-base timer counter. An overflow of the selected bit becomes an interrupt cause.

- Time-base timer control register (TBTC)

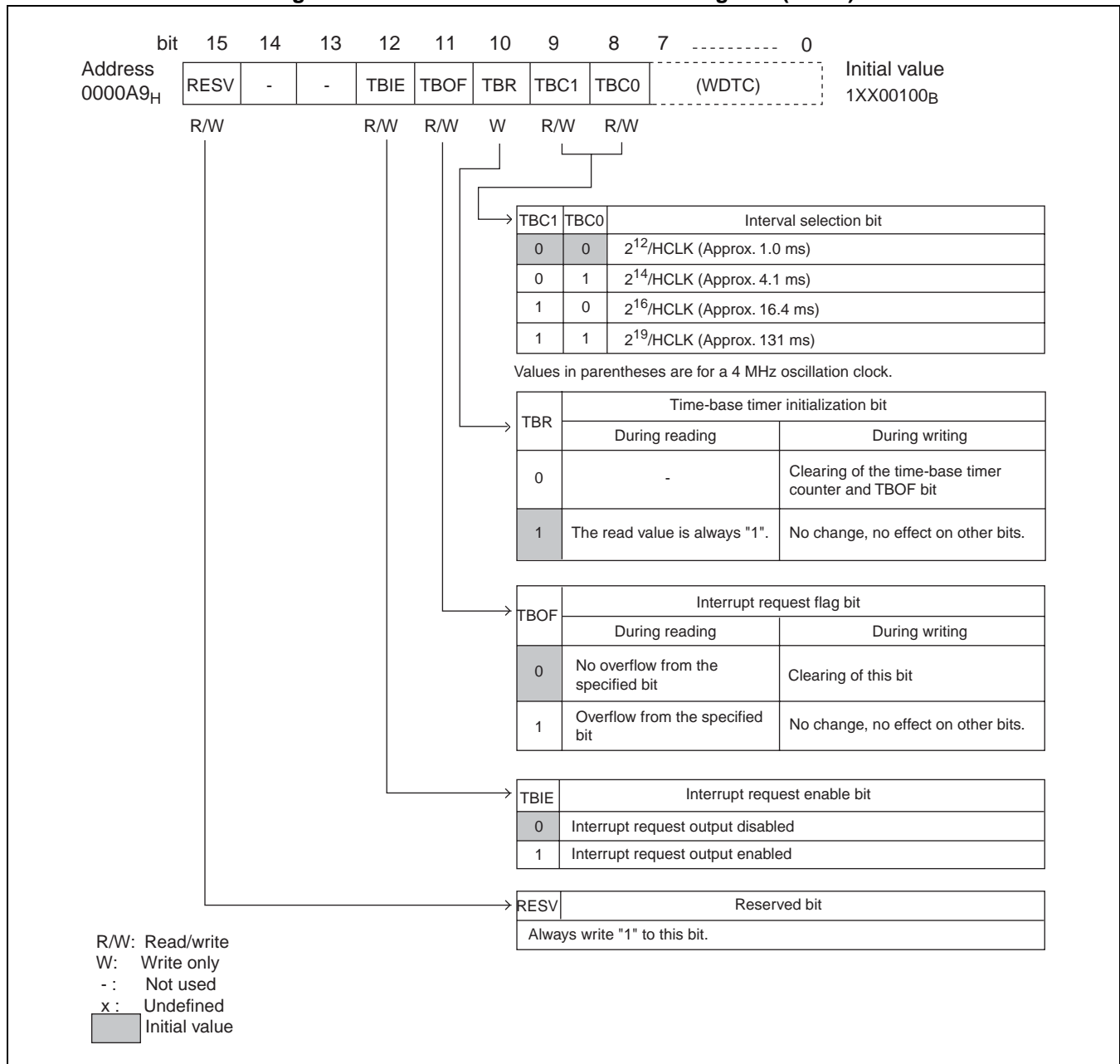
Selects the interval, clears the counter, controls an interrupt request, and checks the status.

## 10.3 Time-base Timer Control Register (TBTC)

The time-base timer control register (TBTC) selects the interval, clears the counter, controls interrupts, and checks the status.

### ■ Time-base Timer Control Register (TBTC)

Figure 10.3-1 Time-base Timer Control Register (TBTC)



**Table 10.3-1 Function Description of Each Bit in the Time-base Timer Control Register (TBTC)**

Bit name		Function
bit15	RESV: Reserved bit	(Note) Always write "1" to this bit.
bit14, bit13	Not used	<ul style="list-style-type: none"> <li>When read, the value is undefined.</li> <li>Writing has no effect on operation.</li> </ul>
bit12	TBIE: Interrupt request enable bit	<ul style="list-style-type: none"> <li>Used to enable or disable the output of an interrupt request to the CPU.</li> <li>When this bit and the interrupt request flag bit (TBOF) are "1", an interrupt request is output.</li> </ul>
bit11	TBOF: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is set to "1" when the bit specifying the time-base timer counter overflows.</li> <li>When this bit and the interrupt request enable bit (TBIE) are 1, an interrupt request is output.</li> <li>During writing, this bit is cleared with "0". If "1" is written, the bit does not change and there is no effect.</li> </ul> (Note) <ul style="list-style-type: none"> <li>To clear the TBOF bit, disable the time-base timer interrupt by specifying the TBIE bit or processor status (PS) ILM bit.</li> <li>The TBOF bit is cleared by writing "0", by a transition to stop mode, by clearing of the time-base timer with the TBR bit or by a reset.</li> </ul>
bit10	TBR: Time-base timer initialization bit	<ul style="list-style-type: none"> <li>Used to clear the time-base timer counter.</li> <li>When "0" is written to this bit, the counter is cleared and the TBOF bit is cleared. If "1" is written, the bit does not change and there is no effect.</li> </ul> (Reference) The read value is always "1".
bit9, bit8	TBC1, TBC0: Interval selection bit	<ul style="list-style-type: none"> <li>Used to select an interval timer cycle.</li> <li>The bit for the interval timer of the time-base timer counter is specified.</li> <li>Four types of interval can be selected.</li> </ul>

## 10.4 Time-base Timer Interrupts

The time-base timer can generate an interrupt request when the bit specifying the time-base timer counter overflows.

### ■ Time-base Timer Interrupts

The interrupt request flag bit (TBTC: TBOF) is set to "1" when the time-base timer counter counts up with the internal count clock and when the bit for the selected interval timer bit overflows. If the interrupt request enable bit has been enabled (TBTC: TBIE = 1), an interrupt request (#36) is generated in the CPU. Write "0" to the TBOF bit in the interrupt handling routine to clear the interrupt request. When the specified bit overflows, the TBOF bit is set regardless of the TBIE bit value.

Note:

Clear the interrupt request flag bit (TBTC: TBOF) while a time-base timer interrupt is disabled by setting the TBIE bit or the processor status (PS) ILM bit.

Reference:

When the TBOF bit is "1", if the TBIE bit status is switched from disable to enable (0 → 1), an interrupt request occurs immediately.

### ■ Time-base Timer Interrupts and EI<sup>2</sup>OS

Table 10.4-1 lists the time-base timer interrupt and EI<sup>2</sup>OS.

**Table 10.4-1 Time-base Interrupts and EI<sup>2</sup>OS**

Interrupt number	Interrupt level setting register		Vector table address			EI <sup>2</sup> OS
	Register name	Address	Lower	Upper	Bank	
#36 (24 <sub>H</sub> )	ICR12	0000BC <sub>H</sub>	FFFF6C <sub>H</sub>	FFFF6D <sub>H</sub>	FFFF6E <sub>H</sub>	Δ

Δ: Usable when an interrupt cause that shares the ICR is not used.

Note:

ICR12 is common to the time-base timer interrupt and input capture channels 2/3 interrupt. Interrupts can be used for two applications, but the interrupt level is the same.

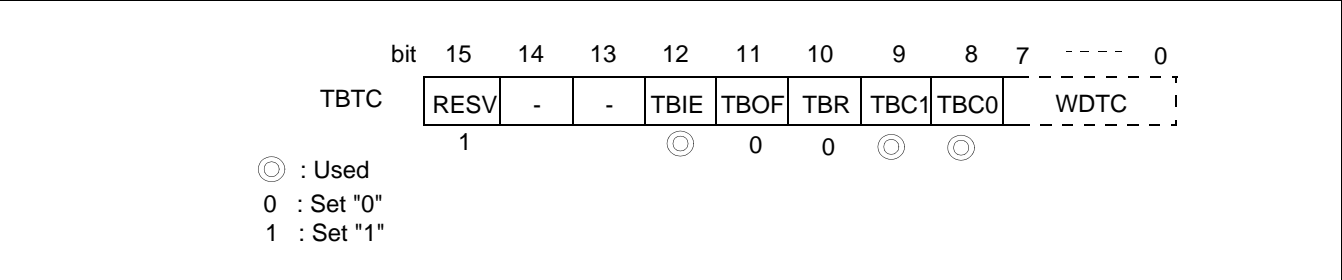
# 10.5     Operation of the Time-base Timer

The time-base timer provides the interval timer function and the clock supply function that supplies clocks to some peripheral functions.

■ **Operation of the Interval Timer Function (Time-base Timer)**

The interval timer function generates an interrupt request for each interval  
The setting in Figure 10.5-1 is required to all the timer to operate as an interval timer.

**Figure 10.5-1   Setting of the Time-base Timer**



- The time-base timer counter continues counting up in synchronization with the internal count clock (one-half of the oscillation clock) as long as the clock is being oscillated.
- When the counter is cleared (TBR = 0), it counts up from "0". When the interval timer bit overflows, the interrupt request flag bit (TBOF) is set to "1". If interrupt request output has been enabled (TBIE = 1), an interrupt is generated for each selected interval based on the cleared time.
- The interval may become longer than the time set because of time-base timer clearing.

## ■ Oscillation Stabilization Time Timer Function

The time-base timer is also used as the oscillation stabilization time timer for oscillation and the PLL clocks.

The oscillation stabilization time is set for the interval from the time the counter counts up from "0" (count clear) until the oscillation stabilization time bit overflows. When control returns from time-base timer mode to PLL clock mode, the oscillation stabilization time starts from the middle of counting because the time-base timer counter has been not cleared. Table 10.5-1 shows the clearing of the time-base counter and the oscillation stabilization times.

**Table 10.5-1 Time-base Timer Counter Clearing and Oscillation Settling Times**

Operation	Counter clear	TBOF clear	Oscillation settling time
TBTC: Writing of 0 to TBR	O	O	
Power-on reset	O	O	Oscillation clock oscillation stabilization time
Watchdog reset			
Releasing of stop mode	O	O	Oscillation clock oscillation stabilization time (at return to main clock mode)
Transition from oscillation clock mode to PLL clock mode (MCS = 1 → 0)	O	O	PLL clock oscillation stabilization time
Releasing of time-base timer mode	X	X	PLL clock oscillation stabilization time (at return to PLL clock mode)
Releasing of sleep mode	X	X	

O: Available

X: Not available

## ■ Clock Supply Function

The time-base timer supplies clocks to the watchdog timer. Clearing of the time-base counter affects operation of the watchdog timer.



## 10.6 Usage Notes on the Time-base Timer

---

**Notes about the effects on peripheral functions of clearing interrupt requests and the time-base timer are given below.**

---

### ■ Time-base Timer Usage Notes

#### ● Clearing interrupt requests

The TBOF bit of the time-base timer control register must be cleared while a time-base timer interrupt is masked by the TBIE bit or the interrupt level mask register (ILM) of the processor status (PS).

#### ● Effects of time-base timer clearing

Clearing of the time-base timer counter affects the following operations:

- When the time-base timer is using the interval timer function (interval interrupt)
- When the watchdog timer is being used

#### ● Use of the time-base timer as the oscillation stabilization time timer

At power-on, the source oscillation of the main clock stops in main stop mode. After oscillator operation starts, the operating clock supplied by the time-base timer is used to take the oscillation stabilization wait time of the main clock. An appropriate oscillation stabilization wait time must be selected based on the type of oscillating element connected to the main clock oscillator (clock generation section). See "5.5 Oscillation Stabilization Wait Interval", for details.

#### ● Notes on peripheral functions to which clocks are supplied from the time-base timer

In the mode in which the main clock source oscillation stops, the counter is cleared and time-base timer operation stops. When the time-base timer counter is cleared, the clock supplied from the time-base timer is supplied from its initial state. As a result, the H level is shortened and the L level lengthened 1/2 cycle. Although the clock for the watchdog timer is also supplied from its initial state; the watchdog timer operates in normal cycles because the watchdog timer counter is cleared at the same time.

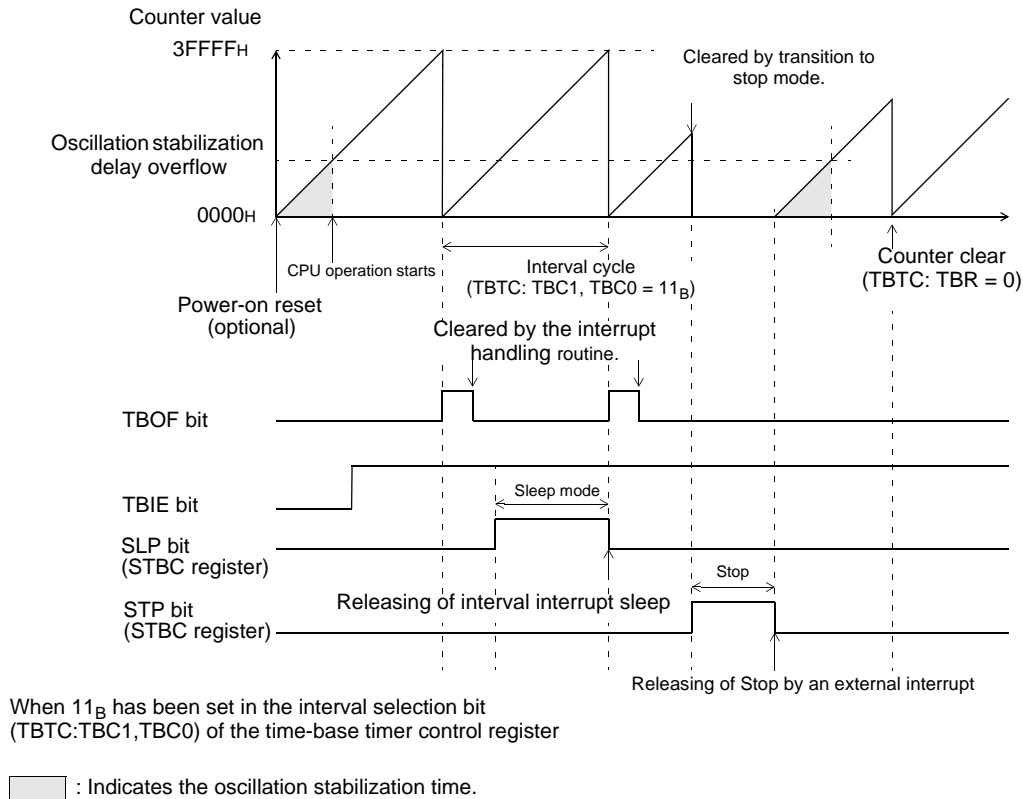
## ■ Operation of the Time-base Timer

The following operations are shown in Figure 10.6-1:

- A power-on reset occurs.
- Sleep mode is entered during operation of the interval timer function.
- A counter clear request is issued.

When stop mode is entered, the time-base timer is cleared and its operation stops. On return from stop mode, the time-base timer counts the oscillation stabilization time.

### Figure 10.6-1 Time-base Timer Operations



## 10.7 Sample Program for the Time-base Timer Program

---

This section contains a sample program for the time-base timer.

---

### ■ Sample Program for the Time-base Timer

#### ● Processing

An interval interrupt of  $2^{12} / \text{HCLK}$  (HCLK: oscillation clock) is repeatedly generated. The interval becomes approx. 1.0 ms (during 4 MHz operation).

#### ● Coding example

```

ICR12    EQU        0000BCH        ;Time-base timer interrupt control register
TBTC     EQU        0000A9H        ;Time-base timer control register
TBOF     EQU        TBTC:3         ;Interrupt request flag bit
;-----Main program-----
CODE     CSEG
START:
;           :                               ;Assumes that stack pointer (SP) has already
;           :                               ;been initialized
;           AND     CCR,#0BFH        ;Disables interrupts
;           MOV     I:ICR12,#00H     ;Interrupt level 0 (highest)
;           MOV     I:TBTC,#10010000B ;Fixes upper 3 bits
;           :                               ;Enables interrupts and clears TBOF
;           :                               ;Clears counter
;           :                               ;Selects interval  $2^{12}/\text{HCLK}$ 
;           MOV     ILM,#07H        ;Sets PS ILM to level 7
;           OR      CCR,#40H        ;Enables interrupts
LOOP:    MOV     A,#00H             ;Endless loop
;           MOV     A,#01H
;           BRA     LOOP
;-----Interrupt program-----
WARI:
;           CLRB    I:TBOF          ;Clears interrupt request flag
;           :
;           :                       ;User handling
;           :
;           RETI                    ;Returns from interrupt
CODE     ENDS

```

```

;-----Vector setting-----
VECT      CSEG      ABS=0FFH
          ORG        0FF6CH          ;Sets vector for interrupt #36 (24H)
          DSL        WARI
          ORG        0FFDCH          ;Sets reset vector
          DSL        START
          DB          00H            ;Sets single-chip mode
VECT      ENDS
          END          START

```



# **CHAPTER 11**

---

# **WATCHDOG TIMER**

**This chapter describes the functions and operation of the watchdog timer.**

- 11.1 Overview of the Watchdog Timer
- 11.2 Configuration of the Watchdog Timer
- 11.3 Watchdog Timer Control Register (WDTC)
- 11.4 Operation of the Watchdog Timer
- 11.5 Usage Notes on the Watchdog Timer
- 11.6 Sample Program for the Watchdog Timer

## 11.1 Overview of the Watchdog Timer

The watchdog timer is a 2-bit counter that uses the time-base timer supply clock as the count clock. After activation, if the watchdog timer is not cleared within a given time, the CPU is reset.

### ■ Watchdog Timer Function

The watchdog timer is a counter for handling program crashes. Once the watchdog timer is activated, it must be regularly cleared within a given time. If the program results in an endless loop and the watchdog timer is not cleared over a given time, a watchdog reset is generated for the CPU.

Table 11.1-1 lists the watchdog timer intervals. If the watchdog timer is not cleared, a watchdog reset is generated between the minimum time and maximum time. Clear the counter within the minimum time listed in this table.

**Table 11.1-1 Intervals for the Watchdog Timer**

Interval		
Minimum*	Maximum*	Oscillation clock cycle count
Approx. 3.58 ms	Approx. 4.61 ms	$2^{14} \pm 2^{11}$ cycle
Approx. 14.33 ms	Approx. 18.3 ms	$2^{16} \pm 2^{13}$ cycle
Approx. 57.23 ms	Approx. 73.73 ms	$2^{18} \pm 2^{15}$ cycle
Approx. 458.75 ms	Approx. 589.82 ms	$2^{21} \pm 2^{18}$ cycle

\* Value during operation of the 4 MHz oscillation clock

The maximum and minimum watchdog timer intervals and the oscillation clock cycle count depend on the clear timing.

The interval is 3.5 to 4.5 times longer than the cycle of the count clock (time-base timer supply clock).

See "11.4 Operation of the Watchdog Timer".

#### Note:

The watchdog counter consists of a 2-bit counter that uses the carry signals of the time-base timer as count clocks. Therefore, if the time-base timer is cleared, the watchdog reset generation time may become longer than the time set.

#### Reference:

At activation, the watchdog timer is initialized by a power-on or watchdog reset, and is placed in stopped status. The watchdog timer is cleared by an external pin reset, software reset, writing to the WTE bit (watchdog timer control register), sleep mode or transition to stop mode. However, It is not stopped.

## 11.2 Configuration of the Watchdog Timer

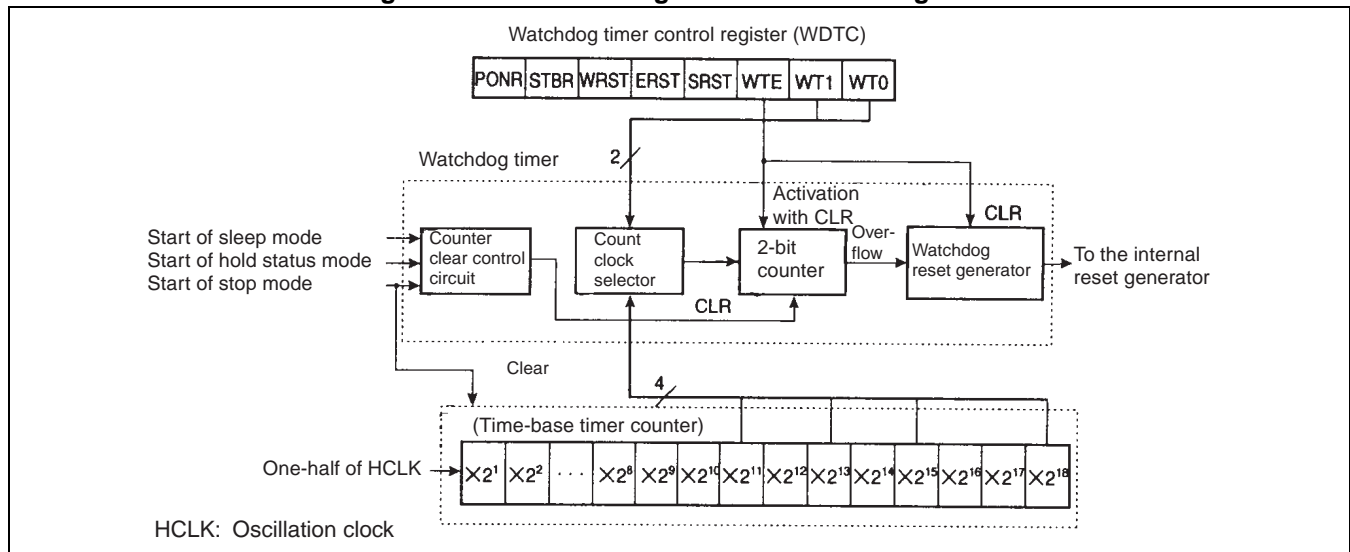
The watchdog timer consists of the following five blocks:

- Count clock selector
- Watchdog counter (2-bit counter)
- Watchdog reset generator
- Counter clear control circuit
- Watchdog timer control register (WDTC)

### ■ Block Diagram of the Watchdog Timer

Figure 11.2-1 shows the block diagram of the watchdog timer.

**Figure 11.2-1 Block diagram of the watchdog timer**



#### ● Count clock selector

This circuit is used to select the count clock of the watchdog timer from four types of time-base timer outputs. This determines the watchdog reset generation time.

#### ● Watchdog counter (2-bit counter)

This 2-bit up counter uses the time-base timer output as the count clock.

#### ● Watchdog reset generator

Used to generate the reset signal by an overflow of the watchdog counter.

#### ● Counter clear circuit

Used to clear the watchdog counter and to control the operation or stopping of the counter.

#### ● Watchdog timer control register (WDTC)

Used to activate or clear the watchdog timer; holds the reset generation cause.



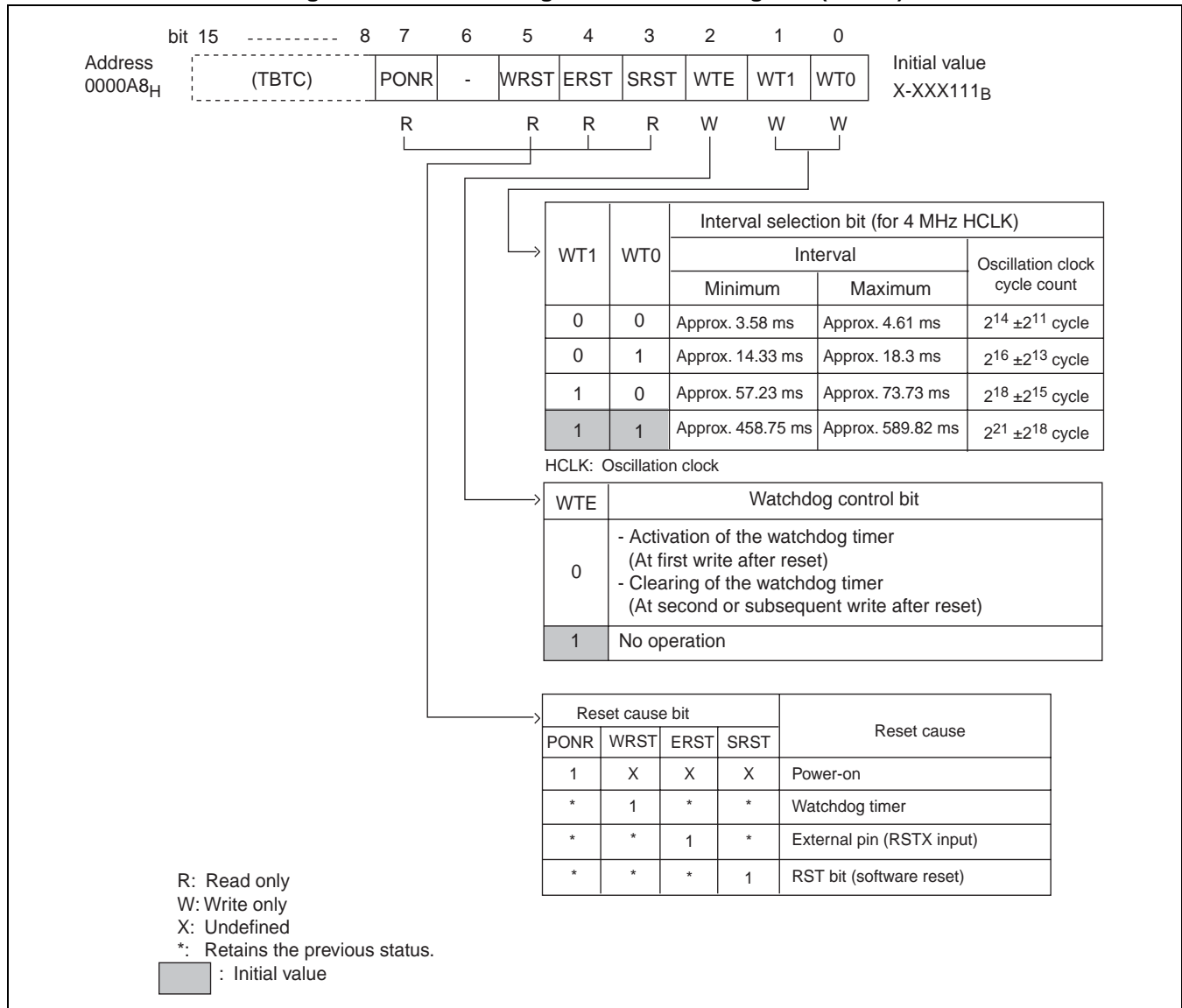
## 11.3 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) activates and clears the watchdog timer, and displays the reset cause.

### ■ Watchdog Timer Control Register (WDTC)

Figure 11.3-1 shows the watchdog timer control register (WDTC). Table 11.3-1 describes the function of each bit of the watchdog timer control register (WDTC).

**Figure 11.3-1 Watchdog Timer Control Register (WDTC)**



The interval becomes 3.5 to 4.5 times longer than the count clock (time-base timer output value) cycle. For details, see "11.4 Operation of the Watchdog Timer".

**Table 11.3-1 Function Description of Each bit of the Watchdog Timer Control Register (WDTC)**

Bit name		Function
bit7, bit5 to bit3	PONR, WRST, ERST, SRST: Reset cause bits	<ul style="list-style-type: none"> <li>Read-only bits for indicating the reset cause. If more than one reset cause occurs, the bit for each reset cause occurring is set to "1".</li> <li>These bits are all cleared to "0" after the watchdog timer control register (WDTC) is read.</li> <li>At power-on, the contents of the bits other than the PONR bit are not guaranteed. Therefore, when the PONR bit is "1", ignore the contents of the bits other than the PONR bit.</li> </ul>
bit2	WTE: Watchdog timer control bit	<ul style="list-style-type: none"> <li>When "0" is written to this bit, the watchdog timer is activated (first write after reset) or the 2-bit counter is cleared (second or subsequent write after reset).</li> <li>Writing "1" does not affect operation.</li> </ul>
bit1, bit0	WT1, WT0: Interval selection bit	<ul style="list-style-type: none"> <li>Used to select the watchdog timer interval.</li> <li>Only data at watchdog timer activation is valid. Data written after watchdog timer activation is ignored.</li> <li>These bits are write-only.</li> </ul>

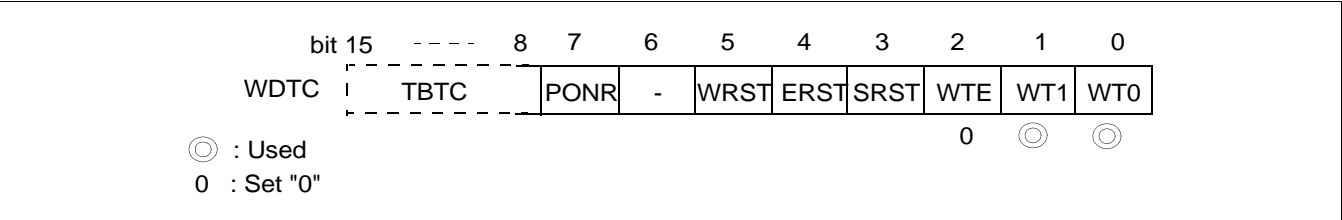
# 11.4 Operation of the Watchdog Timer

The watchdog timer generates a watchdog reset by an overflow of the watchdog counter.

## ■ Watchdog Timer Operation

Operation of the watchdog timer requires the setting in Figure 11.4-1.

Figure 11.4-1 Setting of the Watchdog Timer



### ● Activating the watchdog timer

- The watchdog timer is activated when the first 0 after reset is written to the WTE bit of the watchdog timer control register (WDTC). Specify the interval by specifying the WT1 and WT0 bits of the watchdog timer control register at the same time.
- When watchdog timer activation starts, it can be stopped only by a power-on or its own reset.

### ● Clearing the watchdog timer

- When a second or subsequent "0" is written to the WTE bit, the 2-bit counter of the watchdog timer is cleared. If the counter is not cleared within the time interval, it overflows and a watchdog reset occurs.
- The watchdog counter is cleared by reset generation, sleep mode or stop mode, transition to clock mode.

### ● Intervals for the watchdog timer

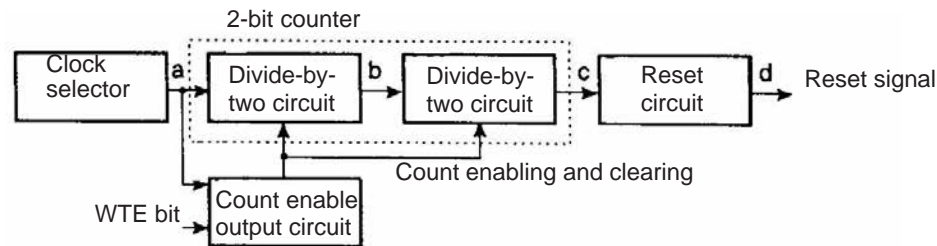
Figure 11.4-2 shows the relationship between the clear timing of the watchdog timer and intervals. The interval changes according to the clear timing of the watchdog timer and requires 3.5 to 4.5 times longer than the count clock cycle.

### ● Checking a reset cause

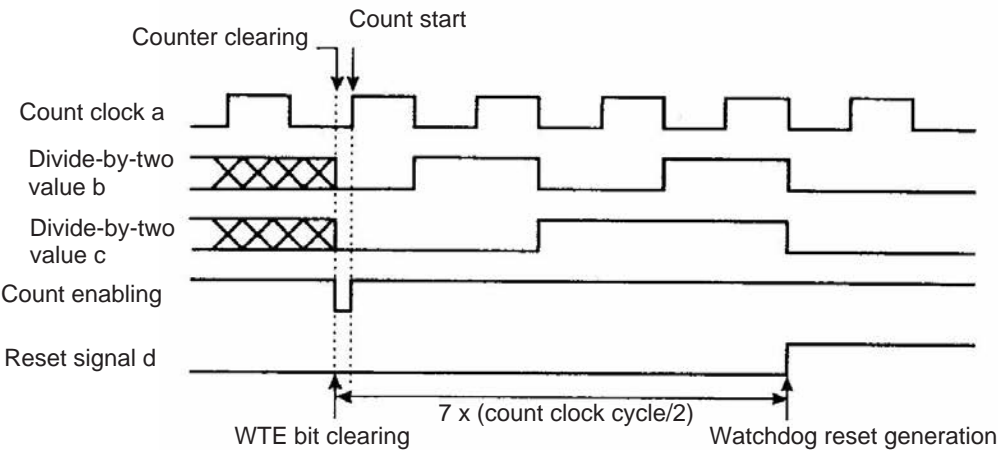
A reset cause can be determined by checking the PONR, WRST, ERST and SRST bits of the watchdog timer control register (WDTC) after a reset.

Figure 11.4-2 Clear Timing and Watchdog Timer Intervals

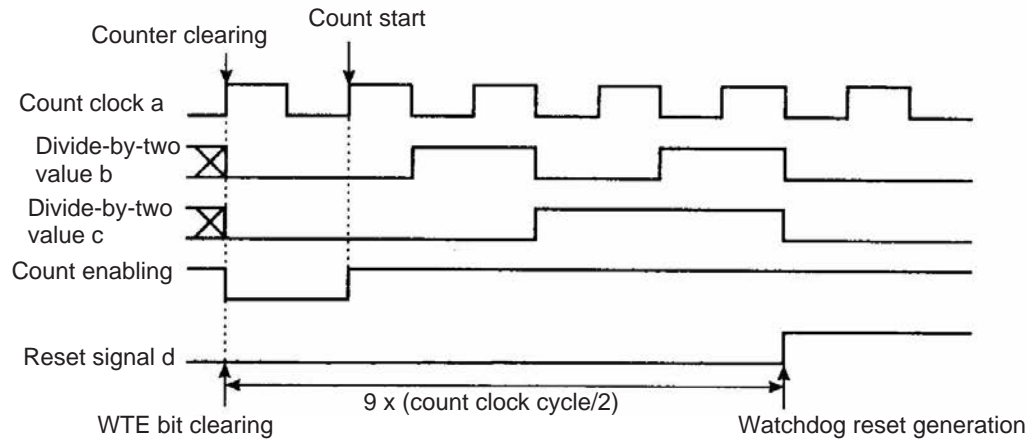
[WDG timer block diagram]



[Minimum interval] When the WTE bit is cleared immediately before the count clock rises:



[Maximum interval] When the WTE bit is cleared immediately after the count clock rises:



## 11.5 Usage Notes on the Watchdog Timer

---

Notes on using the watchdog timer are given below.

---

### ■ Usage Notes on the Watchdog Timer

- Stopping the watchdog timer

Once the watchdog timer is activated, it cannot stop until a power-on or watchdog reset occurs. The watchdog timer counter is cleared by an external reset or software reset; however, the watchdog timer does not stop.

- Intervals

Since a carry signal of the time-base timer is used as the count clock for the interval, the watchdog timer interval may become longer than the setting time when the time-base timer is cleared.

- Selecting the interval

The interval can be set when the watchdog timer is activated. Data written during operations other than activation is ignored.

- Notes on program creation

When a program that repeatedly clears the watchdog timer in the main loop is created, the processing time of the main loop including the interrupt processing must be equal to or less than the minimum watchdog timer interval.

- Watchdog timer operation in time-base timer mode

The time-base timer operates while the time-base timer mode is set. The watchdog timer, however, is temporarily stopped.

## 11.6 Sample Program for the Watchdog Timer

---

This section contains a sample program for the watchdog timer.

---

### ■ Sample Program for the Watchdog Timer

#### ● Processing

- The watchdog timer is cleared every time in the main program loop.
- The main loop must make one iteration within the minimum watchdog timer interval.

#### ● Coding example

```

WDTC      EQU      0000A8H      ;Watchdog timer control register
WTE       EQU      WDTC:2      ;Watchdog control bit
;-----Main program-----
CODE      CSEG
START:
;          :                      ;Assumes that stack pointer (SP) has already
;                               ;been initialized

WDG_START:
          MOV      WDTC,#00000011B ;Activates watchdog timer
;                               ;Selects the interval  $2^{21} \pm 2^{18}$  cycle

;-----Main loop-----
MAIN:     CLRB     I:WTE          ;Clears watchdog timer
;          :                      ;Clears this bit regularly
;          :                      ;User processing
;          :
          JMP      MAIN          ;Loops in less time than the watchdog
;                               ;timer interval

CODE      ENDS

;-----Vector setting-----
VECT      CSEG      ABS=0FFH
          ORG      0FFDCH        ;Sets reset vector
          DSL      START
          DB       00H          ;Sets single-chip mode
VECT      ENDS
          END      START

```



# **CHAPTER 12**

---

## ***16-BIT RELOAD TIMER***

**The chapter describes the functions and operation of the 16-bit reload timer.**

- 12.1 Overview of the 16-bit Reload Timer
- 12.2 Block Diagram of the 16-bit Reload Timer
- 12.3 16-bit Reload Timer Pins
- 12.4 16-bit Reload Timer Registers
- 12.5 16-Bit Reload Timer Interrupts
- 12.6 Operation of the 16-bit Reload Timer
- 12.7 Usage Notes on the 16-bit Reload Timer
- 12.8 Sample Programs for the 16-bit Reload Timer



## 12.1 Overview of the 16-bit Reload Timer

The 16-bit reload timer functions in the two modes shown below, either of which can be selected. It is synchronized with three types of internal clocks for counting down in internal clock mode, and it counts down by detecting an optional edge input to the external pin in event counter mode.

This timer defines that an underflow condition results when the counter value changes from 0000<sub>H</sub> to FFFF<sub>H</sub>. Thus, an underflow will occur after an interval of [reload register setting value + 1] counts.

Either of two modes can be selected for counting. In load mode, the value set for the count is reloaded to repeat counting for an underflow. In single-shot mode, the counting is stopped with an underflow.

The counter underflow allows the occurrence of an interrupt and coordination with the extended intelligent I/O service (EI<sup>2</sup>OS).

The MB90460/465 series 16-bit reload timer has two built-in channels.

### ■ Overview of the 16-bit Reload Timer

#### ● 16-bit reload timer operating modes

Table 12.1-1 lists the operating modes of the 16-bit reload timer.

**Table 12.1-1 16-bit Reload Timer Operating Modes**

Clock mode	Counter operation	Operation of 16-bit reload timer
Internal clock mode	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

Internal clock mode allows selection of one of three types of internal clock for the following operations:

#### ● Software trigger operation

When "1" is written to the TRG bit of the timer control status register (TMCSRL0/TMCSRL1), counting starts. Trigger input by the TRG bit is always valid for external trigger input as well as for external gate input.

#### ● External trigger operation

When selected edges (rising, falling, and both edges) are input to the TIN0 and TIN1 pins, counting starts in channel 0 and 1 respectively.

#### ● External gate input operation

While the selected signal level (L or H) is being input to TIN0 and TIN1 pins, counting continues in channel 0 and 1 respectively.

## ■ Event Count Mode (External Clock Mode)

When selected valid edges (rising, falling and both edges) are input to TIN0 and TIN1 pins, the timer counts down at these edges in channel 0 and 1 respectively.

When an external clock with a constant period is used, this timer can also be used as an interval timer.

## ■ Counter Operation

### ● Reload mode

When an underflow (from 0000<sub>H</sub> to FFFF<sub>H</sub>) occurs during counting down, the count setting value is reloaded to continue counting. Since the 16-bit reload timer causes an interrupt request to occur for an underflow condition, it can be used as an interval timer.

A toggle waveform inverted at each underflow can also be output from TO0 and TO1 pins.

Table 12.1-2 lists the intervals for the 16-bit reload timer.

**Table 12.1-2 Intervals for the 16-bit Reload Timer**

Count clock	Count clock period	Interval
Internal clock	$2^1/\phi$ (0.125 $\mu$ s)	0.125 $\mu$ s to 8.192 ms
	$2^3/\phi$ (0.5 $\mu$ s)	0.5 $\mu$ s to 32.768 ms
	$2^5/\phi$ (2.0 $\mu$ s)	2.0 $\mu$ s to 131.1 ms
External clock	$2^3/\phi$ or more (0.5 $\mu$ s)	0.5 $\mu$ s or more

$\phi$ : Machine clock

Values in parentheses are for a 16 MHz machine clock.

### ● Single-shot mode

When an underflow (from 0000<sub>H</sub> to FFFF<sub>H</sub>) occurs during counting down, counting stops, causing an interrupt to occur for the underflow condition.

During counter operation, a rectangular waveform that indicates when the count is in progress can be output from the TO0 and TO1 pins.

---

### References:

- 16-bit reload timer 0 can be used to create the baud rate of UART0 or enable data transfer in waveform sequencer.
  - 16-bit reload timer 1 can be used as the start trigger of the A/D converter or the baud rate of UART1.
-

## ■ 16-bit Reload Timer Interrupts and EI<sup>2</sup>OS

Table 12.1-3 lists 16-bit reload timer interrupts and EI<sup>2</sup>OS.

**Table 12.1-3 16-bit Reload Timer Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
16-bit reload timer 0 <sup>*1</sup>	#30 (1E <sub>H</sub> )	ICR09	0000B9 <sub>H</sub>	FFFF84 <sub>H</sub>	FFFF85 <sub>H</sub>	FFFF86 <sub>H</sub>	O
16-bit reload timer 1 <sup>*2</sup>	#18 (12 <sub>H</sub> )	ICR03	0000BA <sub>H</sub>	FFFFB4 <sub>H</sub>	FFFFB5 <sub>H</sub>	FFFFB6 <sub>H</sub>	

O: Can be used.

\*1: The same interrupt number as that for 16-bit reload timer 0 underflow is assigned to 16-bit timer 0/1/2 counter borrow.

\*2: The same interrupt number as that for 16-bit reload timer 1 underflow is assigned to 16-bit output compare channel 2.

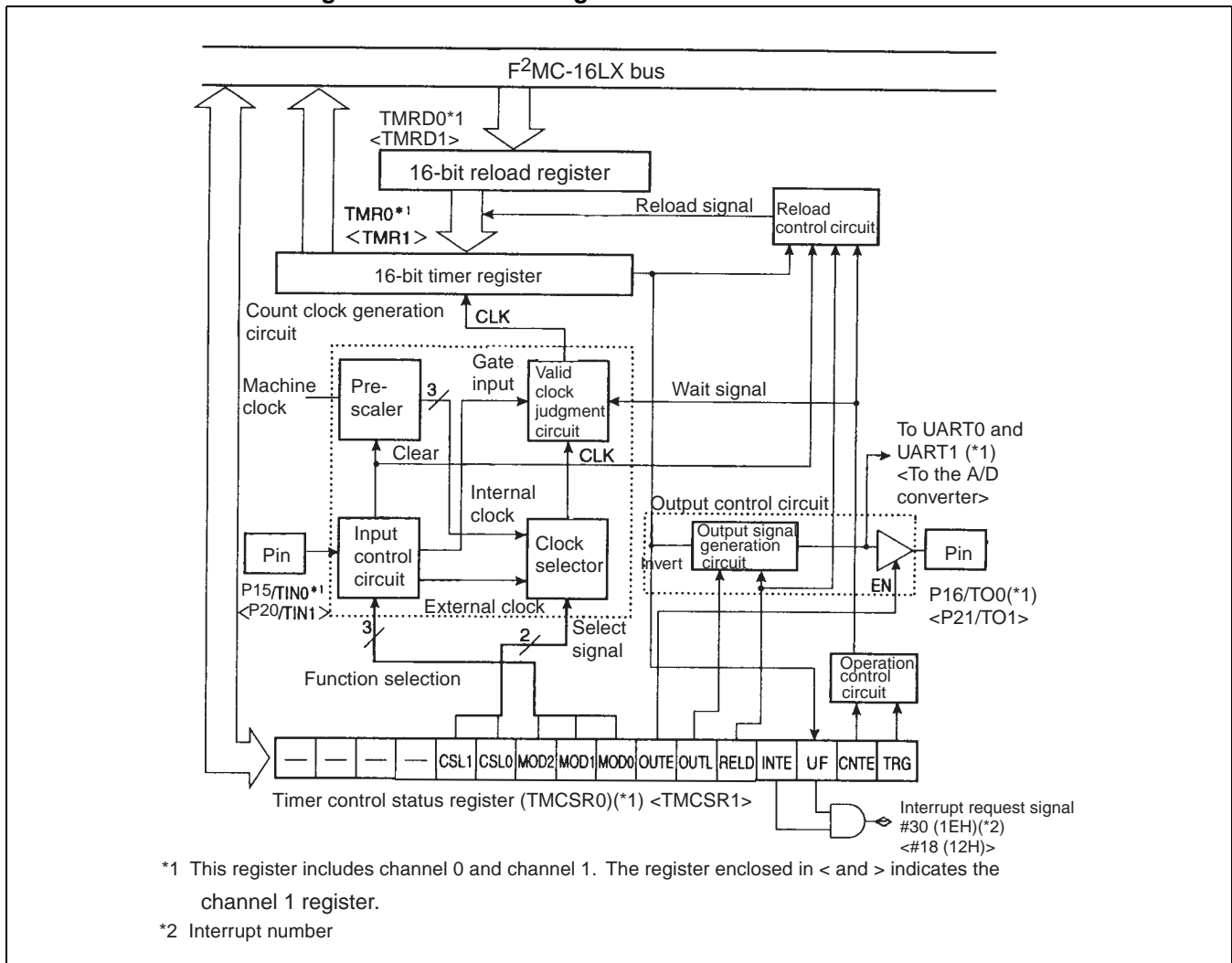
## 12.2 Block Diagram of the 16-bit Reload Timer

There are two 16-bit reload timers in MB90460/465 series. Each of them consists of the seven blocks as shown in the block diagram below.

### ■ Block Diagram of the 16-bit Reload Timer

Figure 12.2-1 shows the block diagram of the 16-bit reload timer

Figure 12.2-1 Block Diagram of the 16-bit Reload Timer



#### ● Count clock generation circuit

This circuit generates the count clock for the 16-bit reload timer from the machine clock or external input clock.

#### ● Reload control circuit

This circuit controls the reload operation when the timer is started and when an underflow occurs.

- Output control circuit

This circuit controls the inversion of the TO pin output by an underflow of the 16-bit timer register and enabling and disabling of TO pin output.

- Operation control circuit

This circuit controls starting and stopping of the 16-bit reload timer.

- 16-bit timer register (TMR0/TMR1)

This register is a 16-bit down counter. The current counter value is read from this register during a read operation.

- 16-bit reload register (TMRD0/TMRD1)

The interval for the 16-bit reload timer is set in this register. The setting value of this register is loaded into the 16-bit timer register and decremented.

- Timer control status register (TMCSR0/TMCSR1)

This register selects the count clock of the 16-bit reload timer and the operating mode, sets operating conditions, starts a trigger with software, enables or disables counting, selects reload or single-shot mode and the pin output level, enables or disables timer output, controls interrupts, and controls the status.

## 12.3 16-bit Reload Timer Pins

This section describes the pins of the 16-bit reload timer and provides a pin block diagram.

### 16-bit Reload Timer Pins

The pins of the 16-bit reload timer are shared with the general-purpose ports. Table 12.3-1 lists the functions of the pins, I/O format, and settings required to use the 16-bit reload timer.

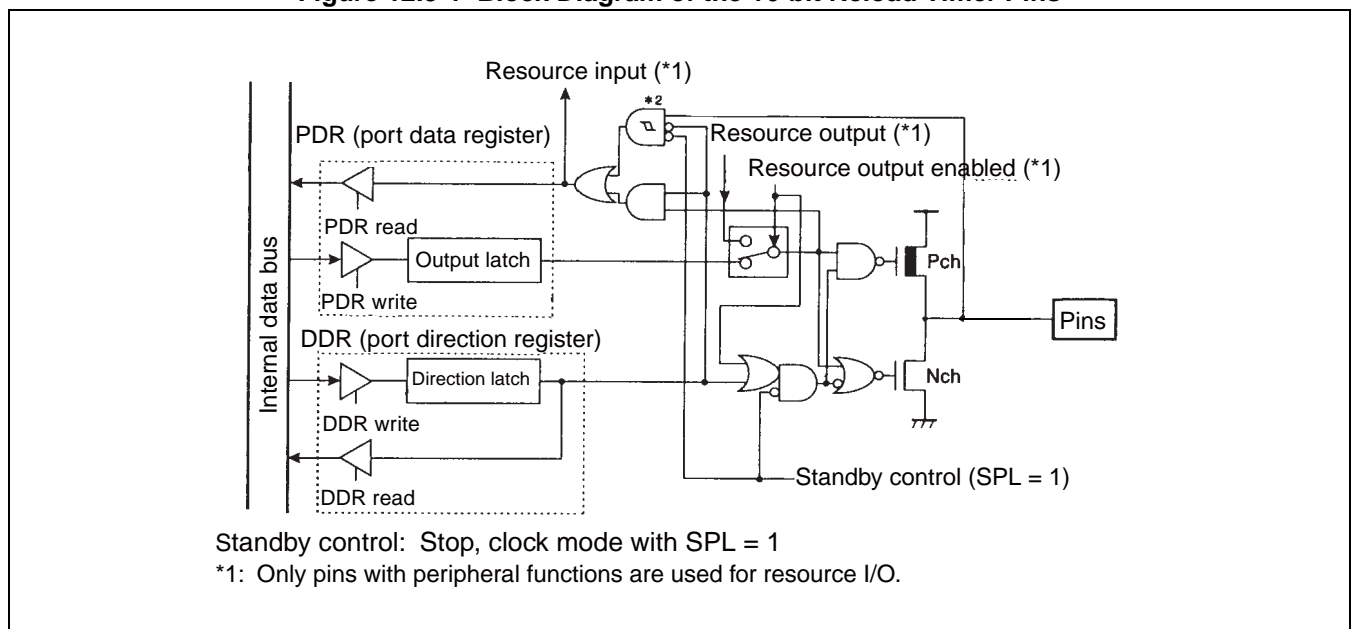
**Table 12.3-1 16-bit reload timer pins**

Pin name	Pin function	I/O format	Pull-up option	Standby control	Settings required for pins
P15/INT5/TIN0	Port 1 input-output / external interrupt input /timer input	CMOS output / CMOS hysteresis input	Selection allowed	Available	Setting for the input port (DDR1: bit15 = 0)
P16/INT6/TO0	Port 1 input-output / external interrupt input /timer output				Setting for timer output enable (TMCSRL0: OUTE = 1)
P20/TIN1	Port 2 input-output / timer input				Setting for the input port (DDR2: bit0 = 0)
P21/TO1	Port 2 input-output / timer output				Setting for timer output enable (TMCSRL1: OUTE = 1)

### Block Diagram of the 16-bit Reload Timer Pins

Figure 12.3-1 shows the block diagram of the 16-bit reload timer pins.

**Figure 12.3-1 Block Diagram of the 16-bit Reload Timer Pins**



## 12.4 16-bit Reload Timer Registers

The 16-bit reload timer registers are as follows.

### ■ 16-bit Reload Timer Registers

Figure 12.4-1 shows 16-bit reload timer registers.

**Figure 12.4-1 16-bit Reload Timer Registers**

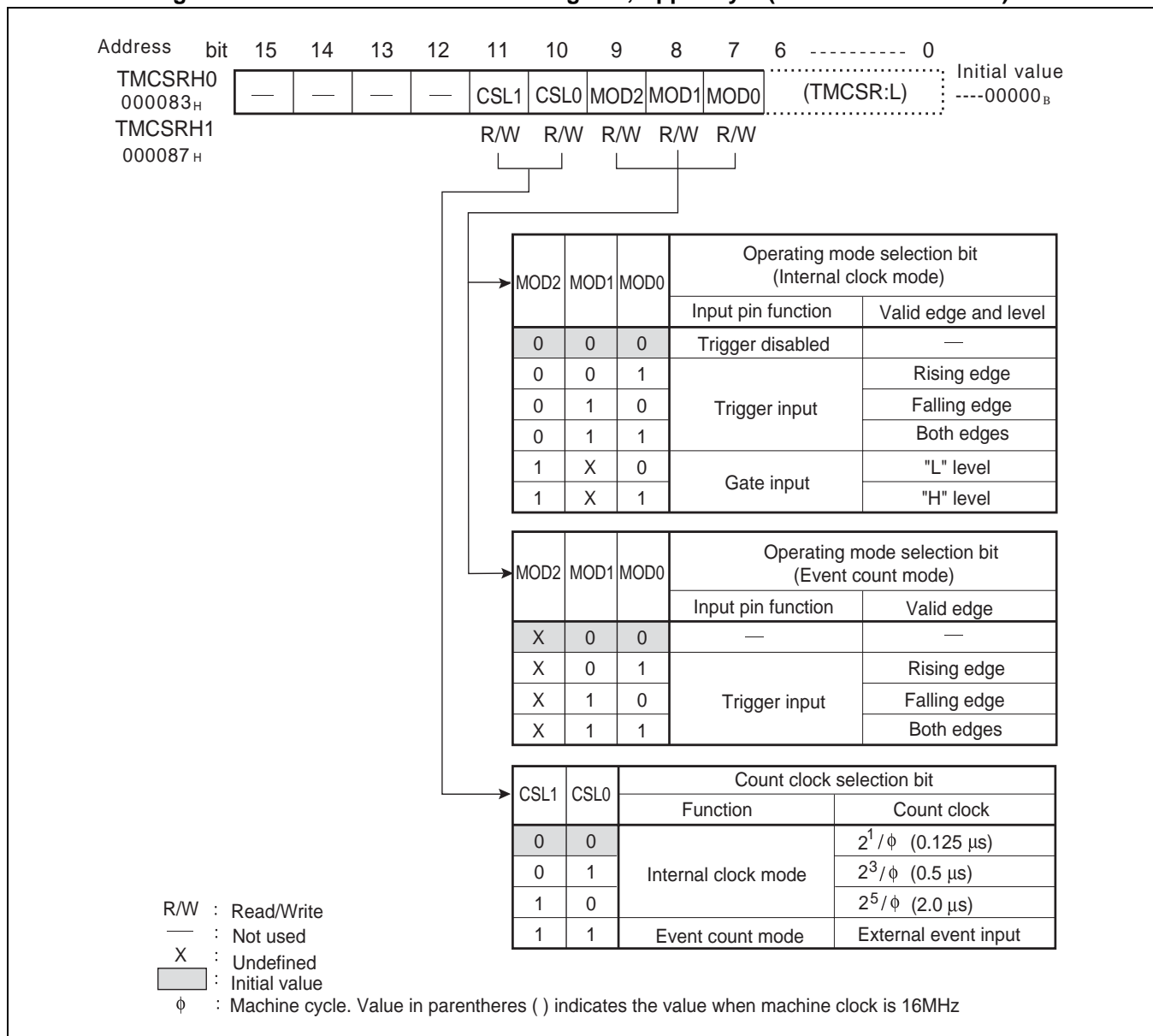
Timer Control Status Register (Upper)									
	bit	15	14	13	12	11	10	9	8
Address: ch.0 000083 <sub>H</sub>		—	—	—	—	CSL1	CSL0	MOD2	MOD1
ch.1 000087 <sub>H</sub>		—	—	—	—	CSL1	CSL0	MOD2	MOD1
Read/write ⇨		—	—	—	—	R/W	R/W	R/W	R/W
Initial value ⇨		—	—	—	—	0	0	0	0
Timer Control Status Register (Lower)									
	bit	7	6	5	4	3	2	1	0
Address: ch.0 000082 <sub>H</sub>		MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
ch.1 000086 <sub>H</sub>		MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
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
16-bit Timer Register / 16-bit Reload Register (Upper)									
	bit	15	14	13	12	11	10	9	8
Address: ch.0 000085 <sub>H</sub>		D15	D14	D13	D12	D11	D10	D09	D08
ch.1000089 <sub>H</sub>		D15	D14	D13	D12	D11	D10	D09	D08
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value ⇨		X	X	X	X	X	X	X	X
16-bit Timer Register / 16-bit Reload Register (Lower)									
	bit	7	6	5	4	3	2	1	0
Address: ch.0 000084 <sub>H</sub>		D07	D06	D05	D04	D03	D02	D01	D00
ch.1000088 <sub>H</sub>		D07	D06	D05	D04	D03	D02	D01	D00
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value ⇨		X	X	X	X	X	X	X	X

## 12.4.1 Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1)

High-order bit11 to bit8 and low-order bit7 of the timer control status registers (TMCSR0/TMCSR1) are used to select the operating mode of the 16-bit reload timer and set the operating conditions. This section describes low-order bit7: the MOD0 bit.

### ■ Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1)

Figure 12.4-2 Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1)





**Table 12.4-1 Timer Control Status Register (TMCSRH0/TMCSRH1)**

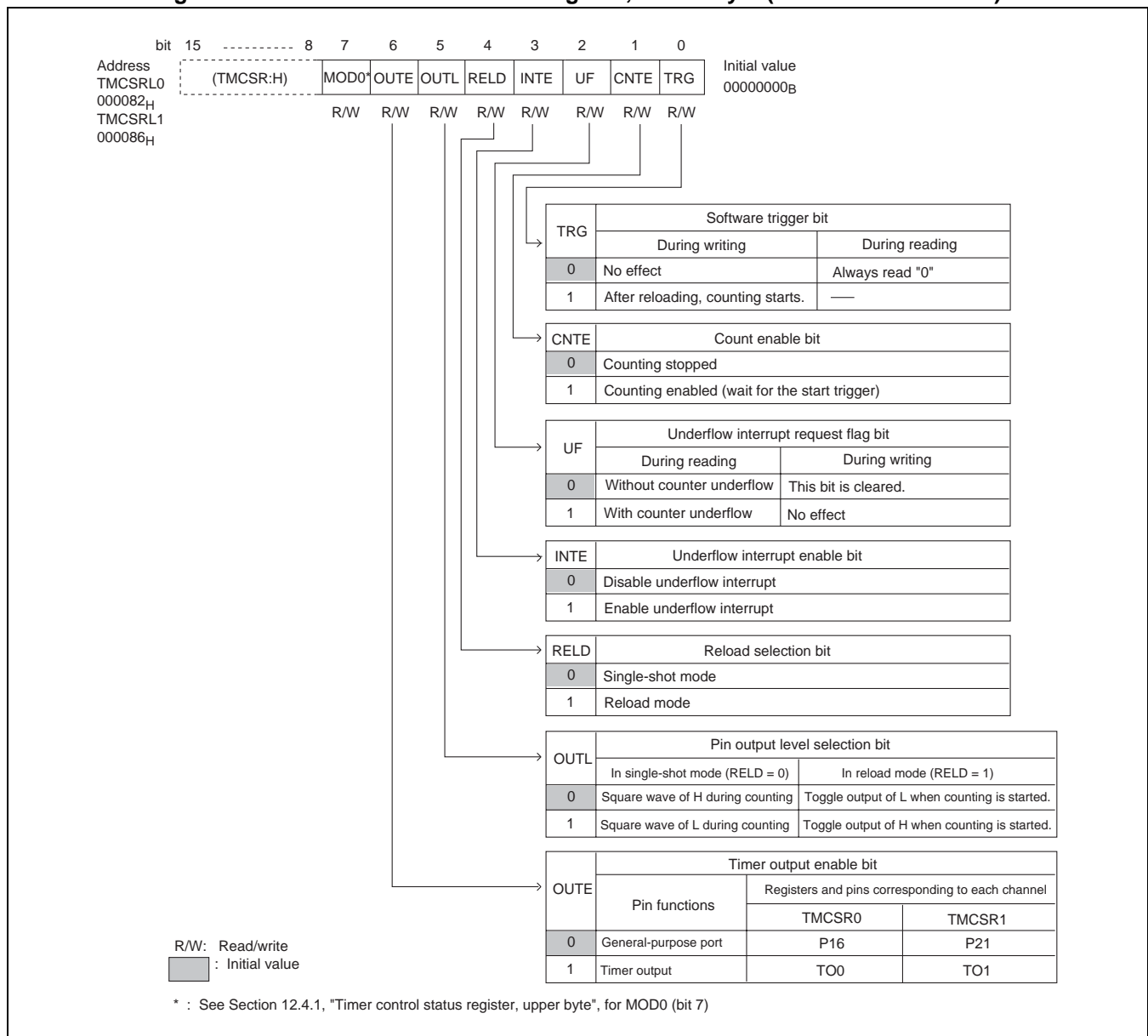
Bit name		Function
bit15 to bit12	Not used	<ul style="list-style-type: none"> <li>When these bits are read, their values are undefined.</li> <li>Writing to these bits has no effect on operation.</li> </ul>
bit11, bit10	CSL1, CSL0: Count clock selection bits	<ul style="list-style-type: none"> <li>Selects the count clock.</li> <li>When these bits are not set to "11<sub>B</sub>", internal clock mode is set. The internal clock is counted in this mode.</li> <li>When these bits are set to "11<sub>B</sub>", event count mode is set. The external clock edges are counted in this mode.</li> </ul>
bit9 to bit7	MOD2 to MOD0: Operating mode selection bits	<p>&lt;In internal clock mode&gt;</p> <ul style="list-style-type: none"> <li>The MOD2 bit is used to select input pin functions.</li> <li>When the MOD2 bit is "0", the input pin is used as a trigger input pin, so that whenever a valid edge is input, the contents of the reload register are loaded into the counter and counting continues. The MOD1 and MOD0 bits are used to select the type of valid edge.</li> <li>When the MOD2 bit is "1", the input pin becomes a gate, meaning that counting will continue only as long as a valid level signal is input. The MOD1 bit is an unused bit, and the MOD0 bit is used to select the valid level.</li> </ul> <p>&lt;In event count mode&gt;</p> <ul style="list-style-type: none"> <li>The MOD2 bit is not used. Set any value ("0" or "1").</li> <li>The input pin is used as a trigger input pin for event input, and the valid edge is selected with MOD1 and MOD0 bits.</li> </ul>

## 12.4.2 Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1)

The lower seven bits of the timer control status registers (TMCSR0/TMCSR1) are used to set operating conditions for the 16-bit reload timer, enable and disable operation, control interrupts, and check the status.

### ■ Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1)

Figure 12.4-3 Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1)



**Table 12.4-2 Timer Control Status Register (TMCSRL0/TMCSRL1)**

Bit name		Function
bit6	OUTE: Timer output enable bit	<ul style="list-style-type: none"> <li>• This bit enables or disables output from the timer output pin.</li> <li>• When this bit is "0", the pin functions as a general-purpose port. When this bit is "1", the pin functions as a timer output pin.</li> <li>• In reload mode, the output waveform of this timer output pin toggles. In single-shot mode, the timer outputs a rectangular waveform that indicates that counting is in progress is output.</li> </ul>
bit5	OUTL: Pin output level selection bit	<ul style="list-style-type: none"> <li>• This register is used to select the output level of the timer output pin.</li> <li>• The output level of the pin is inverted depending on whether this bit is "0" or "1".</li> </ul>
bit4	RELD: Reload selection bit	<ul style="list-style-type: none"> <li>• This bit enables reloading.</li> <li>• When this bit is "1", the timer is in reload mode. At the same time an underflow occurs, the contents of the reload register are loaded into the counter, and counting continues.</li> <li>• When this bit is "0", the timer is in single-shot mode. Counting stops when an underflow occurs.</li> </ul>
bit3	INTE: Interrupt request enable bit	<ul style="list-style-type: none"> <li>• This bit enables or disables underflow interrupt request to the CPU.</li> <li>• When this bit and the underflow interrupt request flag (UF) bit are "1", the timer outputs an interrupt request.</li> </ul>
bit2	UF: Underflow interrupt request flag bit	<ul style="list-style-type: none"> <li>• This bit is set to "1" when a counter underflow occurs.</li> <li>• Writing "0" to this bit clears it. Writing "1" to this bit does not change the bit value and has no effect on other bits.</li> <li>• This bit is also cleared when EI<sup>2</sup>OS is activated.</li> </ul>
bit1	CNTE: Count enable bit	<ul style="list-style-type: none"> <li>• The bit enables or disables counting.</li> <li>• When this bit is set to "1", the counter is placed in trigger standby mode. When a trigger occurs, actual counting starts.</li> </ul>
bit0	TRG: Software trigger bit	<ul style="list-style-type: none"> <li>• This bit starts the interval timer function or counter function with software.</li> <li>• Writing "1" to this bit applies a software trigger, causing the contents of the reload register to be loaded into the counter and starting counter operation. Writing "0" to this bit has no effect.</li> <li>• Trigger input from this trigger is always valid when the CNTE bit is set to "1" regardless of the operating mode.</li> </ul>

### 12.4.3 16-bit Timer Register (TMR0/TMR1)

The 16-bit timer register (TMR0/TMR1) is always able to read the count value from the 16-bit down counter.

#### ■ 16-bit Timer Register (TMR0/TMR1)

Figure 12.4-4 shows the 16-bit timer registers (TMR0/TMR1).

**Figure 12.4-4 16-bit Timer Register (TMR0/TMR1)**

16-bit Timer Register (Upper)										
		bit	15	14	13	12	11	10	9	8
Address:	ch.0	000085 <sub>H</sub>	D15	D14	D13	D12	D11	D10	D09	D08
	ch.1	000089 <sub>H</sub>								
			TMR0/TMR1							
Read/write ⇨			R	R	R	R	R	R	R	R
Initial value ⇨			X	X	X	X	X	X	X	X
16-bit Timer Register (Lower)										
		bit	7	6	5	4	3	2	1	0
Address:	ch.0	000084 <sub>H</sub>	D07	D06	D05	D04	D03	D02	D01	D00
	ch.1	000088 <sub>H</sub>								
			TMR0/TMR1							
Read/write ⇨			R	R	R	R	R	R	R	R
Initial value ⇨			X	X	X	X	X	X	X	X

The 16-bit timer register is able to read the counter value from the 16-bit down counter.

When counting is enabled (TMCSRL0/TMCSRL1:CNTE = 1) to start counting, the value written to the 16-bit reload register is loaded into this register, and counting down starts. This register value is stored in the counter stop status (TMCSRL0/TMCSRL1:CNTE = 0).

#### Notes:

- This register is able to read the count value even during counting. It should always be read with a word transfer instruction (MOVW A, 003AH, etc.).
- Although the 16-bit timer register (TMR0/TMR1) is a read-only register, it is allocated to the same address as the address of the write-only 16-bit reload register (TMRD0/TMRD1). Accordingly, writing to this register has no effect on the TMR0/TMR1 value, although writing to TMRD0/TMRD1 is done.

## 12.4.4 16-bit Reload Register (TMRD0/TMRD1)

The 16-bit reload register (TMRD0/TMRD1) sets a reload value in the 16-bit down counter. The value written to this register is loaded into the down counter, and the value is counted down.

### ■ 16-bit Reload Register (TMRD0/TMRD1)

Figure 12.4-5 shows the 16-bit reload registers (TMRD0/TMRD1).

**Figure 12.4-5 16-bit Reload Register (TMRD0/TMRD1)**

16-bit Reload Register (Upper)										
bit	15	14	13	12	11	10	9	8		
Address: ch.0	000085 <sub>H</sub>								TMRD0/TMRD1	
ch.1	000089 <sub>H</sub>	D15	D14	D13	D12	D11	D10	D09		D08
Read/write	⇒	W	W	W	W	W	W	W		W
Initial value	⇒	X	X	X	X	X	X	X		X
16-bit Reload Register (Lower)										
bit	7	6	5	4	3	2	1	0		
Address: ch.0	000084 <sub>H</sub>								TMRD0/TMRD1	
ch.1	000088 <sub>H</sub>	D07	D06	D05	D04	D03	D02	D01		D00
Read/write	⇒	W	W	W	W	W	W	W		W
Initial value	⇒	X	X	X	X	X	X	X		X

The initial value of the counter is set in this register when counting is disabled (TMCSRL0/TMCSRL1:CNTE = 0), regardless of the operating mode of the 16-bit reload timer. When counting is enabled (TMCSRL0/TMCSRL1:CNTE = 1) and the counter is started, counting down starts from the value written to this register.

In reload mode, the value set in the 16-bit reload register (TMRD0/TMRD1) is reloaded into the counter when an underflow occurs, and counting down continues. In single-shot mode, the counter stops at FFFF<sub>H</sub> when an underflow occurs.

#### Notes:

- Write a value to this register in the counter stop (TMCSRL0/TMCSRL1:CNTE = 0) state. Also, always use a word transfer instruction (MOVW 003AH, A etc.) to write a value to this register.
- Although the 16-bit reload register (TMRD0/TMRD1) is functionally a write-only register, it is allocated to the same address as the read-only 16-bit timer registers (TMR0/TMR1). Accordingly, since the read value is used as the TMR0/TMR1 value, the INC/DEC instruction and other instructions for read-modify-write (RMW) operations cannot be used.

## 12.5 16-Bit Reload Timer Interrupts

The 16-bit reload timer is enabled to generate an interrupt request in an underflow of the counter. It is also coordinated with the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ 16-bit Reload Timer Interrupts

Table 12.5-1 lists the interrupt control bits and interrupt causes of the 16-bit reload timer.

**Table 12.5-1 Interrupt Control Bits and Interrupt Causes of the 16-bit Reload Timer**

	16-bit reload timer 0	16-bit reload timer 1
Interrupt request flag bit	TMCSRL0: UF	TMCSRL1: UF
Interrupt request enable bit	TMCSRL0: INTE	TMCSRL1: INTE
Interrupt cause	Underflow of the 16-bit down counter (TMR0)	Underflow of the 16-bit down counter (TMR1)

In the 16-bit reload timer, the UF bit of the timer control status register (TMCSRL0/TMCSRL1) is set to "1" by an underflow (from 0000<sub>H</sub> to FFFF<sub>H</sub>) of the down counter. If an interrupt request is enabled (TMCSRL0/TMCSRL1:INTE = 1) in this operation, the interrupt request is output to the interrupt controller.

### ■ 16-bit Reload Timer Interrupts and EI<sup>2</sup>OS

Table 12.5-2 lists the 16-bit reload timer interrupts and EI<sup>2</sup>OS.

**Table 12.5-2 16-bit Reload Timer Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
16-bit reload timer 0 <sup>*1</sup>	#30 (1E <sub>H</sub> )	ICR09	0000B9 <sub>H</sub>	FFFF84 <sub>H</sub>	FFFF85 <sub>H</sub>	FFFF86 <sub>H</sub>	O
16-bit reload timer 1 <sup>*2</sup>	#18 (12 <sub>H</sub> )	ICR03	0000B3 <sub>H</sub>	FFFFB4 <sub>H</sub>	FFFFB5 <sub>H</sub>	FFFFB6 <sub>H</sub>	

\*1: The same interrupt number as that for waveform sequencer 16-bit timer counter borrow is assigned to 16-bit reload timer 0.

\*2: The same interrupt number as that for output compare ch 2 match is assigned to 16-bit reload timer 1.

### ■ EI<sup>2</sup>OS Function of the 16-bit Reload Timer

Since the 16-bit reload timer has a circuit that coordinates with EI<sup>2</sup>OS, the counter can start EI<sup>2</sup>OS when an underflow occurs.

However, EI<sup>2</sup>OS is available only when other peripheral functions sharing the interrupt control register (ICR) do not use interrupts. For example, when 16-bit reload timer 0 uses EI<sup>2</sup>OS, interrupts of the waveform generator 16-bit timer 0/1/2 counter borrow must be disabled.

# 12.6 Operation of the 16-bit Reload Timer

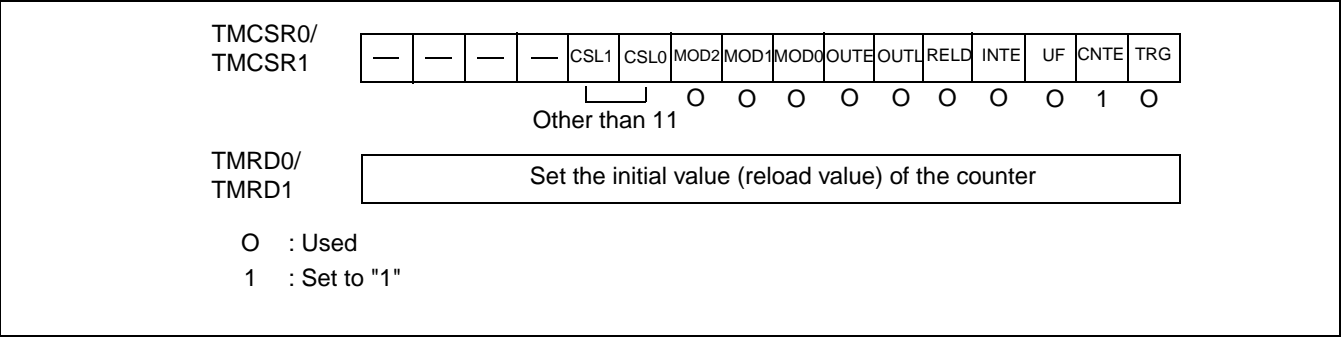
This section describes the 16-bit reload timer settings and counter operating status.

## 16-bit Reload Timer Settings

### Internal clock mode setting

The setting shown in Figure 12.6-1 is required to operate this timer as an interval timer.

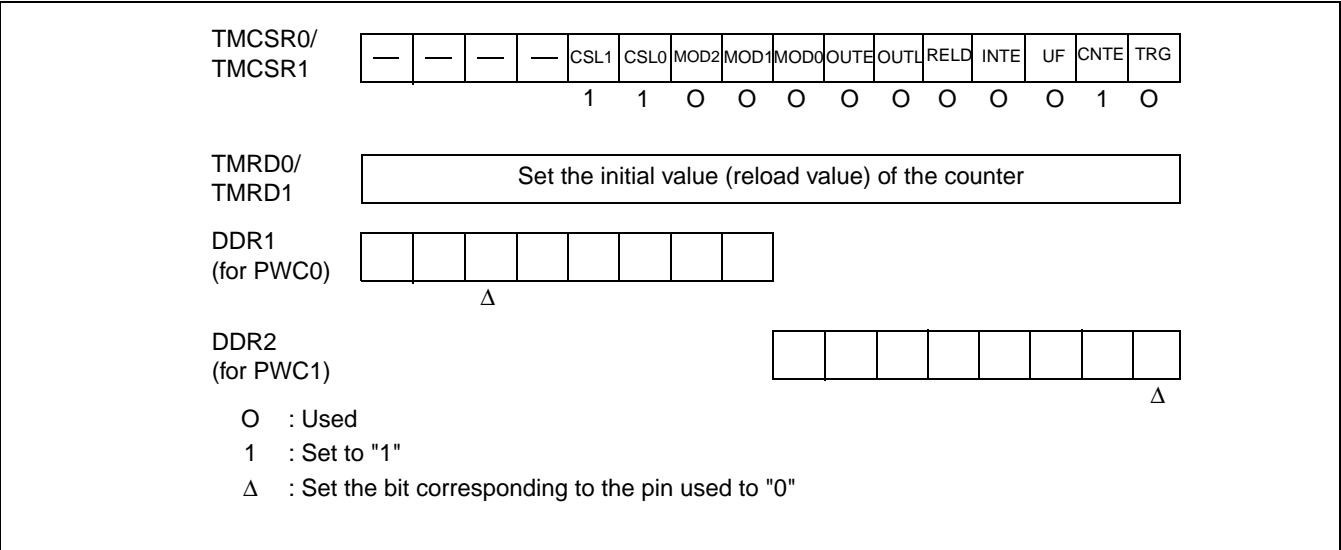
Figure 12.6-1 Internal Clock Mode Setting



### Event count mode setting

The setting shown in Figure 12.6-2 is required to operate this timer as an event counter.

Figure 12.6-2 Event Counter Mode Setting

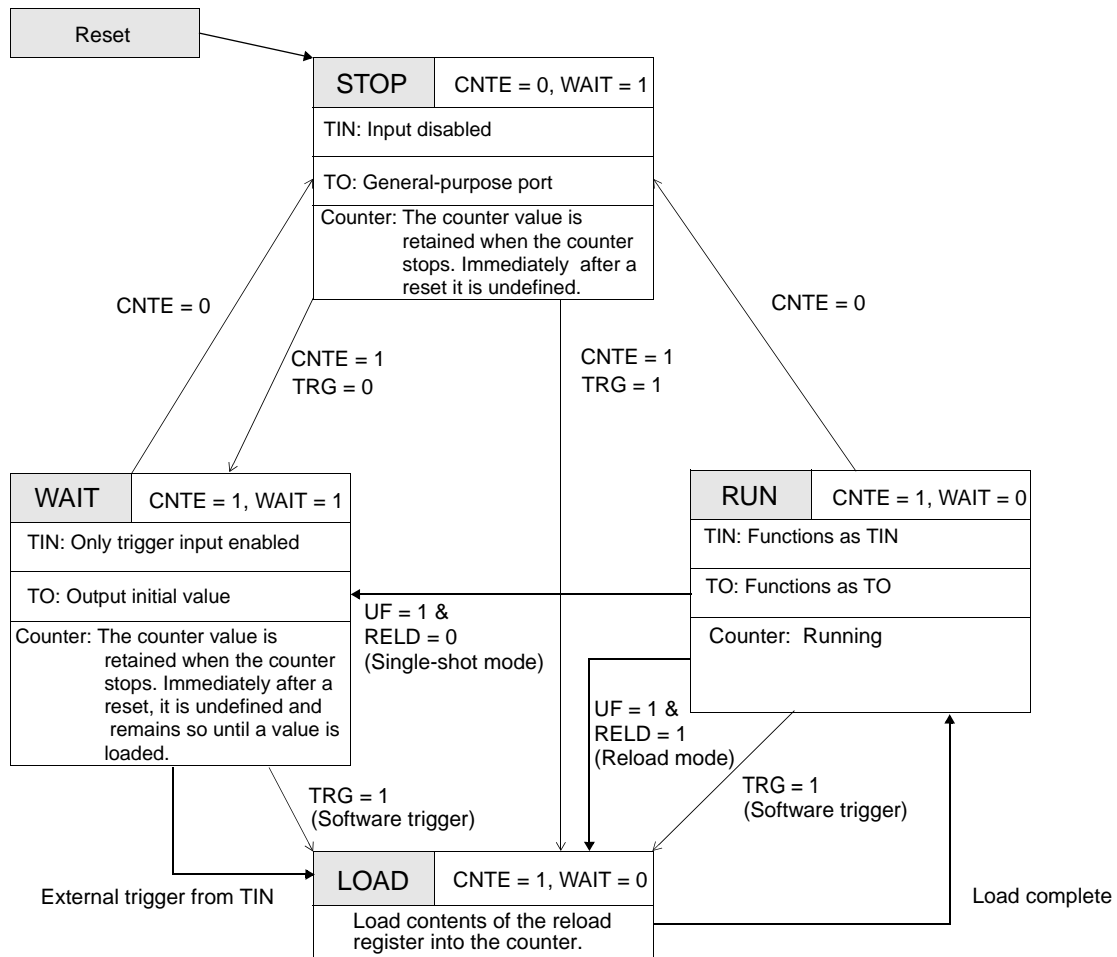


## Counter Operating State

The counter status is determined by the CNTE bit of the timer control status register (TMCSRL0/TMCSRL1) and the internal WAIT signal. Possible settings include the stop status (STOP state), trigger wait state (WAIT state), and running state (RUN state).

Figure 12.6-3 shows the transitions of the counter state.

**Figure 12.6-3 Counter State Transition**



- : State transitions by hardware  
 - - - - - : State transitions by register access  
 WAIT: Wait signal (internal signal)  
 TRG: Software trigger bit of timer control status register (TMCSRL0/TMCSRL1)  
 CNTE: Count enable bit of timer control status register (TMCSRL0/TMCSRL1)  
 UF: Underflow interrupt flag bit of timer control status register (TMCSRL0/TMCSRL1)  
 RELD: Reload selection bit of timer control status register (TMCSRL0/TMCSRL1)



## 12.6.1 Internal Clock Mode (reload mode)

Synchronized with the internal count clock, the 16-bit reload timer is used for counting down of the 16-bit counter, generating an interrupt request to the CPU for a counter underflow. It can also output a toggle waveform from the timer output pin.

### ■ Operation in Internal Clock Mode (reload mode)

When counting is enabled (TMCSRL0/TMCSRL1:CNTE = 1) and the timer is started by the software trigger bit (TMCSRL0/TMCSRL1:TRG) or external trigger, the value of the reload register (TMRD0/TMRD1) is loaded into the counter, and counting starts. If the count enable bit and software trigger bit are set to "1" simultaneously, counting starts simultaneously with the enabling of counting.

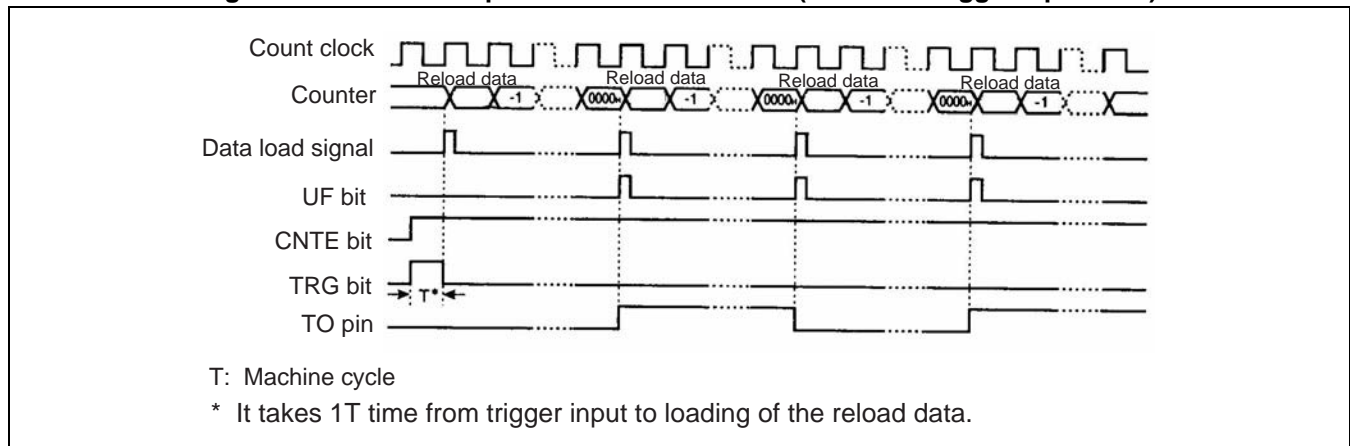
When an underflow of the counter value (from 0000<sub>H</sub> to FFFF<sub>H</sub>) occurs, the value of the 16-bit reload register (TMRD0/TMRD1) is loaded into the counter, and counting continues. If the underflow interrupt flag (TMCSRL0/TMCSRL1:UF) bit is set to "1" and the underflow interrupt enable (TMCSRL0/TMCSRL1:INTE) bit is "1", an interrupt request is generated.

The timer can also output from the TO0/TO1 pin a toggle waveform, which is inverted for each underflow.

#### ● Software trigger operation

When "1" is written to the TRG bit of the timer control status register (TMCSRL0/TMCSRL1), the counter is started. Figure 12.6-4 shows software trigger operation in reload mode.

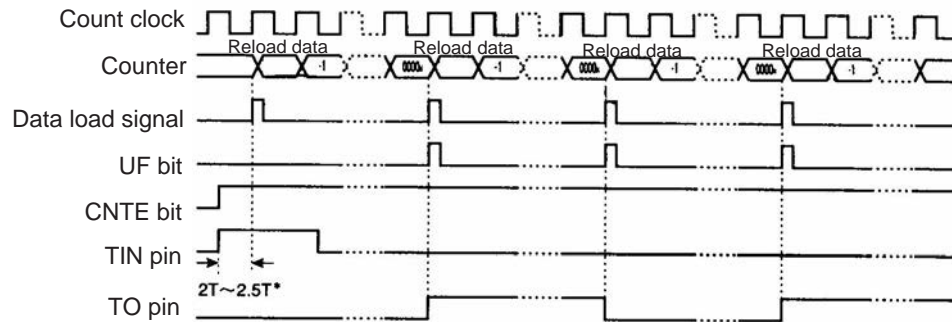
**Figure 12.6-4 Count Operation in Teload Mode (Software Trigger Operation)**



### ● External trigger operation

When a valid edge (rising, falling and both edges can be selected) is input to the TIN pin, the counter is started. Figure 12.6-5 shows external trigger operation in reload mode.

**Figure 12.6-5 Counting in Reload Mode (External Trigger Operation)**



T: Machine cycle

\* It takes  $2T$  to  $2.5T$  time from trigger input to loading of the reload data.

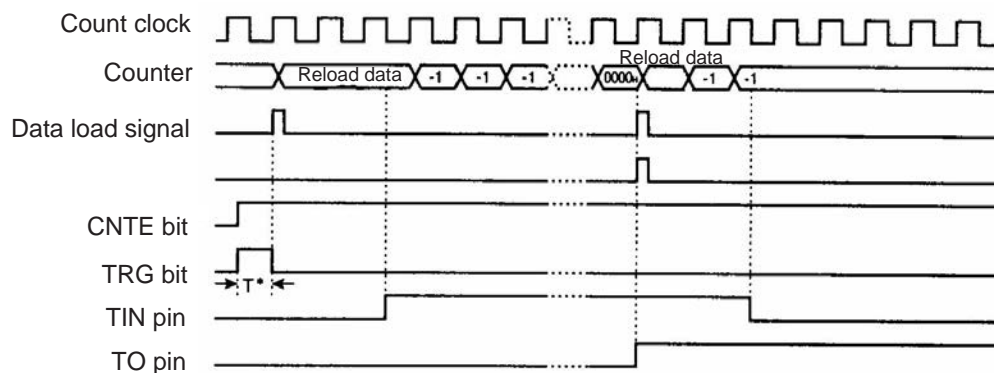
Note:

Specify  $2/\phi$  or more for the width of the trigger pulse input to the TIN0/TIN1 pin.

### ● Gate input operation

While a valid level (H and L levels can be selected) is input to the TIN0/TIN1 pin, counting is done. Figure 12.6-6 shows gate input in reload mode.

**Figure 12.6-6 Count Operation in Reload Mode (Software Trigger and Gate Input Operation)**



T: Machine cycle

\* It takes  $1T$  time from trigger input to loading of the reload data.

Note:

Specify  $2/\phi$  or more for the width of the trigger pulse input to the TIN0/TIN1 pin.

## 12.6.2 Internal Clock Mode (single-shot mode)

Synchronized with the internal count clock, the 16-bit reload timer is used for counting down of the 16-bit counter, generating an interrupt request to the CPU for a counter underflow. It can also output from the TO pin a rectangular waveform indicating that counting is in progress.

### ■ Internal Clock Mode (Single-shot Mode)

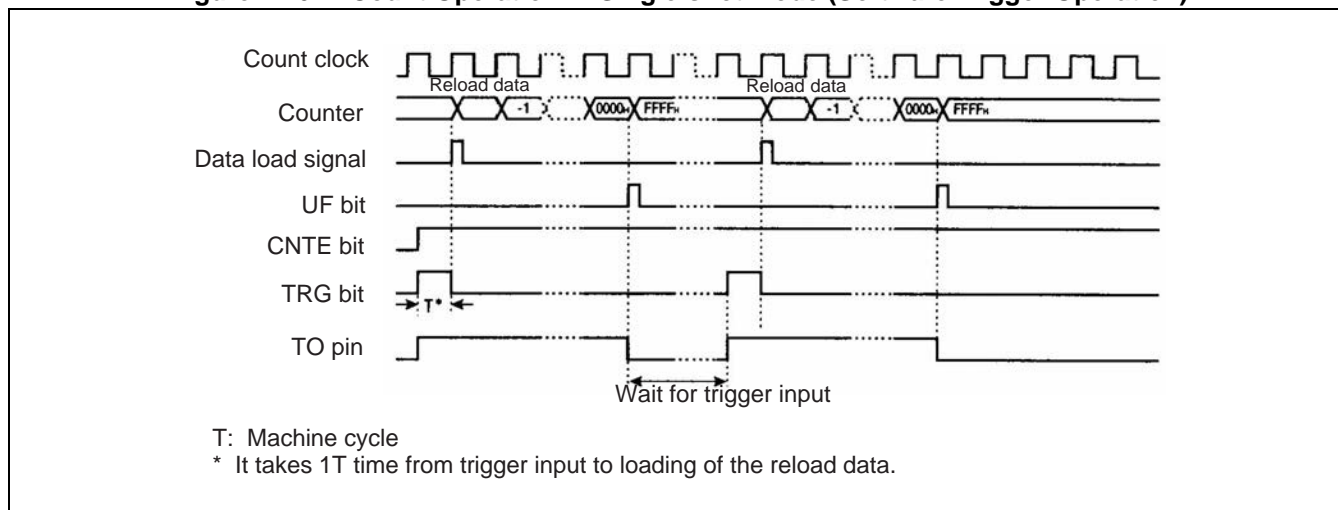
When counting is enabled (TMCSRL0/TMCSRL1:CNTE = 1) and the timer is started with the software trigger bit (TMCSRL0/TMCSRL1:TRG) or external trigger, counting starts. If the count enable bit and software trigger bit are set to "1" simultaneously, counting starts simultaneously with the enabling of counting. When an underflow of the counter value (from 0000<sub>H</sub> to FFFF<sub>H</sub>) occurs, the counter stops in the FFFF<sub>H</sub> state. If the underflow interrupt flag (TMCSRL0/TMCSRL1:UF) bit is set to "1" and the underflow interrupt enable (TMCSRL0/TMCSRL1:INTE) bit is "1", an interrupt request is generated.

The timer can also output from the TO0/TO1 pin a rectangular waveform indicating that counting is in progress.

#### ● Software trigger operation

When "1" is written to the TRG bit of the timer control status register (TMCSRL0/TMCSRL1), the counter is started. Figure 12.6-7 shows the software trigger operation in single-shot mode.

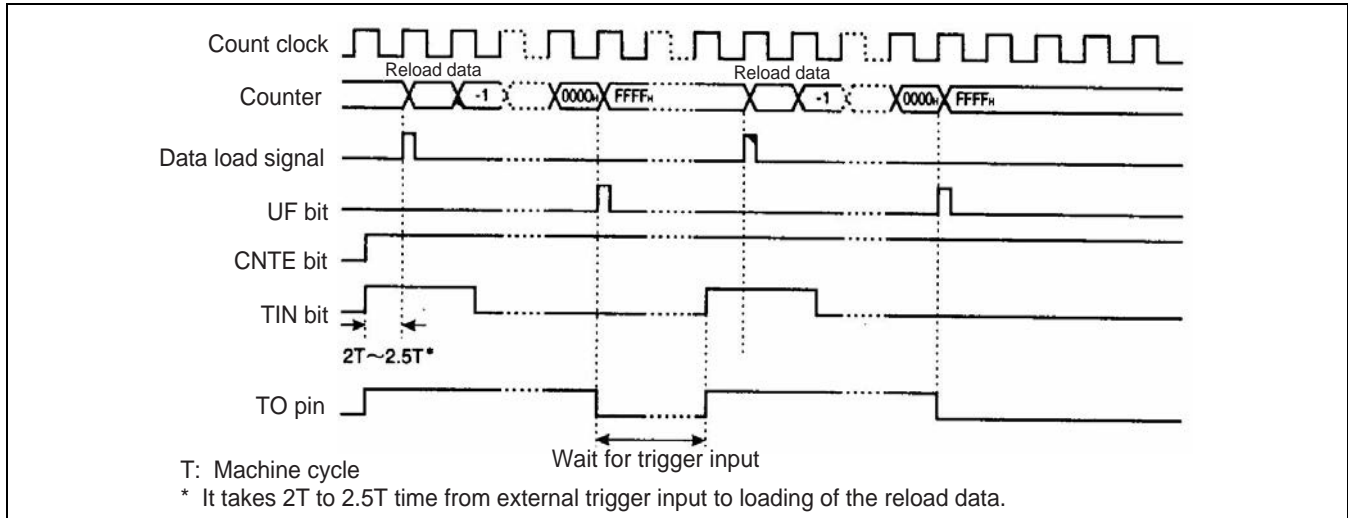
**Figure 12.6-7 Count Operation in Single-shot Mode (Software Trigger Operation)**



### ● External trigger operation

When a valid edge (rising, falling, and both edges can be selected) is input to the TIN0/TIN1 pin, the counter is started. Figure 12.6-8 shows external trigger operation in single-shot mode

**Figure 12.6-8 Specify 2/f or more for the Width of the Trigger Pulse Input to the TIN0/TIN1 Pin.**



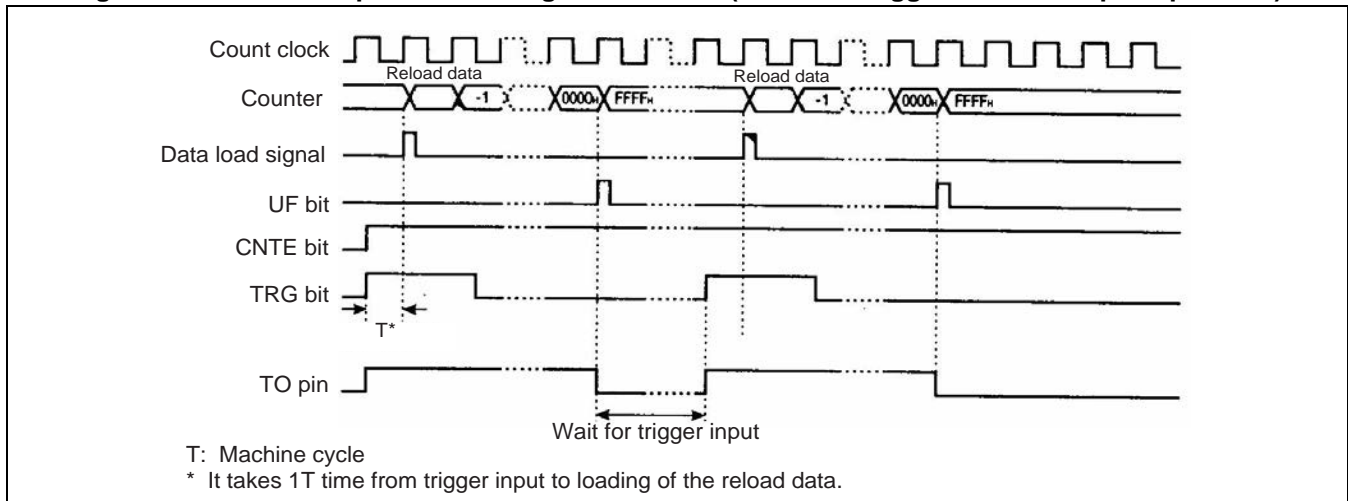
Note:

Specify 2/φ or more for the width of the trigger pulse input to the TIN0/TIN1 pin.

### ● Gate input operation

While a valid level (H and L levels can be selected) is input to the TIN0/TIN1 pin, counting is done. Figure 12.6-9 shows gate input operation in single-shot mode

**Figure 12.6-9 Count Operation in Single-shot Mode (Software Trigger and Gate Input Operation)**



Note:

Specify 2/φ or more for the width of the trigger pulse input to the TIN0/TIN1 pin.

### 12.6.3 Event Count Mode

The 16-bit reload timer counts an input edge from the TIN0/TIN1 pin, counts down the 16-bit counter, and generates an interrupt request to the CPU for a counter underflow. It can also output a toggle waveform or rectangular waveform from the TO0/TO1 pin.

#### ■ Event Count Mode

When counting is enabled (TMCSRL0/TMCSRL1:CNTE = 1) and the counter is started (TMCSRL0/TMCSRL1:TRG = 1), the value of the reload register is loaded into the counter. Counting down is then done every time a valid edge (rising, falling, and both edges can be selected) of the pulse input to the TIN pin (external count clock) is detected.

When the count enable bit and software trigger bit are set to "1" simultaneously, counting is started simultaneously with the enabling of counting.

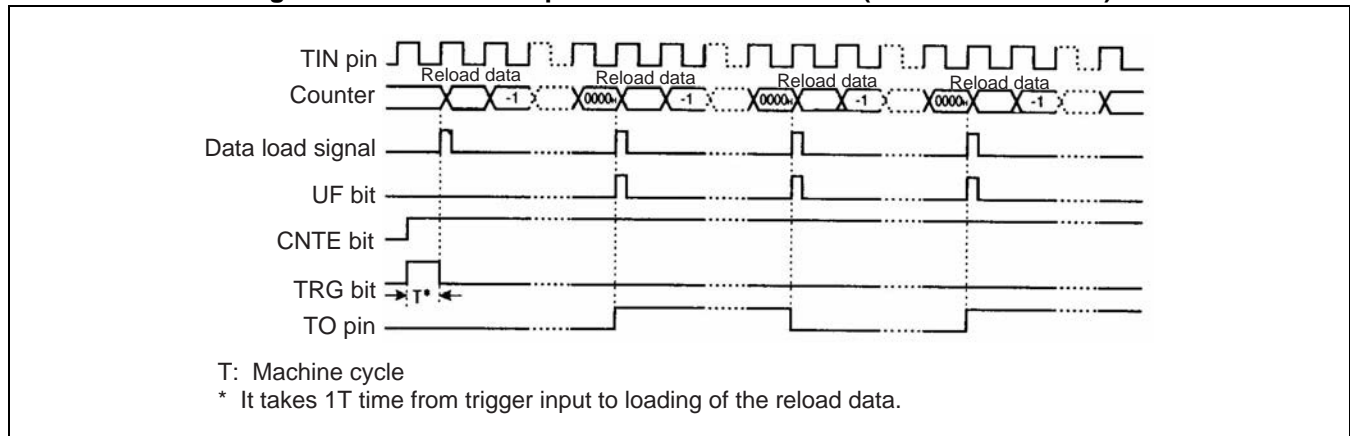
#### ● Operation in reload mode

When an underflow of the counter value (from 0000<sub>H</sub> to FFFF<sub>H</sub>) occurs, the value of the reload register (TMRD0/TMRD1) is loaded into the counter, and counting continues. If the underflow interrupt flag (TMCSRL0/TMCSRL1:UF) bit is set to "1" and the underflow interrupt enable bit (TMCSRL0/TMCSRL1:INTE) is "1", an interrupt request is generated.

The timer can also output from the TO0/TO1 pin a toggle waveform, which is inverted for each underflow.

Figure 12.6-10 shows counting in reload mode.

**Figure 12.6-10 Count Operation in Reload Mode (Event Count Mode)**



Note:

Specify 4/φ or more for the H and L widths of the clock input to the TIN0/TIN1 pin.

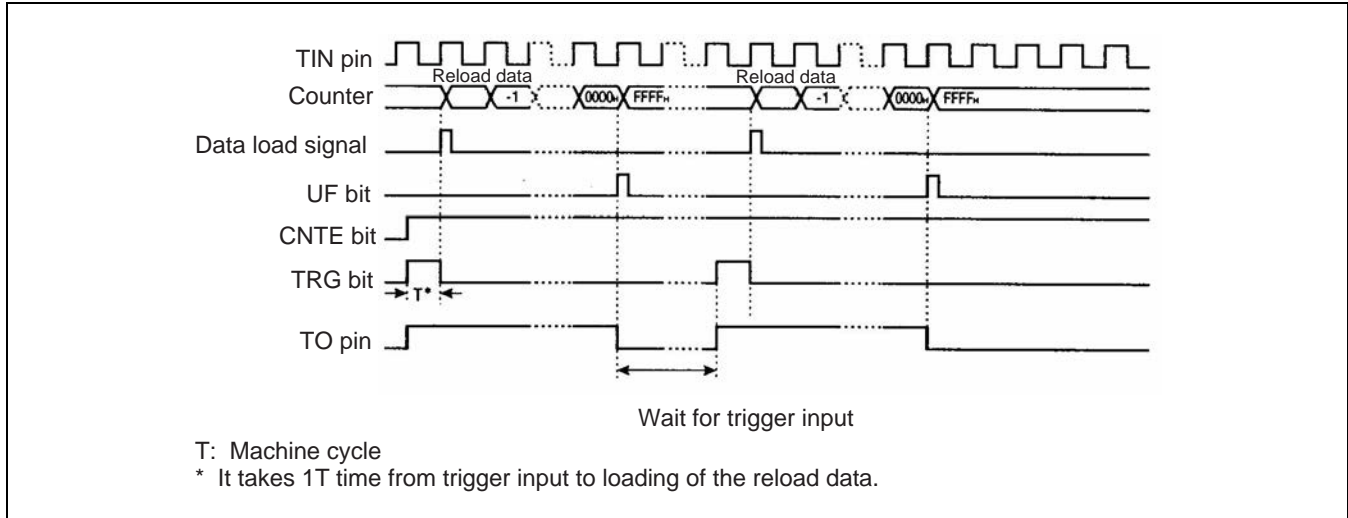
### ● Operation in single-shot mode

When an underflow of the counter value (from 0000<sub>H</sub> to FFFF<sub>H</sub>) occurs, the counter stops in the FFFF<sub>H</sub> state. If the underflow interrupt flag (TMCSRL0/TMCSRL1:UF) bit is set to "1" and the underflow interrupt enable (TMCSRL0/TMCSRL1:INTE) bit is "1", an interrupt request is generated.

The timer can also output from the TO0/TO1 pin a rectangular waveform indicating that counting is in progress.

Figure 12.6-11 shows counting in single-shot mode.

**Figure 12.6-11 Counter Operation in Single-shot Mode (Event Count Mode)**



Note: Specify 4/φ or more for the H and L widths of the clock input to the TIN0/TIN1 pin.

## 12.7 Usage Notes on the 16-bit Reload Timer

---

Notes on using the 16-bit reload timer are given below.

---

### ■ Usage Notes on the 16-bit Reload Timer

#### ● Notes on using a program for setting

- Write a value to the 16-bit reload register (TMRD0/TMRD1) when counting stops (TMCSRL0/TMCSRL1:CNTE = 0). Also, a value can be read from the 16-bit timer register (TMR0/TMR1) even during counting, but always be sure to use a word transfer instruction (MOVW A, dir, etc.).
- Change the CSL1 and CSL0 bits of the timer control status register (TMCSRH0/TMCSRH1) when the counter has stopped (TMCSRL0/TMCSRL1:CNTE = 0).

#### ● Notes about interrupts

- When the UF bit of the timer control status register (TMCSRL0/TMCSRL1) is set to "1" and an interrupt request is enabled (TMCSRL0/TMCSRL1:INTE = 1), control cannot be returned from interrupt processing. Always clear the UF bit.
- Since the 16-bit reload timer shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the 16-bit reload timer, shared resource interrupts must be disabled.

## 12.8 Sample Programs for the 16-bit Reload Timer

This section contains sample programs for the 16-bit reload timer in internal clock mode and event count mode.

### ■ Sample Program in Internal Clock Mode

#### ● Processing

- A 25 ms interval timer interrupt is generated with 16-bit reload timer 0.
- The timer is used in reload mode to repeatedly generate an interrupt.
- The timer is started with a software trigger. External trigger input is not used.
- EI<sup>2</sup>OS is not used.
- 16 MHz is used for the machine clock, and 2  $\mu$ s is used for the count clock.

#### ● Coding example

```

ICR09      EQU      0000B9H      ;Interrupt control register for the 16-bit reload timer
TMCSR      EQU      000082H      ;Timer control status register
TMR        EQU      000084H      ;16-bit timer register
TMRD       EQU      000084H      ;16-bit reload register
UF         EQU      TMCSR:2      ;Interrupt request flag bit
CNTE       EQU      TMCSR:1      ;Counter operation enable bit
TRG        EQU      TMCSR:       ;Software trigger bit

;-----Main program-----
CODE                      CSEG
START:
;           :                      ;Assumes that stack pointer (SP) has already been
;                               ;initialized
;           AND      CCR,#0BFH    ;Interrupt disable
;           MOV      I:ICR09,#00H ;Interrupt level 0 (strongest)
;           CLRB     I:CNTE       ;Temporary stopping of counter
;           MOVW     I:TMRD,#30D3H ;Sets data for 25-ms timer
;           MOVW     I:TMCSR,#00001000000011011B
;                               ;Interval timer operation, 2  $\mu$ s clock
;                               ;Disables external trigger and external output
;                               ;Selects reload mode, and enables interrupts
;                               ;Clears interrupt flag, and starts counter
;           MOV      ILM,#07H    ;Sets ILM in PS to level 7
;           OR       CCR,#40H    ; Interrupt enable

```



```

LOOP:    MOV     A,#00H           ;Endless loop
         MOV     A,#01H           ;
         BRA     LOOP            ;

;-----Interrupt program-----
WARI:
         CLRB    I:UF            ;Clears interrupt request flag
;
;         :
;         User processing
;         :
         RETI                    ;Returns from interrupt

CODE     ENDS

;-----Vector setting-----
VECT     CSEG     ABS=0FFH
         ORG     0FF84H           ;Sets vector for interrupt #30 (1EH)
         DSL     WARI
         ORG     0FFDCH           ;Sets reset vector
         DSL     START
         DB      00H             ;Sets single-chip mode
VECT     ENDS
         END      START

```

## ■ Sample Program in Event Count Mode

### ● Processing

- When the rising edge of the pulse input to the external event input pin is counted 10,000 times with 16-bit reload timer/counter 0, an interrupt is generated.
- The timer operates in single-shot mode.
- External trigger input selects the rising edge.
- EI<sup>2</sup>OS is not used.

### ● Coding example

```

ICR09    EQU     0000B9H         ;Interrupt control register for the 16-bit reload timer
TMCSR    EQU     000082H         ;Timer control status register
TMR      EQU     000084H         ;16-bit timer register
TMRD     EQU     000084H         ;16-bit reload register
DDR1     EQU     000011H         ;Port data register
UF        EQU     TMCSR:2        ;Interrupt request flag bit
CNTE     EQU     TMCSR:1        ;Counter operation enable bit
TRG      EQU     TMCSR:0        ;Software trigger bit

```

```

;-----Main program-----
CODE          CSEG
START:
;          :          ;Assumes that stack pointer (SP) has already been
;                      initialized
;          AND    CCR,#0BFH      ;Interrupt disable
;          MOV    I:ICR09,#00H   ;Interrupt level 0 (strongest)
;          MOV    I:DDR1,#00H    ;Sets P15/INT5/TIN0 pin to input
;          CLRB   I:CNTC         ;Temporary stopping of counter
;          MOVW   I:TMRD,#2710H   ;Sets reload value to 10,000
;          MOVW   I:TMCSR,#0000110010001011B
;                      ;Counter operation, external event input, rising
;                      ;Disables external output
;                      ;Selects single-shot mode and enables interrupts
;                      ;Clears interrupt flag, starts counter
;          MOV    ILM,#07H       ;Sets ILM in PS to level 7
;          OR     CCR,#40H       ;Interrupt enable
LOOP:  MOV    A,#00H             ;Endless loop
;          MOV    A,#01H         ;
;          BRA    LOOP           ;

;-----Interrupt program-----
WARI:
;          CLRB   I:UF           ;Clears interrupt request flag
;          :
;          ; User processing
;
;          RETI                  ;Returns from interrupt
CODE    ENDS

;-----Vector setting-----
VECT    CSEG    ABS=0FFH
;          ORG    0FF84H          ;Sets vector for interrupt #30 (1EH)
;          DSL    WARI
;          ORG    0FFDCH          ;Sets reset vector
;          DSL    START
;          DB     00H             ;Sets single-chip mode
VECT    ENDS
;          END    START

```



# **CHAPTER 13**

---

## ***16-BIT PPG TIMER***

**This chapter describes the functions and operation of the 16-bit PPG Timer.**

- 13.1 Overview of 16-bit PPG Timer
- 13.2 Block Diagram of 16-bit PPG Timer
- 13.3 16-bit PPG Timer Pins
- 13.4 16-bit PPG Timer Registers
- 13.5 16-bit PPG Timer Interrupts
- 13.6 Operation of 16-bit PPG Timer
- 13.7 Usage Notes on the 16-bit PPG Timer
- 13.8 Sample Programs for the 16-bit PPG Timer

## 13.1 Overview of 16-bit PPG Timer

---

**The 16-bit PPG timer consists of a 16-bit down counter, prescaler, 16-bit period setting register, 16-bit duty setting register, 16-bit control register and a PPG output pin.**

---

### ■ 16-bit PPG Timer (× 3, PPG1 is not present in MB90465 Series)

The 16-bit PPG timer consists of a 16-bit down counter, prescaler, 16-bit period setting register, 16-bit duty setting register, 16-bit control register and a PPG output pin. This module can be used to output pulses synchronized by software trigger or GATE signal from Multi-functional timer, refer to "Multi-functional Timer" in chapter 14.

- 8 types of counter operation clock ( $\phi$ ,  $\phi/2$ ,  $\phi/4$ ,  $\phi/8$ ,  $\phi/16$ ,  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ ) can be selected ( $\phi$  is the machine clock).
- An interrupt is generated when there is a trigger or an counter borrow or when PPG rising (normal polarity) / PPG falling (inverted polarity).
- PPG output operation

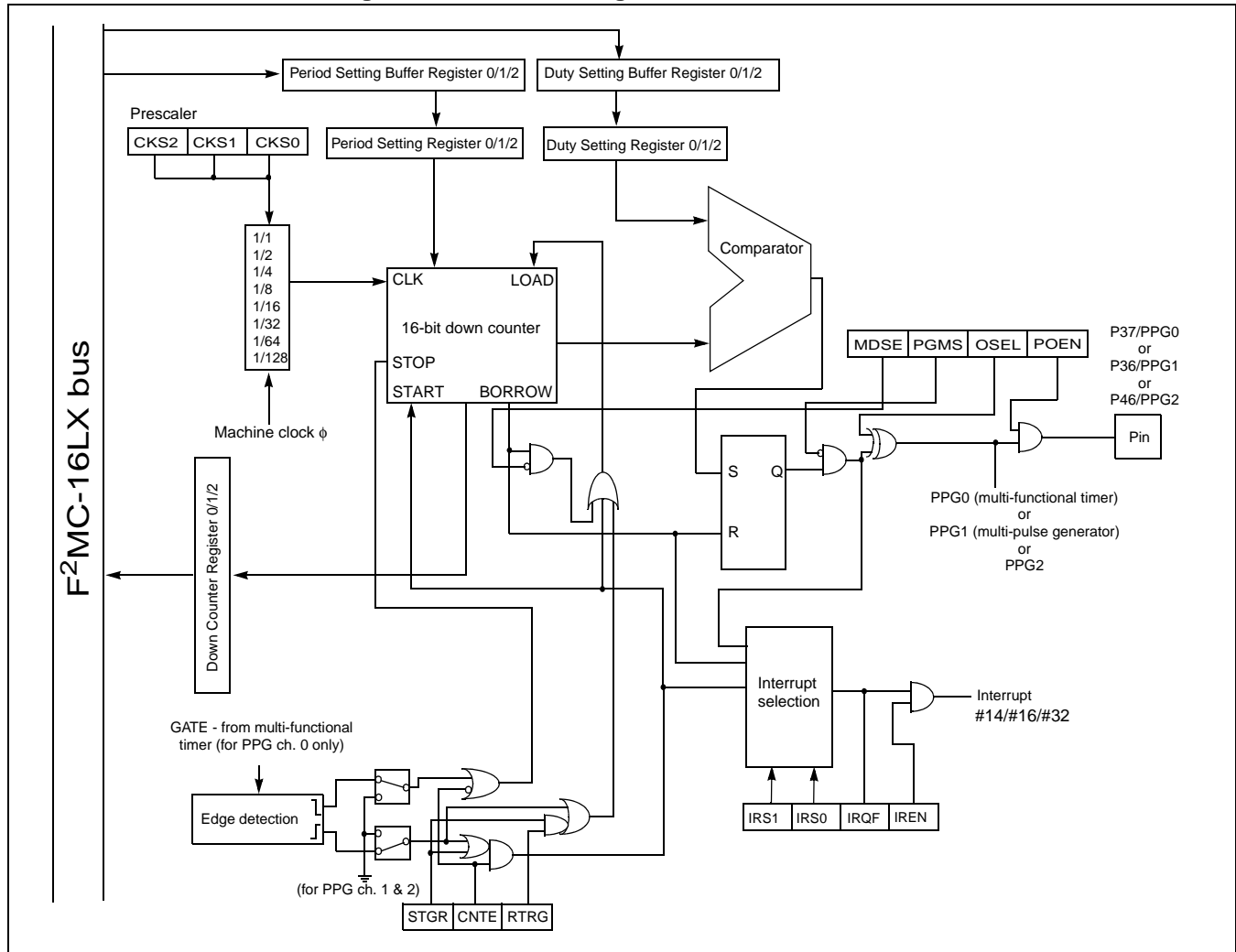
The 16-bit PPG timer can output pulse waveforms with variable period and duty ratio. Also, it can be used as D/A converter in conjunction with an external circuit.

## 13.2 Block Diagram of 16-bit PPG Timer

This section shows the block diagram of 16-bit PPG timer.

### ■ Block Diagram of 16-bit PPG Timer

Figure 13.2-1 Block Diagram of 16-bit PPG Timer



### 13.3 16-bit PPG Timer Pins

This section describes the pins of the 16-bit PPG timer and provides a pin block diagram.

#### 16-bit PPG Timer Pins

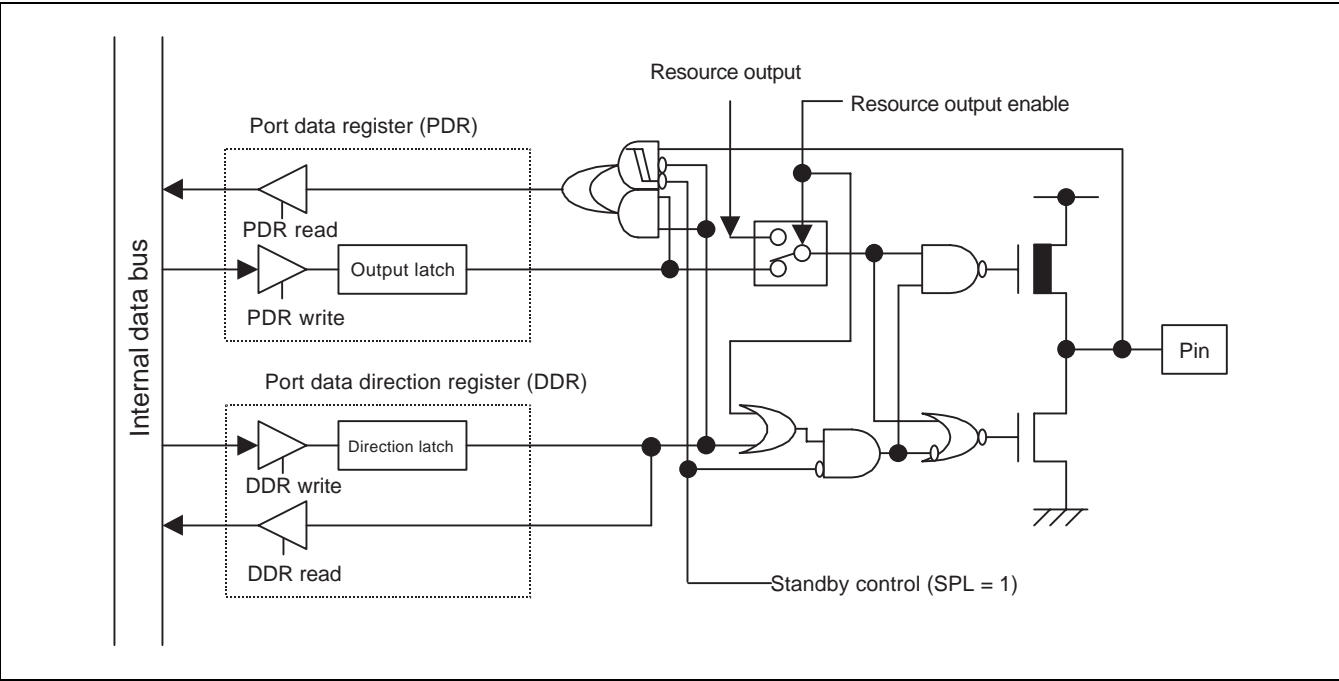
The pins of the 16-bit PPG timer are shared with the general-purpose ports. Table 13.3-1 lists the functions of the pins, I/O format, and settings required to use the 16-bit PPG timer.

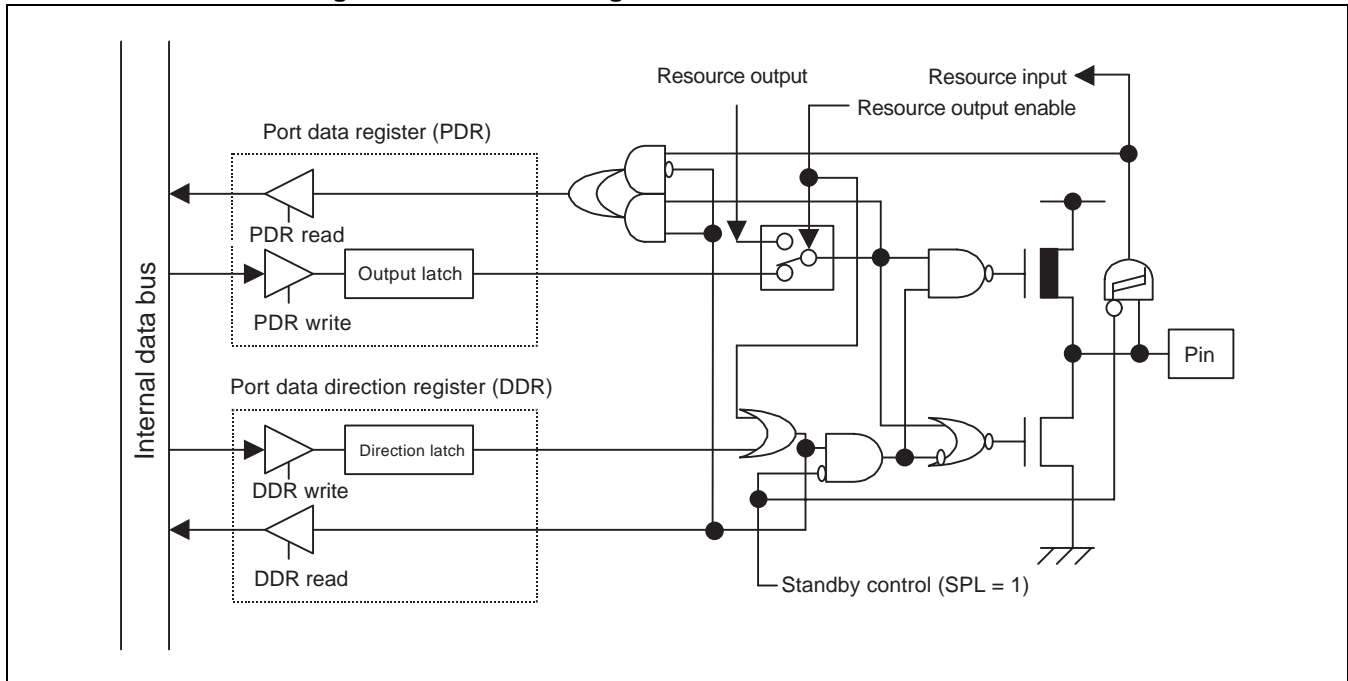
Table 13.3-1 16-bit PPG Timer Pins

Pin name	Pin function	I/O format	Standby control	Settings required for pins
P37/PPG0	Port 3 input-output / PPG0 output	CMOS output / CMOS input	Available	Setting for the PPG timer 0 output (PNCTL0:POEN=1)
P36/PPG1	Port 3 input-output / PPG1 output			Setting for PPG timer 1 output enable (PNCTL1:POEN=1)
P46/PPG2	Port 4 input-output / PPG2 output	CMOS output / CMOS hysteresis input		Setting for PPG timer 2 output enable (PNCTL2:POEN=1)

#### Block Diagram of the 16-bit PPG Timer Pins

Figure 13.3-1 Block Diagram of the 16-bit PPG Timer 0 & 1 Pins



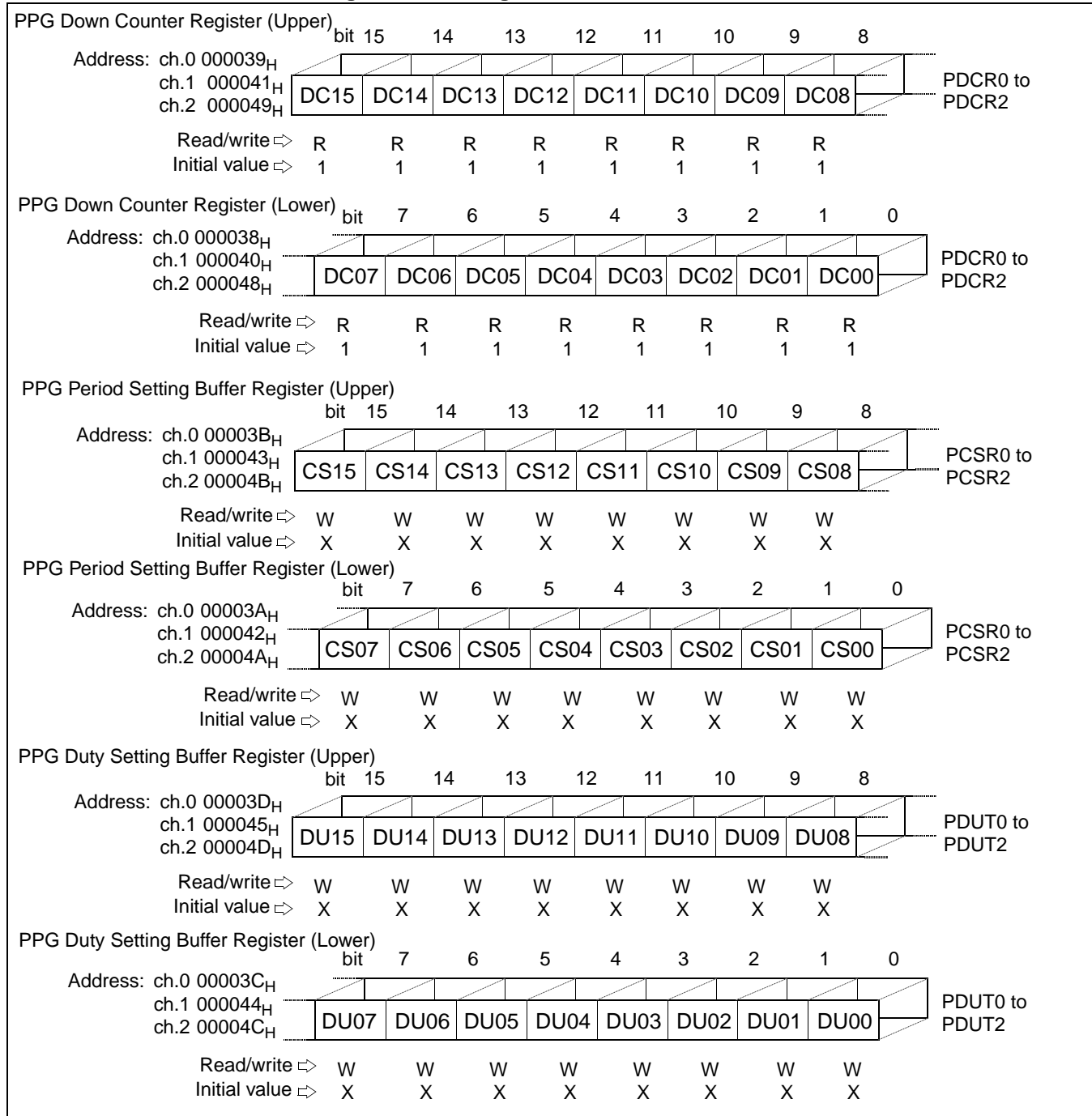
**Figure 13.3-2 Block Diagram of the 16-bit PPG Timer 2 Pin**



## 13.4 16-bit PPG Timer Registers

### ■ 16-bit PPG Timer Registers

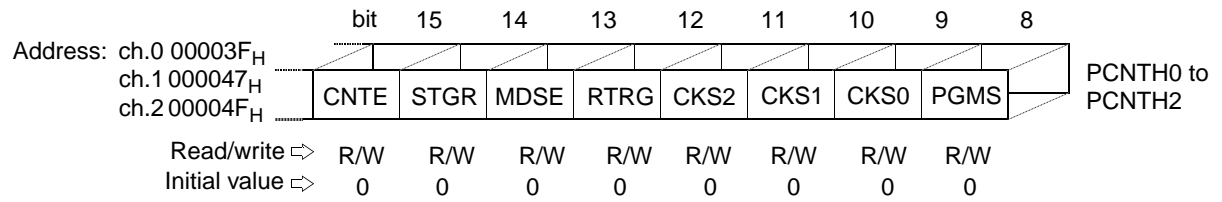
**Figure 13.4-1 Registers of 16-bit PPG Timer**



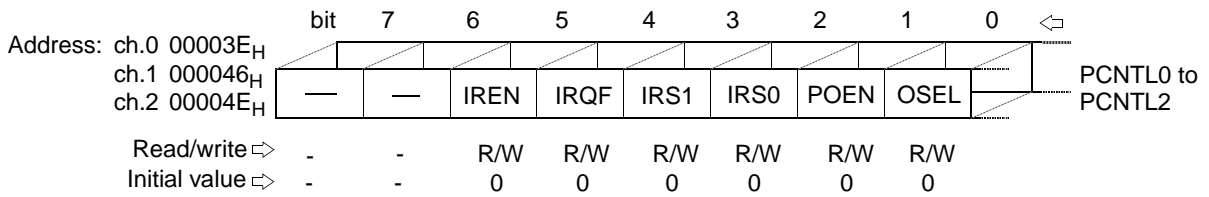
(Continued)

(Continued)

## PPG Control Status Register (Upper)



## PPG Control Status Register (Lower)

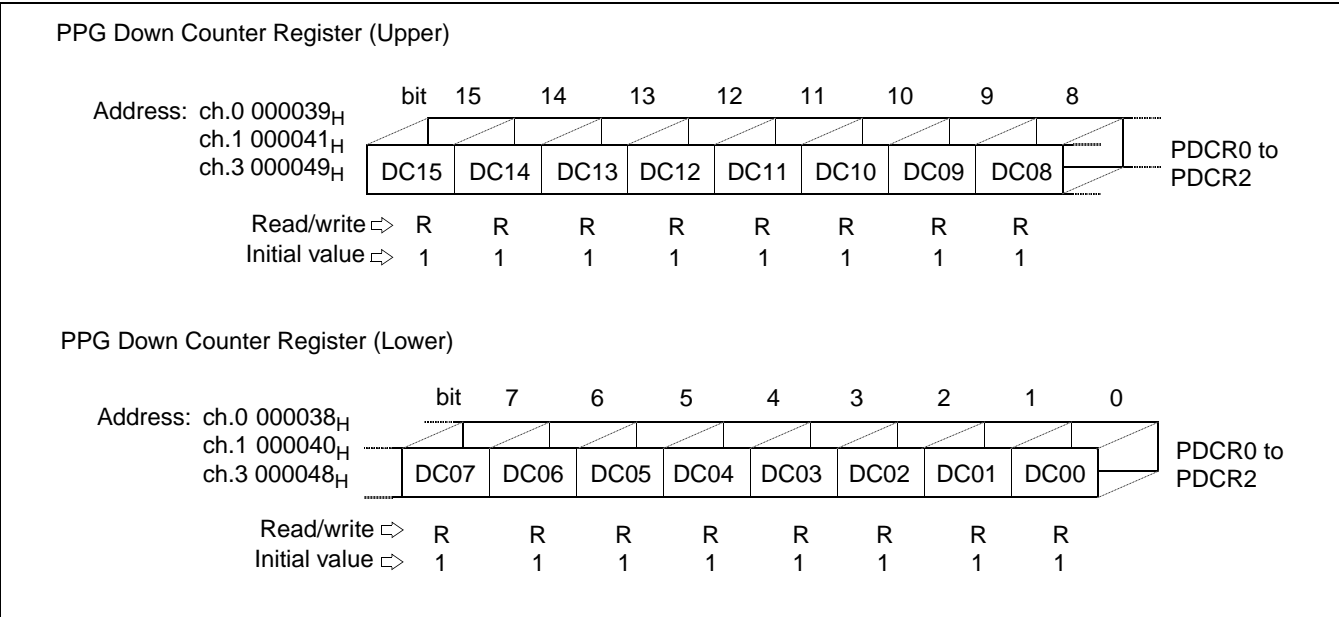


### 13.4.1 PPG Down Counter Register (PDCR0 to PDCR2)

PPG down counter registers (PDCR0 to PDCR2) are 16-bit registers, which are used to read the count value of the 16-bit PPG down counter.

#### ■ PPG Down Counter Register (PDCR0 to PDCR2)

Figure 13.4-2 PPG Down Counter Register (PDCR0 to PDCR2)



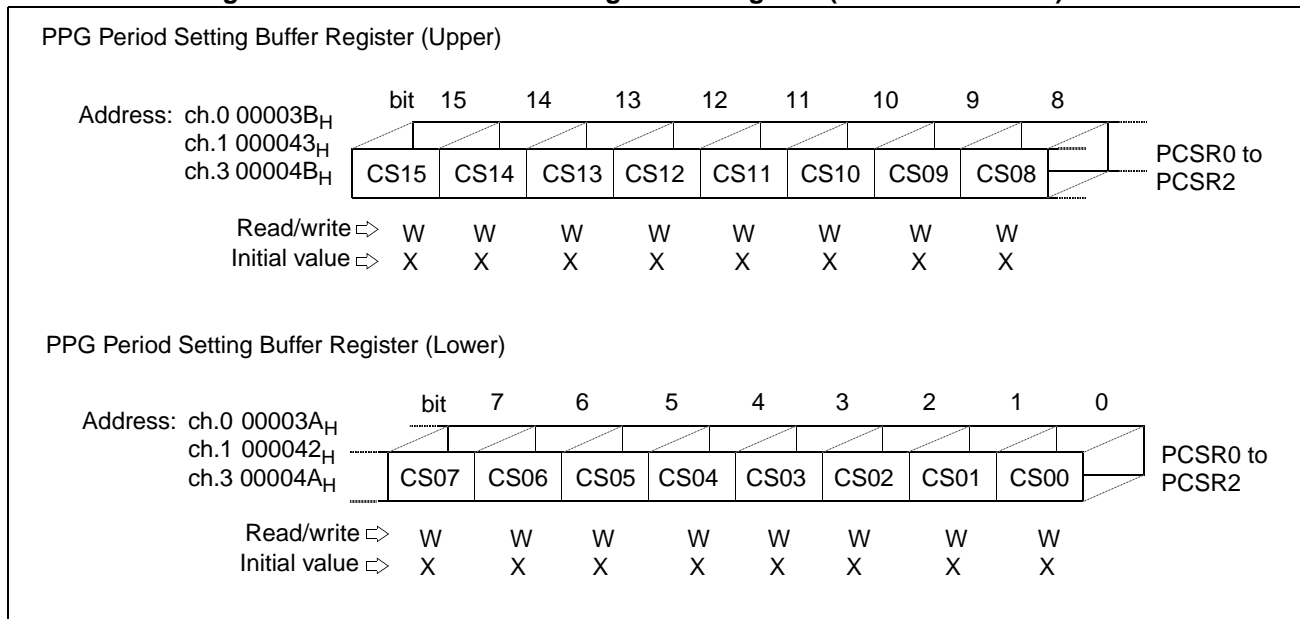
These are 16-bit registers that are used to store the values of the 16-bit down counter. The initial value of them are all 1. Word access to these register are recommended. These registers are read-only.

## 13.4.2 PPG Period Setting Buffer Register (PCSR0 to PCSR2)

PPG period setting buffer register is used to set the period of the output pulses generated by PPG.

### ■ PPG Period Setting Buffer Register (PCSR0 to PCSR2)

**Figure 13.4-3 PPG Period Setting Buffer Register (PCSR0 to PCSR2)**



These are 16-bit registers that are used to set the period of the output pulses generated by PPG. The initial value of them are undetermined, so that these registers must be written before starting an operation. Word access to these registers are recommended. These registers are write-only.

Data transfer from period setting buffer register to period setting register will be at counter borrow or trigger or retrigger if enabled.

#### Note:

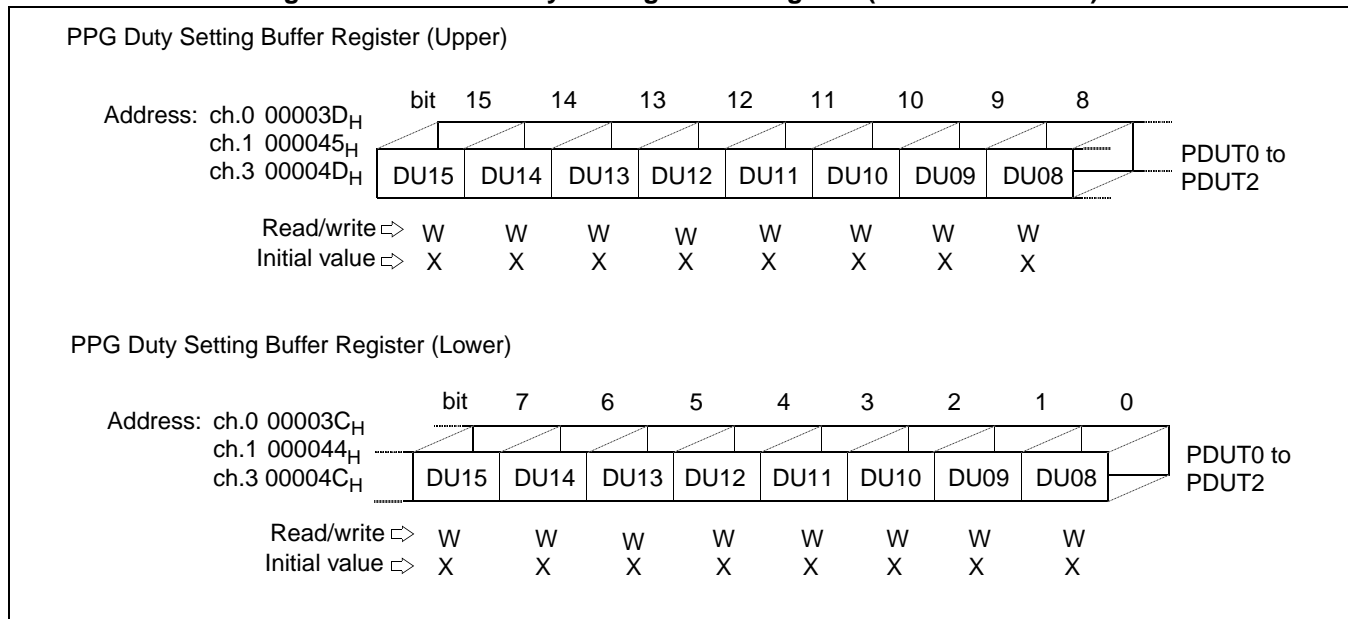
In case of updating period setting buffer register, duty setting buffer register must be written after writing to period setting buffer register. Only updating period setting buffer register is prohibited.

### 13.4.3 PPG Duty Setting Buffer Register (PDUT0 to PDUT2)

PPG duty setting buffer register is used to control the duty ratio of the output pulses generated by PPG.

#### ■ PPG Duty Setting Buffer Register (PDUT0 to PDUT2)

Figure 13.4-4 PPG Duty Setting Buffer Register (PDUT0 to PDUT2)



These are 16-bit registers that are used to control the duty ratio of the output pulses generated by PPG. The initial value of them are undetermined, so that these registers must be set a value before starting an operation. Word access instruction to these registers are recommended. These registers are write-only.

Data transfer from duty setting buffer register to duty setting register is at counter borrow or trigger or retrigger if enabled.

Setting the same value in both the period setting register and duty setting register outputs all "H"s for normal polarity and all "L"s for inverted polarity.

The output of the PPG is indeterminate if  $PCSR < PDUT$ .

Note:

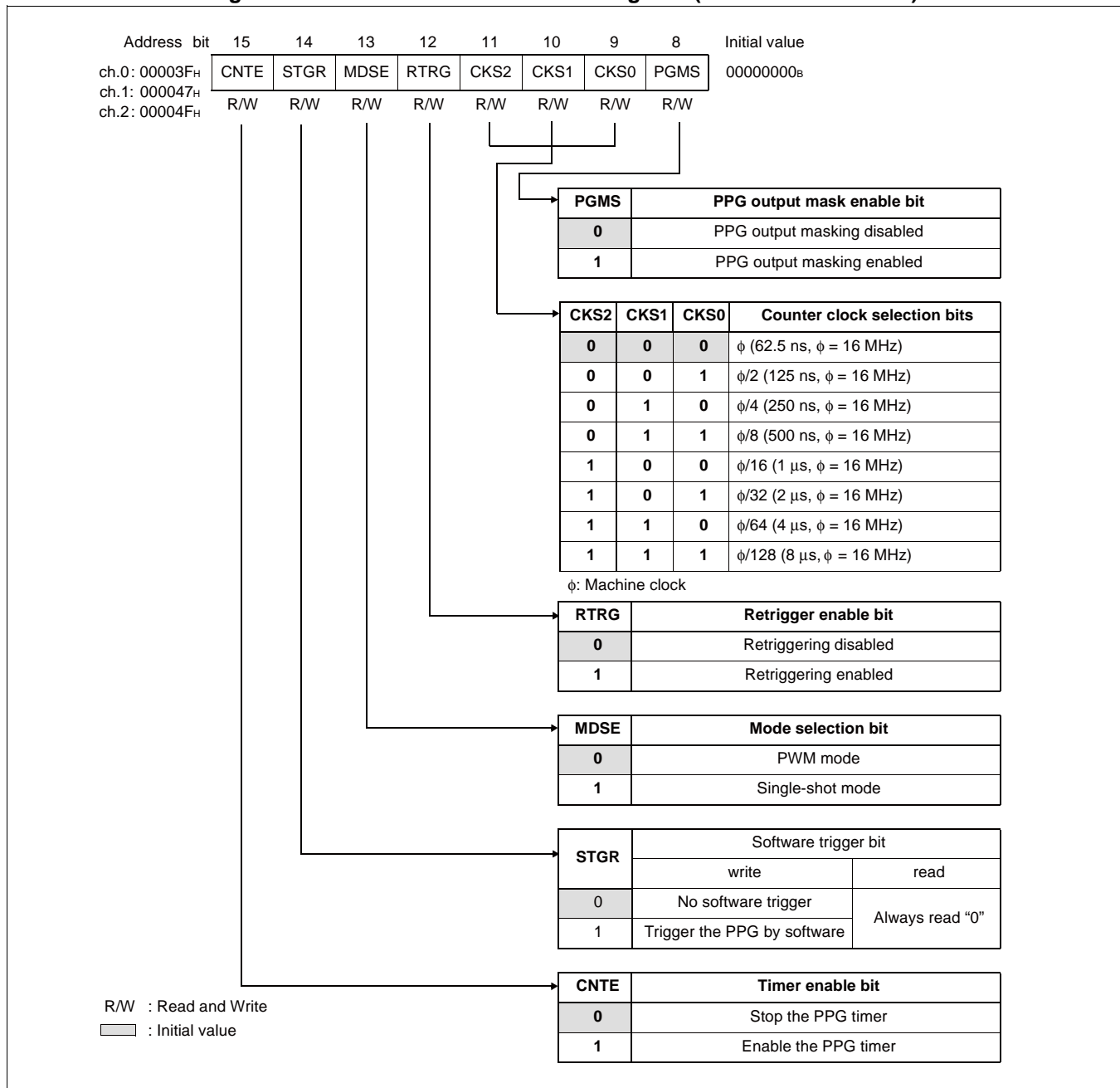
Duty setting buffer register can be written in the case of not updating period setting buffer register.

### 13.4.4 PPG Control Status Register (PCNTL0 to PCNTL2, PCNTH0 to PCNTH2)

PPG control status register is used to set operating conditions for 16-bit PPG timer enable or disable operation, software trigger, retrigger control interrupt, output polarity and check the status

#### ■ PPG Control Status Register, Upper Byte (PCNTH0 to PCNTH2)

Figure 13.4-5 PPG0 to PPG2 Control Register (PCNTH0 to PCNTH2)

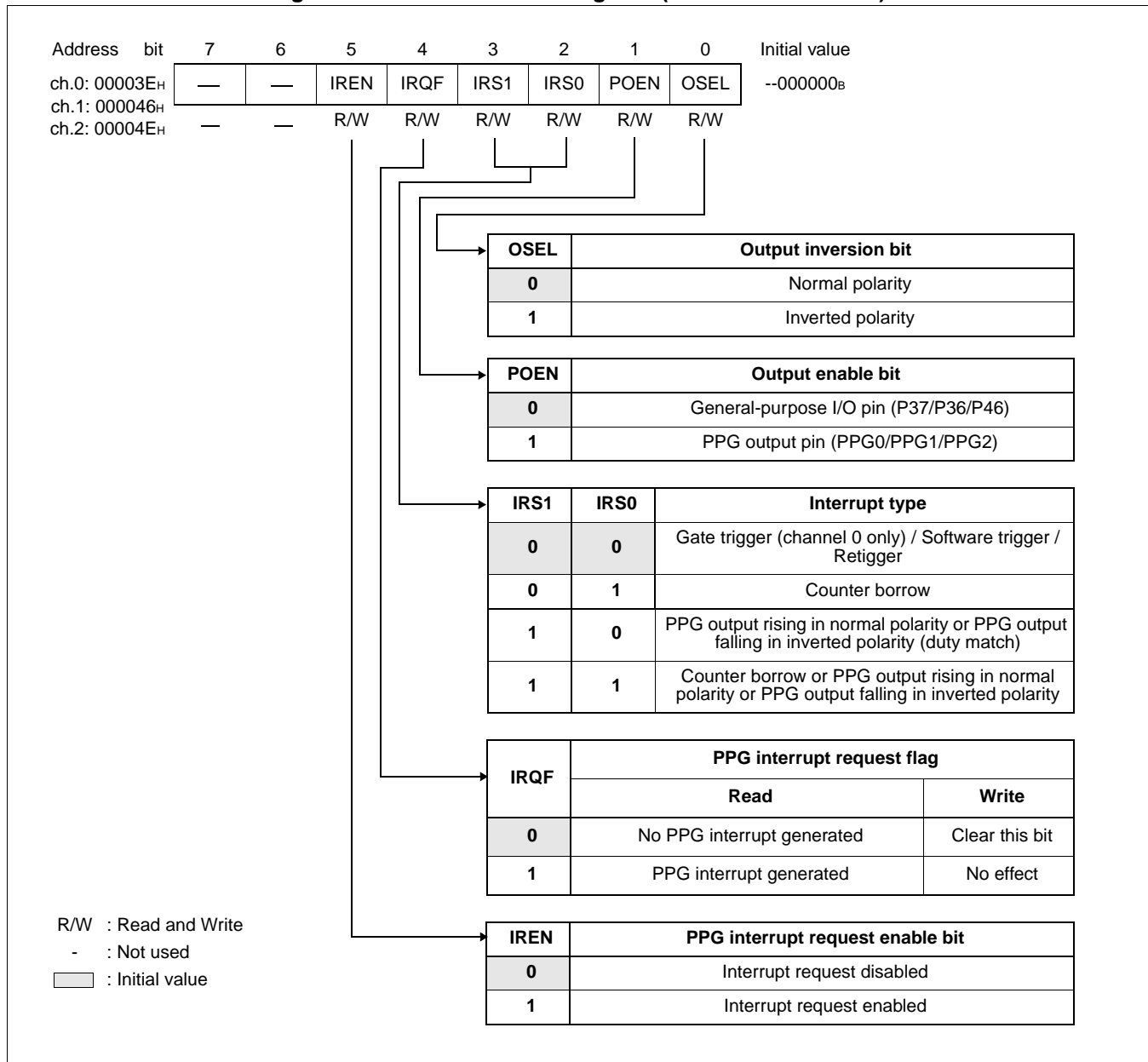


**Table 13.4-1 PPG Control Register (PCNTH0 to PCNTH2) Bit**

Bit name		Function
bit15	CNTE: Timer enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the PPG timer operation.</li> <li>• Writing “1” will enable the PPG operation and wait for trigger to start PPG operation.</li> <li>• Writing “0” will stop the operation.</li> </ul>
bit14	STGR: Software trigger bit	<ul style="list-style-type: none"> <li>• This bit is the software trigger bit for PPG.</li> <li>• Writing “1” to this bit triggers the PPG by software.</li> <li>• This bit is always read as “0”.</li> </ul>
bit13	MDSE: Mode selection bit	<ul style="list-style-type: none"> <li>• When this bit is “0”, PPG operates in PWM mode.</li> <li>• When this bit is “1”, PPG operates in single-shot mode.</li> </ul>
bit12	RTRG: Retrigger enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable retriggering function of PPG during operation.</li> <li>• When this bit is “0”, retriggering function is disabled.</li> <li>• When this bit is “1”, retriggering function is enabled.</li> </ul>
bit11 to bit9	CKS2 to CKS0: Counter clock selection bits	<ul style="list-style-type: none"> <li>• This bit are used to select the operation clock for 16-bit PPG timer.</li> </ul>
bit8	PGMS: PPG output mask enable bit	<ul style="list-style-type: none"> <li>• This bit is used to mask the PPG output to specific level regardless of the mode setting (PCNTH:MDSE), period setting (PCSR) or duty setting (PDUT).</li> <li>• Write “0” will disable PPG output masking function.</li> <li>• Writing “1” to this bit masks the PPG output to always “L” when polarity setting is “Normal” (PCNTL:OSEL=0).</li> <li>• Writing “1” to this bit masks the PPG output to always “H” when polarity setting is “Inverted” (PCNTL:OSEL=1).</li> </ul> <p>(Note) By setting period setting register (PCSR) and duty setting register (PDUT) with same value, all “H” in normal polarity or all “L” in inverted polarity can be outputted when this bit is “1”.</p>

## ■ PPG Control Status Register, Lower Byte (PCNTL0 to PCNTL2)

Figure 13.4-6 PPG Control Register (PCNTL0 to PCNTL2)





**Table 13.4-2 PPG Control Register (PCNTL0 to PCNTL2)**

Bit name		Function
bit7, bit6	Unused bit	<ul style="list-style-type: none"> <li>This read value is indeterminate.</li> <li>Writing to this bit has no effect on the operation.</li> </ul>
bit5	IREN: PPG interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit enable or disables PPG interrupt request to the CPU.</li> <li>When this bit and the interrupt flag (IRQF) bit are “1”, PPG outputs an interrupt request.</li> </ul>
bit4	IRQF: PPG interrupt flag bit	<ul style="list-style-type: none"> <li>This bit is set to "1" when PPG interrupt occurs.</li> <li>Writing "0" will clear this bit.</li> <li>Writing "1" has no effect.</li> <li>In read-modify-write operation, “1” is always read.</li> <li>This bit is also cleared when EI<sup>2</sup>OS is activated.</li> </ul>
bit3, bit2	IRS1, IRS0: Interrupt selection bit	<ul style="list-style-type: none"> <li>These bits are used to select interrupt condition of the PPG timer.</li> </ul>
bit1	POEN: Output enable bit	<ul style="list-style-type: none"> <li>This bit enables or disables output from the PPG output pin.</li> <li>When this bit is “0”, the pin functions as a general purpose port.</li> <li>When this bit is “1”, the pin functions as a PPG timer output pin.</li> </ul>
bit0	OSEL: Output inversion bit	<ul style="list-style-type: none"> <li>This bit selects the polarity of PPG output pin.</li> <li>When this bit is “0”, normal polarity is selected. PPG outputs “L” when 16-bit down count value is greater than PDUT, and outputs “H” when smaller than or equals to PDUT.</li> <li>When this bit is “1”, the PPG output is inverted.</li> </ul>

## 13.5 16-bit PPG Timer Interrupts

The 16-bit PPG timer is enabled to generate an interrupt request when trigger or counter borrow or PPG rising in normal polarity or PPG falling in inverted polarity depending on PCNTL=IRS1, IRS0 setting. It is also coordinated with the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ 16-bit PPG Timer Interrupts

Table 13.5-1 list the interrupt control bits and interrupt causes of the 16-bit PPG timer.

**Table 13.5-1 Interrupt Control Bits and Interrupt Causes of the 16-bit PPG Timer**

	16-bit PPG timer 0	16-bit PPG timer 1	16-bit PPG timer 2
Interrupt flag bit	PCNTL0:IRQF	PCNTL1:IRQF	PCNTL2:IRQF
Interrupt request enable bit	PCNTL0:IREN	PCNTL1:IREN	PCNTL2:IREN
Interrupt type selection bit	PCNTL0:IRS1,IRS0	PCNTL1:IRS1,IRS0	PCNTL2:IRS1,IRS0
Interrupt cause	PCNTL0:IRS1,IRS0=00 gate trigger/software trigger/ retrigger of 16-bit down counter (ch.0)	PCNTL1:IRS1,IRS0=00 software trigger/retrigger of 16-bit down counter (ch.1)	PCNTL2:IRS1,IRS0=00 software trigger/retrigger of 16-bit down counter (ch.2)
	PCNTL0:IRS1,IRS0=01 counter borrow of 16-bit down counter (ch.0)	PCNTL1:IRS1,IRS0=01 counter borrow of 16-bit down counter (ch.1)	PCNTL2:IRS1,IRS0=01 counter borrow of 16-bit down counter (ch.2)
	PCNTL0:IRS1,IRS0=10 PPG0 output rising in normal polarity or PPG0 output falling in inverted polarity	PCNTL1:IRS1,IRS0=10 PPG1 output rising in normal polarity or PPG1 output falling in inverted polarity	PCNTL2:IRS1,IRS0=10 PPG2 output rising in normal polarity or PPG2 output falling in inverted polarity
	PCNTL0:IRS1,IRS0=11 Counter borrow of 16-bit down counter (ch.0) or PPG0 output rising in normal polarity or PPG0 output falling in inverted polarity	PCNTL1:IRS1,IRS0=11 Counter borrow of 16-bit down counter (ch.1) or PPG1 output rising in normal polarity or PPG1 output falling in inverted polarity	PCNTL2:IRS1,IRS0=11 Counter borrow of 16-bit down counter (ch.2) or PPG2 output rising in normal polarity or PPG2 output falling in inverted polarity

In the 16-bit PPG timer, the IRQF bit of the PPG control status register (PCNTL) is set to "1" and an interrupt request is enabled (PCNTL: IREN=1), the interrupt request is outputted to the interrupt controller.

## ■ 16-bit PPG Timer Interrupts and EI<sup>2</sup>OS

Table 13.5-2 lists the 16-bit PPG timer interrupts and EI<sup>2</sup>OS.

**Table 13.5-2 16-bit PPG Timer Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
16-bit PPG timer 0 <sup>*1</sup>	#14 (0E <sub>H</sub> )	ICR01	0000B1 <sub>H</sub>	FFFFC4 <sub>H</sub>	FFFFC5 <sub>H</sub>	FFFFC6 <sub>H</sub>	O
16-bit PPG timer 1 <sup>*2</sup>	#16 (10 <sub>H</sub> )	ICR02	0000B2 <sub>H</sub>	FFFFBC <sub>H</sub>	FFFFBD <sub>H</sub>	FFFFBE <sub>H</sub>	
16-bit PPG timer 2 <sup>*3</sup>	#32 (20 <sub>H</sub> )	ICR10	0000BA <sub>H</sub>	FFFF7C <sub>H</sub>	FFFF7D <sub>H</sub>	FFFF7E <sub>H</sub>	

\*1: The same interrupt control register as that for 16-bit PPG timer 0 is assigned to PWC timer 0.

\*2: The same interrupt control register as that for 16-bit PPG timer 1 is assigned to 16-bit output compare channel 1 match.

\*3: The same interrupt control register as that for 16-bit PPG timer 2 is assigned to 16-bit free-run timer zero detect.

## ■ EI<sup>2</sup>OS Function of the 16-bit PPG Timer

Since the 16-bit PPG timer has a circuit that coordinates with EI<sup>2</sup>OS, the counter can start EI<sup>2</sup>OS when PPG interrupt occurs.

However, EI<sup>2</sup>OS is available only when other peripheral functions sharing the interrupt control register (ICR) do not use interrupts. For example, when 16-bit PPG timer 0 uses EI<sup>2</sup>OS, interrupts of the output compare channel 0 match must be disabled.

## 13.6 Operation of 16-bit PPG Timer

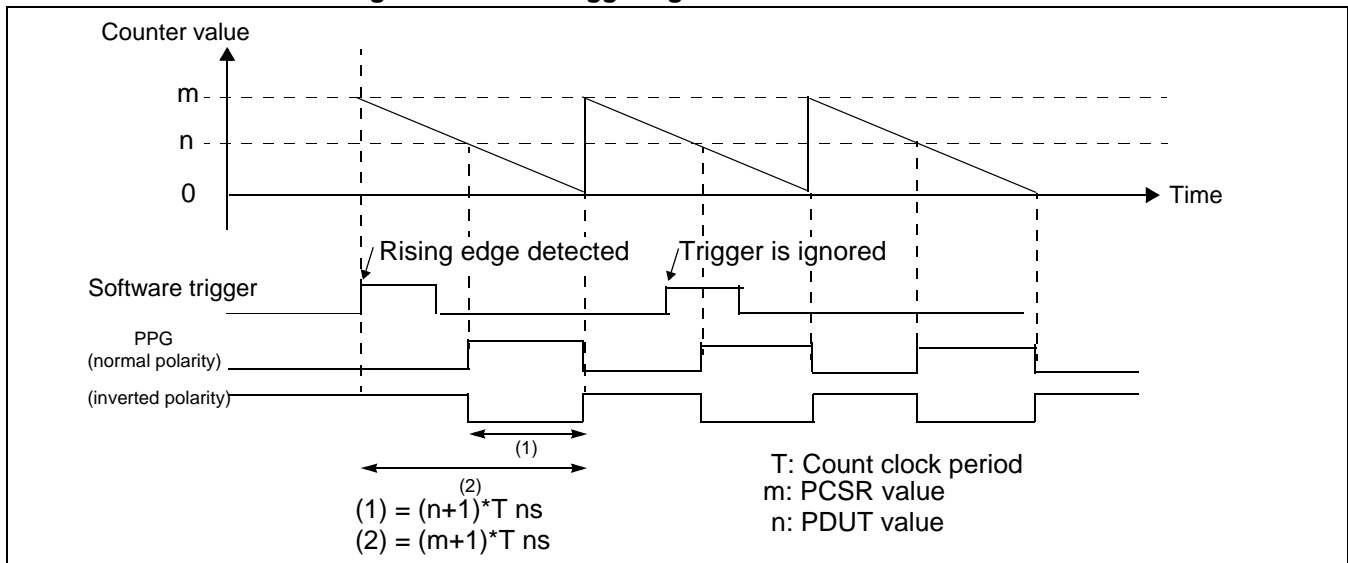
The 16-bit PPG Timer operate in either PWM mode or single shot mode. And Retriggering can be enabled.

### ■ PWM Mode (PCNTL: MDSE = 0)

For PWM operation, the 16-bit down counter will be loaded with PCSR value, starts counting after a valid trigger is detected. And once the 16-bit down counter reached zero, it is reloaded with PCSR value and repeat counting again. PPG output is toggled when 16-bit down counter is reloaded. The period of the output pulses can be controlled by setting PCSR and the duty ratio controlled by setting PDUT.

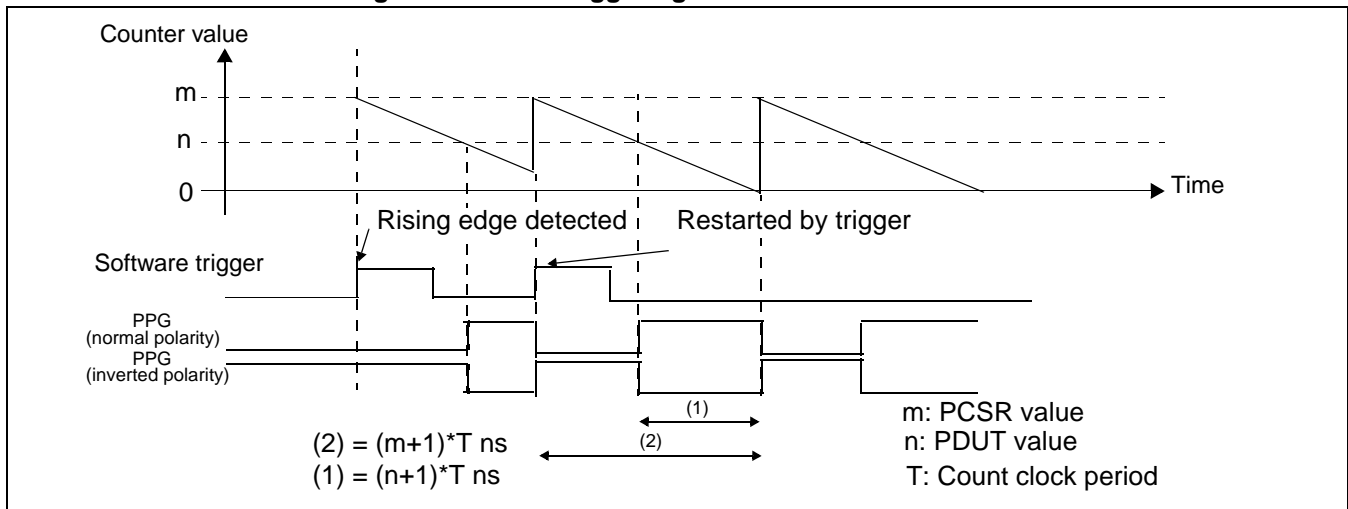
a) Retriggering is disabled (PCNTH: RTRG = 0)

**Figure 13.6-1 Retriggering is disabled in PWM Mode**



b) Retriggering is enabled (PCNTH: RTRG = 1)

**Figure 13.6-2 Retriggering is enabled in PWM Mode**

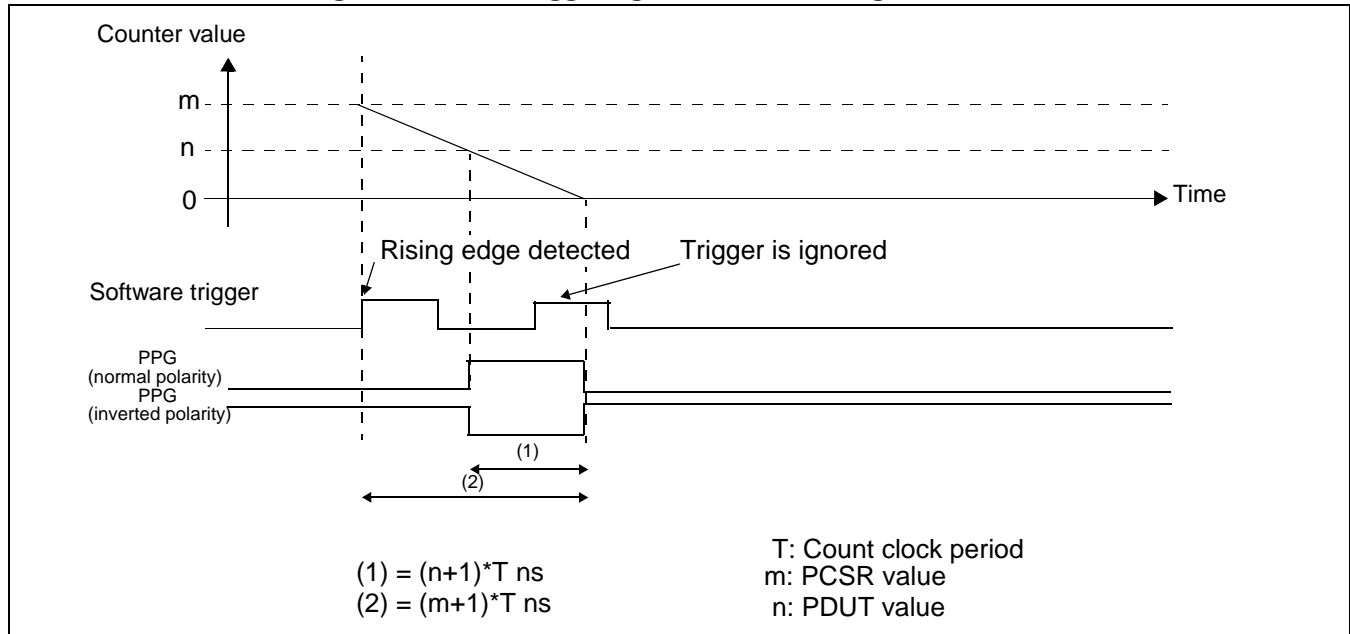


### ■ Single-shot Mode (PCNTL: MDSE = 1)

For single-shot operation, a single pulse of specified width can be output by a valid trigger. When retrigging is enabled, the counter is reloaded if an edge is detected during operation.

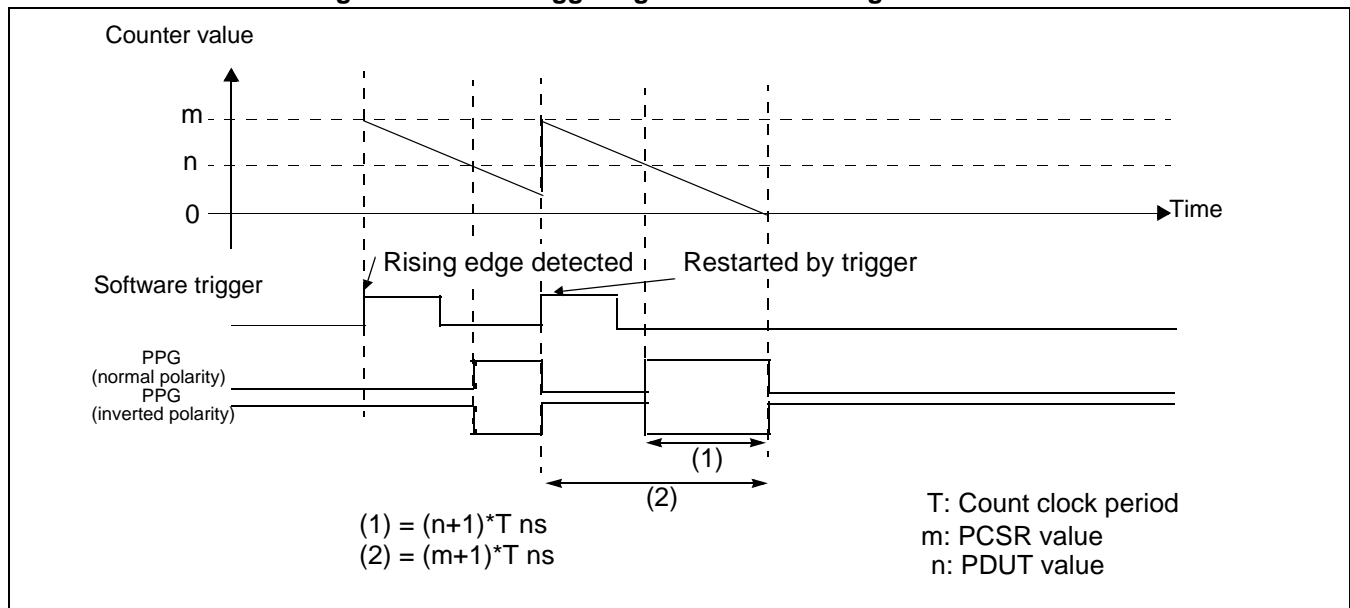
a) Retriggering is disabled (PCNTH: RTRG = 0)

**Figure 13.6-3 Retriggering is disabled in Single-shot Mode**



b) Retriggering is enabled (PCNTH: RTRG = 1)

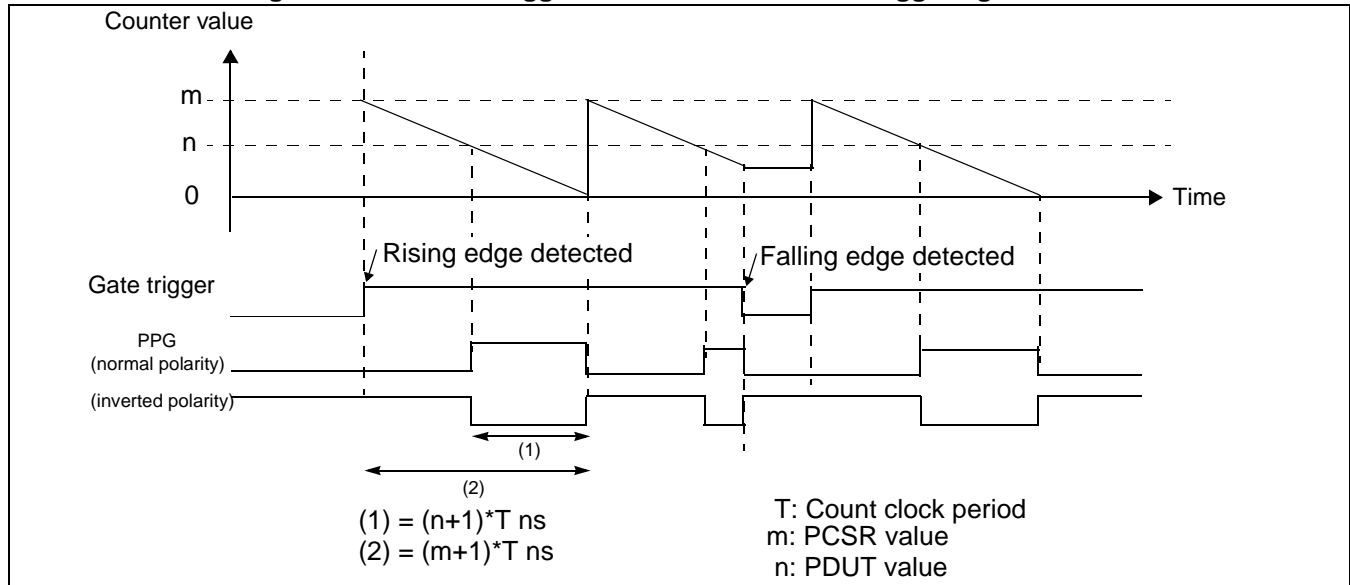
**Figure 13.6-4 Retriggering is enabled in Single-shot Mode**



### ■ Gate Trigger (PPG channel 0 only)

When gate trigger is used, PPG starts operation when rising edge of gate trigger is detected and stops when falling is detected. In next rising edge, PPG restarts operation again.

**Figure 13.6-5 Gate Trigger in PWM Mode when retrigging is enable**

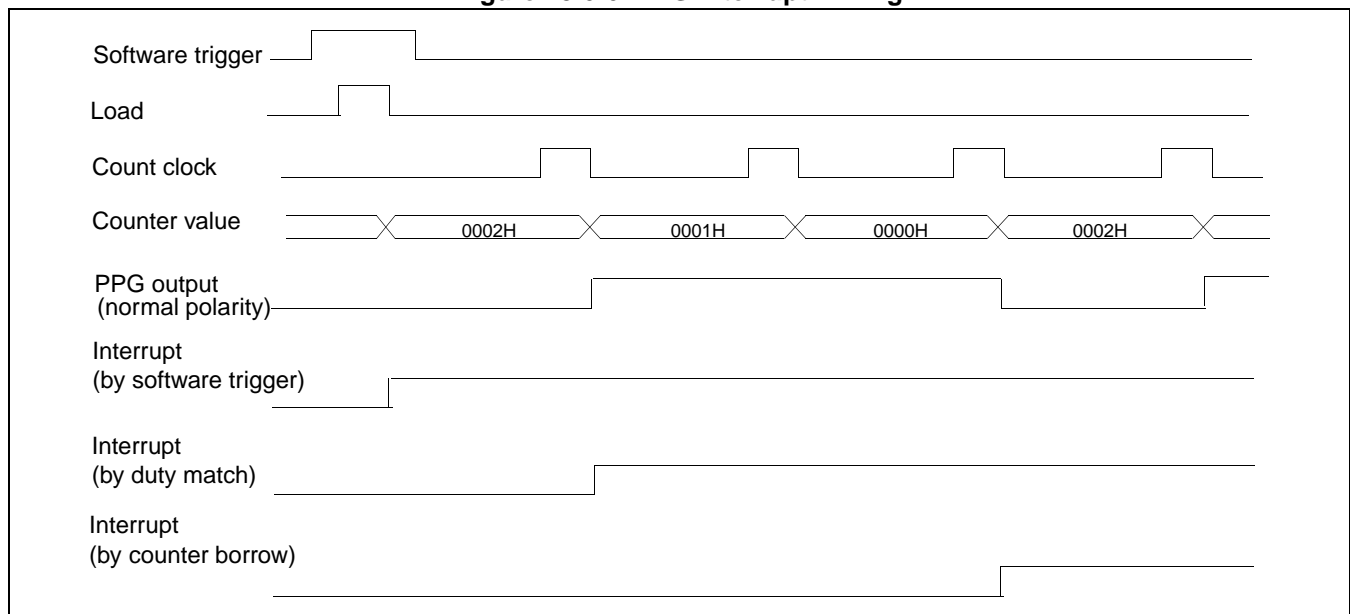


### ■ PPG Interrupts

There are four types of interrupts sharing one interrupt flag (PCNTL:IRQF) selected by interrupt type bits (PCNTL:IRS1,IRS0).

- Gate trigger (for PPG channel 0 only) or software trigger or retrigger
- Counter borrow
- Duty match occurs when PPG output rising in normal polarity or PPG output falling in inverted polarity
- Counter borrow or duty match

**Figure 13.6-6 PPG Interrupt Timing**



## 13.7 Usage Notes on the 16-bit PPG Timer

---

Notes on using the 16-bit PPG timer are given below.

---

### ■ Usage Notes on the 16-bit PPG Timer

#### ● Notes on using a program for setting

- Write a value to the period setting buffer register (PCSR), duty setting buffer register (PDUT) must be written after writing to PCSR. Only updating PCSR is prohibited. Be sure to use a word transfer instruction (MOVW A, dir, etc.) to access PCSR and PDUT.
- Always set the value of duty setting buffer register (PDUT) not greater than period setting buffer register (PCSR), otherwise the output of PPG is indeterminate.
- Change the CKS2, CKS1 and CKS0 bits of the control status register (PCNTH) when the PPG is stopped (PCNTH: CNTE=0).

#### ● Notes about interrupts

- When the IRQF bit of the PPG control status register (PCNTL) is set to "1" and an interrupt request is enabled (PCNTL: IREN = 1), control cannot be returned from interrupt processing. Always clear the IRQF bit.
- Since the 16-bit PPG timer shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.  
Also, when EI<sup>2</sup>OS is used by the 16-bit PPG timer, shared resource interrupts must be disabled.

### 13.8 Sample Programs for the 16-bit PPG Timer

**This section contains sample programs for the 16-bit PPG timer.**

## ■ Sample Program for the 16-bit PPG Timer

- Processing

- An output in 160 kHz with 60% duty is generated with 16-bit PPG timer 0.
- The timer is used in PWM mode to repeatedly generate an interrupt.
- The timer is started with a software trigger.
- EI<sup>2</sup>OS is not used.
- 16 MHz is used for the machine clock, and 62.5 ns is used for the count clock.

- Coding example

ICR01	EQU	0000B1H	;Interrupt control register for the 16-bit PPG timer
PCSR0	EQU	00003AH	;PPG period setting register
PDUT0	EQU	00003CH	;PPG duty setting register
PCNT0	EQU	00003EH	;PPG control status register
IRQF	EQU	PCNT0:4	;Interrupt request flag bit
;-----Main program-----			
CODE		CSEG	
START:			
;	:		;Assumes that stack pointer (SP) has already been initialized
	AND	CCR,#0BFH	;Interrupt disable
	MOV	I:ICR01,#00H	;Interrupt level 0 (strongest)
	MOVW	I:PCSR0,#0063H	;Sets the period of the PPG output
	MOVW	I:PDUT0,#003BH	;Sets the duty ratio of the PPG output
	MOVW	I:PCNT0,#01100000000100110B	;Enables PPG output in normal polarity
			;Enables 16-bit PPG timer, and 62.5 ns clock
			;Software triggers PPG
			;Select PWM mode and enable interrupt
			;Clears interrupt flag, and starts counter
	MOV	ILM,#07H	;Sets ILM in PS to level 7
	OR	CCR,#40H	;Interrupt enable
LOOP:	MOV	A,#00H	;Endless loop
	MOV	A,#01H	;
	BRA	LOOP	;



```

;-----Interrupt program-----
WARI:
        CLRB    I:IRQF                ;Clears interrupt request flag
;
;        :
;        User processing
;        :
        RETI                ;Returns from interrupt

CODE    ENDS

;-----Vector setting-----
VECT    CSEG    ABS=0FFH
        ORG     0FFC4H                ;Sets vector for interrupt #14 (0EH)
        DSL     WARI
        ORG     0FFDCH                ;Sets reset vector
        DSL     START
        DB      00H                ;Sets single-chip mode
VECT    ENDS
        END     START

```

# **CHAPTER 14**

---

## ***MULTI-FUNCTIONAL TIMER***

**This chapter describes the functions and operation of the multi-functional timer.**

- 14.1 Overview of Multi-functional Timer
- 14.2 Block Diagram of Multi-functional Timer
- 14.3 Multi-functional Timer Pins
- 14.4 Registers of Multi-functional Timer
- 14.5 Multi-functional Timer Interrupts
- 14.6 Operation of Multi-functional Timer
- 14.7 Usage Notes on the Multi-functional Timer
- 14.8 Sample Programs for the Multi-functional Timer

## 14.1 Overview of Multi-functional Timer

---

The multi-functional timer consists of a 16-bit free-run timer, six 16-bit output compare, four 16-bit input capture, 1 channel of 16-bit PPG timer and a waveform generator. By using this waveform generator, 12 independent waveform can be outputted through 16-bit free-run timer. Furthermore input pulse width measurement and external clock cycle measurement can be done.

---

### ■ 16-bit Free-run Timer (× 1)

- The 16-bit free-run timer consists of a 16-bit up/up-down counter, control register, 16-bit compare clear register (with buffer register) and a prescaler.
- 8 types of counter operation clock ( $\phi$ ,  $\phi/2$ ,  $\phi/4$ ,  $\phi/8$ ,  $\phi/16$ ,  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ ) can be selected ( $\phi$  is the machine clock).
- Compare clear interrupt is generated when there is a compare match with compare clear register and 16-bit free-run timer. Zero detection interrupt is generated while 16-bit free-run timer is detected as zero in count value.
- The compare clear register has a selectable buffer register, into which data is written for transfer to the compare clear register. When the timer is stopped, transfer occurs immediately when the data is written to the buffer. When the timer is operation, data transfer from the buffer occurs when the timer value is detected to be zero.
- Reset, software clear, compare match with compare clear register in up-count mode will reset the counter value to "0000<sub>H</sub>".
- The output value of this counter can be used as the count clock of the output compares and input captures in multi-functional timer.

### ■ 16-bit Output Compare (× 6)

- The output compare consists of six 16-bit compare registers (with selectable buffer register), compare output latch and compare control registers. An interrupt is generated and output level is inverted when the value of 16-bit free-run timer and compare register are matched.
- 6 compare registers can be operated independently.  
Output pins and interrupt flag are corresponding to each compare register.
- 2 compare registers can be paired to control the output pins.  
Inverts output pins by using 2 compare registers together.
- Setting the initial value for each output pin is possible.
- An interrupt is generated when output compare register is matched with 16-bit free-run timer.

### ■ 16-bit Input Capture (× 4)

Input capture consists of 4 independent external input pins, the corresponding capture register and capture control register. By detecting any edge of the input signal from the external pin, the value of the 16-bit free-run timer can be stored in the capture register and an interrupt is generated simultaneously.

- 3 types of trigger edge (rising edge, falling edge and both edge) of the external input signal can be selected and there is indication bit to show the trigger edge is rising or falling.
- 4 input captures can be operated independently.
- An interrupt is generated by detecting a valid edge from external input.
- Channel 0 and 1 share interrupt #33.
- Channel 2 and 3 share interrupt #35.

### ■ 16-bit PPG Timer (× 1)

The 16-bit PPG timer 0 is used to provide a PPG signal for waveform generator. The detail of 16-bit PPG timer 0 is described in Chapter 13.

### ■ Waveform Generator

The waveform generator consists of three 16-bit timer registers, three timer control registers and 16-bit waveform control register.

With waveform generator, it is possible to generate real-time output, 16-bit PPG waveform output, non-overlap 3-phase waveform output for inverter control and DC chopper waveform output.

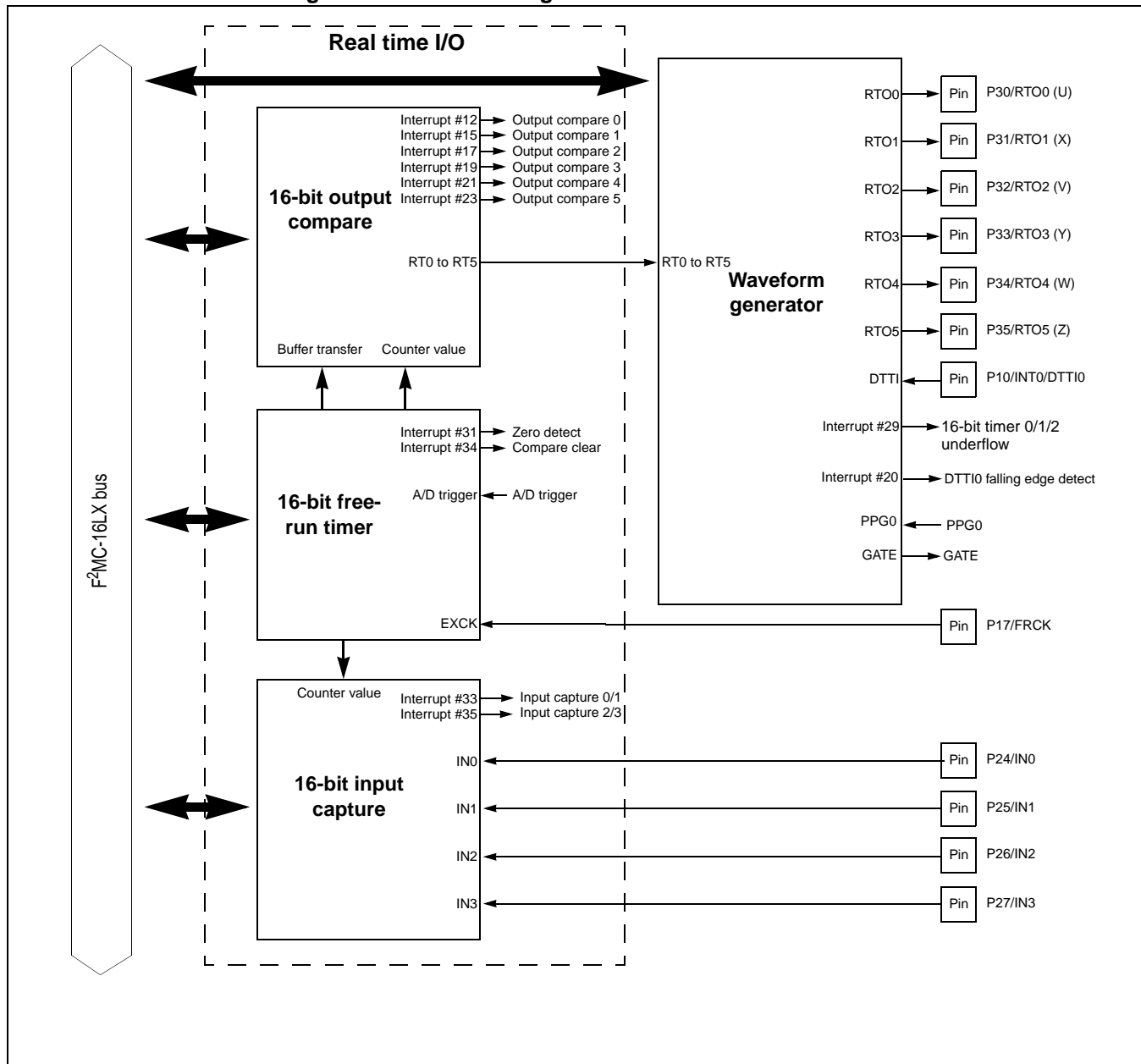
- It is possible to generate a non-overlap waveform output based on dead-time of 16-bit timer. (Dead-time timer function)
- It is possible to generate a non-overlap waveform output when real-time output is operated in 2-channel mode. (Dead-time timer function)
- By detecting real-time output compare match, GATE signal of the PPG timer operation will be generated to start or stop PPG timer operation. (GATE function)
- When a match is detected by real-time output compare, the 16-bit timer is activated. The PPG timer can be started or stopped easily by generating a GATE signal for PPG operation until the 16-bit timer stops. (GATE function)
- Forced stop control using DTTI0 pin input

## 14.2 Block Diagram of Multi-functional Timer

The block diagram of the multi-functional timer will be described in the following sections.

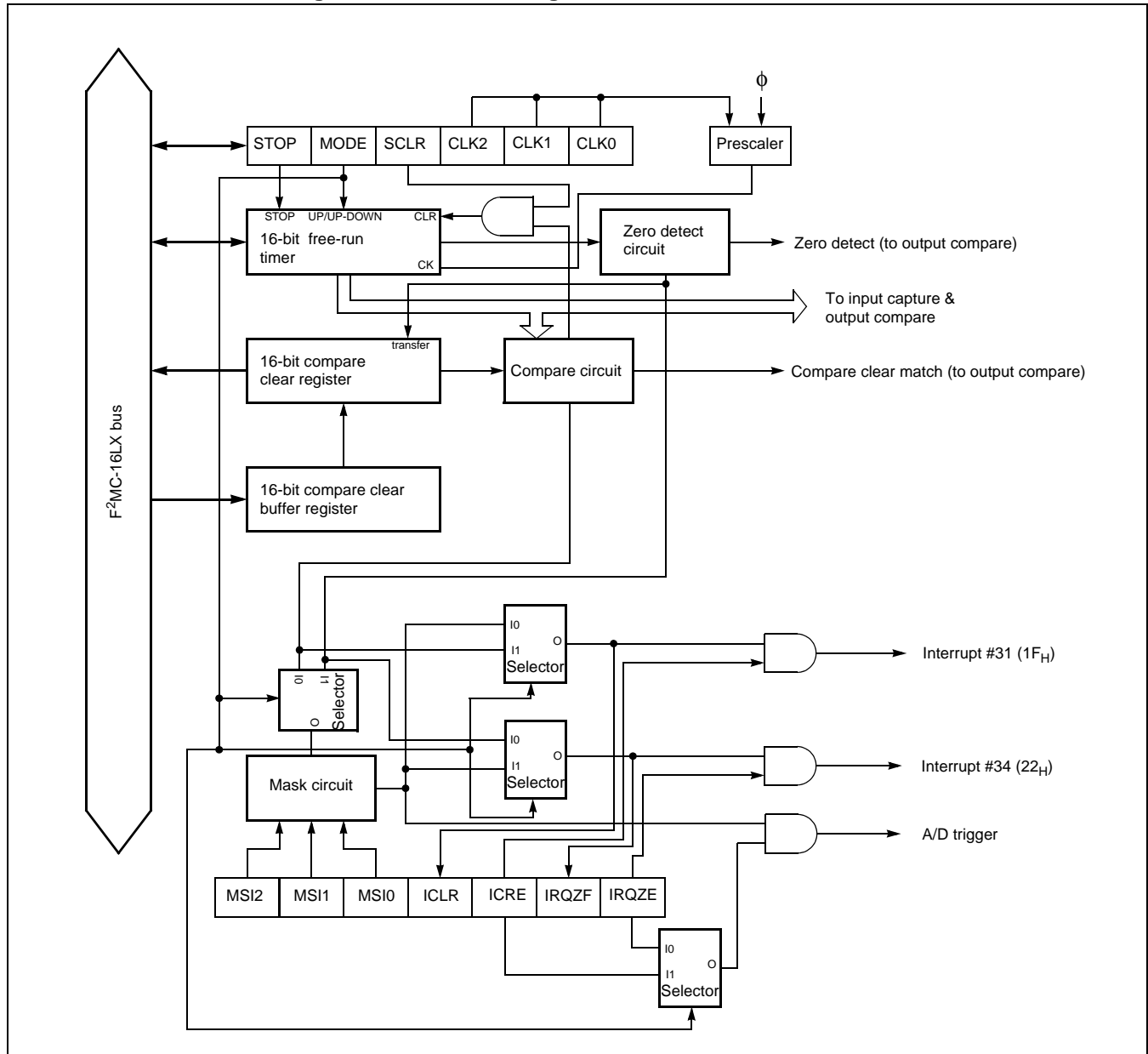
### ■ Block Diagram of Multi-functional Timer

Figure 14.2-1 Block Diagram of Multi-functional Timer



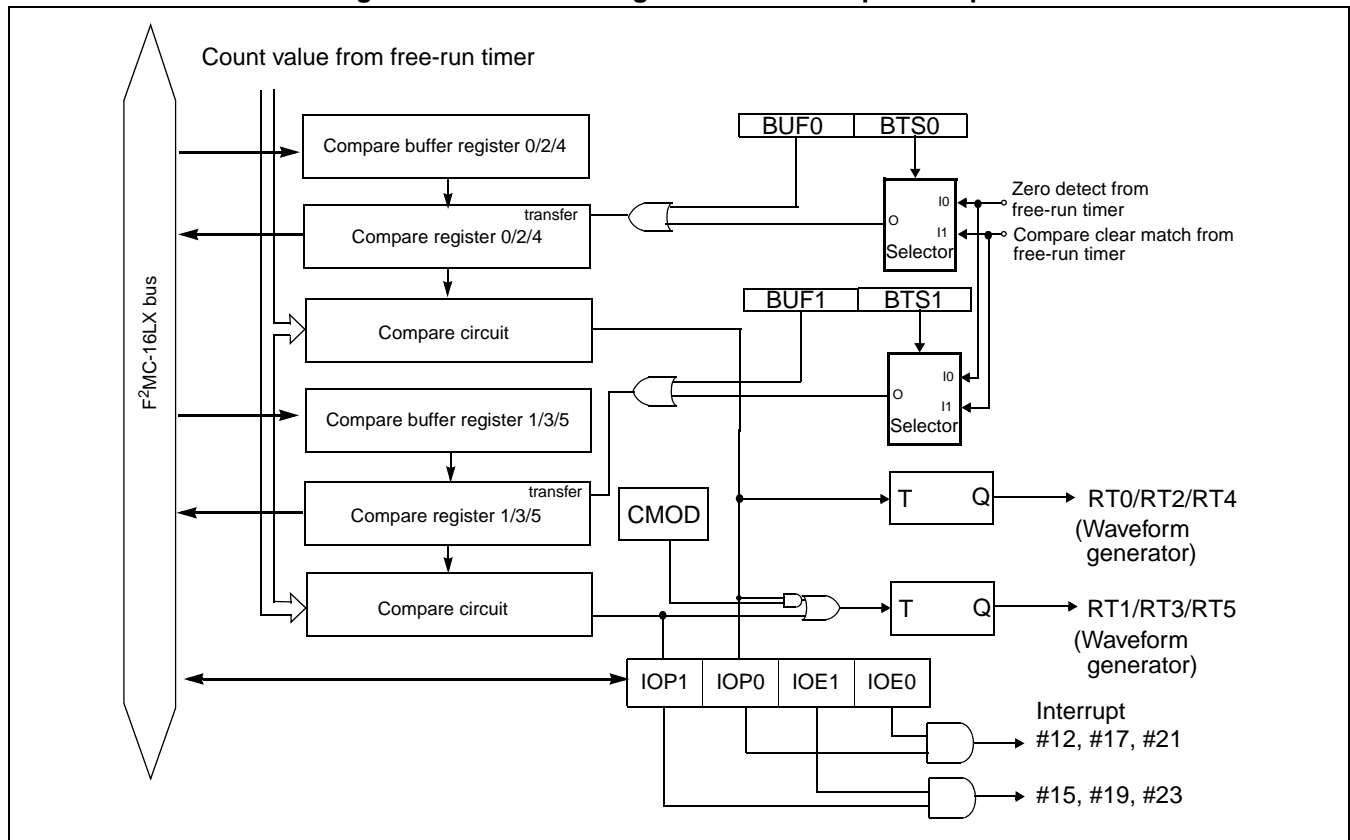
## ■ Block Diagram of 16-bit Free-run Timer

Figure 14.2-2 Block Diagram of 16-bit Free-run Timer



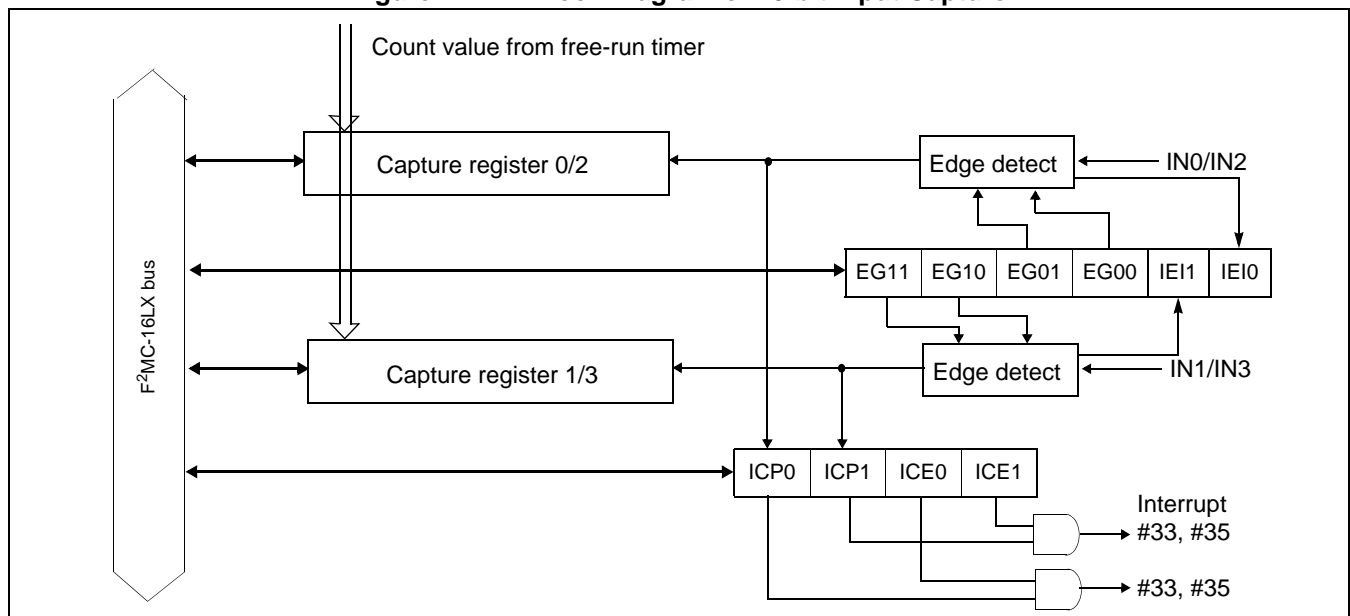
## ■ Block Diagram of 16-bit Output Compare

Figure 14.2-3 Block Diagram of 16-bit Output Compare



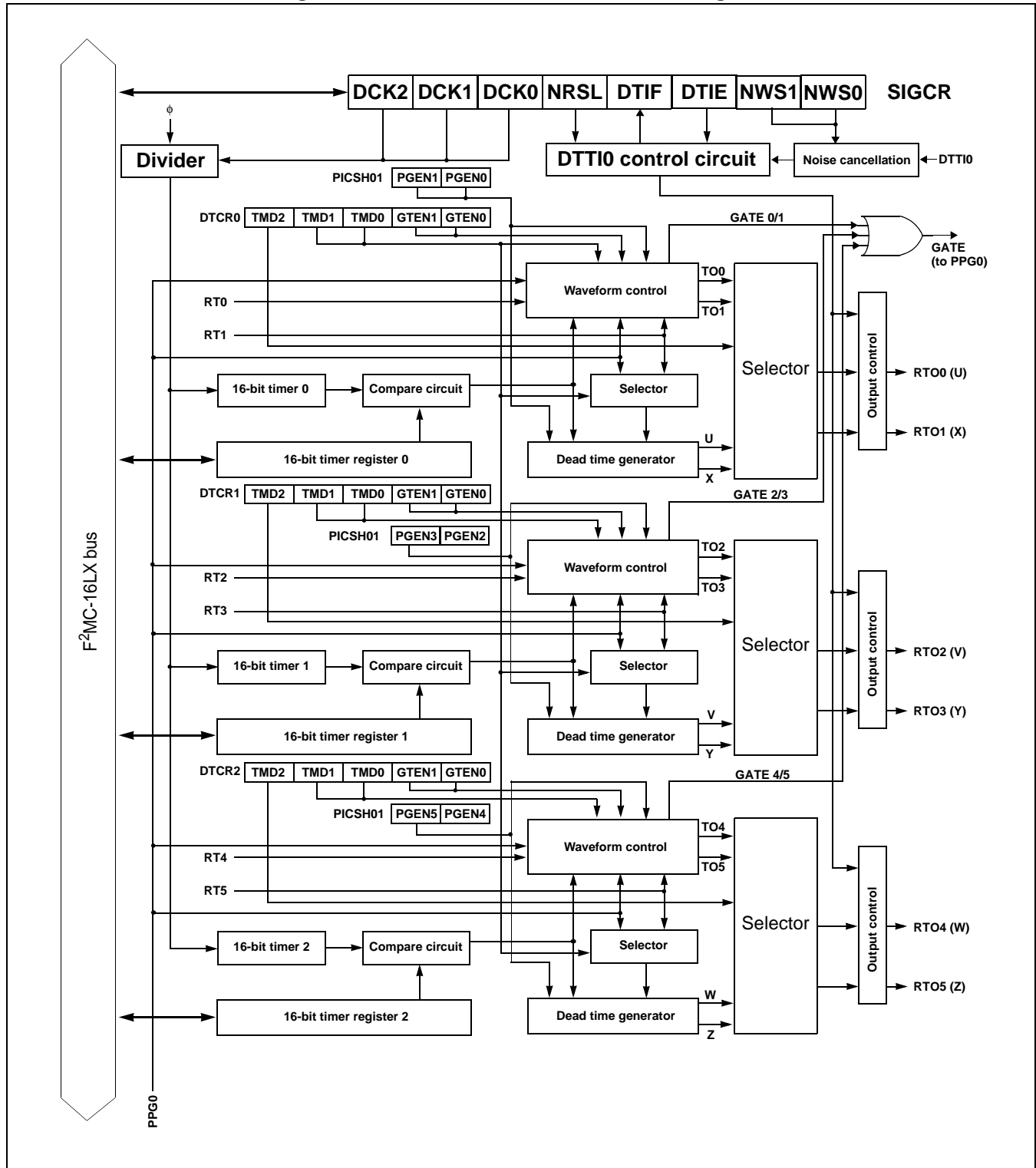
## ■ Block Diagram of 16-bit Input Capture

Figure 14.2-4 Block Diagram of 16-bit Input Capture



## ■ Block Diagram of Waveform Generator

Figure 14.2-5 Waveform Generator Block Diagram





## 14.3 Multi-functional Timer Pins

This section describes the pins of the multi-functional timer and provides a pin block diagram.

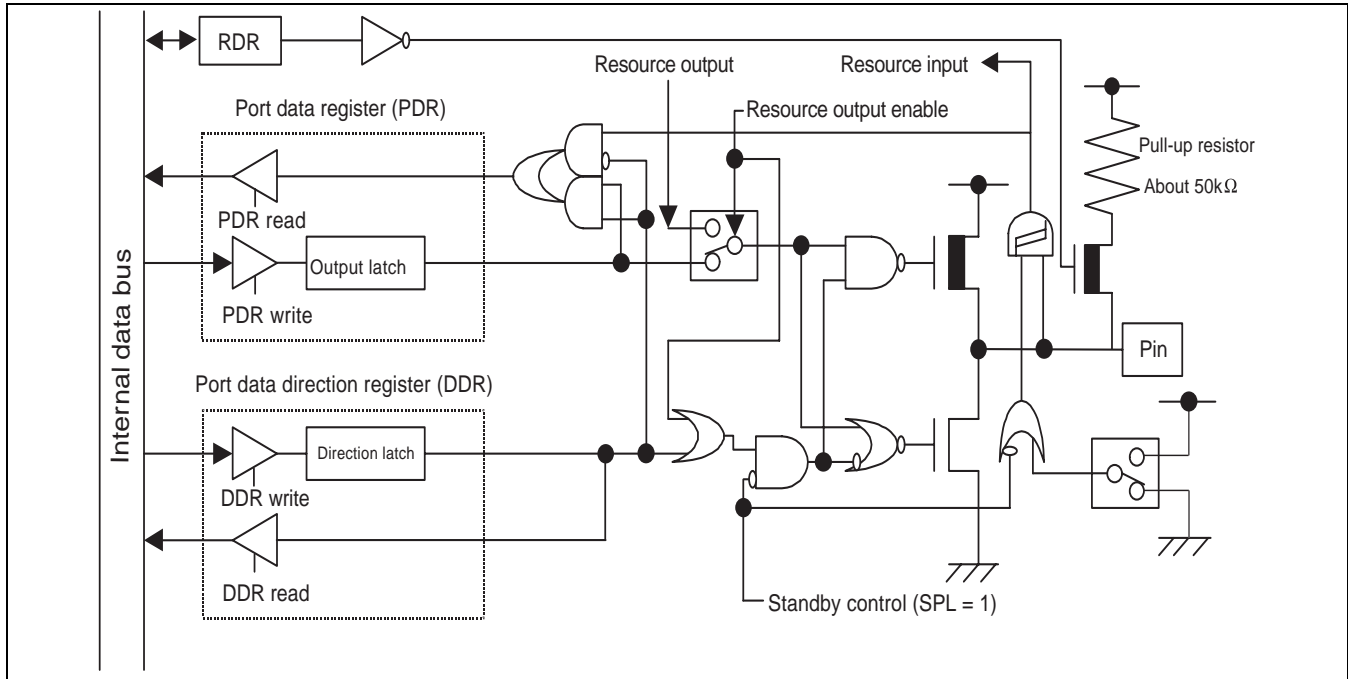
### ■ Multi-functional Timer Pins

Table 14.3-1 Multi-functional Timer Pins

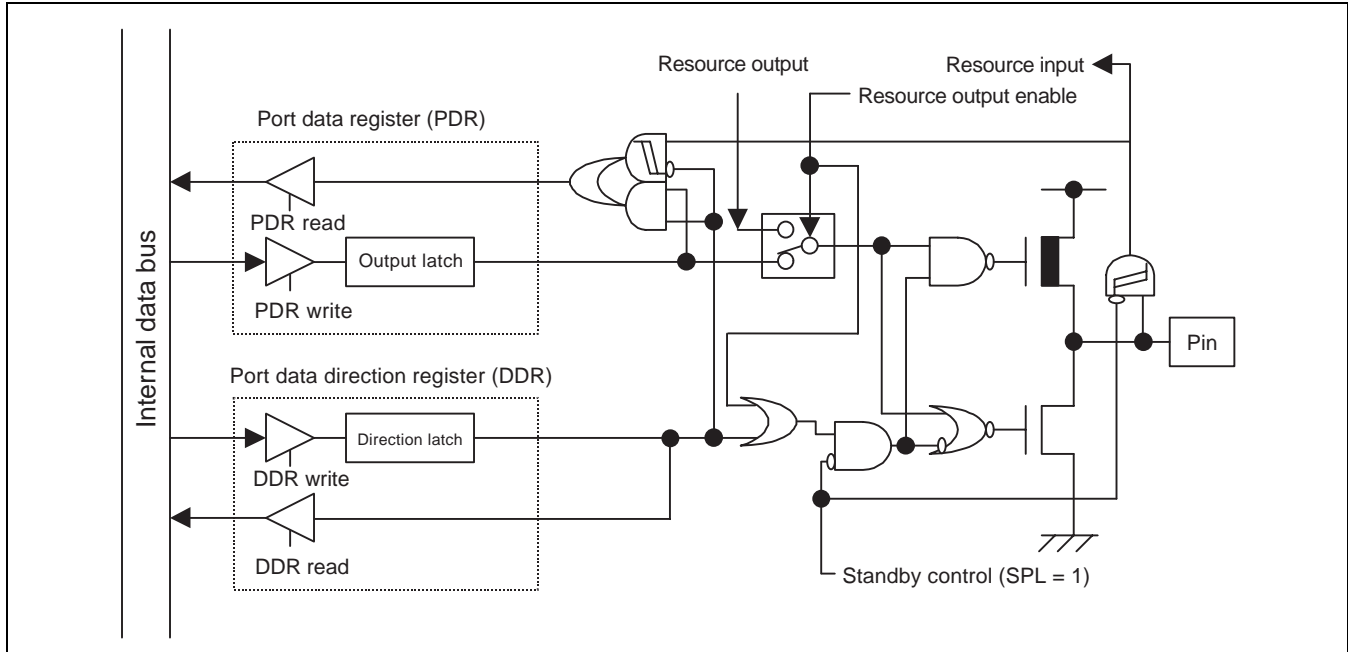
Pin Name	Pin function	I/O format	Pull-up option	Standby control	Setting required for pins
P10/INT0/ DTTI0	Port 1 input-output/ external interrupt input/ DTTI0	CMOS output/ CMOS hysteresis input	Selectable	Provided	Set the pin as an input port (DDR1:bit8 = 0)
P17/FRCK	Port 1 input-output/ external clock				Set the pin as an input port (DDR1:bit15 = 0)
P24/IN0	Port 2 input-output/ input capture 0		Set the pin as an input port (DDR2:bit4 = 0)		
P25/IN1	Port 2 input-output/ input capture 1		Set the pin as an input port (DDR2:bit5 = 0)		
P26/IN2	Port 2 input-output/ input capture 2		Set the pin as an input port (DDR2:bit6 = 0)		
P27/IN3	Port 2 input-output/ input capture 3		Set the pin as an input port (DDR2:bit7 = 0)		
P30/RTO0 (U)	Port 3 input-output/ RTO0	CMOS output/ CMOS input	Not provided		Set RTO0 output (OCS1:OTE0 = 1)
P31/RTO1 (X)	Port 3 input-output/ RTO1				Set RTO1 output (OCS1:OTE1 = 1)
P32/RTO2 (V)	Port 3 input-output/ RTO2				Set RTO2 output (OCS3:OTE0 = 1)
P33/RTO3 (Y)	Port 3 input-output/ RTO3				Set RTO3 output (OCS3:OTE1 = 1)
P34/RTO4 (W)	Port 3 input-output/ RTO4				Set RTO4 output (OCS5:OTE0 = 1)
P35/RTO5 (Z)	Port 3 input-output/ RTO5				Set RTO5 output (OCS5:OTE1 = 1)

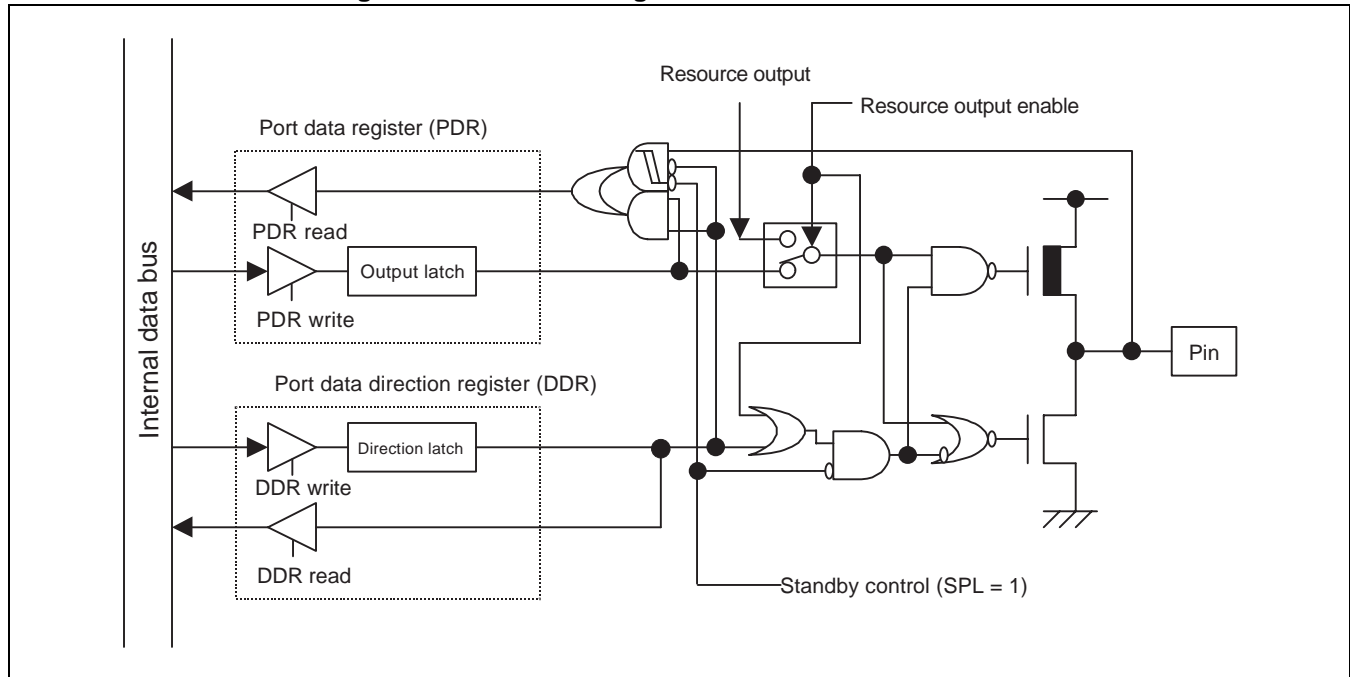
## ■ Block Diagram of Multi-functional Timer Pins

**Figure 14.3-1 Block Diagram of P10/INT0/DTT10, P17/FRCK**



**Figure 14.3-2 Block Diagram of P24/IN0 to P27/IN3**



**Figure 14.3-3 Block Diagram of P30/RT00 to P35/RT05**

## 14.4 Registers of Multi-functional Timer

This section describes registers of multi-functional timer.

### ■ 16-bit Free-run Timer Registers

Figure 14.4-1 Registers of 16-bit Free-run Timer

Compare Clear Buffer Register / Compare Clear Register (Upper)										
	bit	15	14	13	12	11	10	9	8	
Address: 00005B <sub>H</sub>		CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	CPCLRB/CPCLR
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		1	1	1	1	1	1	1	1	

Compare Clear Buffer Register / Compare Clear Register (Lower)										
	bit	7	6	5	4	3	2	1	0	
Address: 00005A <sub>H</sub>		CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	CPCLRB/CPCLR
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		1	1	1	1	1	1	1	1	

Timer Data Register (Upper)										
	bit	15	14	13	12	11	10	9	8	
Address: 00005D <sub>H</sub>		T15	T14	T13	T12	T11	T10	T09	T08	TCDT
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	

Timer Data Register (Lower)										
	bit	7	6	5	4	3	2	1	0	
Address: 00005C <sub>H</sub>		T07	T06	T05	T04	T03	T02	T01	T00	TCDT
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	

Timer Control Status Register (Upper)										
	bit	15	14	13	12	11	10	9	8	
Address: 00005F <sub>H</sub>		ECKE	IRQZF	IRQZE	MSI2	MSI1	MSI0	ICLR	ICRE	TCCSH
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	

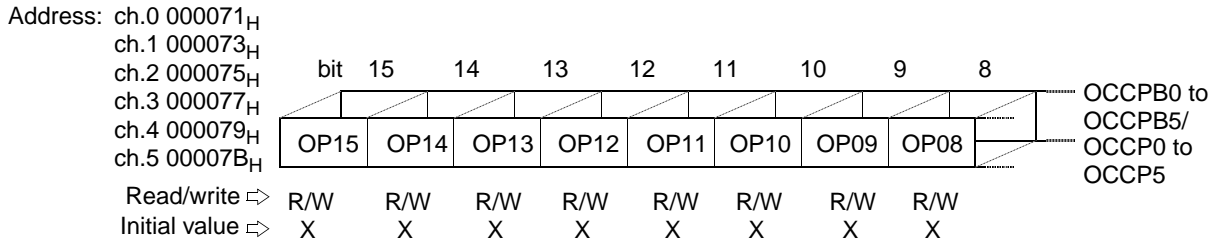
  

Timer Control Status Register (Lower)										
	bit	7	6	5	4	3	2	1	0	
Address: 00005E <sub>H</sub>		—	BFE	STOP	MODE	SCLR	CLK2	CLK1	CLK0	TCCSL
Read/write ⇨		-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		-	0	1	0	0	0	0	0	

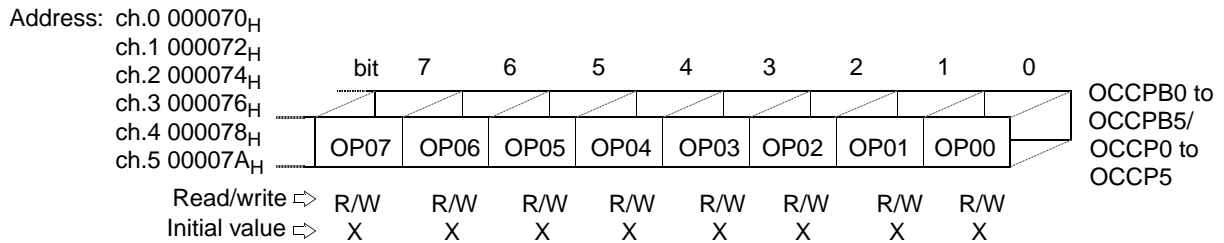
## ■ 16-bit Output Compare Registers

**Figure 14.4-2 Registers of Output Compare**

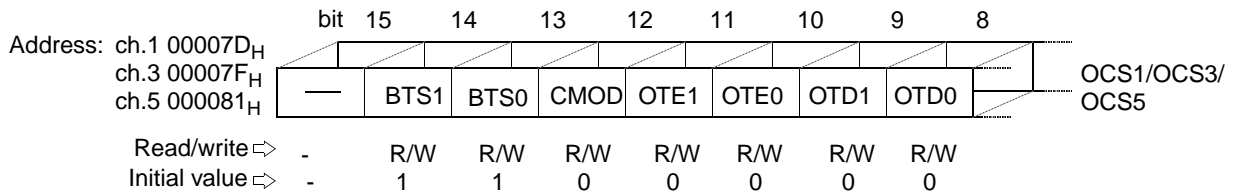
### Output Compare Buffer Register / Output Compare Register (Upper)



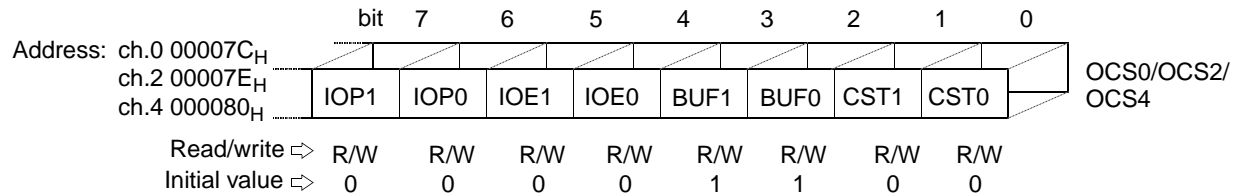
### Output Compare Buffer Register / Output Compare Register (Lower)



### Compare Control Register (Upper)



### Compare Control Register (Lower)



## Input Capture Registers

**Figure 14.4-3 Registers of 16-bit Input Capture**

### Input Capture Data Register (Upper)

Address: ch.0 000061 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
ch.1 000063 <sub>H</sub>									
ch.2 000065 <sub>H</sub>									
ch.3 000067 <sub>H</sub>									
	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	IPCP0 to IPCP3
Read/write ⇒	R	R	R	R	R	R	R	R	
Initial value ⇒	X	X	X	X	X	X	X	X	

### Input Capture Data Register (Lower)

Address: ch.0 000060 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	
ch.1 000062 <sub>H</sub>									
ch.2 000064 <sub>H</sub>									
ch.3 000066 <sub>H</sub>									
	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	IPCP0 to IPCP3
Read/write ⇒	R	R	R	R	R	R	R	R	
Initial value ⇒	X	X	X	X	X	X	X	X	

### Input Capture Control Status Register (2/3) (Upper)

Address: 00006B <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	—	—	—	—	—	—	IEI3	IEI2	ICSH23
Read/write ⇒	-	-	-	-	-	-	R	R	
Initial value ⇒	-	-	-	-	-	-	0	0	

### Input Capture Control Status Register (2/3) (Lower)

Address: 00006A <sub>H</sub>	bit 7	6	5	4	3	2	1	0	
	ICP3	ICP2	ICE3	ICE2	EG31	EG30	EG21	EG20	ICSL23
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	

### PPG output control/ Input Capture Control Status Register (0/1) (Upper)

Address: 000069 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	PGEN5	PGEN4	PGEN3	PGEN2	PGEN1	PGEN0	IEI1	IEI0	PICSH01
Read/write ⇒	R/W	R/W	R/W	R/W	R/W	R/W	R	R	
Initial value ⇒	0	0	0	0	0	0	0	0	

### Input Capture Control Register (0/1)

Address: 000068 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	
	ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	PICSL01
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	

## ■ Waveform Generator Registers

**Figure 14.4-4 Registers of Waveform Generator**

### 16-bit Timer Register (Upper)

	bit	15	14	13	12	11	10	9	8	
Address: ch.0 000051 <sub>H</sub> ch.1 000053 <sub>H</sub> ch.2 000055 <sub>H</sub>		TR15	TR14	TR13	TR12	TR11	TR10	TR09	TR08	TMRR0/TMRR1/ TMRR2
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		X	X	X	X	X	X	X	X	

### 16-bit Timer Register (Lower)

	bit	7	6	5	4	3	2	1	0	
Address: ch.0 000050 <sub>H</sub> ch.1 000052 <sub>H</sub> ch.2 000054 <sub>H</sub>		TR07	TR06	TR05	TR04	TR03	TR02	TR01	TR00	TMRR0/TMRR1/ TMRR2
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		X	X	X	X	X	X	X	X	

### 16-bit Timer Control Register

	bit	15	14	13	12	11	10	9	8	
Address: ch.1 000057 <sub>H</sub>		DMOD	GTEN1	GTEN0	TMIF	TMIE	TMD2	TMD1	TMD0	DTCR1
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	

### 16-bit Timer Control Register

	bit	7	6	5	4	3	2	1	0	
Address: ch.0 000056 <sub>H</sub> ch.2 000058 <sub>H</sub>		DMOD	GTEN1	GTEN0	TMIF	TMIE	TMD2	TMD1	TMD0	DTCR0/DTCR2
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	

### Waveform Control Register

	bit	15	14	13	12	11	10	9	8	
Address: 000059 <sub>H</sub>		DTIE	DTIF	NRSL	DCK2	DCK1	DCK0	NWS1	NWS0	SIGCR
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	

### 14.4.1 Compare Clear Buffer Register (CPCLRB) and Compare Clear Register (CPCLR)

Compare clear buffer register (CPCLRB) is a 16-bit buffer register of compare clear register (CPCLR). Both CPCLRB and CPCLR registers are located in the same address.

#### ■ Compare Clear Buffer Register (CPCLRB)

**Figure 14.4-5 Compare Clear Buffer Register (CPCLRB)**

Compare Clear Buffer Register (Upper)								
bit	15	14	13	12	11	10	9	8
Address: 00005B <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08
Read/write ⇨	W	W	W	W	W	W	W	W
Initial value ⇨	1	1	1	1	1	1	1	1
Compare Clear Buffer Register (Lower)								
bit	7	6	5	4	3	2	1	0
Address: 00005A <sub>H</sub>	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00
Read/write ⇨	W	W	W	W	W	W	W	W
Initial value ⇨	1	1	1	1	1	1	1	1

Compare Clear Buffer Register is the buffer register for Compare Clear Register. When buffer function is disabled (TCCSL:BFE=0) or when free-run timer is stopped, value in Compare clear buffer register is transferred to Compare clear register immediately. When buffer function is enabled, value is transferred when the count value of 16-bit free-run timer is detected as zero.

Word access to this register is recommended.

#### ■ Compare Clear Register (CPCLR)

**Figure 14.4-6 Compare Clear Register (CPCLR)**

Compare Clear Register (Upper)								
bit	15	14	13	12	11	10	9	8
Address: 00005B <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08
Read/write ⇨	R	R	R	R	R	R	R	R
Initial value ⇨	1	1	1	1	1	1	1	1
Compare Clear Register (Lower)								
bit	7	6	5	4	3	2	1	0
Address: 00005A <sub>H</sub>	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00
Read/write ⇨	R	R	R	R	R	R	R	R
Initial value ⇨	1	1	1	1	1	1	1	1

The Compare Clear Register is used to compare with the count value of the 16-bit free-run timer. In up-count mode, when this register is matched with the count value of 16-bit free-run timer, timer will be reset to "0000<sub>H</sub>". In up-down count mode, when this register is matched with the count value of the 16-bit free-run timer, the timer changes from up-count to down-count and changes from down-count to up-count at zero detect.

Word access to this register is recommended.



## 14.4.2 Timer Data Register (TCDT)

The timer data register (TCDT) is used to read the count value of 16-bit free-run timer.

### ■ Timer Data Register (TCDT)

Figure 14.4-7 Timer Data Register

Timer Data Register (Upper)									
	bit	15	14	13	12	11	10	9	8
Address: 00005D <sub>H</sub>		T15	T14	T13	T12	T11	T10	T09	T08
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
Timer Data Register (Lower)									
	bit	7	6	5	4	3	2	1	0
Address: 00005C <sub>H</sub>		T07	T06	T05	T04	T03	T02	T01	T00
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

The timer data register is used to read the count value of the 16-bit free-run timer. The counter value is cleared to "0000<sub>H</sub>" upon a reset. The timer value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (STOP = 1). Word access instruction to the timer data register is recommended.

The 16-bit free-run timer is initialized upon the following factors:

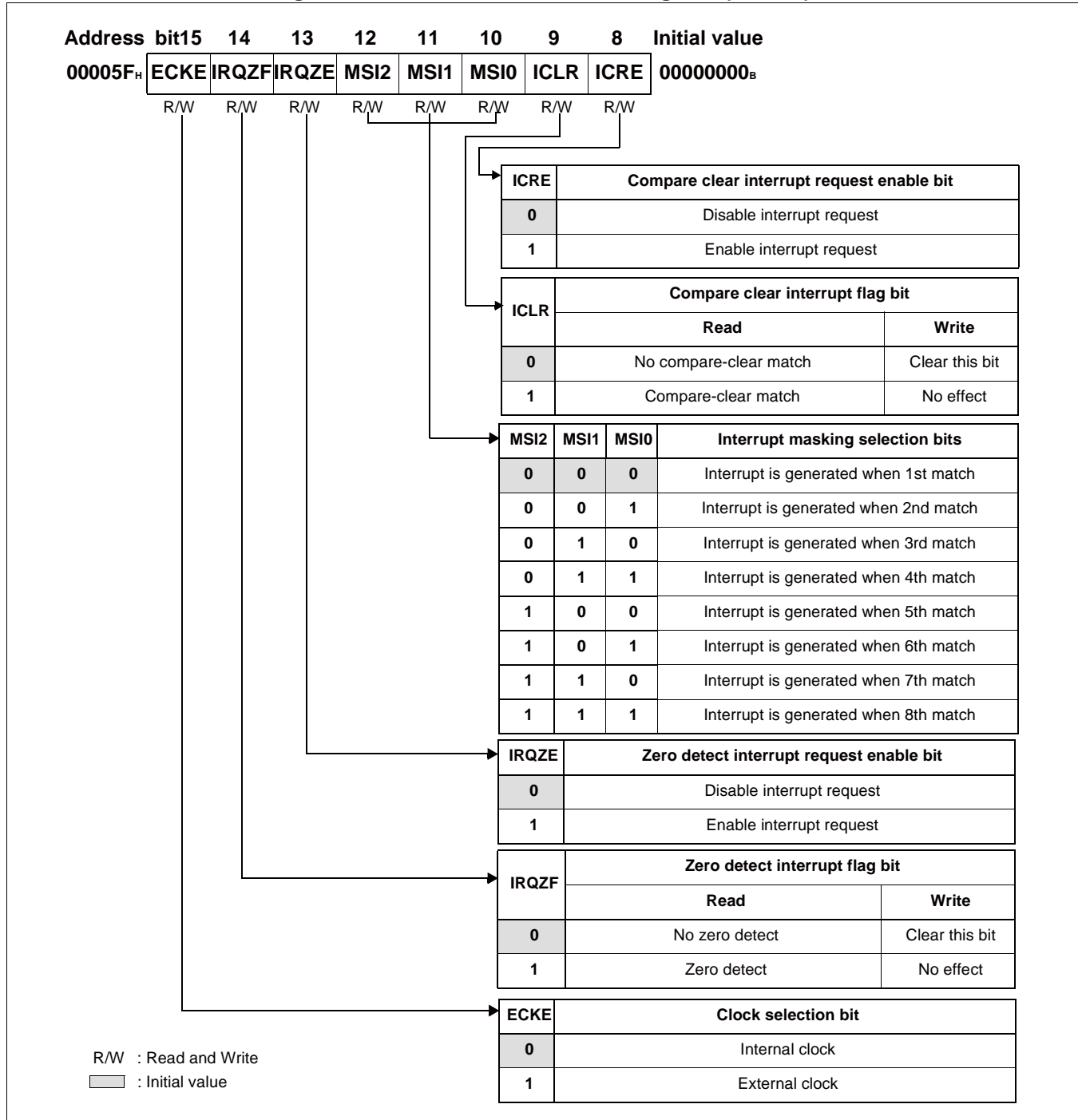
- Reset
- Clear bit (SCLR) of control status register
- A match between compare clear register and the timer counter value in up-count mode (TCCSL:MODE=0)

### 14.4.3 Timer Control Status Register (TCCSH, TCCSL)

The timer control status register (TCCS) is a 16-bit register and used to control the operation of 16-bit free-run timer.

#### ■ Timer Control Status Register, Upper Byte (TCCSH)

Figure 14.4-8 Timer Control Status Register (TCCSH)



**Table 14.4-1 Timer Control Status Register (TCCSH)**

Bit name		Function
bit15	ECKE: Clock selection bit	<ul style="list-style-type: none"> <li>This bit is used to select internal or external clock as count clock for 16-bit free-run timer.</li> <li>Writing “0” selects internal clock. The clock frequency selection bits (CK2 to CK0) should also be set to select the count clock frequency.</li> <li>Writing “1” selects external clock. External clock is input from pin “P17/FRCK”, so DDR1:7 should be set as “0” to enable external clock input.</li> </ul> <p>(Note) The count clock is changed immediately after this bit is set. So change this bit while the output compare and input capture units are stopped.</p>
bit14	IRQZF: Zero detect interrupt flag bit	<ul style="list-style-type: none"> <li>This bit is an interrupt flag for zero detect.</li> <li>When the count value of 16-bit free-run timer is “0000<sub>H</sub>”, this bit is set to “1”.</li> <li>Writing “0” will clear this bit.</li> <li>Writing “1” has no effect.</li> <li>In read-modify-write operation, “1” is always read.</li> </ul> <p>(Note)</p> <ul style="list-style-type: none"> <li>In software clear, (writing TCCSL:SCLR “1”) will not set this bit.</li> <li>In up-down count mode (MODE=1) and interrupt mask function is selected (MSI2 to MSI0 not equals 000<sub>B</sub>), this bit will only be set after the number of zero detect is masked.</li> <li>In up-count mode (MODE=0), this bit is set at every zero detect disregarding the value of MSI2 to MSI0.</li> </ul>
bit13	IRQZE: Zero detect interrupt request enable bit	<ul style="list-style-type: none"> <li>This is the interrupt request enable bit for the zero detect.</li> <li>When this bit is “1” and the interrupt flag (bit14: IRQZF) is set to “1”, an interrupt request will be generated to CPU.</li> </ul>
bit12 to bit10	MSI2 to MSI0: Interrupt mask selection bits	<ul style="list-style-type: none"> <li>These bits are used to set the number of times of masking the compare clear interrupt in up-count mode (MODE=0) or zero detect interrupt in up-down count mode (MODE=1).</li> <li>No interrupt cause is masked when MSI2 to MSI0 equals zero.</li> </ul> <p>(Note) To mask the interrupt cause twice and perform interrupt processing at the third time, MSI2 to MSI0 should be set as 010<sub>B</sub>.</p>
bit9	ICLR: Compare clear interrupt flag bit	<ul style="list-style-type: none"> <li>This bit is an interrupt flag for compare clear.</li> <li>When the compare clear value and 16-bit free-run timer value are matched, this bit is set to “1”.</li> <li>Writing “0” will clear this bit.</li> <li>Writing “1” has no effect.</li> <li>In read-modify-write operation, “1” is always read.</li> </ul> <p>(Note)</p> <ul style="list-style-type: none"> <li>In up-count mode (MODE=0) and interrupt mask function is selected (MSI2 to MSI0 not equals 000<sub>B</sub>), this bit will only be set after the number of compare clear is masked.</li> <li>In up-down count mode (MODE=1), this bit is set at every compare clear disregarding the value of MSI2 to MSI0.</li> </ul>
bit8	ICRE: Compare clear interrupt request enable bit	<ul style="list-style-type: none"> <li>This is the interrupt request enable bit for the compare clear.</li> <li>When this bit is “1” and the interrupt flag (bit9: ICLR) is set to “1”, an interrupt request will be generated to CPU.</li> </ul>

## ■ Timer Control Status Register, Lower Byte (TCCSL)

Figure 14.4-9 Timer Control Status Register (TCCSL)

Address	bit7	6	5	4	3	2	1	0	Initial value
00005E <sub>H</sub>	—	BFE	STOP	MODE	SCLR	CLK2	CLK1	CLK0	-0100000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

CLK2	CLK1	CLK0	Clock frequency selection bit				
			Count clock	$\phi = 16 \text{ MHz}$	$\phi = 8 \text{ MHz}$	$\phi = 4 \text{ MHz}$	$\phi = 1 \text{ MHz}$
0	0	0	$\phi$	62.5 ns	125 ns	0.25 $\mu\text{s}$	1 $\mu\text{s}$
0	0	1	$\phi/2$	125 ns	0.25 $\mu\text{s}$	0.5 $\mu\text{s}$	2 $\mu\text{s}$
0	1	0	$\phi/4$	0.25 $\mu\text{s}$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	4 $\mu\text{s}$
0	1	1	$\phi/8$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	2 $\mu\text{s}$	8 $\mu\text{s}$
1	0	0	$\phi/16$	1 $\mu\text{s}$	2 $\mu\text{s}$	4 $\mu\text{s}$	16 $\mu\text{s}$
1	0	1	$\phi/32$	2 $\mu\text{s}$	4 $\mu\text{s}$	8 $\mu\text{s}$	32 $\mu\text{s}$
1	1	0	$\phi/64$	4 $\mu\text{s}$	8 $\mu\text{s}$	16 $\mu\text{s}$	64 $\mu\text{s}$
1	1	1	$\phi/128$	8 $\mu\text{s}$	16 $\mu\text{s}$	32 $\mu\text{s}$	128 $\mu\text{s}$

$\phi$ : Machine cycle

SCLR	Timer clear bit	
	Write	Read
	0: Clear SCLR bit 1: Initialize counter to "0000 <sub>H</sub> "	Always read as "0"

MODE	Timer counting mode
0	up-count mode
1	up-down count mode

STOP	Timer enable bit
0	Counting is enabled (operation)
1	Counting is disabled (stop)

BFE	Compare clear buffer enable bit
0	Disable compare clear buffer
1	Enable compare clear buffer

R/W : Read and write  
 : Initial value  
 — : Not used

**Table 14.4-2 Timer control status register (TCCSL)**

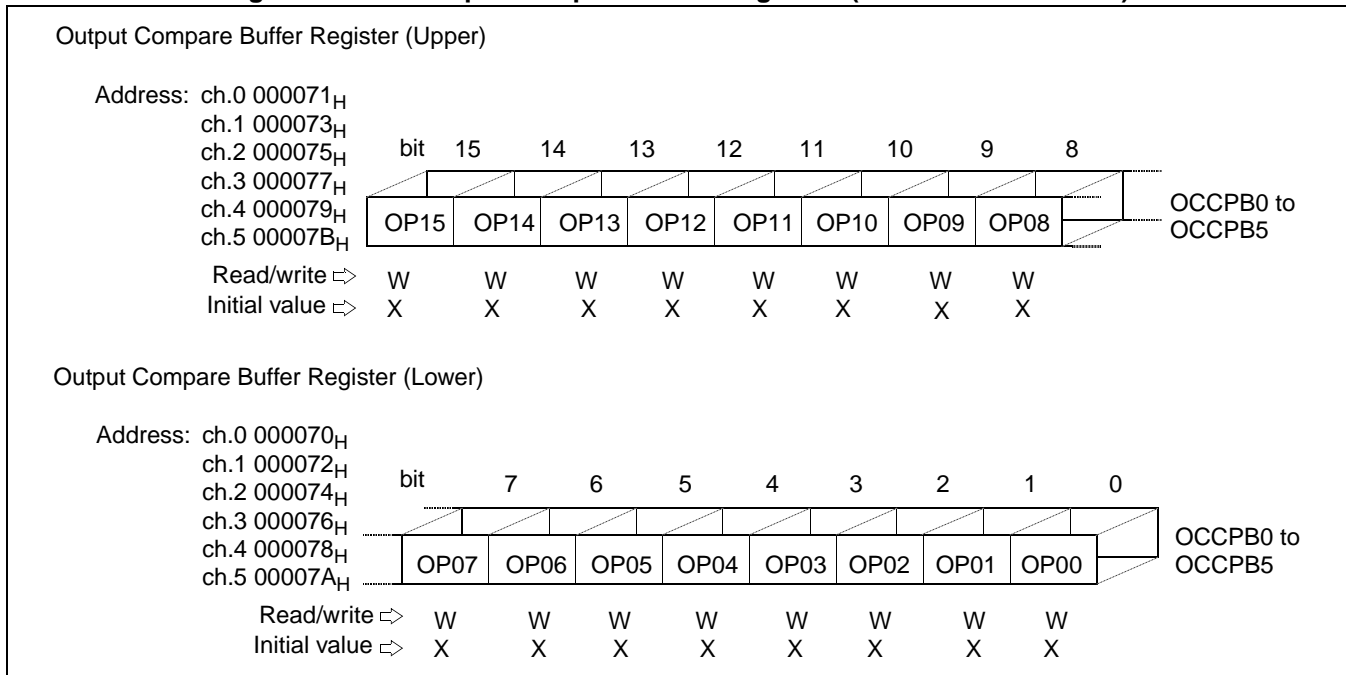
Bit name		Function
bit7	Unused bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to this bit has no effect on the operation.</li> </ul>
bit6	BFE: Compare clear buffer enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable compare clear buffer.</li> <li>Writing “0” disables compare clear buffer. Directly write in compare clear register is possible.</li> <li>Writing “1” enables compare clear buffer. Data written in compare clear buffer register will be held and transfer to compare clear register when the count value of 16-bit free-run timer is detected as zero.</li> </ul>
bit5	STOP: Timer enable bit	<ul style="list-style-type: none"> <li>This bit is used to stop/start the counting of the 16-bit free-run timer.</li> <li>Writing “1” stops the counting of the 16-bit free-run timer.</li> <li>Writing “0” starts the counting of the 16-bit free-run timer.</li> </ul> <p>(Note) When the 16-bit free-run timer is stopped, the output compare operation will also be stopped.</p>
bit4	MODE: Timer counting mode bit	<ul style="list-style-type: none"> <li>This bit is used to select the count mode of the 16-bit free-run timer.</li> <li>Writing “0” selects up-count mode. Timer counts up until counter value matches with compare clear register and reset to “0000<sub>H</sub>” and then counts up again.</li> <li>Writing “1” selects up-down count mode. In up-down count mode, whenever zero in the timer data register is detected, the timer counting direction will always be reseted to up-counting. The timer will reverse its counting direction whenever the timer value matches with compare clear register.</li> <li>This bit can be written at any time whether the timer is operating or stopped. The value written to this bit is buffered and the count mode will be changed when timer value is “0000<sub>H</sub>”.</li> </ul> <p>(Note) Because the timer will reverse its counting direction when compare-match is detected in up-down count mode (MODE = 1), it should be careful to set the compare clear register and timer data register when the timer is being counted down.</p>
bit3	SCLR: Timer clear bit	<ul style="list-style-type: none"> <li>This bit is used to initialize the 16-bit free-run timer to “0000<sub>H</sub>”.</li> <li>Writing “1” initializes 16-bit free-run timer to “0000<sub>H</sub>” at the next count clock.</li> <li>Writing “0” will clear the bit SCLR if it is “1”.</li> <li>Read value is always “0”.</li> </ul> <p>(Note)</p> <ul style="list-style-type: none"> <li>This bit cannot be used to initialize the timer when timer stops (STOP=1). Writing “0000<sub>H</sub>” to timer data register (TCDT) can initialize the timer.</li> <li>Writing “1” will not generate zero detect interrupt.</li> <li>This bit will be cleared by hardware after the timer is initialized to “0000”. If “0” is written to the bit before timer initialization, the bit is cleared and the timer did not init</li> <li>Even after “1” is written, the counter value is not initialized if “0” is written to this bit before the next count clock.</li> </ul>
bit2 to bit0	CLK2 to CLK0: Clock frequency selection bit	<ul style="list-style-type: none"> <li>This bit is used to select count clock for the 16-bit free-run timer.</li> <li>The count clock is changed immediately after these bits are set. So change them while the output compare and input capture units are stopped.</li> </ul>

## 14.4.4 Output Compare Buffer Registers (OCCPB0 to OCCPB5) / Output Compare Registers (OCCP0 to OCCP5)

Output compare buffer register (OCCPB) is a 16-bit buffer register of output compare register (OCCP). Both OCCPB and OCCP registers are located in the same address.

### ■ Output Compare Buffer Registers (OCCPB0 to OCCPB5)

Figure 14.4-10 Output Compare Buffer Registers (OCCPB0 to OCCPB5)

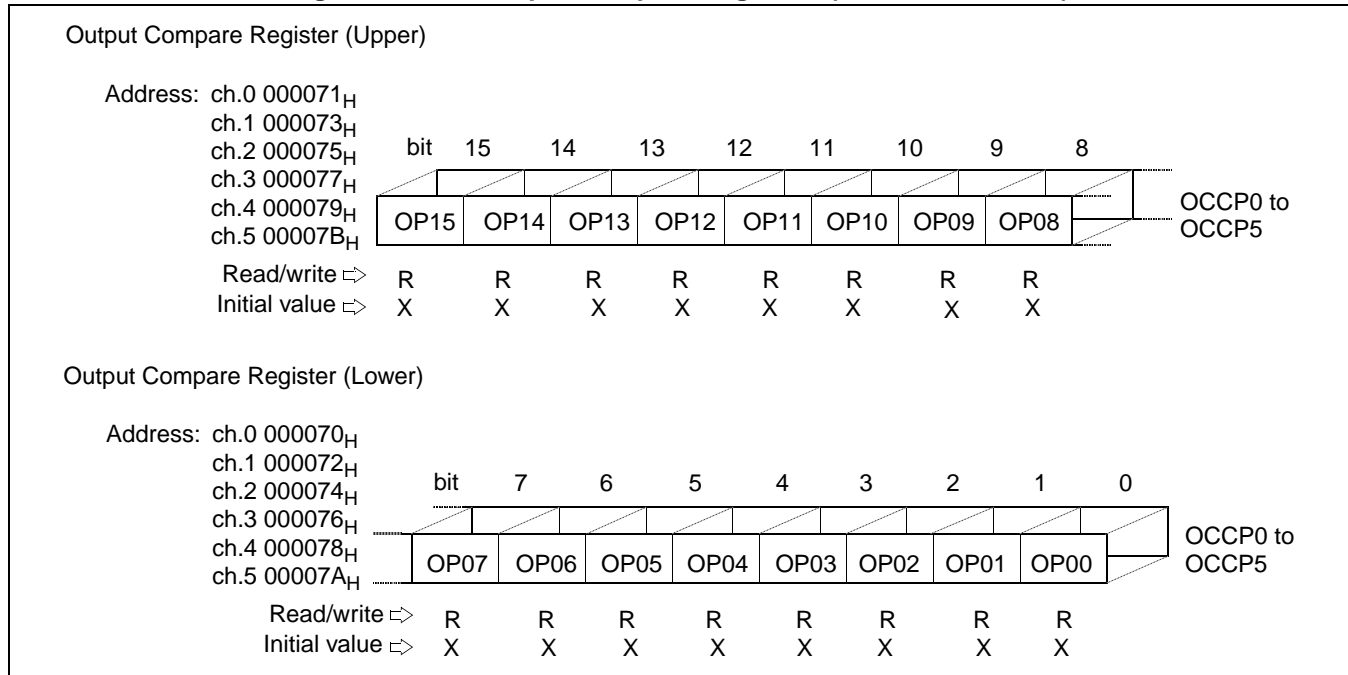


Output compare buffer register is the buffer register of output compare register (OCCP). When buffer function is disabled (OCS0/OCS2/OCS4:BUF0/BUF1=1) or when free-run timer is stopped, value in output compare buffer register is transferred to output compare register immediately. When buffer function is enabled (OCS0/OCS2/OCS4:BUF0/BUF1=0), value is transferred at compare clear match or zero detection depending on transfer selection bit BTS in compare control register (OCS1/OCS3/OCS5).

Word access to this register is recommended.

## ■ Output Compare Registers (OCCP0 to OCCP5)

**Figure 14.4-11 Output Compare Registers (OCCP0 to OCCP5)**



The output compare register is a 16-bit register which is used to compare the count value of 16-bit free-run timer. The initial value of the output compare register is undetermined, so output compare buffer register (OCCPB) must be set with a value before enabling the operation.

When the value of the output compare register matches the count value of 16-bit free-run timer, a compare signal is generated to set the output compare interrupt flag (OCS0/OCS2/OCS4:IOP0/IOP1). If output level is set (OCS1/OCS3/OCS5:OTD0/OTD1), the output level of RT0 to RT5 corresponding to the output compare register (OCCP0 to OCCP5) can be reversed.

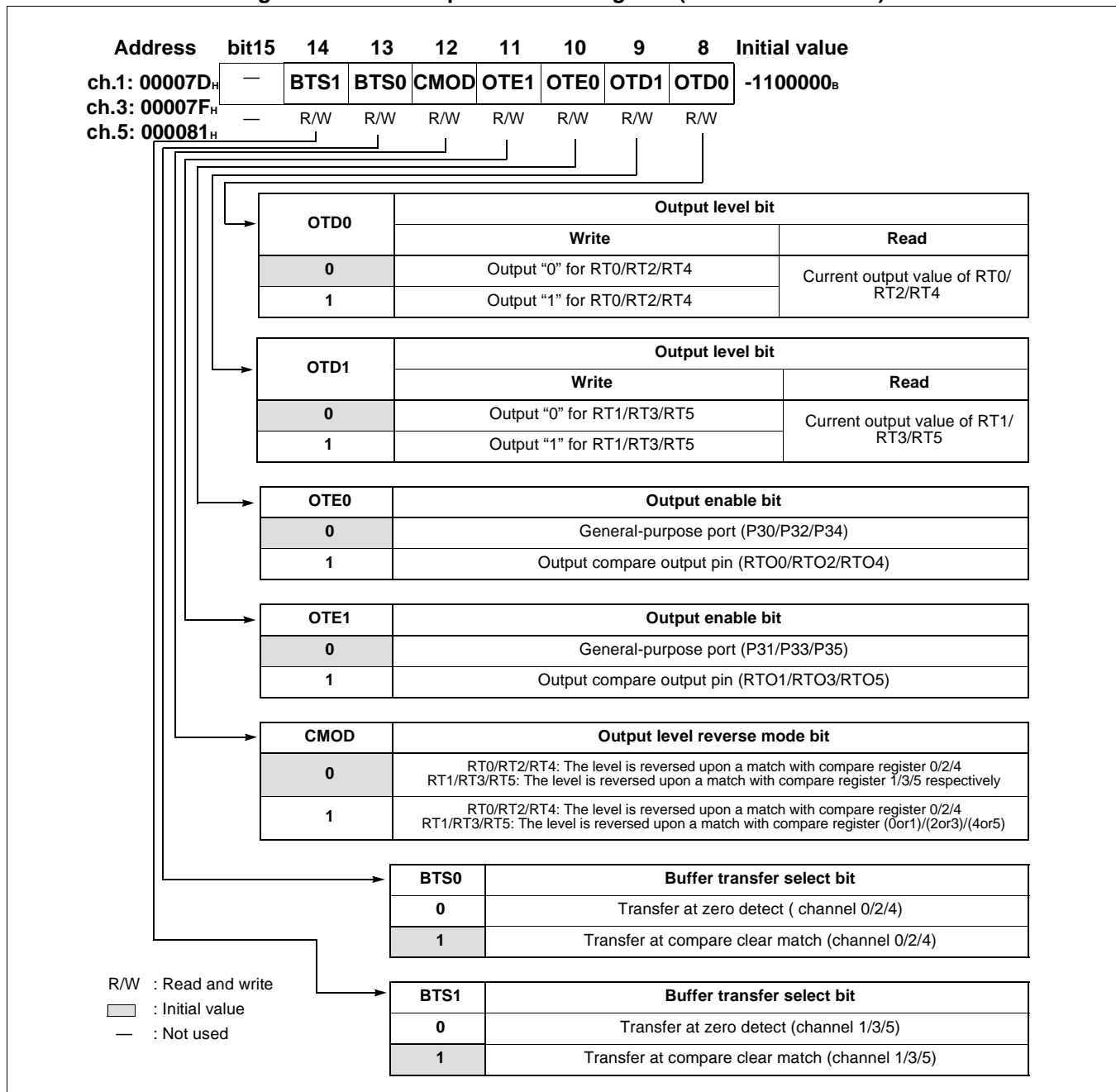
Word access to this register is recommended.

## 14.4.5 Compare Control Registers (OCS0 to OCS5)

Compare control register is used to control the output level, output enable, output reverse mode, compare operation enable, compare match interrupt enable and compare match interrupt flag for RTO0 to RTO5.

### ■ Compare Control Register, Upper Byte (OCS1/OCS3/OCS5)

Figure 14.4-12 Compare Control Register (OCS1/OCS3/OCS5)



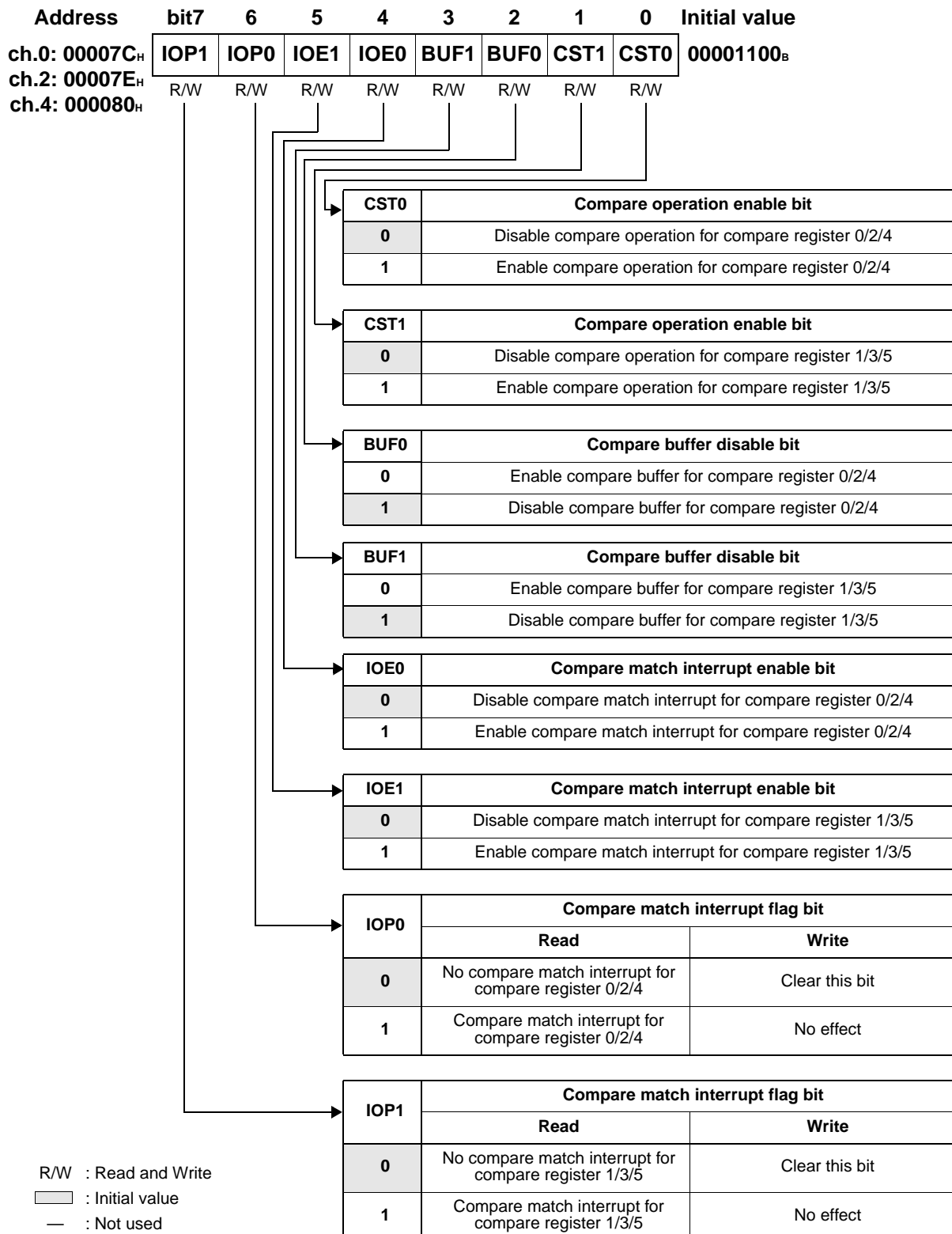


**Table 14.4-3 Compare Control Register (OCS1/OCS3/OCS5) bit**

Bit name		Function
bit15	Unused bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to this bit has no effect on the operation.</li> </ul>
bit14	BTS1	<ul style="list-style-type: none"> <li>This bit is used to select when data transfer from output compare buffer register (OCCPB1/OCCPB3/OCCPB5) to output compare register (OCCP1/OCCP3/OCCP5).</li> <li>When BTS1=0, buffer transfer is occurred when count value of 16-bit free-run timer is detected as zero.</li> <li>When BTS1=1, buffer transfer is occurred when compare clear match is occurred in 16-bit free-run timer.</li> </ul>
bit13	BTS0	<ul style="list-style-type: none"> <li>This bit is used to select when data transfer from output compare buffer register (OCCPB0/OCCPB2/OCCPB4) to output compare register (OCCP0/OCCP2/OCCP4).</li> <li>When BTS0=0, buffer transfer is occurred when count value of 16-bit free-run timer is detected as zero.</li> <li>When BTS0=1, buffer transfer is occurred when compare clear match is occurred in 16-bit free-run timer.</li> </ul>
bit12	CMOD: Output level reverse mode bit	<ul style="list-style-type: none"> <li>CMOD is used to switch the pin output level reverse mode upon a match while pin output is enabled (OTE1 = 1 or OTE0 = 1).</li> <li>When CMOD = 0, the output level of the pin is reversed upon a match with corresponding compare register. RT0/RT2/RT4: The level is reversed upon a match between the 16-bit free-run timer and compare register 0/2/4. RT1/RT3/RT5: The level is reversed upon a match between the 16-bit free-run timer and compare register 1/3/5.</li> <li>When CMOD = 1, the output level of the pin RT0/RT2/RT4 corresponding to compare register is reversed as same as when CMOD = 0. However, the output level of the pin (RT1/RT3/RT5) corresponding to compare register 1/3/5 is reversed when a match is detected in compare register 0/2/4 or 1/3/5. If compare registers 0/2/4 and 1/3/5 have the same value, the same operation as when only one compare register is used. RT0/RT2/RT4: The level is reversed upon a match between the 16-bit free-run timer and compare register 0/2/4. RT1/RT3/RT5: The level is reversed upon a match between the 16-bit free-run timer and compare register (0 or 1)/(2 or 3)/(4 or 5).</li> </ul>
bit11	OTE1: Output enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable waveform generator output RTO1/RTO3/RTO5 to P31/P33/P35.</li> <li>The initial value for these bits is "0".</li> </ul> <p>(Note) If waveform generator is disabled (DTCR:TMD2 to TMD0=000<sub>B</sub>) RTO1/RTO3/RTO5 output the same value in output compare RT1/RT3/RT5.</p>
bit10	OTE0: Output enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable waveform generator output RTO0/RTO2/RTO4 to P30/P32/P34.</li> <li>The initial value for these bits is "0".</li> </ul> <p>(Note) If waveform generator is disabled (DTCR:TMD2 to TMD0=000<sub>B</sub>) RTO0/RTO2/RTO4 output the same value in output compare RT0/RT2/RT4.</p>
bit9	OTD1: Output level bit	<ul style="list-style-type: none"> <li>This bit is used to change the output level for output compare 1/3/5 (RT1/RT3/RT5).</li> <li>The initial value of the compare output is "0".</li> <li>Ensure that the compare operation is stopped before a value is written. When reading this bit, this bit indicate the output compare value in RT1/RT3/RT5.</li> </ul>
bit8	OTD0: Output level bit	<ul style="list-style-type: none"> <li>This bit is used to change the output level for output compare 0/2/4 (RT0/RT2/RT4).</li> <li>The initial value of the compare output is "0".</li> <li>Ensure that the compare operation is stopped before a value is written. When reading this bit, this bit indicates the output compare value in RT0/RT2/RT4.</li> </ul>

## ■ Compare Vontrol Tegister, Lower Byte (OCS0/OCS2/OCS4)

Figure 14.4-13 Compare Vontrol Register (OCS0/OCS2/OCS4)



**Table 14.4-4 Compare Control Register (OCS0/OCS2/OCS4)**

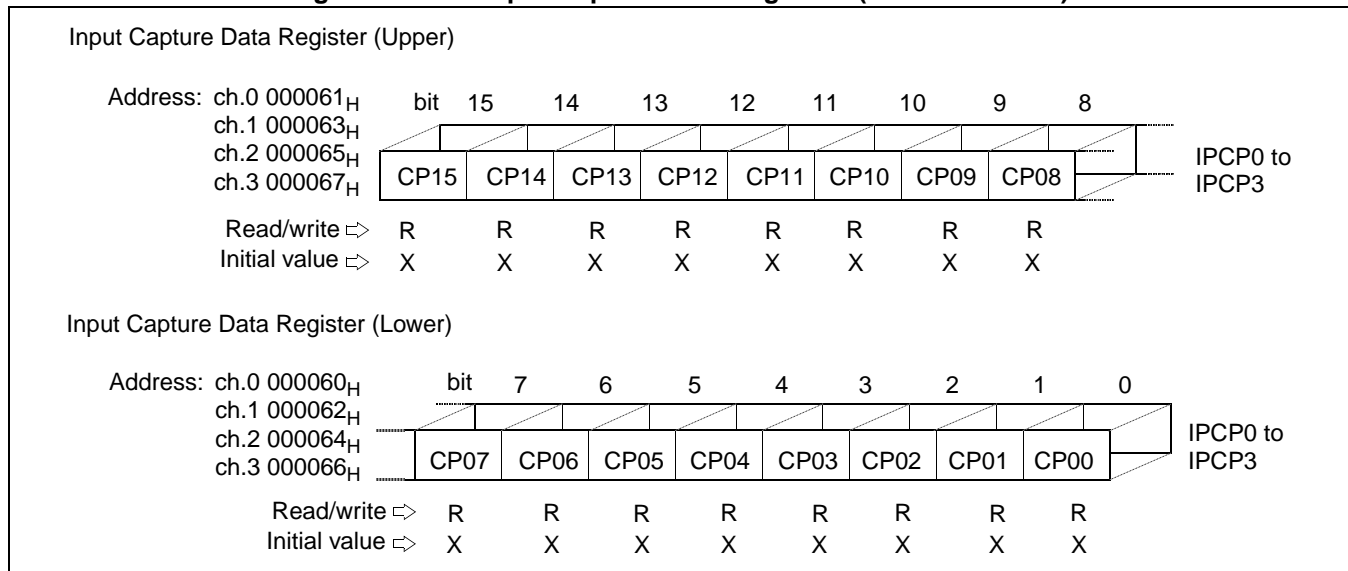
Bit name		Function
bit7	IOP1: Compare match interrupt flag bit	<ul style="list-style-type: none"> <li>• This bit is an interrupt flag for when compare register 1/3/5 is matched with the value of 16-bit free-run timer.</li> <li>• “1” is set to this bit when the compare register value matches the 16-bit free-run timer value.</li> <li>• While the interrupt request bits (IOE1) is enabled, an output compare interrupt occurs when the IOP1 bit is set.</li> <li>• Writing “0” will clear this bit.</li> <li>• Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit6	IOP0: Compare match interrupt flag bit	<ul style="list-style-type: none"> <li>• This bit is an interrupt flag for when compare register 0/2/4 is matched with the value of 16-bit free-run timer.</li> <li>• “1” is set to this bit when the compare register value matches the 16-bit free-run timer value.</li> <li>• While the interrupt request bits (IOE0) is enabled, an output compare interrupt occurs when the IOP0 bit is set.</li> <li>• Writing “0” will clear this bit.</li> <li>• Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit5	IOE1: Compare match interrupt enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable output compare interrupt for compare register 1/3/5.</li> <li>• While the “1” is written to this bit, an output compare interrupt occurs when an interrupt flag (IOP1) is set.</li> </ul>
bit4	IOE0: Compare match interrupt enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable output compare interrupt for compare register 0/2/4.</li> <li>• While the “1” is written to this bit, an output compare interrupt occurs when an interrupt flag (IOP0) is set.</li> </ul>
bit3	BUF1: Compare buffer disable bit	<ul style="list-style-type: none"> <li>• This bit is used to disable buffer function for output compare register 1/3/5.</li> <li>• Writing “0” will enable the buffer function.</li> </ul>
bit2	BUF0: Compare buffer disable bit	<ul style="list-style-type: none"> <li>• This bit is used to disable buffer function for output compare register 0/2/4.</li> <li>• Writing “0” will enable the buffer function.</li> </ul>
bit1	CST1: Compare operation enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the compare operation between 16-bit free-run timer and compare register 1/3/5.</li> <li>• Ensure that a value is written into the compare register and timer data register before the compare operation is enabled.</li> </ul> <p>Note: Since output compare is synchronized with the 16-bit free-run timer clock, stopping the 16-bit free-run timer stops compare operation.</p>
bit0	CST0: Compare operation enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the compare operation between 16-bit free-run timer and compare register 0/2/4.</li> <li>• Ensure that a value is written into the compare register and timer data register before the compare operation is enabled.</li> </ul> <p>(Note) Since output compare is synchronized with the 16-bit free-run timer clock, stopping the 16-bit free-run timer stops compare operation.</p>

## 14.4.6 Input Capture Register (IPCP0 to IPCP3)

Input capture registers are used to hold the count value of 16-bit timer when a valid edge of the input waveform is detected.

### ■ Input Capture Register (IPCP0 to IPCP3)

Figure 14.4-14 Input Capture Data Registers (IPCP0 to IPCP3)



This register is used to store the value of the 16-bit timer when a valid edge of the corresponding external pin input waveform is detected. (Word access instruction to this register is recommended. No data can be written to this register.)

## 14.4.7 Input Capture Control Status Registers (ICS23, PICS01)

Input capture control status registers (ICS23, PICS01) are used to control edge selection, interrupt request enable, interrupt request flag and to indicate valid edge detected for input capture 0 to 3.

### ■ Input Capture Control Status Register, Upper Byte (ICSH23)

Figure 14.4-15 Input Capture Control Status Register (ICSH23)

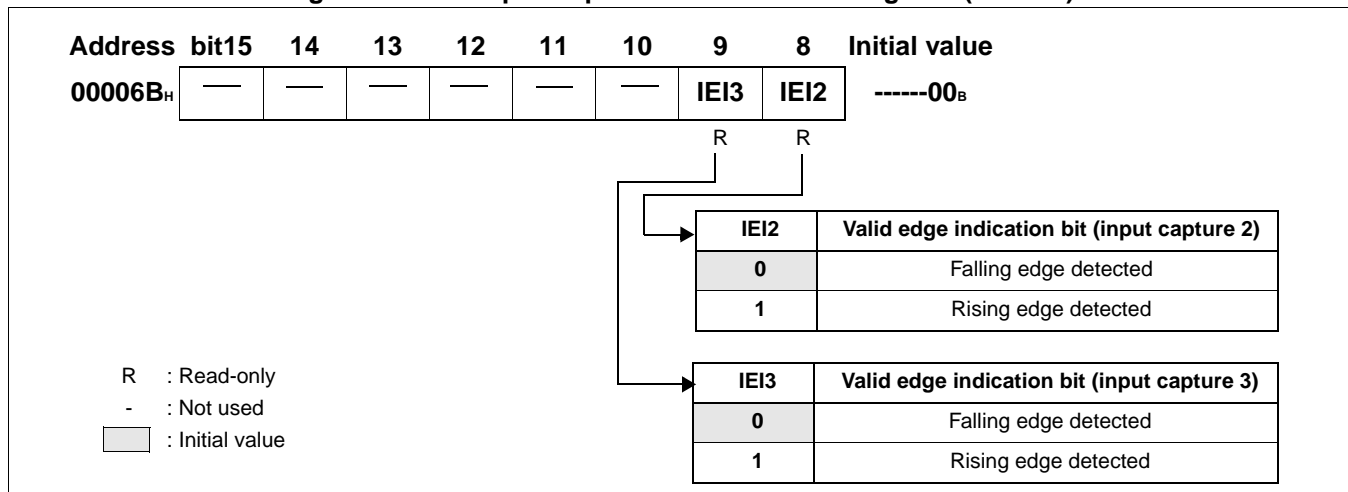
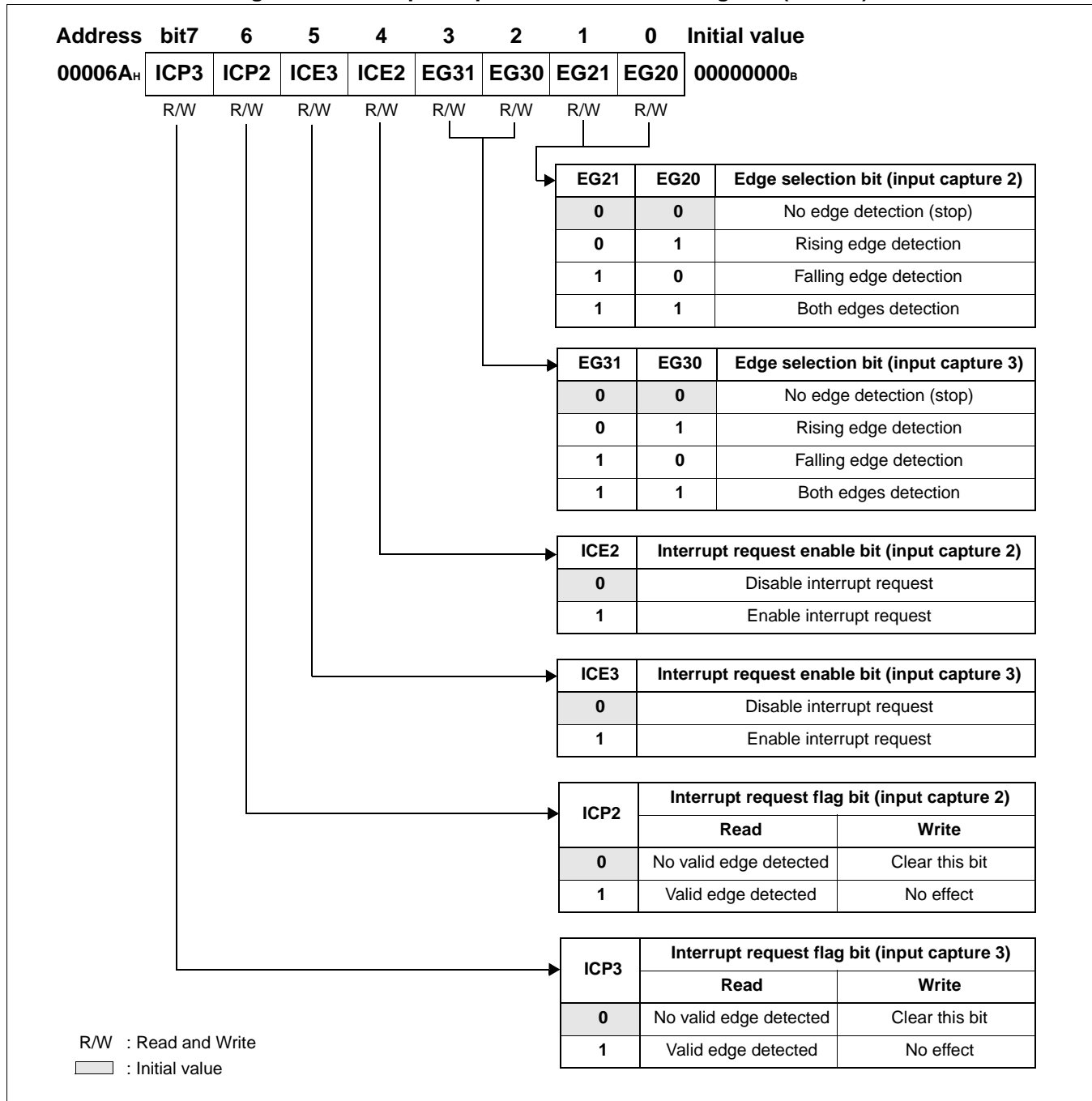


Table 14.4-5 Input Capture Control Status Register (ICSH23)

Bit name		Function
bit15 to bit10	Unused bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to this bit has no effect on the operation.</li> </ul>
bit9	IEI3: Valid edge indication bit	<ul style="list-style-type: none"> <li>This bit is an valid edge indication bit for capture register 3, to indicate a rising or falling edge is detected.</li> <li>“0” is written to this bit when falling edge is detected.</li> <li>“1” is written to this bit when rising edge is detected.</li> <li>This bit is read-only.</li> </ul> <p>(Note)</p> <p>The read value is meaningless when EG31, EG30 = 00.</p>
bit8	IEI2: Valid edge indication bit	<ul style="list-style-type: none"> <li>This bit is an valid edge indication bit for capture register 2, to indicate a rising or falling edge is detected.</li> <li>“0” is written to this bit when falling edge is detected.</li> <li>“1” is written to this bit when rising edge is detected.</li> <li>This bit is read-only.</li> </ul> <p>(Note)</p> <p>The read value is meaningless when EG21, EG20 = 00.</p>

## ■ Input Capture Control Status Register, Lower Byte (ICSL23)

Figure 14.4-16 Input Capture Control Status Register (ICSL23)

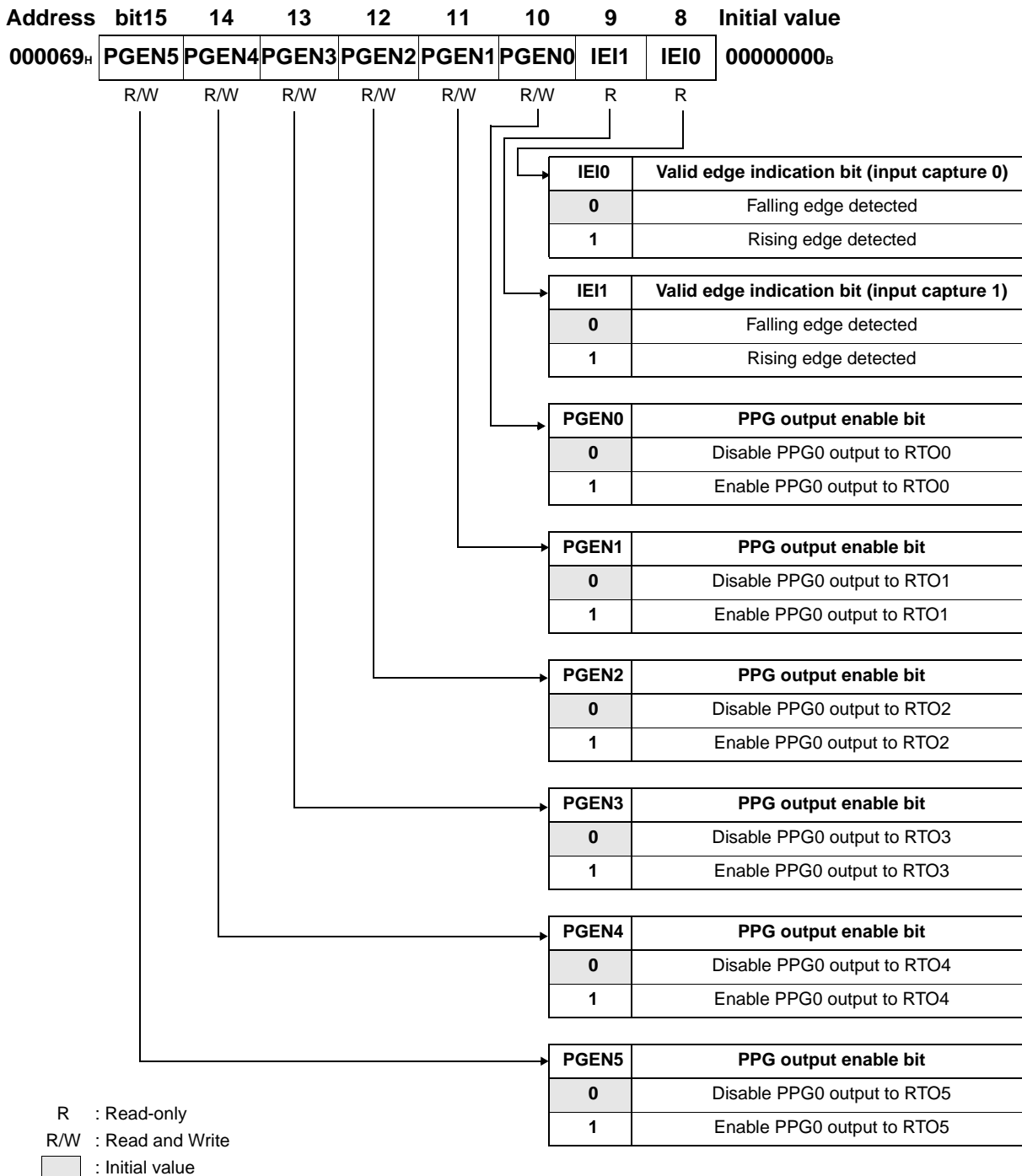


**Table 14.4-6 Input Capture Control Status Register (ICSL23)**

Bit name		Function
bit7	ICP3: Interrupt request flag bit (Input capture 3)	<ul style="list-style-type: none"> <li>• This bit is used as interrupt request flag for input capture 3.</li> <li>• “1” is set to this bit upon detection of a valid edge in an external input pin.</li> <li>• While the interrupt enable bit (ICE3) is set, an interrupt can be generated upon detection of a valid edge.</li> <li>• Writing “0” will clear this bit.</li> <li>• Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit6	ICP2: Interrupt request flag (Input capture 2)	<ul style="list-style-type: none"> <li>• This bit is used as interrupt request flag for input capture 2.</li> <li>• “1” is set to this bit upon detection of a valid edge in an external input pin.</li> <li>• While the interrupt enable bit (ICE2) is set, an interrupt can be generated upon detection of a valid edge.</li> <li>• Writing “0” will clear this bit.</li> <li>• Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit5	ICE3: Interrupt request enable bit (Input capture 3)	<ul style="list-style-type: none"> <li>• This bit is used to enable input capture interrupt request for input capture 3.</li> <li>• While “1” is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP3) is set.</li> </ul>
bit4	ICE2: Interrupt request enable bit (Input capture 2)	<ul style="list-style-type: none"> <li>• This bit is used to enable input capture interrupt request for input capture 2.</li> <li>• While “1” is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP2) is set.</li> </ul>
bit3, bit2	EG31, EG30: Edge selection bit (Input capture 3)	<ul style="list-style-type: none"> <li>• These bits are used to specify the valid edge polarity of an external input for input capture 3.</li> <li>• These bits are also used to enable input capture operation.</li> </ul>
bit1, bit0	EG21, EG20: Edge selection bit (Input capture 2)	<ul style="list-style-type: none"> <li>• These bits are used to specify the valid edge polarity of an external input for input capture 2.</li> <li>• These bits are also used to enable input capture operation.</li> </ul>

## ■ PPG Output Control / Input Capture Control Status Register, Upper Byte (PICSH01)

Figure 14.4-17 PPG Output Control/Input Capture Control Status Register (PICSH01)



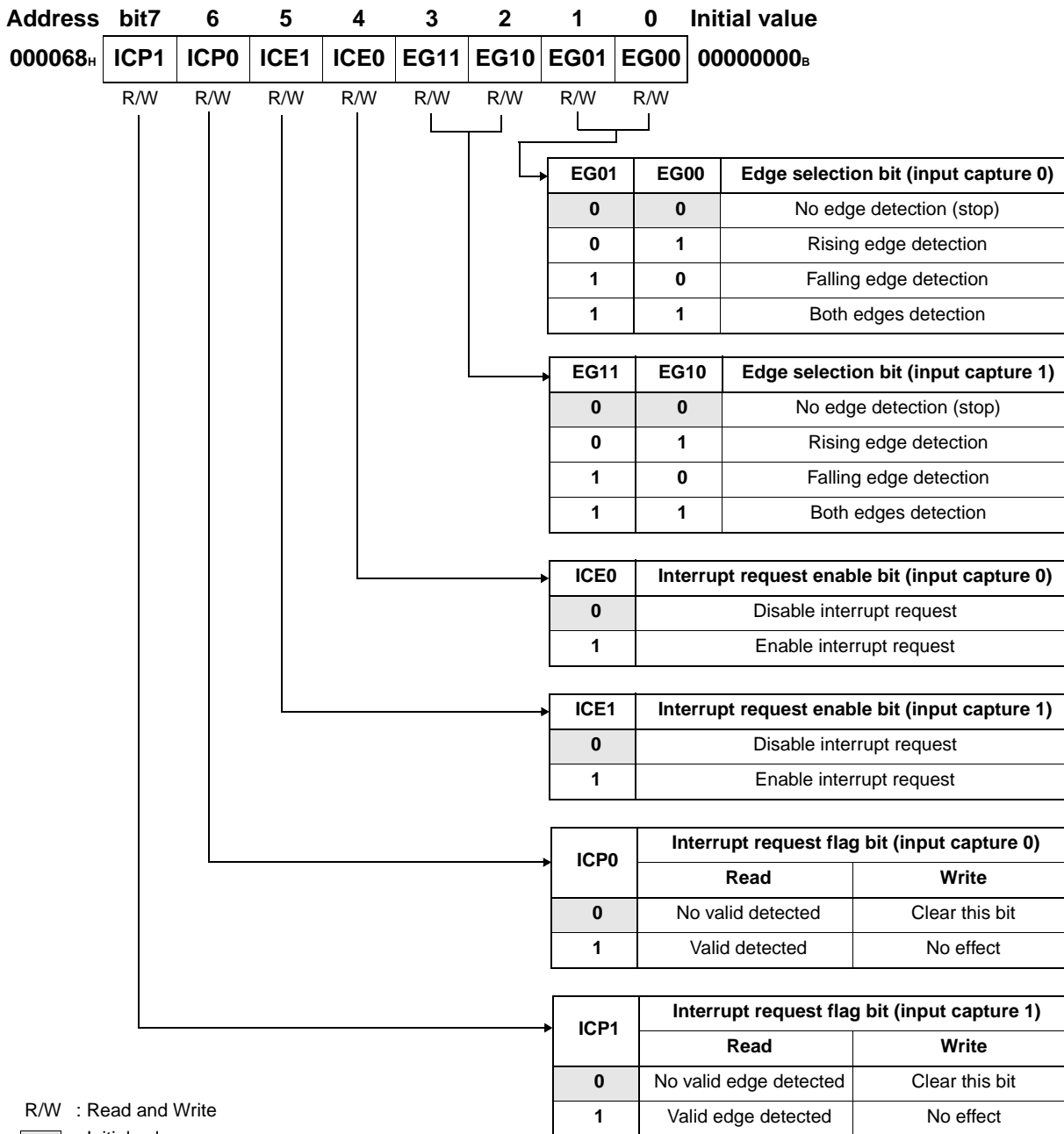


**Table 14.4-7 PPG output control/input capture control status register (PICSH01)**

Bit name		Function
bit15 to bit10	PGEN5 to PGEN0: PPG output enable bits	<ul style="list-style-type: none"> <li>This bit is used to select PPG0 output to RTO0 to RTO5.</li> </ul>
bit9	IEI1: Valid edge indication bit	<ul style="list-style-type: none"> <li>This bit is an valid edge indication bit for capture register 1, to indicate a rising or falling edge is detected.</li> <li>“0” is written to this bit when falling edge is detected.</li> <li>“1” is written to this bit when rising edge is detected.</li> <li>This bit is read-only.</li> </ul> <p>(Note) The read value is meaningless when EG11, EG10 = 00<sub>B</sub>.</p>
bit8	IEI0: Valid edge indication bit	<ul style="list-style-type: none"> <li>This bit is an value edge indication bit for capture register 0, to indicate a rising or falling edge is detected.</li> <li>“0” is written to this bit when falling edge is detected.</li> <li>“1” is written to this bit when rising edge is detected.</li> <li>This bit is read-only.</li> </ul> <p>(Note) The read value is meaningless when EG01, EG00 = 00<sub>B</sub>.</p>

## ■ Input Capture Control Status Register, Lower Byte (PICSL01)

Figure 14.4-18 Input Capture Control Status Register (PICSL01)



**Table 14.4-8 Input Capture Control Status Register (PICSL01)**

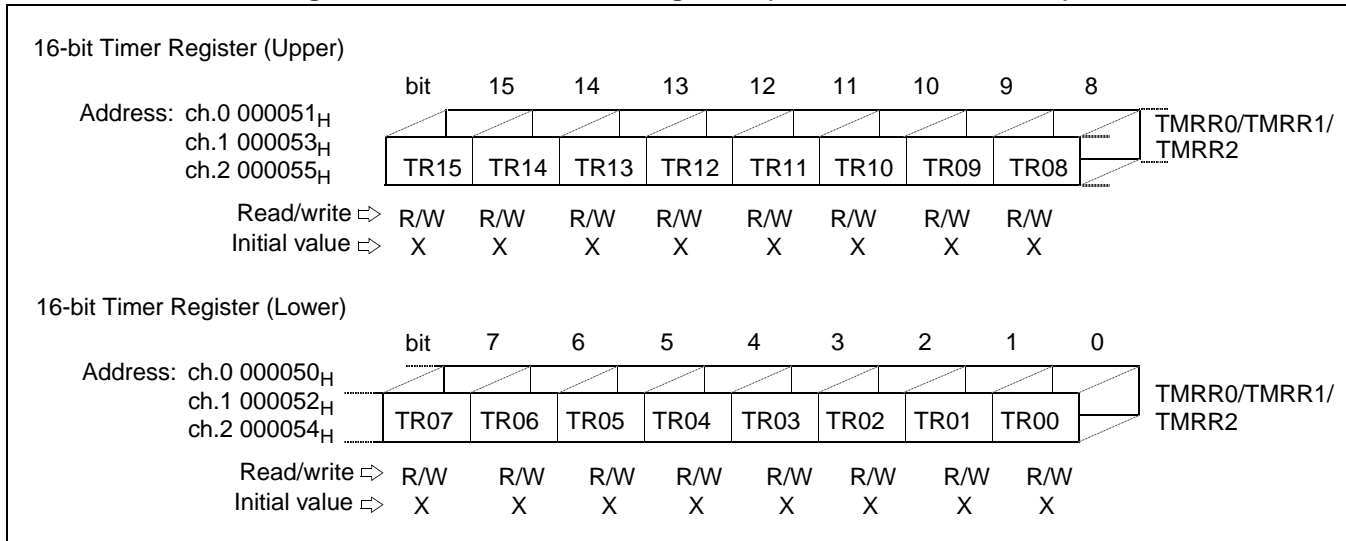
Bit name		Function
bit7	ICP1: Interrupt request flag bit (Input capture 1)	<ul style="list-style-type: none"> <li>• This bit is used as interrupt request flag for input capture 1.</li> <li>• “1” is set to this bit upon detection of a valid edge of an external input pin.</li> <li>• While the interrupt enable bit (ICE1) is set, an interrupt can be generated upon detection of a valid edge.</li> <li>• Writing “0” will clear this bit.</li> <li>• Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit6	ICP0: Interrupt request flag bit (Input capture 0)	<ul style="list-style-type: none"> <li>• This bit is used as interrupt request flag for input capture 0.</li> <li>• “1” is set to this bit upon detection of a valid edge of an external input pin.</li> <li>• While the interrupt enable bit (ICE0) is set, an interrupt can be generated upon detection of a valid edge.</li> <li>• Writing “0” will clear this bit.</li> <li>• Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit5	ICE1: Interrupt request enable bit (Input capture 1)	<ul style="list-style-type: none"> <li>• This bit is used to enable input capture interrupt request for input capture 1.</li> <li>• While “1” is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP1) is set.</li> </ul>
bit4	ICE0: Interrupt request enable bit (Input capture 0)	<ul style="list-style-type: none"> <li>• This bit is used to enable input capture interrupt request for input capture 0.</li> <li>• While “1” is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP0) is set.</li> </ul>
bit3, bit2	EG11, EG10: Edge selection bit (Input capture 1)	<ul style="list-style-type: none"> <li>• These bits are used to specify the valid edge polarity of an external input for input capture 1.</li> <li>• These bits are also used to enable input capture operation.</li> </ul>
bit1, bit0	EG01, EG00: Edge selection bit (Input capture 0)	<ul style="list-style-type: none"> <li>• These bits are used to specify the valid edge polarity of an external input for input capture 0.</li> <li>• These bits are also used to enable input capture operation.</li> </ul>

## 14.4.8 16-bit Timer Register (TMRR0/TMRR1/TMRR2)

16-bit timer registers hold the compare value of 16-bit timers.

### ■ 16-bit Timer Registers (TMRR0/TMRR1/TMRR2)

Figure 14.4-19 16-bit Timer Registers (TMRR0/TMRR1/TMRR2)



These registers are used to store the comparison value of 16-bit timers. The value in these registers will be reloaded when the 16-bit timer is started to operate. Therefore, if the value is re-written into these registers during timer operation, these value will be valid at the next timer initiation/operation.

In dead-time timer mode, these registers are used to set the non-overlap time.

- Non-overlap time = (set value + 1) × selected clock.

#### Notes:

- The value of "0000<sub>H</sub>" cannot be set.
- The maximum offset of non-overlap time is -1 selected clock.

In timer mode, these registers are used to set the GATE time for PPG timer 0 operation.

- GATE time = (set value + 1) × selected clock.

#### Notes:

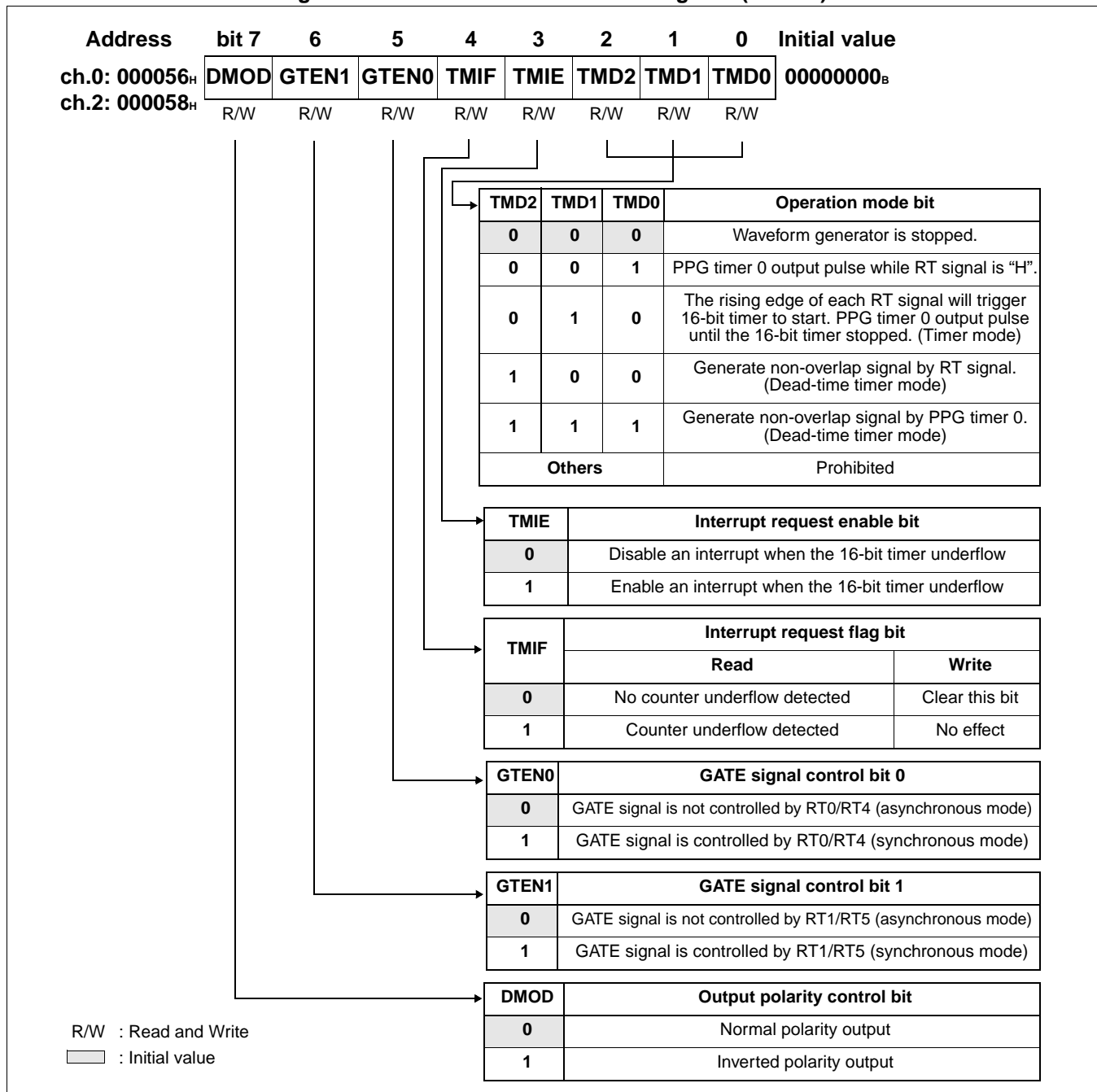
- The value of "0000<sub>H</sub>" cannot be set and maximum offset is -1 selected clock.
- The maximum offset of GATE time is -1 selected clock.

## 14.4.9 16-bit Timer Control Register (DTCR0/DTCR1/DTCR2)

16-bit timer control registers (DTCR0/DTCR1/DTCR2) are used to control the operation mode, interrupt request enable, interrupt request flag, GATE signal enable and output level polarity for the waveform generator.

### ■ 16-bit Timer Control Register (DTCR0/DTCR2)

Figure 14.4-20 16-bit Timer Control Register (DTCR1)

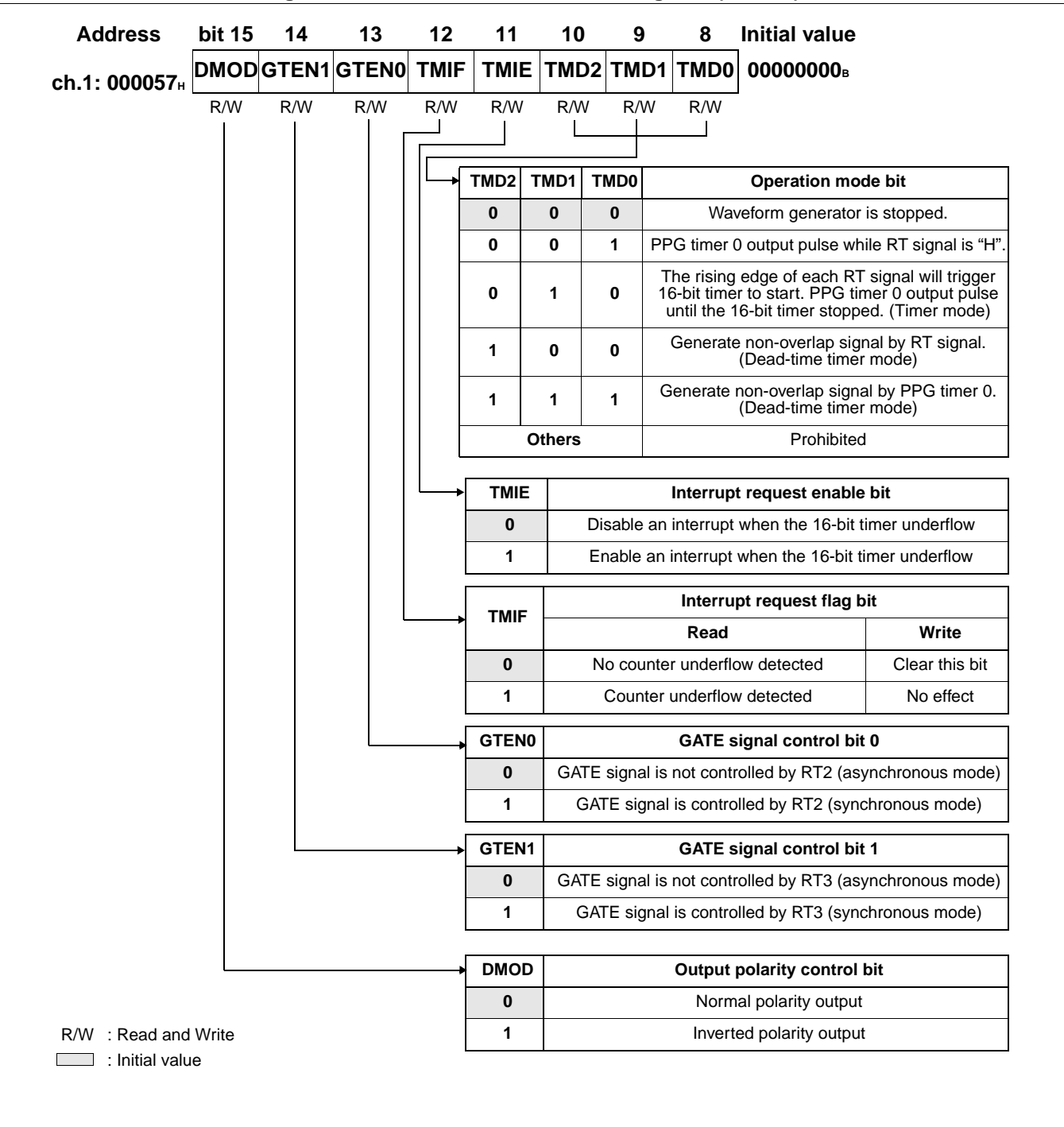


**Table 14.4-9 16-bit Timer Control Registers (DTCR0/DTCR2) bit**

Bit name		Function
bit7	DMOD: Output polarity control bit	<ul style="list-style-type: none"> <li>This bit is used to set the output polarity of U/V/W in dead-time timer mode.</li> <li>By setting this bit, the output polarity of U/V/W is inverted.</li> </ul> <p>(Note) This bit is meaningless when dead-time timer mode is not selected (bit2: TMD2 = 0).</p>
bit6	GTEN1: GATE signal control bit 1	<ul style="list-style-type: none"> <li>This bit is used to control the GATE signal output for PPG timer 0 by RT1/RT5.</li> </ul>
bit5	GTEN0: GATE signal control bit 0	<ul style="list-style-type: none"> <li>This bit is used to control the GATE signal output of PPG timer 0 by RT0/RT4.</li> </ul>
bit4	TMIF: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is used as an interrupt request flag for 16-bit timers.</li> <li>This bit will be set to "1" when 16-bit timer 0/2 is underflow.</li> <li>Writing "0" will clear this bit.</li> <li>Writing "1" has no effect.</li> <li>In read-modify-write operation, "1" is always read.</li> </ul> <p>(Notes)</p> <ul style="list-style-type: none"> <li>This bit functions only in mode TMD2 to TMD0=000<sub>B</sub> or 001<sub>B</sub>. In other modes, this bit is always "0".</li> <li>If both software clear (writing "0") and hardware set (16-bit timer 0/2 underflow) occurs simultaneously, software clear takes the higher priority to clear this bit.</li> </ul>
bit3	TMIE: Interrupt request enable / software trigger bit	<ul style="list-style-type: none"> <li>This bit is used as the software trigger bit and interrupt enable bit for the 16-bit timer 0/2.</li> <li>When TMD22 to TMD0=000<sub>B</sub> or 001<sub>B</sub>, this bit is used as software trigger for 16-bit timer. Setting this bit from "0" to "1" trigger the 16-bit timer to reload and starts down-counting.</li> <li>When this bit is "1" and the interrupt flag bit (bit4: TIMF) is "1", an interrupt request is sent to CPU.</li> </ul> <p>(Note) To retrigger the 16-bit timer, be sure to write "0" before write "1" to this bit.</p>
bit2 to bit0	TMD2 to TMD0: Operation mode bits	<ul style="list-style-type: none"> <li>These bits are used to select the operation mode of the waveform generator.</li> <li>When TMD2 to TMD0=000<sub>B</sub>, output compare RT0/RT4 and RT1/RT5 outputs to RTO0/RTO4 and RTO1/RTO5 respectively. And 16-bit timer can be used as reload timer.</li> <li>When TMD2 to TMD0=001<sub>B</sub>, output compare RT0/RT4 and RT1/RT5 outputs to RTO0/RTO4 and RTO1/RTO5 respectively if PPG0 output is disabled (PICSH01:PGEN0/PGEN4=0, PGEN1/PGEN5=0). And 16-bit timer can be used as reload timer.</li> </ul> <p>(Notes)</p> <ul style="list-style-type: none"> <li>To operate the waveform generator in dead-time timer mode, be sure to select 2-channel mode for RT1/RT5 (OCS1/OCS5:CMOD=1)</li> <li>When TMD2 to TMD0=111<sub>B</sub> is selected, RTO0/RTO4 and RTO1/RTO5 output are independent of setting in PICSH01:PGEN0/PGEN4,PGEN1/PGEN5.</li> </ul>

■ 16-bit Timer Control Register (DTCR1)

Figure 14.4-21 16-bit Timer Control Register (DTCR1)



R/W : Read and Write

Initial value

**Table 14.4-10 16-bit Timer Control Registers (DTCR1) bit**

Bit name		Function
bit15	DMOD: Output polarity control bit	<ul style="list-style-type: none"> <li>This bit is used to set the output polarity of U/V/W in dead-time timer mode.</li> <li>By setting this bit, the output polarity of U/V/W is inverted.</li> </ul> (Note) This bit is meaningless when dead-time timer mode is not selected (bit10: TMD2 = 0).
bit14	GTEN1: GATE signal control bit 1	<ul style="list-style-type: none"> <li>This bit is used to control the GATE signal output for PPG timer 0 by RT3.</li> </ul>
bit13	GTEN0: GATE signal control bit 0	<ul style="list-style-type: none"> <li>This bit is used to control the GATE signal output of PPG timer 0 by RT2.</li> </ul>
bit12	TMIF: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is used as an interrupt request flag for 16-bit timers.</li> <li>This bit will be set to "1" when 16-bit timer 1 is underflow.</li> <li>Writing "0" will clear this bit.</li> <li>Writing "1" has no effect.</li> <li>In read-modify-write operation, "1" is always read.</li> </ul> (Notes) <ul style="list-style-type: none"> <li>This bit functions only in mode TMD2 to TMD0=000<sub>B</sub> or 001<sub>B</sub>. In other modes, this bit is always "0".</li> <li>If both software clear (writing "0") and hardware set (16-bit timer 1 underflow) occurs simultaneously, software clear takes the higher priority to clear this bit.</li> </ul>
bit11	TMIE: Interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit is used as the software trigger bit and interrupt enable bit for the 16-bit timer.</li> <li>When TMD2 to TMD0=000<sub>B</sub> or 001<sub>B</sub>, this bit is used as software trigger for 16-bit timer. Setting this bit from "0" to "1" trigger the 16-bit timer to reload and starts down-counting.</li> <li>When this bit is "1" and the interrupt flag bit (bit12: TMIF) is "1", an interrupt request is sent to CPU.</li> </ul> (Note) To retrigger the 16-bit timer, be sure to write "0" before write "1" to this bit.
bit10 to bit8	TMD2 to TMD0: Operation mode bit	<ul style="list-style-type: none"> <li>These bits are used to select the operation mode of the waveform generator.</li> <li>When TMD2 to TMD0=000<sub>B</sub>, output compare RT2 and RT3 outputs to RTO2 and RTO3 respectively. And 16-bit timer can be used as reload timer.</li> <li>When TMD2 to TMD0=001<sub>B</sub>, output compare RT2 and RT3 outputs to RTO2 and RTO3 respectively if PPG0 output is disabled (PICSH01:PGEN2=0, PGEN3=0). And 16-bit timer can be used as reload timer.</li> </ul> (Notes) <ul style="list-style-type: none"> <li>To operate the waveform generator in dead-time timer mode, be sure to select 2-channel mode for RT3 (OCS3:CMOD=1)</li> <li>When TMD2 to TMD0=111<sub>B</sub> is selected, RTO2 and RTO3 output are independent of setting in PICSH01:PGEN2,PGEN3.</li> </ul>

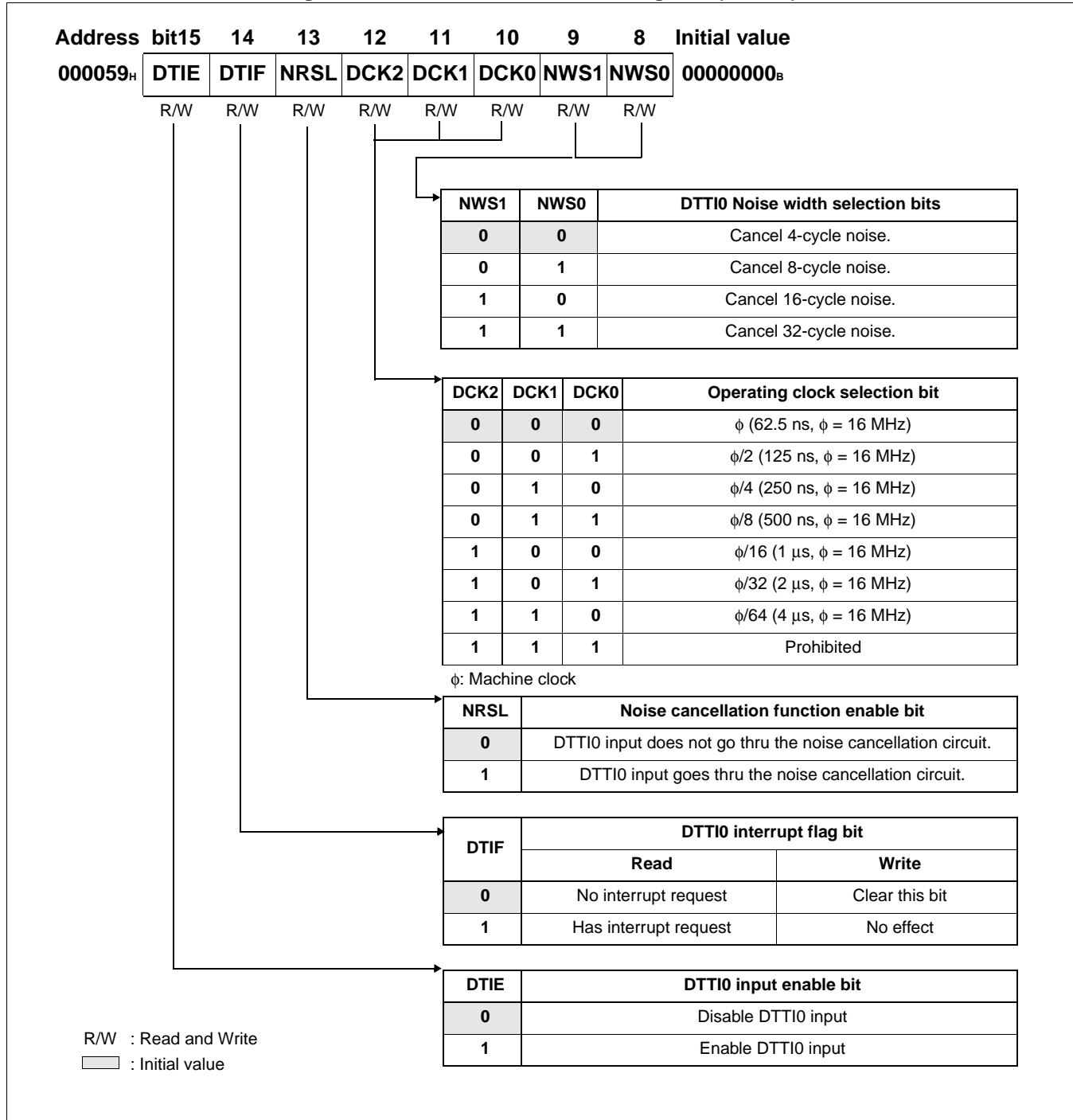


## 14.4.10 Waveform Control Register (SIGCR)

Waveform control register is used to control how the operating clock frequencies, noise cancellation function enable, DTTI0 input enable and DTTI0 interrupt.

### ■ Waveform Control Register (SIGCR)

Figure 14.4-22 Waveform Control Register (SIGCR)



**Table 14.4-11 Waveform Control Register (SIGCR)**

Bit name		Function
bit15	DTIE: DTTI0 input enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable the DTTI0 pin to control the output level of RTO0 to RTO5 pin.</li> </ul>
bit14	DTIF: DTTI0 interrupt flag bit	<ul style="list-style-type: none"> <li>This bit is an interrupt flag for DTTI0.</li> <li>When DTTI0 input is enabled (DTIE=1) and low level of DTTI0 is detected, this bit will be set and interrupt request will send to CPU.</li> <li>Writing "0" will clear this bit.</li> <li>Writing "1" has no effect.</li> <li>In read-modify-write operation, "1" is always read.</li> </ul> <p>(Notes)</p> <ul style="list-style-type: none"> <li>If noise cancellation function is enabled (NRSL=1), this bit will be set to "1" when noise pulse width is passed.</li> <li>If both software clear (writing "0") and hardware set (low level of DTTI0 is detected) occurs simultaneously, software clear takes the higher priority to clear this bit.</li> </ul>
bit13	NRSL: Noise cancellation function enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable the noise cancellation function.</li> <li>Noise cancellation circuit will receive DTTI0 input signal when the low level is held until the counter overflows. The counter is n-bit counter which is operated by the low level input. The value of n can be 2, 3, 4 and 5 which depends on the setting of NWS1 and NWS0.</li> </ul> <p>(Notes)</p> <ul style="list-style-type: none"> <li>To cancel the noise pulse width, it takes approximately 2n machine cycles.</li> <li>When the noise cancellation circuit is selected, the input will become invalid in a mode such as STOP mode in which the internal clock is stopped.</li> </ul>
bit12 to bit10	DCK2 to DCK0: Operating clock selection bit	<ul style="list-style-type: none"> <li>These bits are used to select the operating clock for the 16-bit timer.</li> </ul>
bit9, bit8	NWS1, NWS0: DTTI0 Noise width selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the noise pulse width to be removed for DTTI0 pin.</li> </ul>

## 14.5 Multi-functional Timer Interrupts

The multi-functional timer is enabled to generate interrupts in 16-bit free-run timer, 16-bit output compare, 16-bit input capture and waveform generator.

### ■ 16-bit Free-run Timer Interrupts

Table 14.5-1 lists the interrupt control bits and interrupt causes of the 16-bit free-run timer.

**Table 14.5-1 Interrupt Control Bits and Interrupt Causes of the 16-bit Free-run Timer**

	16-bit free-run timer	
	Compare Clear	Zero Detect
Interrupt request flag bit	TCCSH:ICLR	TCCSH:IRQZF
Interrupt request enable bit	TCCSH:ICRE	TCCSH:IRQZE
Interrupt cause	16-bit free-run timer value matches with compare clear register (CPCLR)	16-bit free-run timer value equals zero

In the 16-bit free-run timer, the ICLR bit of the timer control status register (TCCSH) is set to "1" when timer value matches compare clear register (CPCLR). If an interrupt request is enabled (TCCSH:ICRE = 1) in this operation, the interrupt request is output to the interrupt controller.

The IRQZF bit of the timer control status register (TCCSH) is set to "1" when timer value equals "0000<sub>H</sub>". If an interrupt request is enabled (TCCSH:IRQZE = 1) in this operation, the interrupt request is output to the interrupt controller.

### ■ 16-bit Free-run Timer Interrupts and EI<sup>2</sup>OS

Table 14.5-2 lists the 16-bit free-run timer interrupts and EI<sup>2</sup>OS.

**Table 14.5-2 16-bit Free-run Timer Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
Compare clear <sup>*1</sup>	#34 (22 <sub>H</sub> )	ICR11	0000BB <sub>H</sub>	FFFF74 <sub>H</sub>	FFFF75 <sub>H</sub>	FFFF76 <sub>H</sub>	Δ
Zero detect <sup>*2</sup>	#31 (1F <sub>H</sub> )	ICR10	0000BA <sub>H</sub>	FFFF80 <sub>H</sub>	FFFF81 <sub>H</sub>	FFFF82 <sub>H</sub>	

\*1: The same interrupt control register as that for 16-bit free-run timer compare clear is assigned to 16-bit input capture channels 0/1.

\*2: The same interrupt control register as that for 16-bit free-run timer zero detect is assigned to 16-bit PPG timer 2.

## ■ 16-bit Output Compare Interrupts

Table 14.5-3 lists the interrupt control bits and interrupt causes of the 16-bit output compare.

**Table 14.5-3 Interrupt Control Bits and Interrupt Causes of the 16-bit Output Compare 0 to 5**

	16-bit output compare 0/1	16-bit output compare 2/3	16-bit output compare 4/5
Interrupt request flag bit	OCS0:IOP0/IOP1	OCS2:IOP0/IOP1	OCS4:IOP0/IOP1
Interrupt request enable bit	OCS0:IOE0/IOE1	OCS2:IOE0/IOE1	OCS4:IOE0/IOE1
Interrupt cause	16-bit free-run timer value matches with output compare register (OCCP0/OCCP1)	16-bit free-run timer value matches with output compare register (OCCP2/OCCP3)	16-bit free-run timer value matches with output compare register (OCCP4/OCCP5)

In the 16-bit output compare, the IOP0/IOP1 bit of the compare control register (OCS0/OCS2/OCS4) is set to "1" when 16-bit free-run timer value matches output compare register (OCCP0 to OCCP5). If an interrupt request is enabled (OCS0/OCS2/OCS4:IOE0/IOE1 = 1) in this operation, the interrupt request is output to the interrupt controller.

## ■ 16-bit Output Compare Interrupts and EI<sup>2</sup>OS

Table 14.5-4 lists the 16-bit output compare interrupts and EI<sup>2</sup>OS

**Table 14.5-4 16-bit Output Compare Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
Output compare 0 match *1	#12 (0C <sub>H</sub> )	ICR00	0000B0 <sub>H</sub>	FFFFCC <sub>H</sub>	FFFFCD <sub>H</sub>	FFFFCE <sub>H</sub>	O
Output compare 1 match *2	#15 (0F <sub>H</sub> )	ICR02	0000B2 <sub>H</sub>	FFFFC0 <sub>H</sub>	FFFFC1 <sub>H</sub>	FFFFC2 <sub>H</sub>	
Output compare 2 match *3	#17 (11 <sub>H</sub> )	ICR03	0000B3 <sub>H</sub>	FFFFB8 <sub>H</sub>	FFFFB9 <sub>H</sub>	FFFFBA <sub>H</sub>	
Output compare 3 match *4	#19 (13 <sub>H</sub> )	ICR04	0000B4 <sub>H</sub>	FFFFB0 <sub>H</sub>	FFFFB1 <sub>H</sub>	FFFFB2 <sub>H</sub>	
Output compare 4 match *5	#21 (15 <sub>H</sub> )	ICR05	0000B5 <sub>H</sub>	FFFA8 <sub>H</sub>	FFFA9 <sub>H</sub>	FFFAA <sub>H</sub>	
Output compare 5 match *6	#23 (17 <sub>H</sub> )	ICR06	0000B6 <sub>H</sub>	FFFA0 <sub>H</sub>	FFFA1 <sub>H</sub>	FFFA2 <sub>H</sub>	

\*1: The same interrupt control register as that for 16-bit output compare 0 is assigned to A/D conversion termination.

\*2: The same interrupt control register as that for 16-bit output compare 1 is assigned to 16-bit PPG timer 1.

\*3: The same interrupt control register as that for 16-bit output compare 2 is assigned to 16-bit reload timer 1 underflow.

\*4: The same interrupt control register as that for 16-bit output compare 3 is assigned to DTP/external interrupt channels 0/1 detection DTTI0.

\*5: The same interrupt control register as that for 16-bit output compare 4 is assigned to DTP/external interrupt channels 2/3 detection / DTTI1.

\*6: The same interrupt control register as that for 16-bit output compare 5 is assigned to PWC timer 1.

## ■ 16-bit Input Capture Interrupts

Table 14.5-5 lists the interrupt control bits and interrupt causes of the 16-bit input capture.

**Table 14.5-5 Interrupt Control Bits and Interrupt Causes of the 16-bit Input Capture 0 to 3**

	16-bit input capture 0/1	16-bit input capture 2/3
Interrupt request flag bit	PICSL01:ICP0/ICP1	ICSL23:ICP2/ICP3
Interrupt request enable bit	PICSL01:ICE0/ICE1	ICSL23:ICE2/ICE3
Interrupt cause	Valid edge is detected in IN0/IN1	Valid edge is detected in IN2/IN3

In the 16-bit input capture, the ICP0 to ICP3 bit of the input capture control register (PICSL01/ICSL23) is set to "1" when valid edge is detected in IN0 to IN3. If an interrupt request is enabled (PICSL01/ICSL23:ICE0/ICE1 = 1) in this operation, the interrupt request is output to the interrupt controller.

## ■ 16-bit Input Capture Interrupts and EI<sup>2</sup>OS

Table 14.5-6 lists the 16-bit input capture interrupts and EI<sup>2</sup>OS.

**Table 14.5-6 16-bit Input Capture Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
Input capture 0/1 <sup>*1</sup>	#33 (21 <sub>H</sub> )	ICR11	0000BB <sub>H</sub>	FFFF78 <sub>H</sub>	FFFF79 <sub>H</sub>	FFFF7A <sub>H</sub>	O
Input capture 2/3 <sup>*2</sup>	#35 (23 <sub>H</sub> )	ICR12	0000BC <sub>H</sub>	FFFF70 <sub>H</sub>	FFFF71 <sub>H</sub>	FFFF72 <sub>H</sub>	

\*1: The same interrupt control register as that for 16-bit input capture 0/1 is assigned to 16-bit free-run timer compare clear.

\*2: The same interrupt control register as that for 16-bit input capture 2/3 is assigned to Time-base timer.

## ■ Waveform Generator Interrupts

lists the interrupt control bits and interrupt causes of the waveform generator.

**Table 14.5-7 Interrupt Control Bits and Interrupt Causes of the Waveform Generator**

	Waveform generator	
	16-bit timer 0/1/2	DTT10
Interrupt request flag bit	DTCR0/DTCR1/DTCR2:TMIF	SIGCR:DTIF
Interrupt request enable bit	DTCR0/DTCR1/DTCR2:TMIE	--
Interrupt cause	16-bit timer 0/1/2 underflow	Low level is detected in DTT10

In the waveform generator, the TMIF bit of the 16-bit timer control register (DTCR0/DTCR1/DTCR2) is set to "1" when 16-bit timer underflow and DTCR0/DTCR1/DTCR2:TMD2 to TMD0=000<sub>B</sub> or 001<sub>B</sub>. If an interrupt request is enabled (DTCR0/DTCR1/DTCR2:TMIE = 1) in this operation, the interrupt request is output to the interrupt controller.

## ■ Waveform Generator Interrupts and EI<sup>2</sup>OS

Table 14.5-8 lists the waveform generator interrupts and EI<sup>2</sup>OS.

**Table 14.5-8 Waveform Generator Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
16-bit timer 0/1/2 underflow <sup>*1</sup>	#29 (1D <sub>H</sub> )	ICR09	0000B9 <sub>H</sub>	FFFF88 <sub>H</sub>	FFFF89 <sub>H</sub>	FFFF8A <sub>H</sub>	Δ
DTTI0 <sup>*2</sup>	#20 (14 <sub>H</sub> )	ICR04	0000B4 <sub>H</sub>	FFFFAC <sub>H</sub>	FFFFAD <sub>H</sub>	FFFFAE <sub>H</sub>	

\*1: The same interrupt control register as that for 16-bit timer 0/1/2 underflow is assigned to 16-bit reload timer 0 underflow.

\*2: The same interrupt control register as that for DTTI0 is assigned to DTP/external interrupt channels 0/1 detection and 16-bit output compare 3.

## ■ EI<sup>2</sup>OS Function of the Multi-functional Timer

Since the multi-functional timer has a circuit that coordinates with EI<sup>2</sup>OS, the interrupt generated can start EI<sup>2</sup>OS.

However, EI<sup>2</sup>OS is available only when other peripheral functions sharing the interrupt control register (ICR) do not use interrupts. For example, when 16-bit free-run timer compare clear uses EI<sup>2</sup>OS, interrupts of 16-bit input capture channels 0/1 must be disabled.

## 14.6 Operation of Multi-functional Timer

---

**This section describes the operation of the multi-functional timer.**

---

### ■ Operation of Multi-functional Timer

- 16-bit free-run timer

The 16-bit free-run timer starts counting up from value set in timer data register (TCDT) after a reset has been completed. The counter value is used as the reference time for 16-bit output compare and 16-bit input capture.

- 16-bit output compare

The 16-bit output compare is used to compare the value set in the specified output compare register with the value of the 16-bit free-run timer. If a match is detected, the interrupt flag is set and the output level is inverted.

- 16-bit input capture

The 16-bit input capture is used to detect a specified valid edge. If a valid edge is detected, the interrupt flag is set and the value of 16-bit free-run timer is fetched and stored into the input capture data register.

- Waveform generator

Waveform generator can produce various waveform such as dead-time, by using the real-time outputs (RT0 to RT5), 16-bit PPG timer 0 and 16-bit timers.

## 14.6.1 Operation of 16-bit free-run timer

The 16-bit free-run timer starts counting up from counter value specified in timer data register (TCDT) after a reset has been completed. The counter value is used as the reference time for 16-bit output compare and 16-bit input capture.

### ■ Timer Clear

The counter value of 16-bit free-run timer is cleared in the following conditions:

- When a match with compare clear register is detected in up-count mode (TCCSL:MODE=0)
- When "1" is written to the SCLR bit of the TCCSL register during operation. The timer will be cleared at the valid edge of count clock.

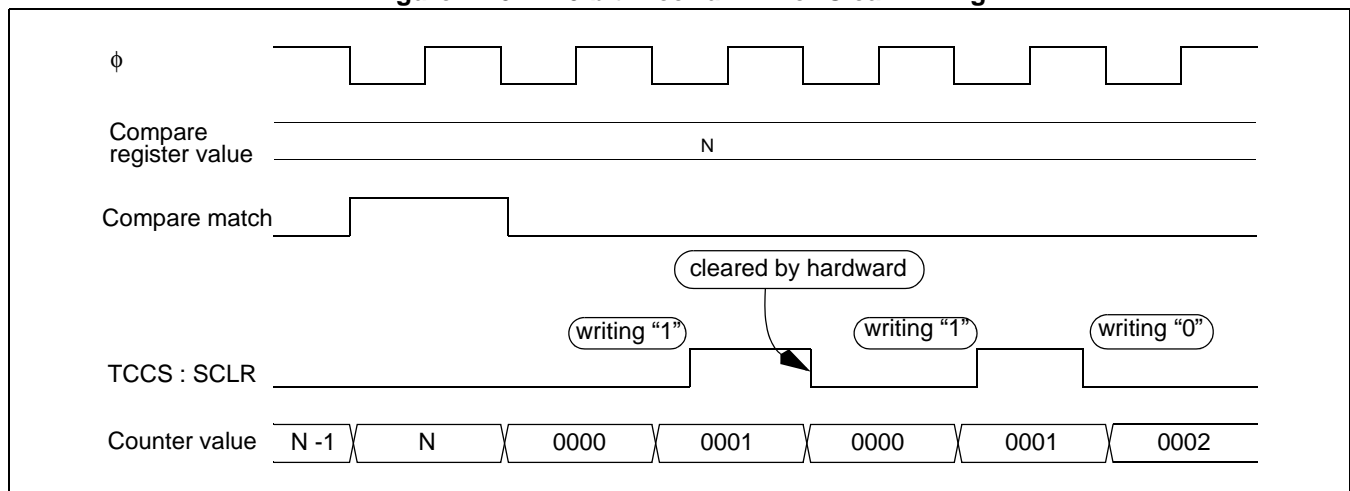
Note:

If writing "0" to the SCLR bit before a valid edge of count clock, the SCLR bit is cleared and the timer would not be cleared to "0000<sub>H</sub>".

- When "0000<sub>H</sub>" is written to the TCDT register during stop.
- Reset

By a reset, the counter is immediately cleared. By a software clear or a match with compare clear register, the counter is cleared in synchronization with the count timing. If the

**Figure 14.6-1 16-bit Free-run Timer Clear Timing**





## ■ Timer Mode

Two count modes can be selected in 16-bit free-run timer

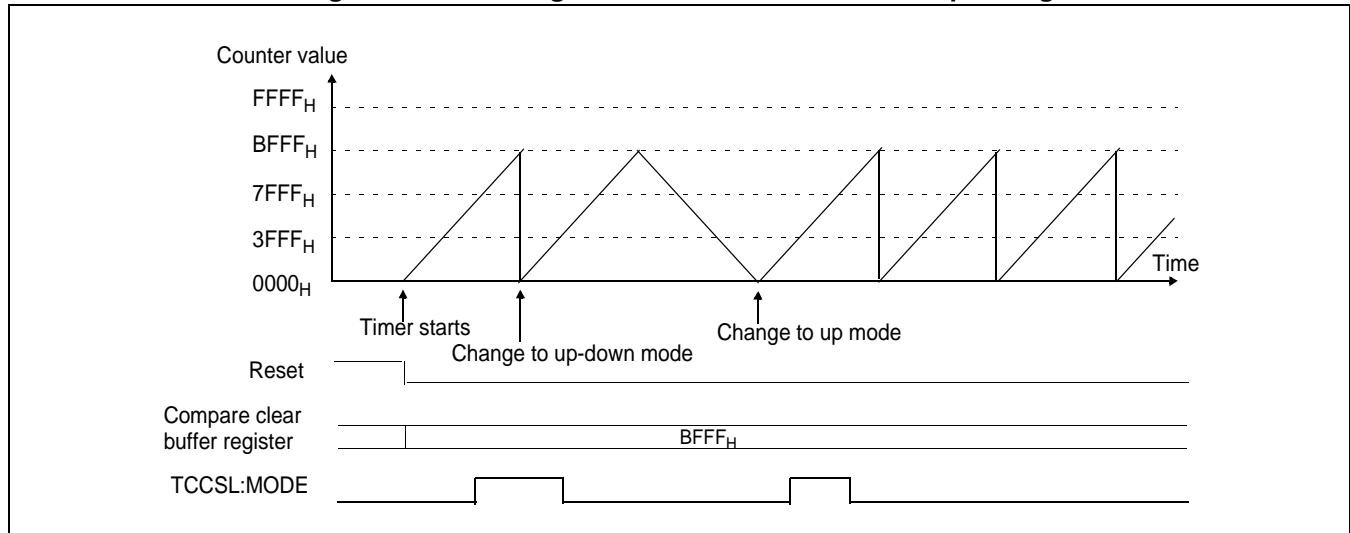
- up-count mode (TCCSL:MODE=0)
- up-down count mode (TCCSL:MODE=1)

In up-count mode, counter starts counting from pre-set timer data register (TCDT), counts up until counter value matches value of compare clear register (CPCLR), then counter is cleared to "0000<sub>H</sub>" and then counts up again.

In up-down count mode, counter starts counting from pre-set timer data register (TCDT), counts up until counter value matches value of compare clear register (CPCLR), then counter changes from up-count to down-count, counts down until counter value reaches "0000<sub>H</sub>" and then counts up again.

There is a buffer in mode bit, TCCSL:MODE, it can be written at any time no matter the timer is operating or stopped. While the timer is operating, value written to this bit is buffered and the count mode will be changed when timer value is "0000<sub>H</sub>".

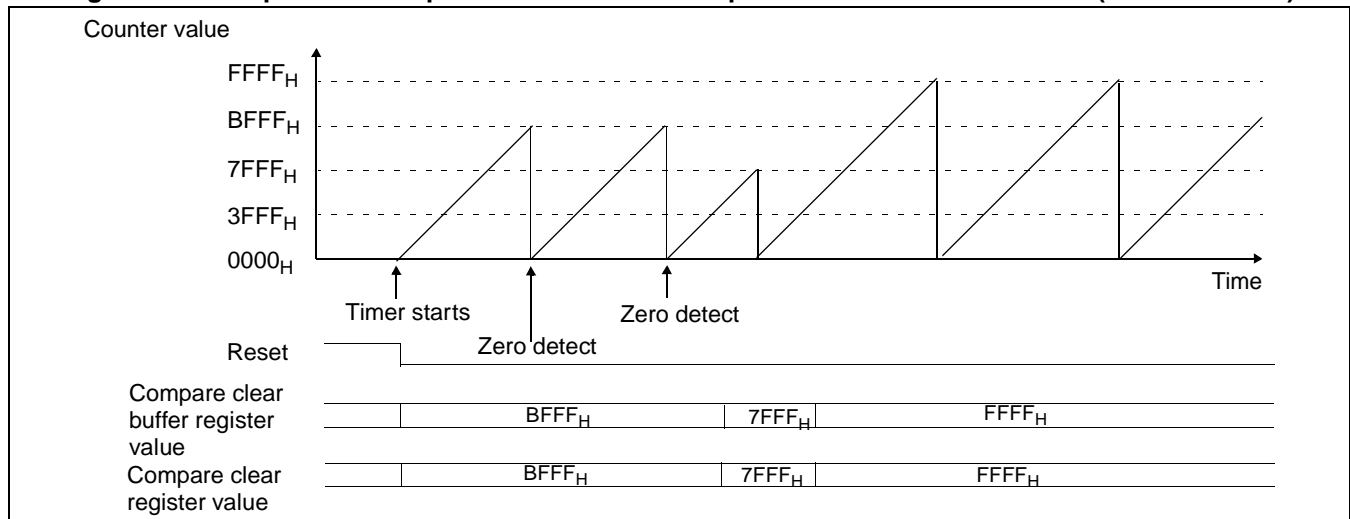
**Figure 14.6-2 Change Timer Mode while Timer is Operating**



## ■ Compare Clear Buffer

There is a selected buffer function on compare clear register (CPCLR). In buffer enable (TCCSL:BFE=1), data written in compare clear buffer register (CPCLRB) will transfer to CPCLR at zero detection of the 16-bit free-run timer. In buffer disable (TCCSL:BFE=0), CPCLRB is transparent, data can directly be written into CPCLR.

**Figure 14.6-3 Operation in Up-count Mode with Compare Clear Buffer is disabled (TCCSL:BFE=0)**



**Figure 14.6-4 Operation in Up-count Mode with Compare Clear Buffer is enabled (TCCSL:BFE=1)**

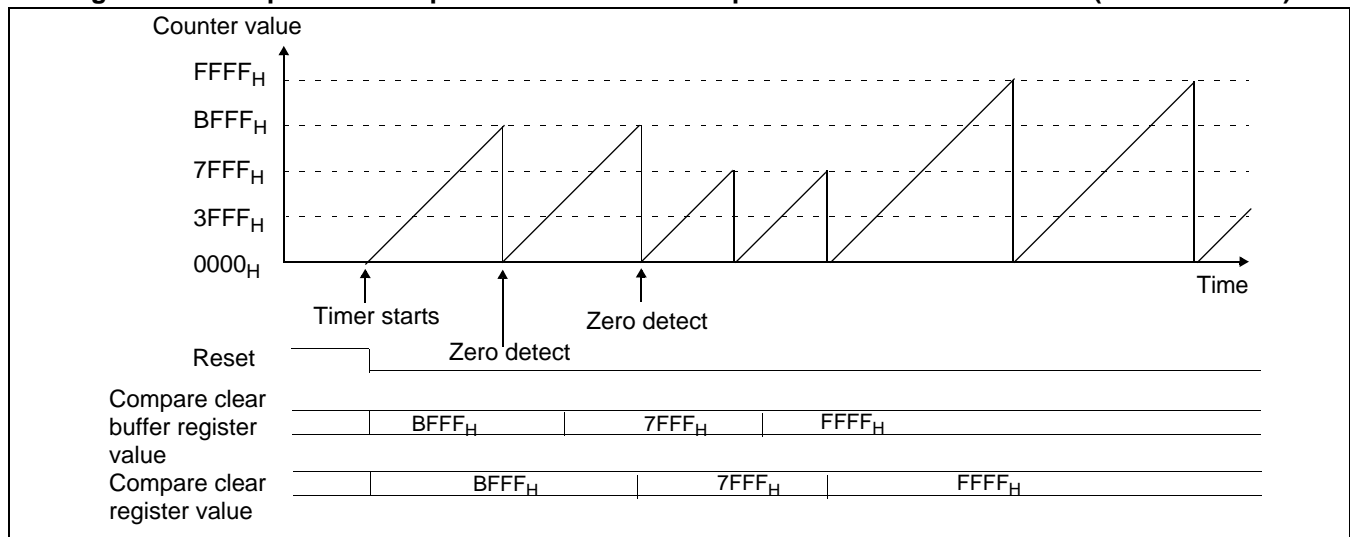
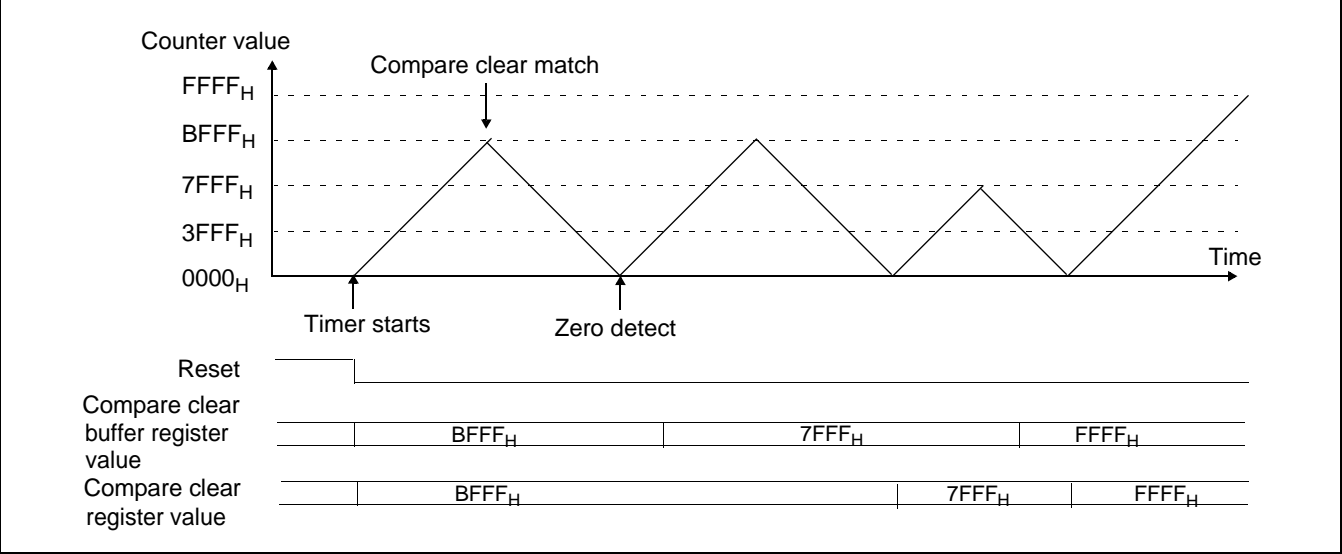


Figure 14.6-5 Operation in Up-down Count Mode with Compare Clear Buffer enabled (TCCSL:BFE=1)



## ■ Timer Interrupts

Two interrupts can be generated from 16-bit free-run timer:

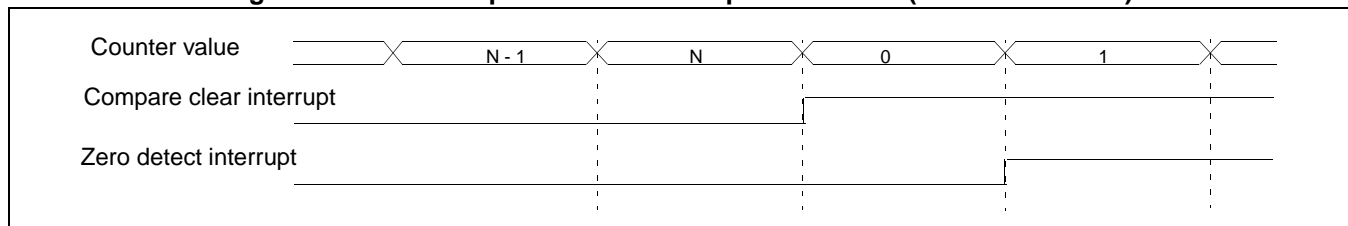
- Compare clear interrupt
- Zero detect interrupt

Compare clear interrupt is generated when the timer value matches compare clear register (CPCLR). Zero detect interrupt is generated when the timer value reaches "0000<sub>H</sub>".

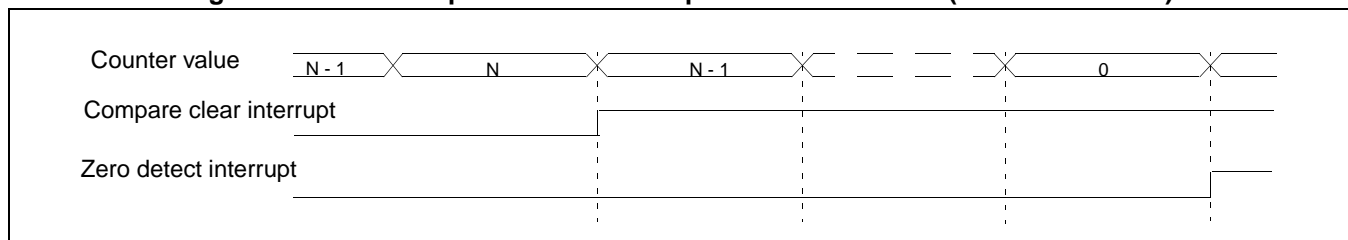
Note:

Software clear (TCCSL:SCLR=1) will not generate zero detect interrupt.

**Figure 14.6-6 Interrupts Generated in Up-count Mode (TCCSL:MODE=0)**



**Figure 14.6-7 Interrupts Generated in Up-down Count Mode (TCCSL:MODE=1)**



## ■ Interrupt Mask Function

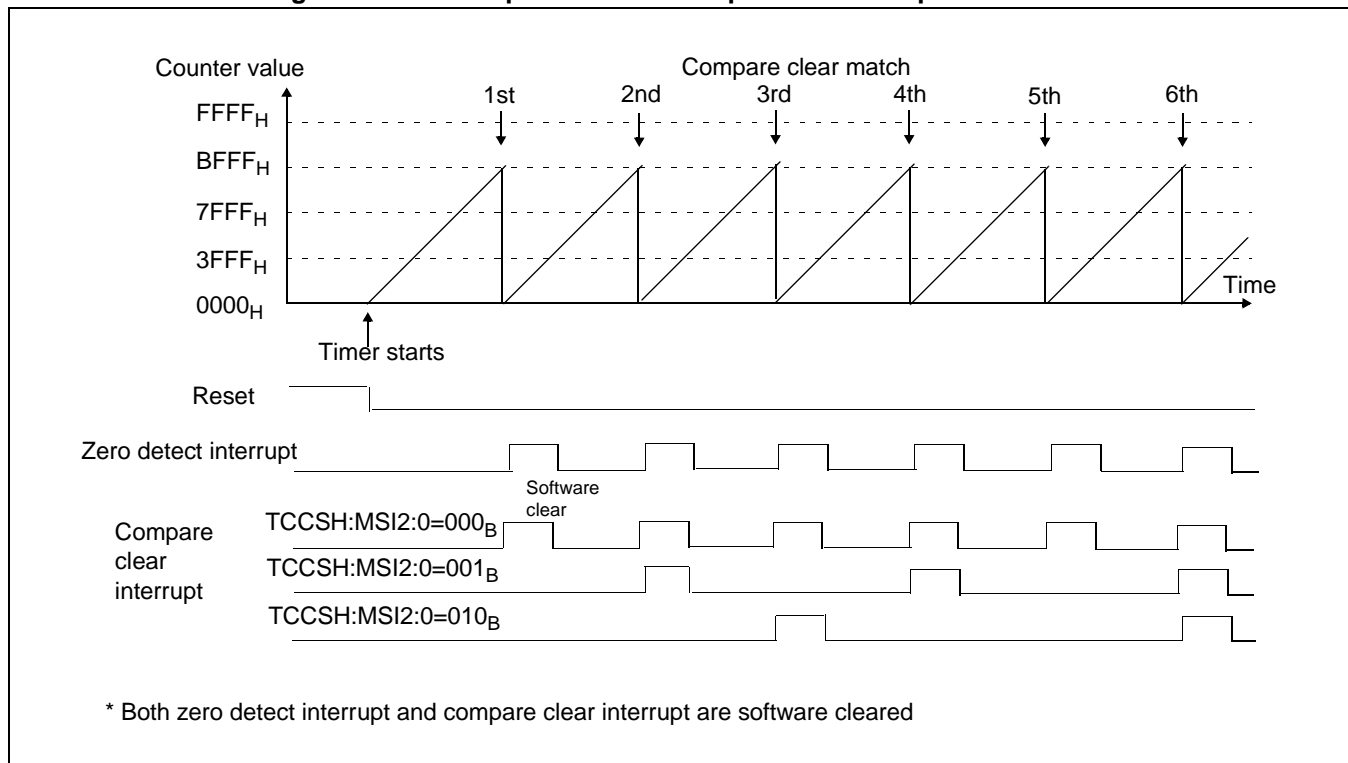
The number of times of Interrupt source can be masked by setting TCCSH:MSI2 to MSI0. MSI2 to MSI0 configure a 3-bit reload down counter, which reloads when its count value reaches "000<sub>B</sub>". Count value can also be loaded by writing directly to MSI2 to MSI0. The mask count equals the value set in MSI2 to MSI0 and there is no interrupt source will be masked when MSI2 to MSI0 equals "000<sub>B</sub>".

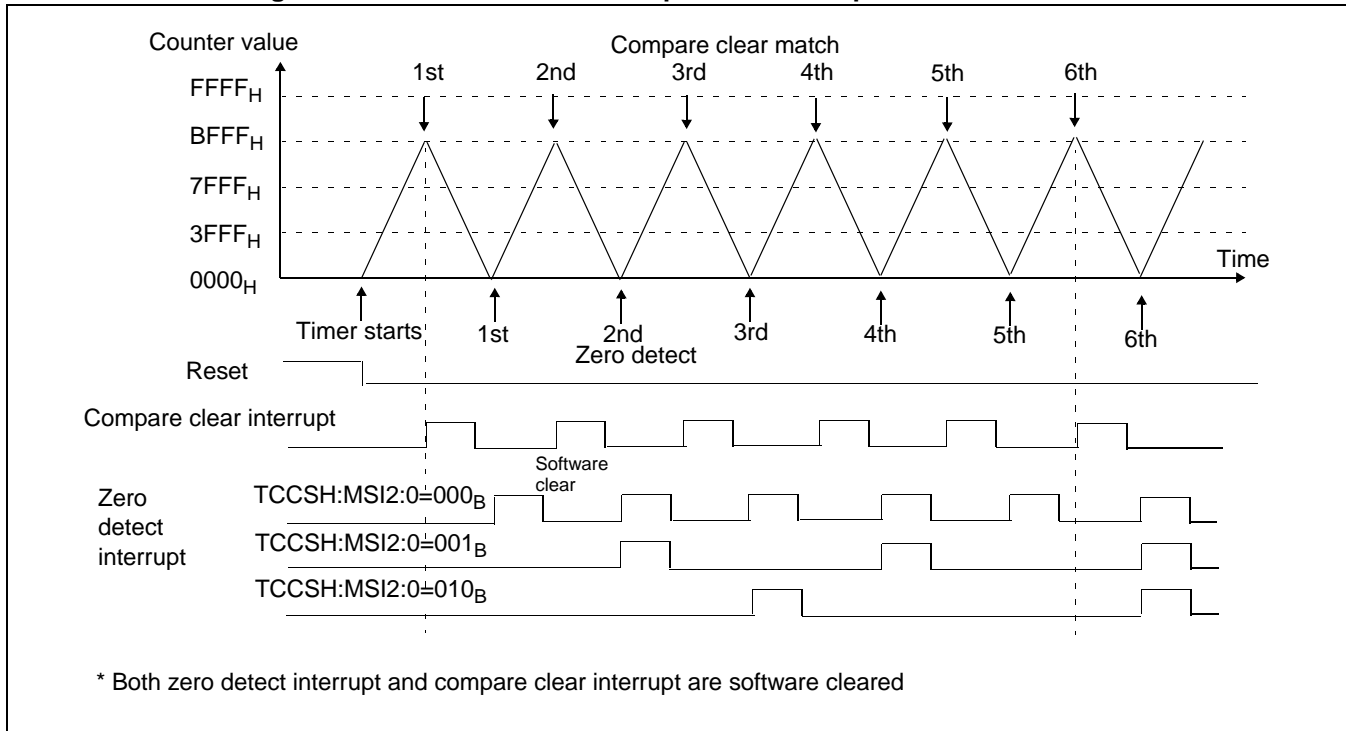
The interrupt source depends on the count mode (TCCSL:MODE). In up-count mode, only compare clear interrupt can be masked, zero detect interrupt is generated in every zero detection. In up-down count mode, only zero detect interrupt can be masked, compare clear interrupt is generated in every compare clear.

Note:

Software clear (TCCSL:SCLR=1) will not generate zero detection.

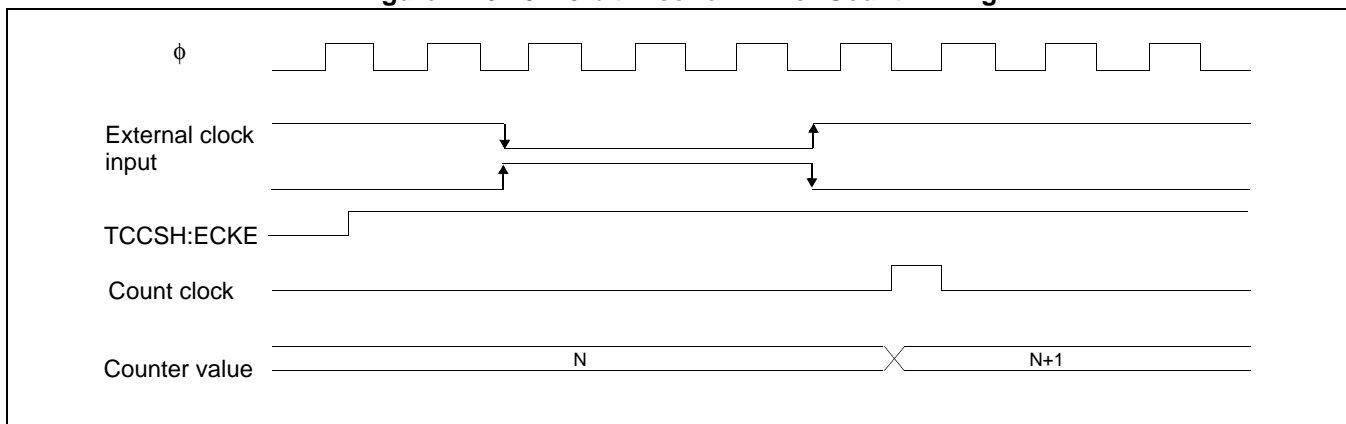
**Figure 14.6-8 Compare Clear Interrupt masked in Up-count Mode**



**Figure 14.6-9 Zero Detect Interrupt masked in Up-down Count Mode**

### ■ External Count Clock Selected

The 16-bit free-run timer is incremented based on the input clock (internal or external clock). When external clock is selected, the 16-bit free-run timer counts up at a rising edge when the initial value of external input is "1" or at a falling edge when initial value of external clock input is "0" after external clock mode is selected (TCCSH:ECKE=1).

**Figure 14.6-10 16-bit Free-run Timer Count Timing**

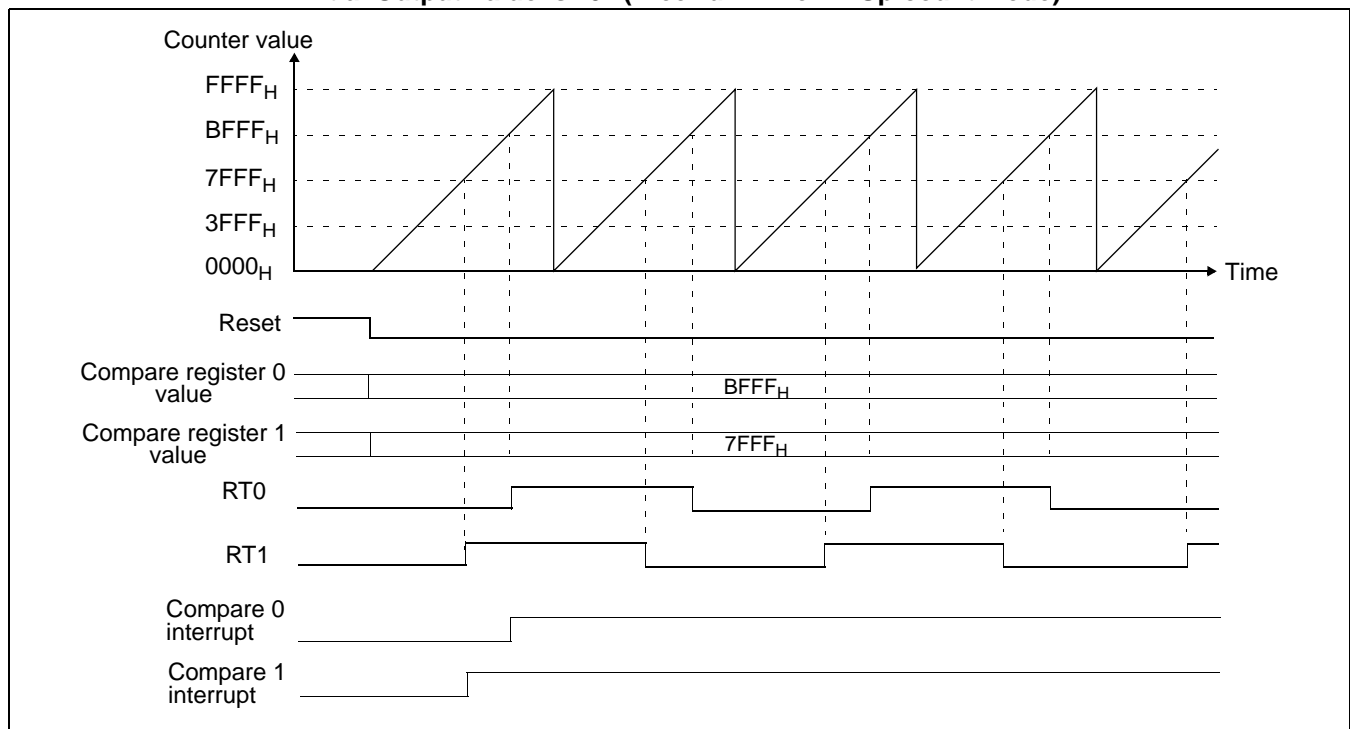
## 14.6.2 Operation of 16-bit Output Compare

The output compare unit is used to compare the value set in the specified compare register with the value of the 16-bit free-run timer. If a match is detected, the interrupt flag is set and the output level is inverted.

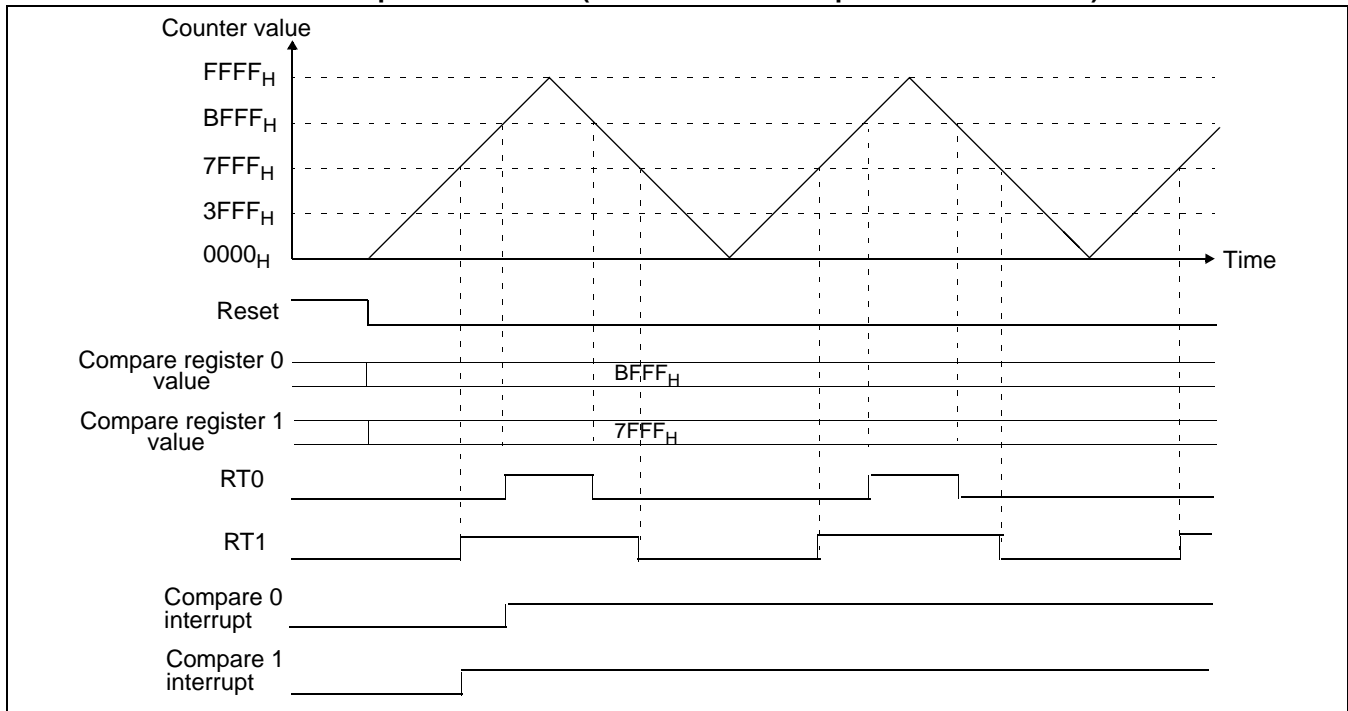
### ■ 16-bit Output Compare Operation

- Compare operation can be performed for individual channel (OCS1/OCS3/OCS5:CMOD = 0)

**Figure 14.6-11 Sample Output Waveform when Compare Registers 0 and 1 are used individually when the Initial Output Value is "0" (Free-run Timer in Up-count Mode)**

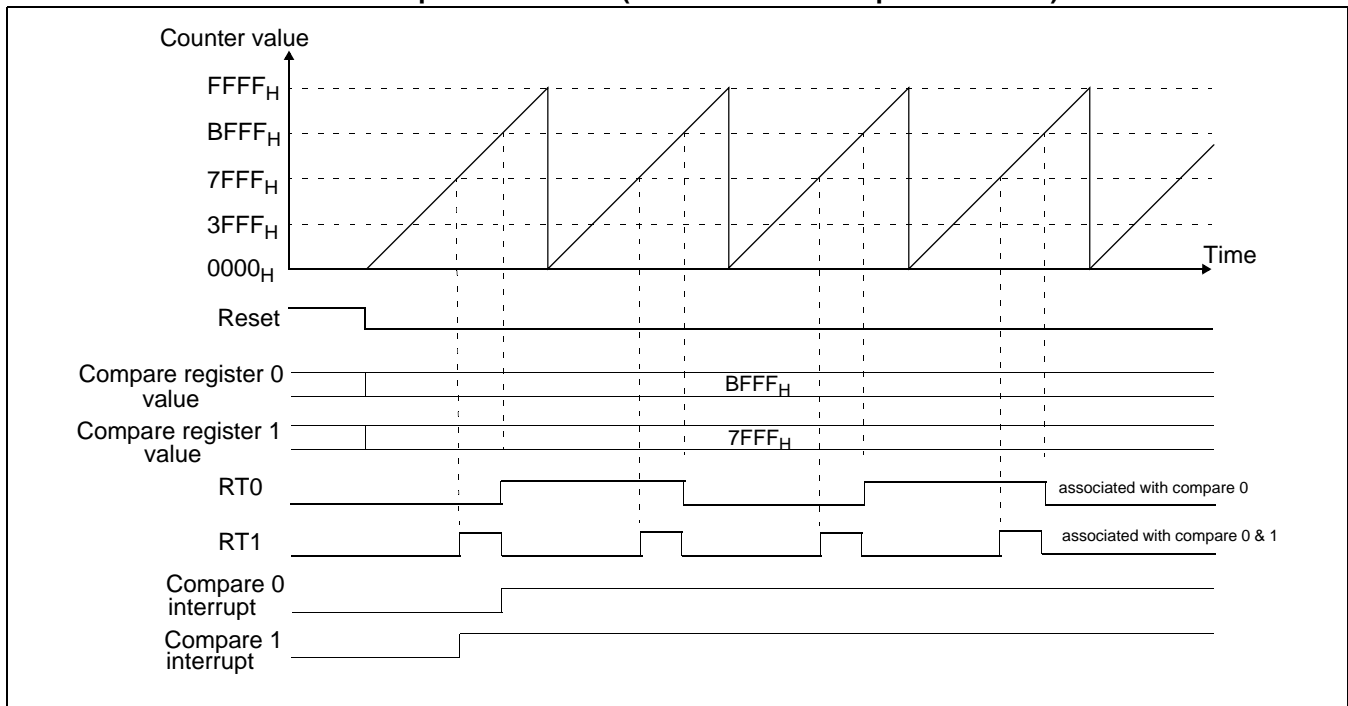


**Figure 14.6-12 Sample Output Waveform when Compare Registers 0 and 1 are used individually when the Initial Output Value is "0" (Free-run Timer in Up-down Count Mode)**



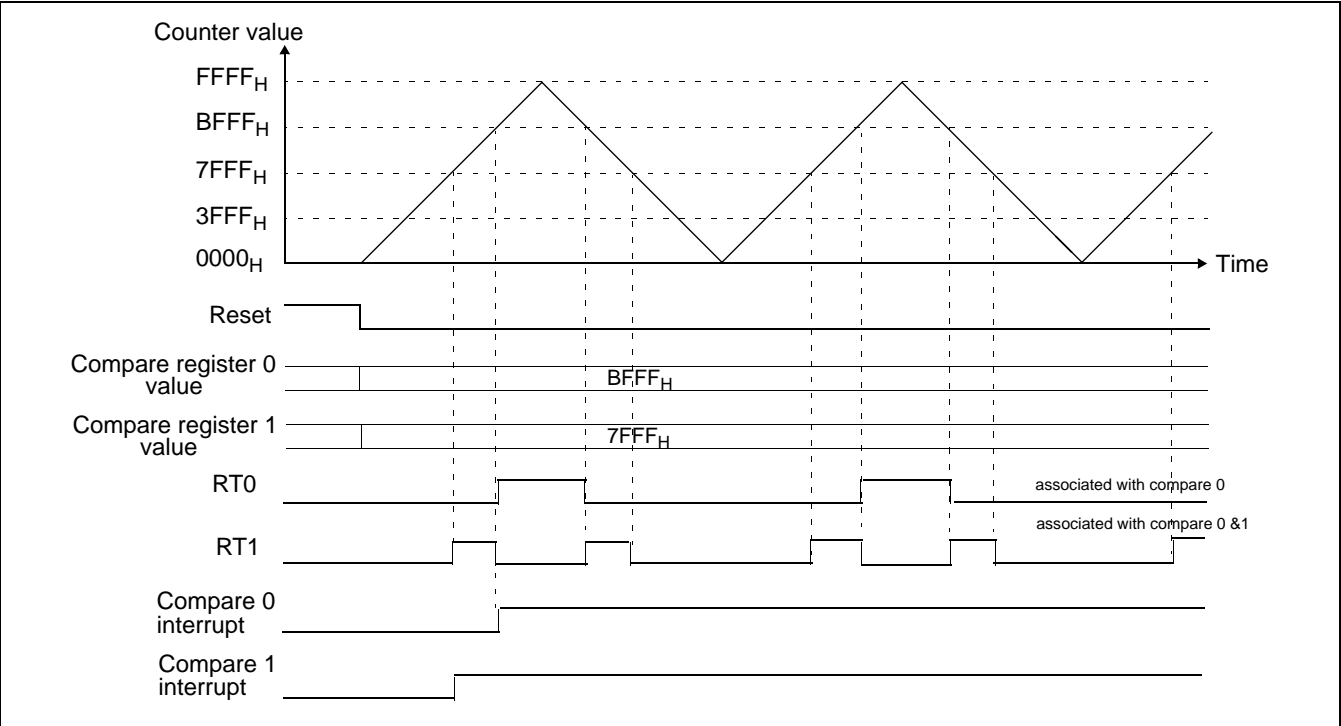
- Output level can be changed by using a pair of compare registers (OCS1/OCS3/OCS5:CMOD = 1)

**Figure 14.6-13 Sample Output Waveform when Compare Registers 0 and 1 are used in a Pair when the Initial Output Value is "0" (Free-run Timer in Up-count Mode)**



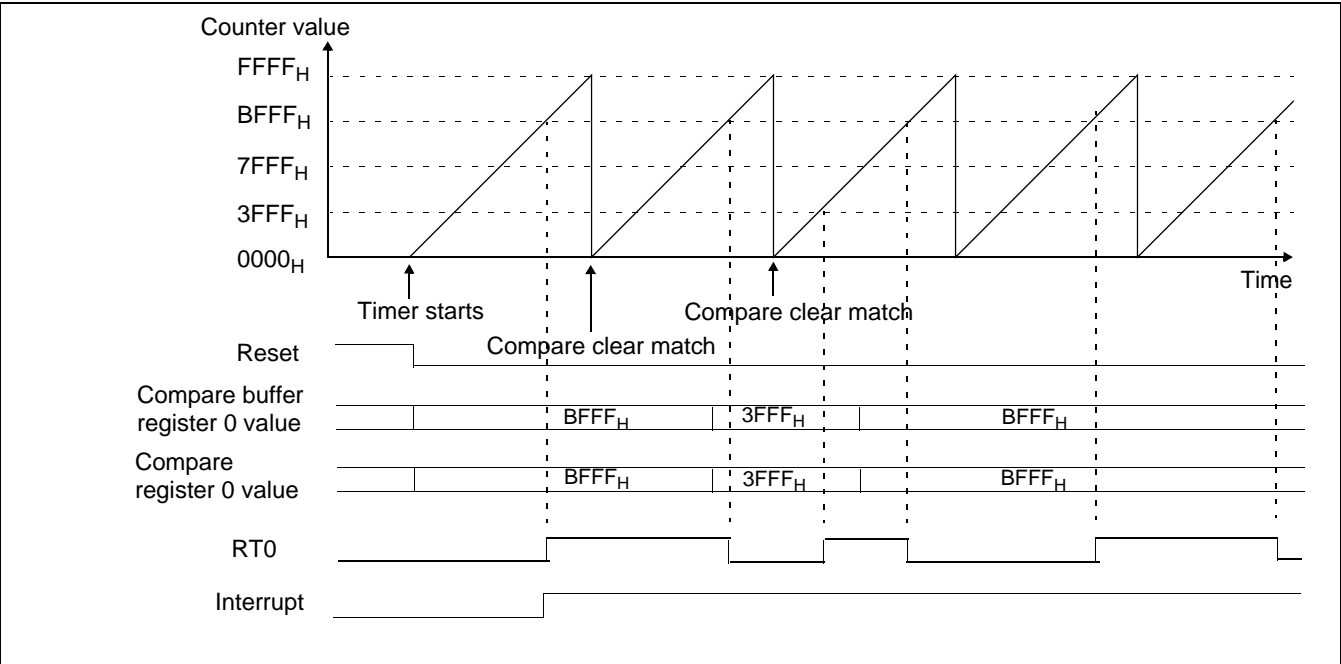


**Figure 14.6-14 Sample Output Waveform when Compare Register 0 and 1 are used in a Pair when the Initial Output Value is "0" (Free-run Timer in Up-down Count Mode)**



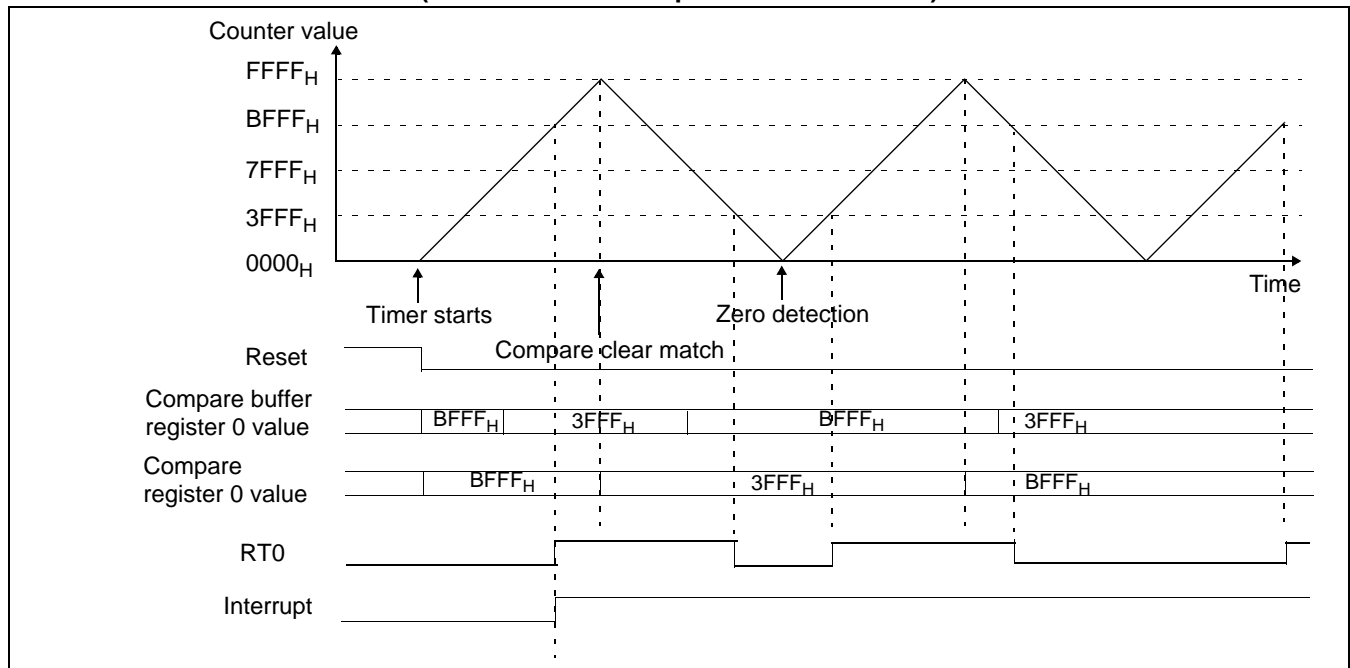
● Output level when compare buffer is disabled

**Figure 14.6-15 Sample Output Waveform when Compare Buffer is disabled (Free-run Timer in Up-count Mode)**



- Output level when compare buffer is selected at compare clear match

**Figure 14.6-16 Sample Output Waveform when Compare Buffer is enable  
(Free-run Timer in Up-down Count Mode)**



■ 16-bit Output Compare Timing

When the free-run timer matches the value set in the compare register, the output compare unit generates a compare match signal to invert the output and generate an interrupt. When a compare match occurs, the output is inverted in synchronization with the count timing of the counter.

Note: When the compare register is updated, comparison with the counter value is not performed.

Figure 14.6-17 Compare Operation upon Update of Compare Registers

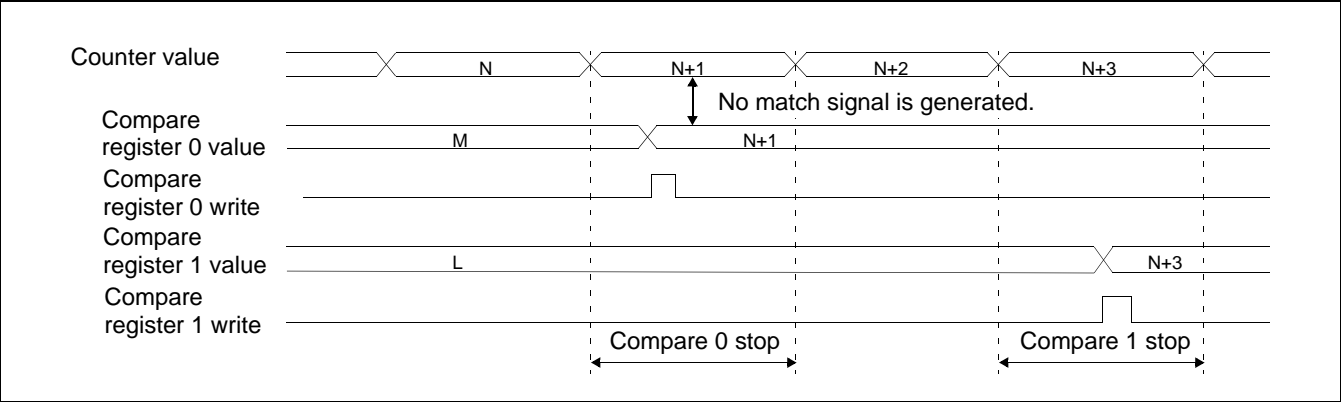


Figure 14.6-18 Compare Interrupt Timing

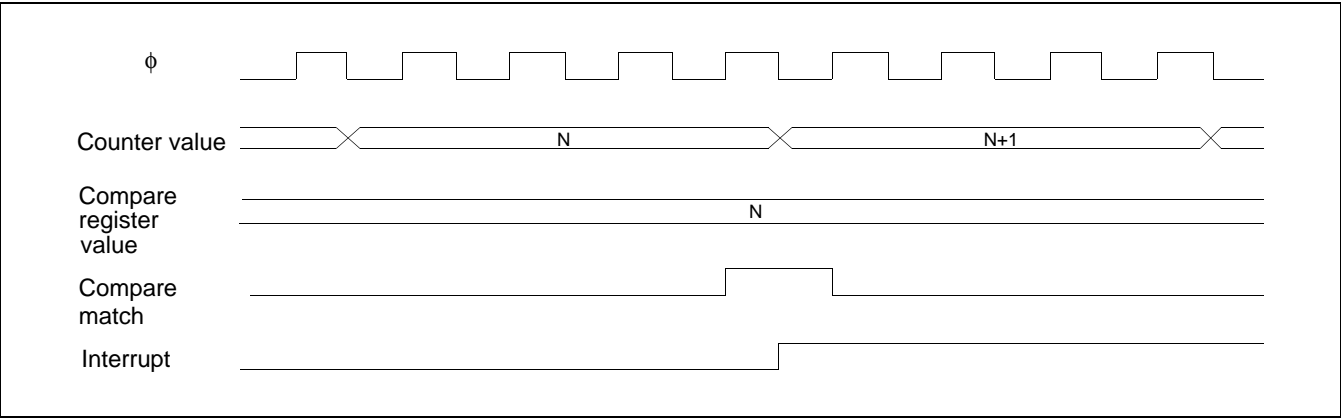
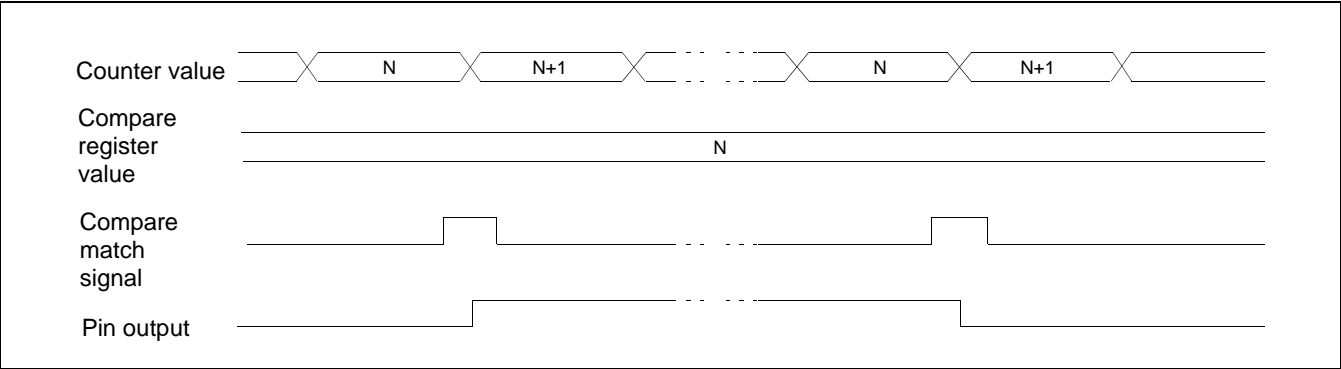


Figure 14.6-19 Output Pin Change Timing

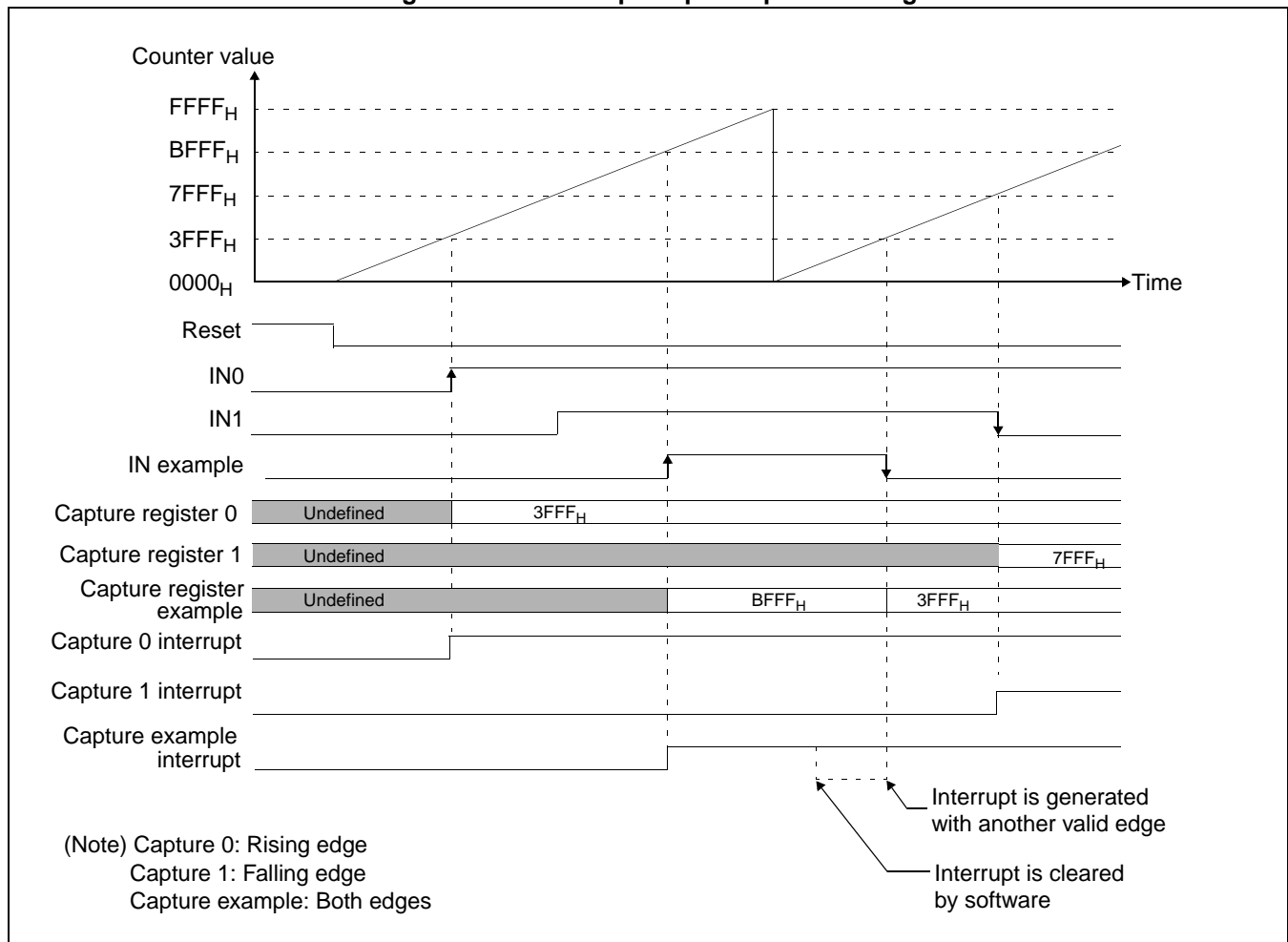


### 14.6.3 Operation of 16-bit Input Capture

The input capture unit is used to detect a specified valid edge. If a valid edge is detected, the interrupt flag is set and the value of 16-bit free-run timer is loaded into the capture register.

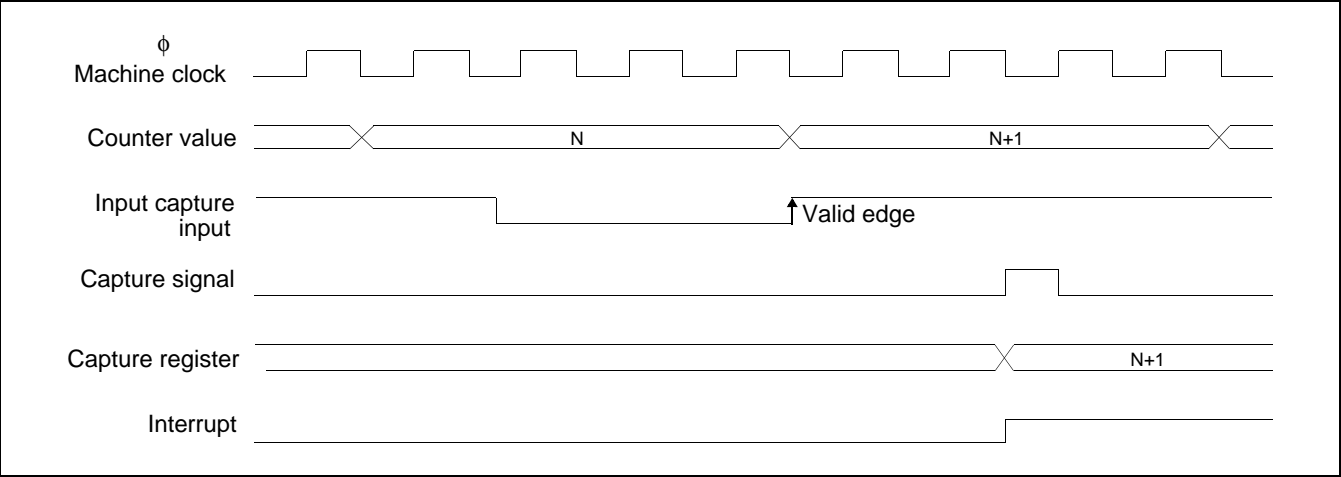
#### ■ 16-bit Input Capture Operation

Figure 14.6-20 Sample Input Capture Timing



■ 16-bit Input Capture Input Timing

Figure 14.6-21 16-bit Input Capture Timing for Input Signals



## 14.6.4 Operation of Waveform Generator

Waveform generator can produce various waveform such as dead-time, by using the real-time outputs (RT0 to RT5), 16-bit PPG timer 0 and 16-bit timers 0/1/2.

### ■ Output Condition of RTO0 to RTO5 and GATE

Table 14.6-1 Output Condition of RTO0 to RTO5, GATE and Register Bit Setting

TMD2	TMD1	TMD0	GTENx	PGENx	RTOx	GATE
0	0	0	X	X	Real-time output, RTx	Always “0”
0	0	1	X	0	Real-time output, RTx	OR(RTx & GTENx)
0	0	1	0	1	PPG0 output pulse when RTx is high	Always “0”
0	0	1	1	1	Gate triggered PPG0 output pulse when RTx is high	OR(RTx)
0	1	0	X	0	Output “H” from rising edge of RTx to 16-bit timer 0 underflow (x=0,1)	OR(RTOx & GTENx)
					Output “H” from rising edge of RTx to 16-bit timer 1 underflow (x=2,3)	
					Output “H” from rising edge of RTx to 16-bit timer 2 underflow (x=4,5)	
0	1	0	0	1	PPG0 output pulse from rising edge of RTx to 16-bit timer 0 underflow (x=0,1)	Always “0”
					PPG0 output pulse from rising edge of RTx to 16-bit timer 1 underflow (x=2,3)	
					PPG0 output pulse from rising edge of RTx to 16-bit timer 2 underflow (x=4,5)	
0	1	0	1	1	Gate triggered PPG0 output pulse from rising edge of RTx to 16-bit timer 0 underflow (x=0,1)	OR(output “H” from RTx/y/z rising edge to timer 0/1/2 underflow) x=0,1 y=2,3 z=4,5
					Gate triggered PPG0 output pulse from rising edge of RTx to 16-bit timer 1 underflow (x=2,3)	
					Gate triggered PPG0 output pulse from rising edge of RTx to 16-bit timer 2 underflow (x=4,5)	
1	0	0	X	X	Generate non-overlap signal by RT1 (x=0,1) *1	Always “0”
					Generate non-overlap signal by RT3 (x=2,3) *1	
					Generate non-overlap signal by RT5 (x=4,5) *1	
1	1	1	0	X	Generate non-overlap signal by PPG0	Always “0”
1	1	1	1	X	Generate non-overlap signal by gate triggered PPG0	OR(RTx)
Others					Always “0”	Always “0”

\*1 In order to generate non-overlap signal, be sure to select 2-channel mode for RT1/RT3/RT5 (OCS1/OCS3/OCS5:CMOD=1)

\*2 RTO0/RTO1 is controlled by DTCR0:TMD2 to TMD0, RTO2/RTO3 is controlled by DTCR1:TMD2 to TMD0 and RTO4/RTO5 is controlled by DTCR2:TMD2 to TMD0.

## ■ PPG0 Output Control

PPG0 output to RTO0 to RTO5 can be enabled by PGEN0 to PGEN5 in PPG output control/input capture control status register (PICS01).

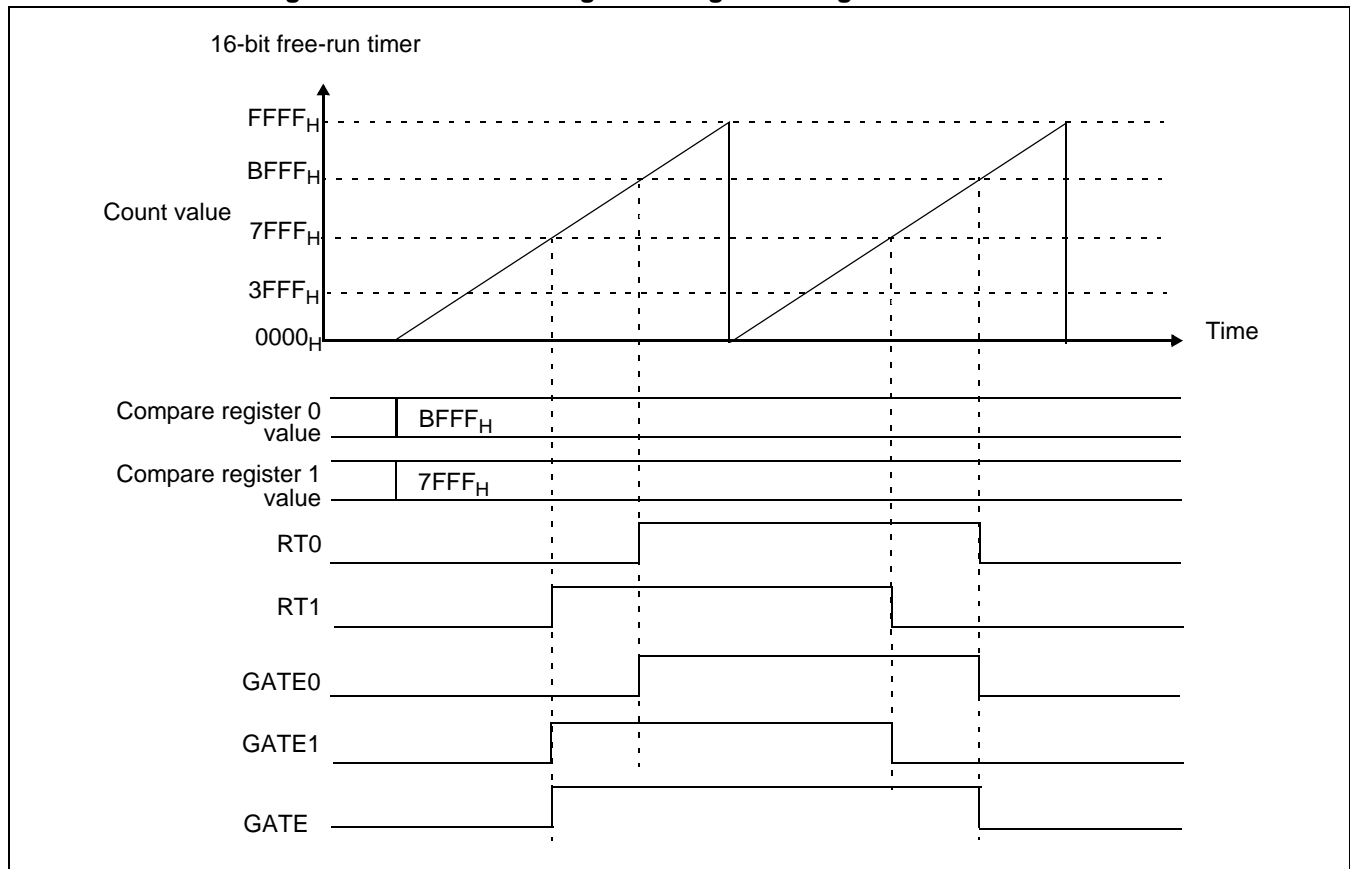
## ■ Gate Triggered PPG0 Output

In waveform generator, a GATE signal can be generated by using real-time outputs RT0 to RT5 or cope with 16-bit timers 0/1/2 to trigger PPG0 counting. When 16-bit timer is used, two real-time outputs RT0/RT2/RT4 and RT1/RT3/RT5 is operated with one 16-bit timer 0/1/2 to generate six individual gate signal. And these six gate signals are logically OR to generate a GATE signal to trigger PPG0 counting.

If PGEN0 to PGEN5 signal is also used, six different waveforms can be output to RTO0 to RTO5 by using one PPG0 only.

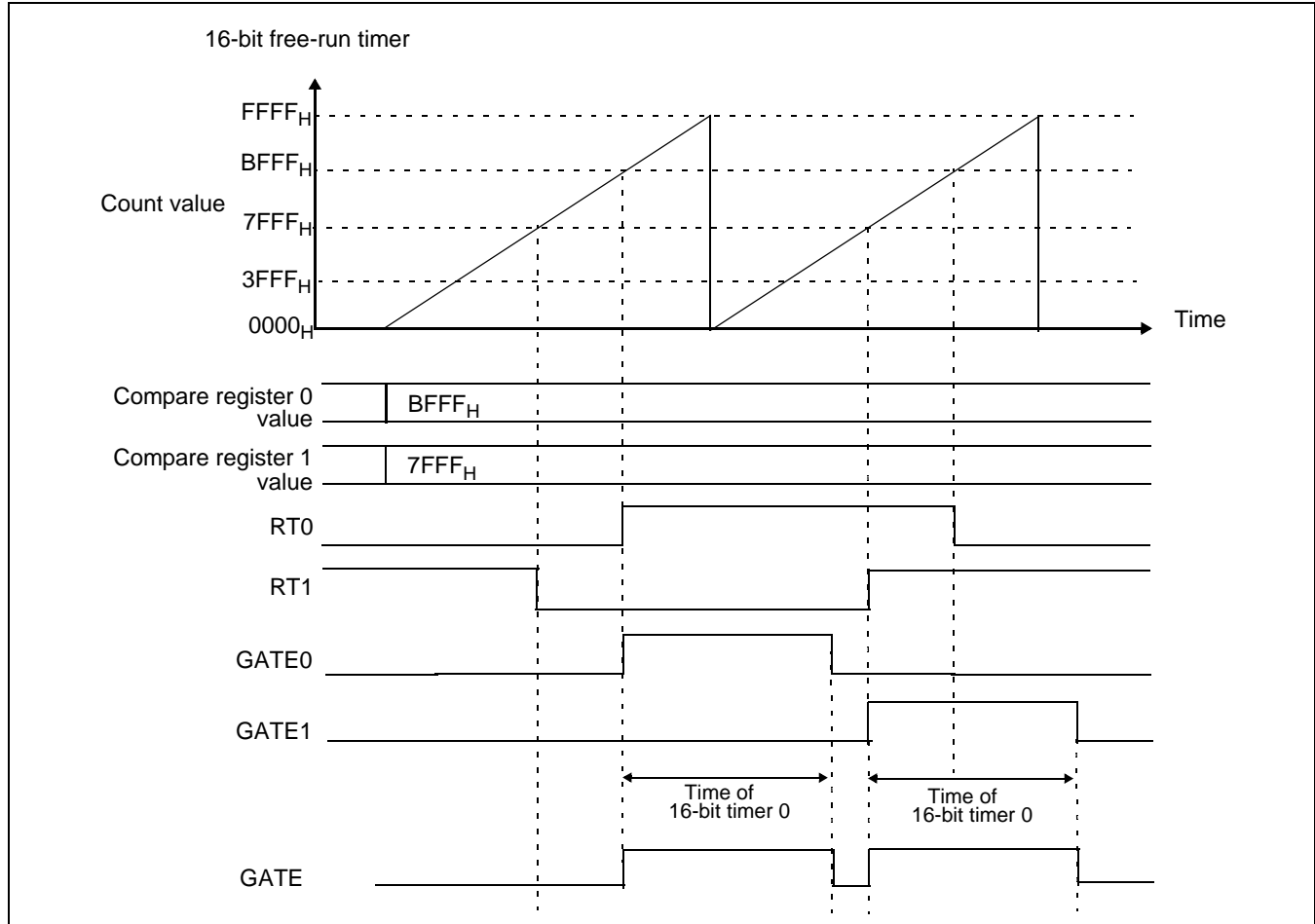
## ■ Generating GATE Signal during Each RTx is at "H" Level when GTENx is active (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=001<sub>B</sub> or 111<sub>B</sub>)

**Figure 14.6-22 Generating GATE Signal during RTx is at "H" Level**



■ **Generating GATE Signal from Rising Edge of Each RTx until 16-bit Timer 0/1/2 Underflow when GTENx is active (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=010<sub>B</sub>)**

**Figure 14.6-23 Generating GATE Signal from Rising Edge of RTx until 16-bit Timer Underflow**



**Note:**

Each 16-bit timer is used for two RTs. i.e. 16-bit timer 0 is used for RT0 and RT1; 16-bit timer 1 is used for RT2 and RT3; 16-bit timer 2 is used for RT4 and RT5. Therefore, do not use an RT and attempt to start the corresponding timer that is already operating. Doing so may cause that the outputting GATE signal will be extended and malfunction will be occurred.

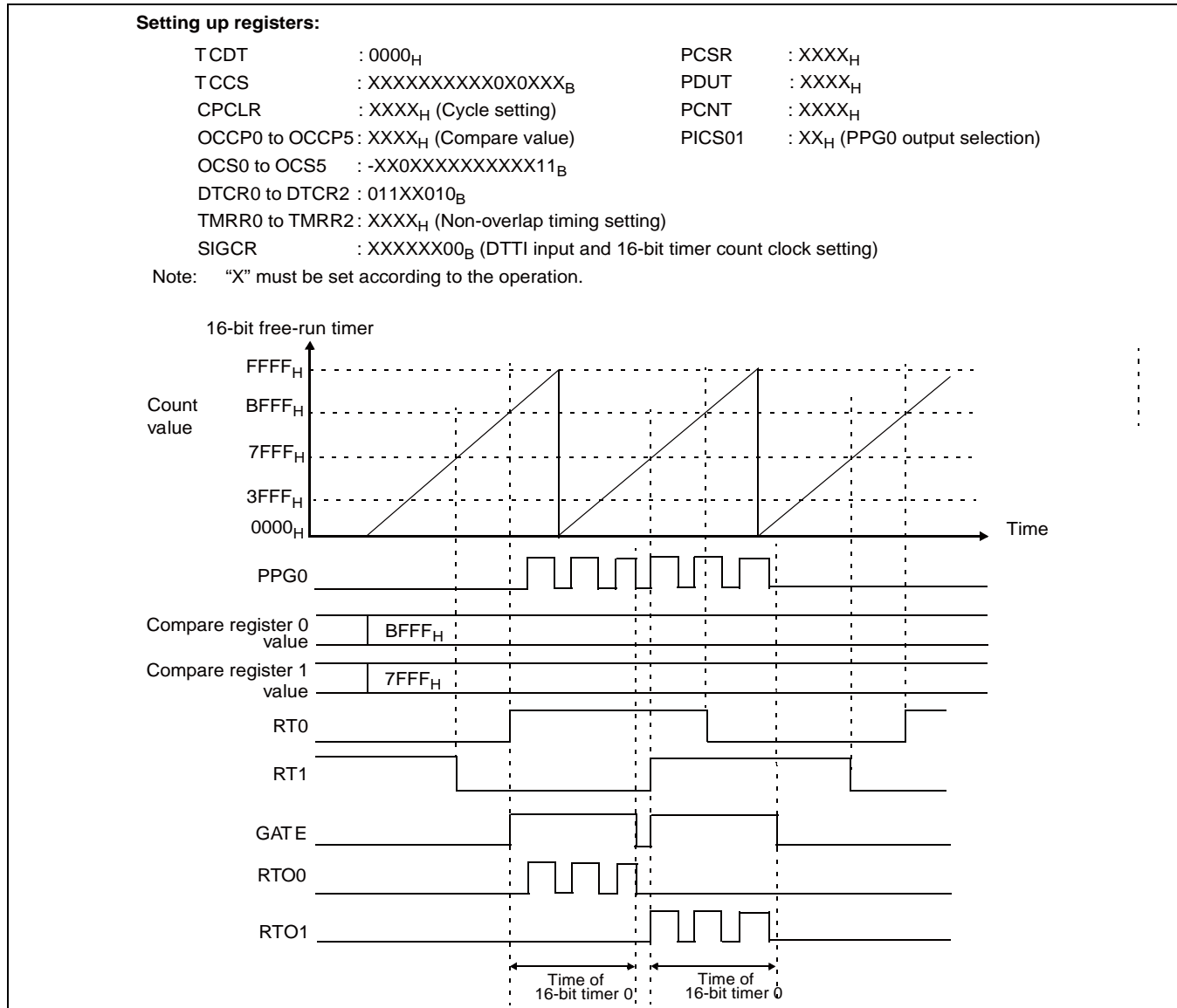


### 14.6.4.1 Operation in Timer Mode

With RT0 to RT5 rising edge, the 16-bit timer is reloaded, starts down-counting and the PPG timer 0 keeps outputting to RTO0 to RTO5 until the 16-bit timer is underflow.

#### ■ PPG0 Output Pulse from Rising Edge of RT to 16-bit Timer Underflow (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=010<sub>B</sub>)

Figure 14.6-24 Waveform generated when TMD2 to TMD0=010<sub>B</sub>



Note:

Each 16-bit timer is used for two RTs. i.e. 16-bit timer 0 is used for RT0 and RT1; 16-bit timer 1 is used for RT2 and RT3; 16-bit timer 2 is used for RT4 and RT5. Therefore, do not use an RT and attempt to start PPG0 which is under operation. Doing so may cause that the outputting GATE signal will be extended and malfunction will be occurred.

### 14.6.4.2 Operation in Dead-time Timer Mode

The dead-time generator will input the real-time output (RT1/RT3/RT5), select PPG timer 0 pulse output, and output non-overlap signals (inverted signals) to external pins (RTO0 to RTO5).

#### ■ Making Non-overlap Signals by using RT1/RT3/RT5 in Normal Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=100<sub>B</sub>)

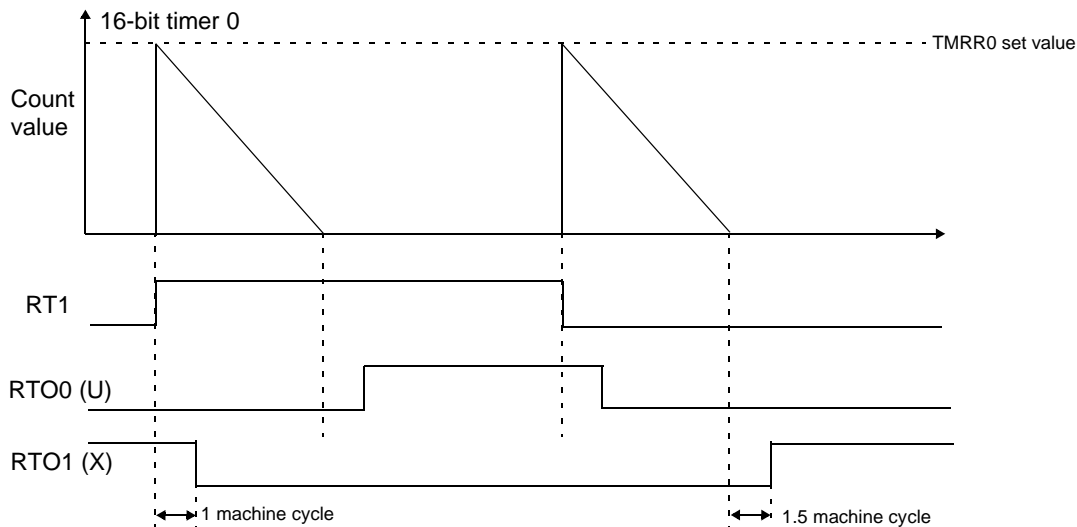
When selecting non-overlap signal for an active level "0" (normal polarity) in DTCR0/DTCR1/DTCR2:DMOD, a delay corresponding to the non-overlap time set in the TMRR0/TMRR1/TMRR2 register (16-bit timer register) is applied. The delay is applied at a rising edge of RT1/RT3/RT5 or its falling edge. If RT1/RT3/RT5 pulse width is smaller than the set non-overlap time, the 16-bit timer will restart down-counting from TMRR0/TMRR1/TMRR2 value at the next RT's edge.

**Figure 14.6-25 Non-overlap Signal Generation by RT1/RT3/RT5 in Normal Polarity**

Setting up registers:

- TCDT : 0000<sub>H</sub>
- TCCS : X-XXXXXX0X0XXX<sub>B</sub>
- OCCP0 to OCCP5 : XXXX<sub>H</sub> (Compare value)
- TMRR0 to TMRR2 : XXXX<sub>H</sub> (Non-overlap timing setting)
- SIGCR : XXXXXXXX<sub>B</sub> (DTTI0 input and 16-bit timer count clock setting)
- CPCLR : XXXX<sub>H</sub> (Cycle setting)
- OCS0 to OCS5 : -XX1XXXXXXXXXX11<sub>B</sub>
- DTCR0 to DTCR2 : 0XXXX100<sub>B</sub>

Note: "X" must be set according to the operation.



Pin name	Output signal
RTO0 (U)	Signal with delay is applied at RT1 rising edge
RTO2 (V)	Signal with delay is applied at RT3 rising edge
RTO4 (W)	Signal with delay is applied at RT5 rising edge
RTO1 (X)	Inverted signal with delay is applied at RT1 falling edge
RTO3 (Y)	Inverted signal with delay is applied at RT3 falling edge
RTO5 (Z)	Inverted signal with delay is applied at RT5 falling edge

### ■ Making Non-overlap Signals by using RT1/RT3/RT5 in Inverted Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=100<sub>B</sub>)

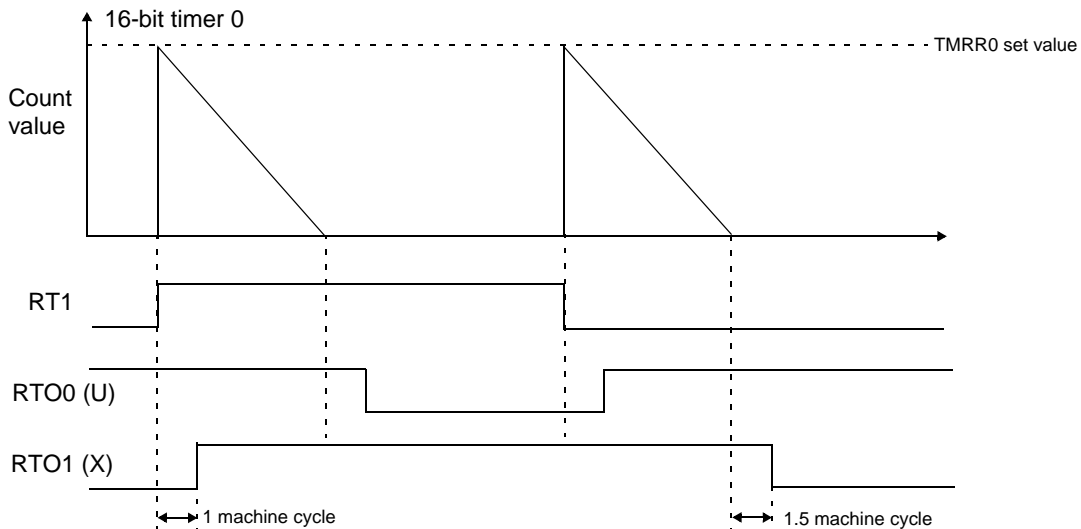
When selecting non-overlap signal for a active level "1" (inverted polarity) in DTCR0/DTCR1/DTCR2:DMOD, a delay corresponding to the non-overlap time set in the TMRR0/TMRR1/TMRR2 register (16-bit timer register) is applied. The delay is applied at a rising edge of RT1/RT3/RT5 or its falling edge. If RT1/RT3/RT5 pulse width is smaller than the set non-overlap time, the 16-bit timer will restart down-counting from TMRR0/TMRR1/TMRR2 value at the next RT's edge.

**Figure 14.6-26 Non-overlap Signal Generation by RT1/RT3/RT5 in Inverted Polarity**

**Setting up registers:**

- TCDT : 0000<sub>H</sub>
- TCCS : XXXXXXXXXXX0X0XX<sub>B</sub>
- OCCP0 to OCCP5: XXXX<sub>H</sub> (Compare value)
- TMRR0 to MRR2 : XXXX<sub>H</sub> (Non-overlap timing setting)
- SIGCR : XXXXXXXX<sub>B</sub> (DTT10 input and 16-bit timer count clock setting)
- CPCLR : XXXX<sub>H</sub> (Cycle setting)
- OCS0 to OCS5 : -XX1XXXXXXXXXX11<sub>B</sub>
- DTCR0 to DTCR2 : 1XXXX100<sub>B</sub>

Note: "X" must be set according to the operation.



Pin name	Output signal
RTO0 (U)	Inverted signal with delay is applied at RT1 rising edge
RTO2 (V)	Inverted signal with delay is applied at RT3 rising edge
RTO4 (W)	Inverted signal with delay is applied at RT5 rising edge
RTO1 (X)	Signal with delay is applied at RT1 falling edge
RTO3 (Y)	Signal with delay is applied at RT3 falling edge
RTO5 (Z)	Signal with delay is applied at RT5 falling edge

### ■ Making Non-overlap Signals by using PPG in Normal Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111<sub>B</sub>)

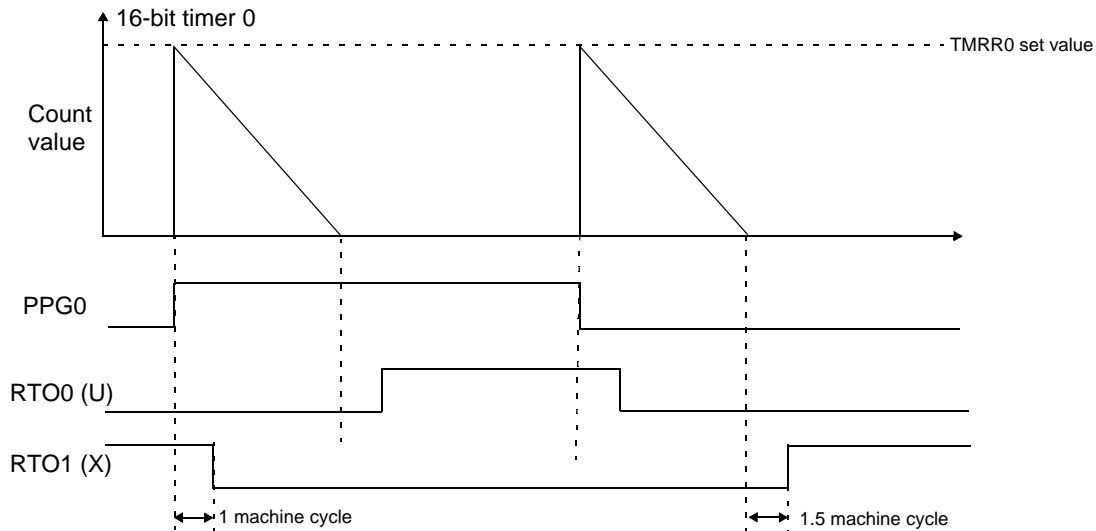
When selecting non-overlap signal for a active level "0" (normal polarity) in DTCR0/DTCR1/DTCR2:DMOD, a delay corresponding to the non-overlap time set in the TMRR0/TMRR1/TMRR2 register (16-bit timer register) is applied. The delay is applied at a rising edge of PPG timer 0 pulse signal or its inverted signal. If PPG timer pulse width is smaller than the set non-overlap time, the 16-bit timer will start down-counting from TMMR0/TMMR1/TMMR2 value at the next edge of PPG0 pulse.

**Figure 14.6-27 Non-overlap Signal Generation by PPG0 in Normal Polarity**

#### Setting up registers:

- TCDT : 0000<sub>H</sub>
- TCCS : XXXXXXXXXXXX0X0XX<sub>B</sub>
- CPCLR : XXX<sub>H</sub> (Cycle setting)
- OCCP0 to OCCP5: XXX<sub>H</sub> (Compare value)
- OCS0 to OCS5 : -XX1XXXXXXXXXX11<sub>B</sub>
- DTCR0 to DTCR2 : 0XXXX111<sub>B</sub>
- TMRR0 to TMRR2: XXX<sub>H</sub> (Non-overlap timing setting)
- SIGCR : XXXXXXX<sub>B</sub> (DTT10 input and 16-bit timer count clock setting)
- PCSR : XXX<sub>H</sub>
- PDUT : XXX<sub>H</sub>
- PCNT : XXX<sub>H</sub>

Note: "X" must be set according to the operation.



Pin name	Output signal
RTO0 (U)	Signal with delay is applied at PPG0 rising edge
RTO2 (V)	Signal with delay is applied at PPG0 rising edge
RTO4 (W)	Signal with delay is applied at PPG0 rising edge
RTO1 (X)	Inverted signal with delay is applied at PPG0 falling edge
RTO3 (Y)	Inverted signal with delay is applied at PPG0 falling edge
RTO5 (Z)	Inverted signal with delay is applied at PPG0 falling edge

## ■ Making Non-overlap Signals by using PPG in Inverted Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111<sub>B</sub>)

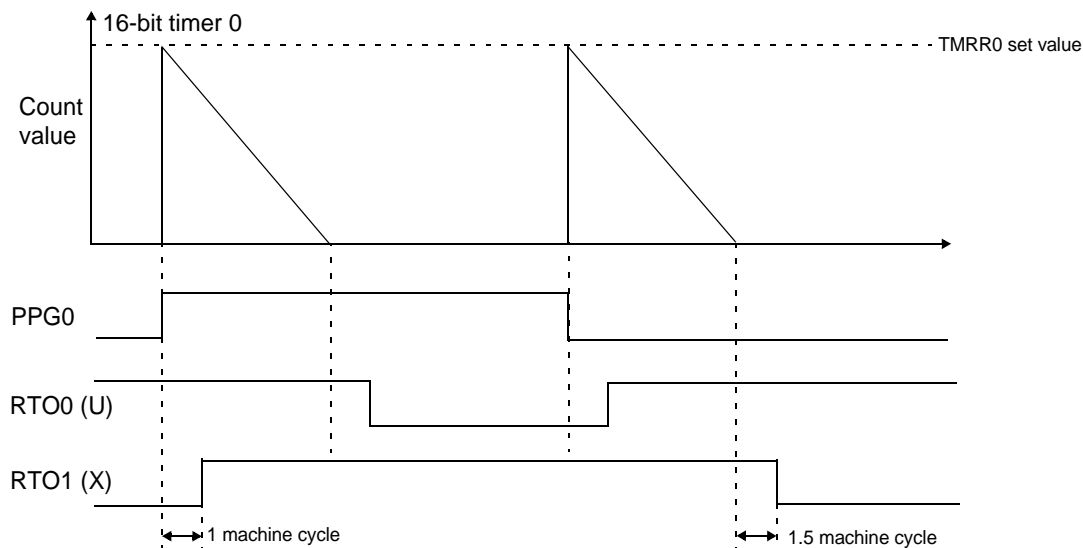
When selecting non-overlap signal for an active level "1" (inverted polarity) in DTCR0/DTCR1/DTCR2:DMOD, a delay corresponding to the non-overlap time set in the TMRR0/TMRR1/TMRR2 register (16-bit timer register) is applied. The delay is applied at a rising edge of PPG timer 0 pulse signal or its inverted signal. If PPG timer 0 pulse width is smaller than the set non-overlap time, the 16-bit timer will start down-counting from TMMR0/TMMR1/TMMR2 value at the next edge of PPG0 pulse.

**Figure 14.6-28 Non-overlap Signal Generation by PPG0 in Inverted Polarity**

### Setting up registers:

- TCDT : 0000<sub>H</sub>
- TCCS : XXXXXXXXXXXX0X0XXX<sub>B</sub>
- CPCLR : XXXX<sub>H</sub> (Cycle setting)
- OCCP0 to OCCP5 : XXXX<sub>H</sub> (Compare value)
- OCS0 to OCS5 : -XX1XXXXXXXXXX11<sub>B</sub>
- DTCR0 to DTCR2 : 1XXXX111<sub>B</sub>
- TMRR0 to TMRR2 : XXXX<sub>H</sub> (Non-overlap timing setting)
- SIGCR : XXXXXXXX<sub>B</sub> (DTTIO input and 16-bit timer count clock setting)
- PCSR : XXXX<sub>H</sub>
- PDUT : XXXX<sub>H</sub>
- PCNT : XXXX<sub>H</sub>

Note: "X" must be set according to the operation.



Pin name	Output signal
RTO0 (U)	Inverted signal with delay is applied at PPG0 rising edge
RTO2 (V)	Inverted signal with delay is applied at PPG0 rising edge
RTO4 (W)	Inverted signal with delay is applied at PPG0 rising edge
RTO1 (X)	Signal with delay is applied at PPG0 falling edge
RTO3 (Y)	Signal with delay is applied at PPG0 falling edge
RTO5 (Z)	Signal with delay is applied at PPG0 falling edge

### 14.6.4.3 Operation of DTTI0 Pin Control

By setting "1" to waveform control register, SIGCR: bit7 (DTIE), the output of RTO0 to RTO5 can be controlled by the DTTI0 pin. When "L" level in DTTI0 is detected, the output of RTO0 to RTO5 will be fixed to an inactive level until the interrupt flag, SIGCR: bit6 (DTIF) is cleared. The inactive level of RTO0 to RTO5 can be set by PDR3 in port 3 by software.

#### ■ DTTI0 Pin Input Operation

Even when the "L" level of DTTI0 pin input is detected, the timer will keep running for the waveform generator operation, but no waveform will outputted to external pins P30/RTO0 to P35/RTO5.

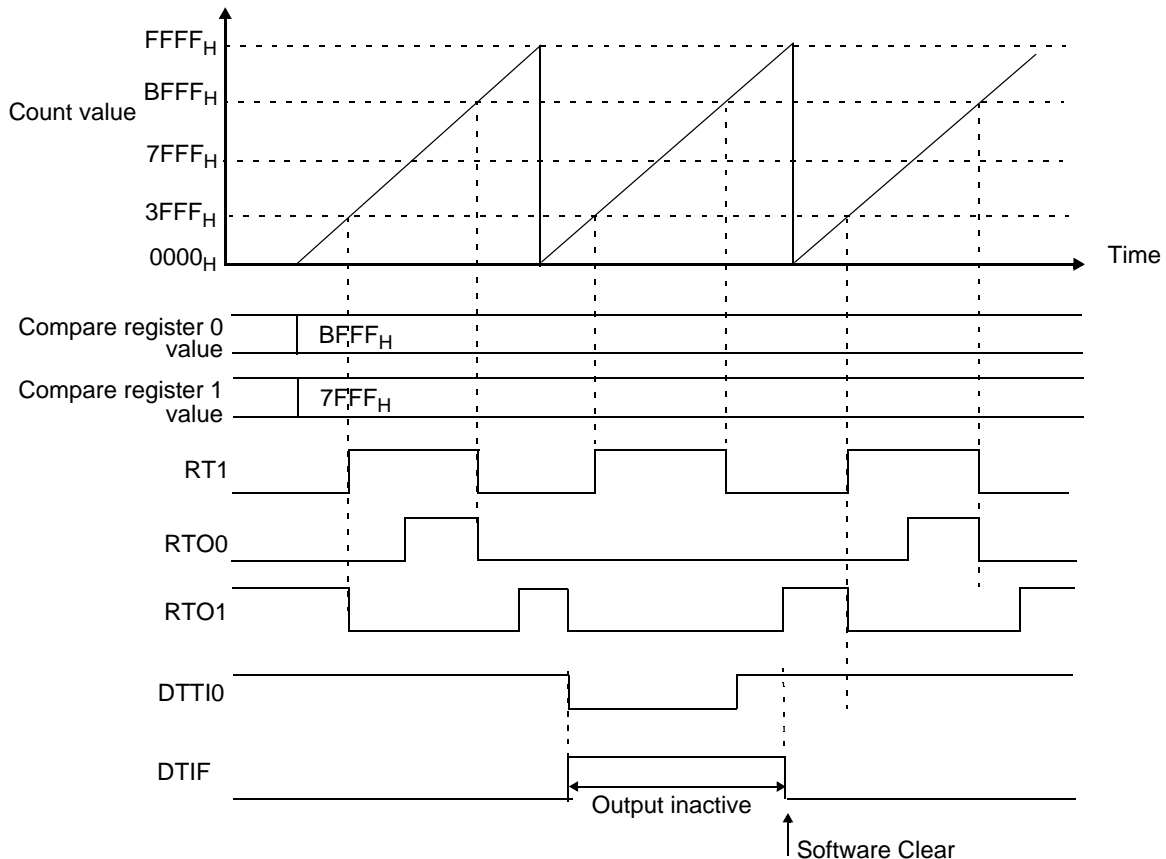
Figure 14.6-29 Operation when DTTI0 Input is enabled

##### Setting up registers:

- |                  |   |                  |                                     |
|------------------|---|------------------|-------------------------------------|
| • TCDT           | : 0000 <sub>H</sub>   | • CPCLR          | : XXXX <sub>H</sub> (Cycle setting) |
| • TCCS           | : XXXXXXXXXX0X0XX <sub>B</sub>  | • OCS0 to OCS5   | : -XX1XXXXXXXXXX11 <sub>B</sub>     |
| • OCCP0 to OCCP5 | : XXXX <sub>H</sub> (Compare value)                                       | • DTCR0 to DTCR2 | : 0XXXX100 <sub>B</sub>             |
| • PDR3           | : XXXXXX00 <sub>B</sub> (Inactive level setting)                          |                  |                                     |
| • TMRR0 to TMRR2 | : XXXX <sub>H</sub> (Non-overlap timing setting)                          |                  |                                     |
| • SIGCR          | : 1XXXXXX <sub>B</sub> (DTTI0 input and 16-bit timer count clock setting) |                  |                                     |

Note: "X" must be set according to the operation.

16-bit free-run timer



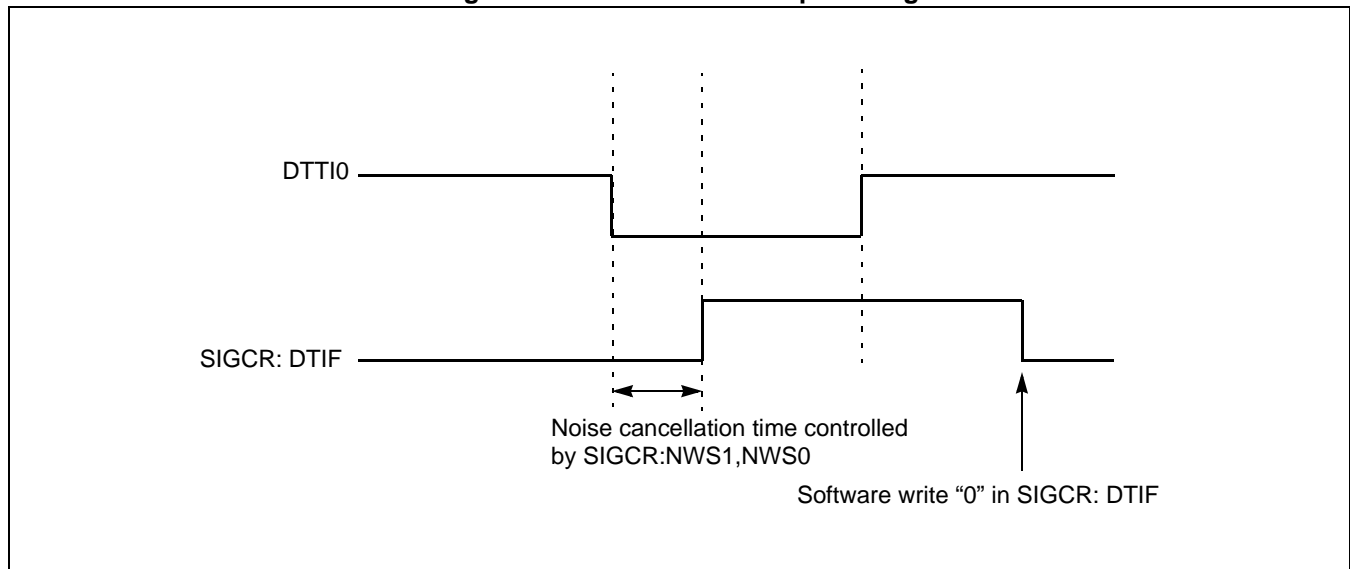
## ■ DTTI0 Pin Noise Cancellation Function

By setting bit5 (NRSL) of the waveform control register (SIGCR) to "1", the noise cancellation function for DTTI0 pin input is enabled. When noise cancellation function is enabled, the time for fixing an output pin RTO0 to RTO5 to inactive level is delayed for about 4, 8, 16 or 32 machine cycles (selected by SIGCR:NWS1,NWS0). Since the noise cancellation circuit uses a peripheral clock, input is invalidated even if the DTTI0 input is enabled in a mode such as STOP mode in which the oscillation stops.

## ■ DTTI0 Interrupt

When low level of DTTI0 is detected, DTTI0 interrupt flag (SIGCR:DTIF) is set to "1" after noise cancellation time is passed and an interrupt request is sent to interrupt controller.

**Figure 14.6-30 DTTI0 Interrupt Timing**



Notes:

- If SIGCR:NWS1,NWS0 is changed within noise cancellation time, the larger value of NWS1, NWS0 will take effect.
- SIGCR:DTIF can only be software clear.

## 14.7 Usage Notes on the Multi-functional Timer

---

Notes on using the multi-functional timer are given below.

---

### ■ Usage Notes on the 16-bit Free-run Timer

#### ● Notes about using a program for setting

- After reset, the timer value is "0000<sub>H</sub>", zero detect interrupt flag will be set to "1" in next count clock after timer enable (TCCSL:STOP=0).
- Since the timer mode bit (TCCSL:MODE) has a buffer, changing timer mode will take effect in next count cycle. Zero detect interrupt is always generated when timer mode is changed from up-count to up-down count mode.
- Software clear timer (TCCSL:SCLR=1) will initialize the timer but not generate zero detect interrupt.

#### ● Notes about interrupts

- When the IRQZF bit of the timer control status register (TCCSH) is set to "1" and an interrupt request is enabled (TCCSH:IRQZE=1), control cannot be returned from interrupt processing. Always clear the IRQZF bit.
- When the ICLR bit of the timer control status register (TCCSH) is set to "1" and an interrupt request is enabled (TCCSH:ICRE=1), control cannot be returned from interrupt processing. Always clear the ICLR bit.
- Since the 16-bit free-run timer shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the 16-bit free-run timer, shared resource interrupts must be disabled.

### ■ Usage Notes on the 16-bit Output Compare

#### ● Notes about interrupts

- When the IOP bit of the compare control register (OCS0/OCS2/OCS4) is set to "1" and an interrupt request is enabled (OCS0/OCS2/OCS4:IOE=1), control cannot be returned from interrupt processing. Always clear the IOP bit.
- Since the 16-bit output compare shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the 16-bit output compare, shared resource interrupts must be disabled.



## ■ Usage Notes on the 16-bit Input Capture

### ● Notes about interrupts

- When the ICP bit of the input capture control status register (PICSL01/ICSL23) is set to "1" and an interrupt request is enabled (PICSL01/ICSL23:ICE=1), control cannot be returned from interrupt processing. Always clear the ICP bit.
- If input capture pins IN is toggled after ICP bit is set but before interrupt routine is processed, the edge indication bit (ICSH23:IEI3,IEI2 or PICSH01:IEI1,IEI0) will show the latest edge detected.
- Since the 16-bit input capture shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the 16-bit input capture, shared resource interrupts must be disabled.

## ■ Usage Notes on the Waveform Generator

### ● Notes on using a program for setting

- Change the TMD2, TMD1 and TMD0 bits of the 16-bit timer control register (DTCR0/DTCR1/DTCR2) when the waveform generator is under operation (TMD2 to TMD0=001<sub>B</sub>, 010<sub>B</sub>, 100<sub>B</sub> or 111<sub>B</sub>), always be sure no trigger source and timer is not counting. Otherwise unexpected waveform in RTO will be occurred due to prescheduled output by previous trigger. But RTO output becomes normal once after timer is underflow or retriggered by new trigger source in new mode setting. Trigger source is H level of RT when TMD2 to TMD0=001<sub>B</sub>, rising edge of RT when TMD2 to TMD0=010<sub>B</sub>, rising/falling edge of RT when TMD2 to TMD0=100<sub>B</sub> or rising/falling edge of PPG0 when TMD2 to TMD0=111<sub>B</sub>.

For example, changing TMD2 to TMD0 from 100<sub>B</sub> to 111<sub>B</sub>, you can set in following procedures

- 1) set TMRR0/TMRR1/TMRR2 to a very small value like 0001<sub>H</sub>
  - 2) set RT1/RT3/RT5 to output "L"/"H" and wait until timer 0/1/2 underflow
  - 3) change mode bits TMD2, TMD1 and TMD0 and corresponding setting
  - 4) corrected output waveform will appear in RTO pins one machine cycle later
- Writing a value in 16-bit timer register (TMRR0/TMRR1/TMRR2) during timer counting, new value will be valid at the next timer trigger. And always be sure to use a word transfer instruction (MOVW A, dir, etc.) to access timer register.
  - Change the DCK2, DCK1 and DCK0 bits of the waveform control register (SIGCR) when the timer is not counting.
  - Change the NWS1 and NWS0 bits of waveform control register (SIGCR) when the noise cancellation function is disabled (SIGCR: NRSL=0).

### ● Notes about interrupts

- When the TMIF bit of the timer control register (DTCR) is set to "1" and an interrupt request is enabled (DTCR:TMIE=1), control cannot be returned from interrupt processing. Always clear the TMIF bit.
- When the DTIF bit of the waveform control register (SIGCR) is set to "1", control cannot be returned from interrupt processing. Always clear the DTIF bit.
- Since the waveform generator shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the waveform generator, shared resource interrupts must be disabled.

## 14.8 Sample Programs for the Multi-functional Timer

This section contains sample programs for the multi-functional timer.

### ■ Sample Program for 16-bit Free-run Timer

#### ● Processing

- A 4 ms compare clear interrupt is generated with 16-bit free-run timer 0.
- The timer is used in up-down mode to repeatedly generate a compare clear interrupt.
- EI<sup>2</sup>OS is not used.
- 16 MHz is used for the machine clock, and 62.5 ns is used for the count clock.

#### ● Coding example

```

ICR11      EQU      0000BBH      ;Interrupt control register for the
                                   16-bit free-run timer
TCCS       EQU      00005EH      ;Timer control status register
CPCLRB     EQU      000058H      ;Compare clear buffer register
ICLR       EQU      TCCS:9       ;Interrupt request flag bit
;-----Main program-----
CODE                               CSEG
START:
;           :                       ;Assumes that stack pointer (SP) has already been
                                   ;initialized
                                   AND      CCR,#0BFH      ;Interrupt disable
                                   MOV      I:ICR11,#00H   ;Interrupt level 0 (strongest)
                                   MOVW     I:CPCLRB,#0FFFFH ;Set compare clear value to change
                                   ;16-bit free-run timer from up-count to
                                   ;down-count
                                   MOVW     I:TCCS,#0110H  ;Sets up-down mode, 62.5 ns count clock
                                   ;Enables compare clear interrupt
                                   ;Disables interrupt mask
                                   ;Clears interrupt flag and enable timer
                                   MOV      ILM,#07H       ;Sets ILM in PS to level 7
                                   OR       CCR,#40H       ; Interrupt enable
LOOP:      MOV      A,#00H      ;Endless loop
                                   MOV      A,#01H
                                   BRA      LOOP
;-----Interrupt program-----

```

```

WARI:
        CLRB      I:ICLR      ;Clears interrupt request flag
;
;      :
;      User processing
;      :
        RETI      ;Returns from interrupt

CODE    ENDS
;-----Vector setting-----
VECT    CSEG      ABS=0FFH
        ORG      0FF74H      ;Sets vector for interrupt #34 (22H)
        DSL      WARI
        ORG      0FFDCH      ;Sets reset vector
        DSL      START
        DB      00H      ;Sets single-chip mode
VECT    ENDS
        END      START

```

## ■ Sample Program for 16-bit Output Compare

### ● Processing

A output compare match interrupt is generated when the count value of 16-bit free-run timer is matched with output compare 0.

- The 16-bit free-run timer is used in up-count mode.
- EI<sup>2</sup>OS is not used.
- 16 MHz is used for the machine clock, and 62.5 ns is used for the count clock of 16-bit free-run timer.

### ● Coding example

```

ICR00    EQU      0000B0H      ;Interrupt control register for the output compare 0
TCCS     EQU      00005EH      ;Timer control status register
CPCLRB   EQU      000058H      ;Compare clear buffer register
OCCP0    EQU      000070H      ;Output compare register 0
OCCP1    EQU      000072H      ;Output compare register 1
OCS01    EQU      00007CH      ;Compare control register
IOP      EQU      OCS01:6      ;Interrupt request flag bit
;-----Main program-----
CODE     CSEG
START:
;      :
;      ;Assumes that stack pointer (SP) has already been
;      initialized

```

```

        AND        CCR,#0BFH        ;Interrupt disable
        MOV        I:ICR00,#00H      ;Interrupt level 0 (strongest)
        MOVW       I:TCCS,#0000H     ;Enables 16-bit free-run timer
                                   ;Sets up-count mode
        MOVW       I:OCCP0,#0BFFFH   ;Set output compare register 0
        MOVW       I:OCCP1,#07FFFH   ;Set output compare register 1
        MOVW       I:OCS01,#0C1FH    ;Enables output compare output
                                   ;Enables compare match interrupt 0
                                   ;Clears interrupt flag and enable output compare
        MOV        ILM,#07H          ;Sets ILM in PS to level 7
        OR         CCR,#40H          ; Interrupt enable
LOOP:    MOV        A,#00H            ;Endless loop
        MOV        A,#01H            ;
        BRA        LOOP              ;
;-----Interrupt program-----
WARI:
        CLRB       I:IOP             ;Clears interrupt request flag
;
;      :
;      User processing
;      :
        RETI                     ;Returns from interrupt

CODE     ENDS
;-----Vector setting-----
VECT     CSEG      ABS=0FFH
        ORG        0FFCCH            ;Sets vector for interrupt #12 (0CH)
        DSL        WARI
        ORG        0FFDCH            ;Sets reset vector
        DSL        START
        DB         00H              ;Sets single-chip mode
VECT     ENDS
        END        START

```



# **CHAPTER 15**

---

# ***MULTI-PULSE GENERATOR***

**This chapter describes the specification and operation of the Multi-pulse Generator.**

- 15.1 Overview of Multi-pulse Generator
- 15.2 Block Diagram of Multi-pulse Generator
- 15.3 Multi-pulse Generator Pins
- 15.4 Registers of Multi-pulse Generator
- 15.5 Multi-pulse Generator Interrupts
- 15.6 Operation of Multi-pulse Generator
- 15.7 Usage Notes on the Multi-pulse Generator
- 15.8 Sample Programs for the Multi-pulse Generator

## 15.1 Overview of Multi-pulse Generator

The Multi-pulse Generator consists of a 16-bit PPG timer, a 16-bit reload timer and a waveform sequencer. By using the waveform sequencer, 16-bit PPG timer output signal can be directed to Multi-pulse Generator output (OPT5 to OPT0) according to the input signal of Multi-pulse Generator (SNI2 to SNI0). Meanwhile, the OPT5 to OPT0 output signal can be hardware terminated by DTTI1 input in case of emergency. The OPT5 to OPT0 output signals are synchronized with the PPG signal in order to eliminate the unwanted glitch.

**Note:**

In MB90465 series, 16-bit PPG timer and waveform sequencer are not present in Multi-pulse Generator, but 16-bit Reload Timer can be used individually.

For the detail information about 16-bit Reload Timer and 16-bit PPG Timer, please refer to Chapter 12, 16-bit Reload Timer and Chapter 13, 16-bit PPG Timer respectively.

### ■ Function of Waveform Sequencer

#### ● Output Signal Control

With waveform sequencer, it is possible to generate 16-bit PPG waveform output and DC chopper waveform output at the Multi-pulse Generator output (OPT5 to OPT0).

- When an effective edge of the input signal from Multi-pulse Generator position detect input (SNI2 to SNI0) or when the 16-bit reload timer is underflow or when the OPDBR0 register is written, one of Output Data Buffer Registers (OPDBRB to OPDBR0) will be loaded into the Output Data Register (OPDR).
- The Output Data Register (OPDR) determines the 16-bit PPG timer output to which OPT output (OPT5 to OPT0). By loading different Output Data Buffer Registers (OPDBRB to OPDBR0) into the Output Data Register (OPDR). Various combination of OPT outputs (OPT5 to OPT0) can be obtained.
- Therefore, the 16-bit PPG timer output can be presented/absented at Multi-pulse Generator output (OPT5 to OPT0) or switch the PPG timer output signal from one OPT output to another OPT output according to the sequence set in the Output Data Register (OPDR) and 12 Output Data Buffer Registers (OPDBRB to OPDBR0). Meanwhile, the 16-bit reload timer can insert a delay when switch OPT output.
- Table 15.1-1 shows the combination the data transfer from OPDBRB to OPDBR0 registers to OPDR register.

**Table 15.1-1 Data Transfer from OPDBRB to OPDBR0 Registers to OPDR Register**

Combination	Data transfer from OPDBRB to OPDBR0 to OPDR
1	Data transfer from OPDBR0 to OPDR after OPDBR0 is written by software.
2	Triggered by the 16-bit reload timer 0 underflow.
3	Triggered by the position detection input (SNI2 to SNI0).
4	Triggered by the 16-bit reload timer 0 underflow. The 16-bit timer is started by the position detection comparison circuit.
5	Triggered either by the 16-bit reload timer 0 underflow, or by the position detection input.

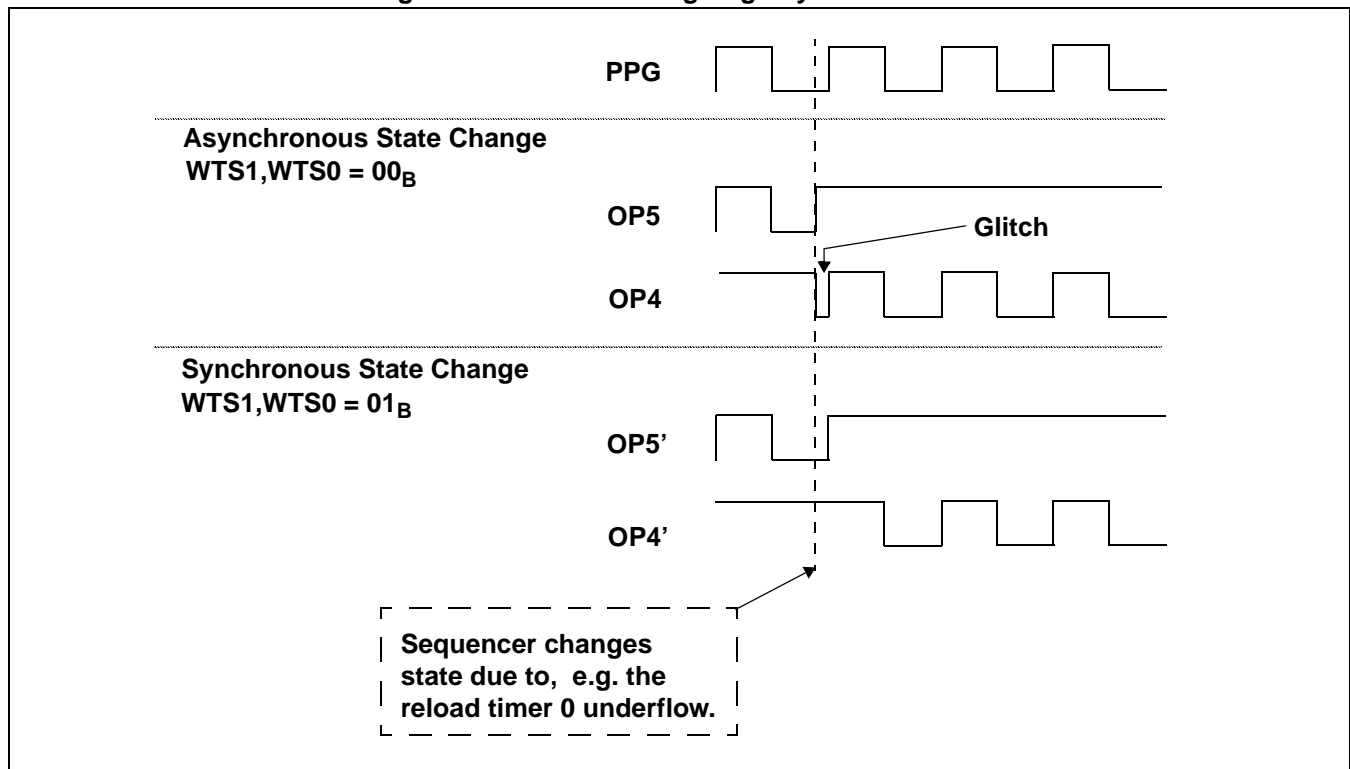
- In the waveform sequencer, there is a 16-bit timer that can be used to measure the speed of the motor and disable the OPT output in case of position detect missing.
- Forced stop control using DTTI1 pin input  
External pin control can be performed through clockless DTTI1 pin input even when oscillation is stopped. (The pin level can be set by each pin or software.) There is selectable noise filter for DTTI1 input. Table 15.1-2 shows the noise width for noise filter of DTTI1 pin.

**Table 15.1-2 Noise Width for Noise Filter**

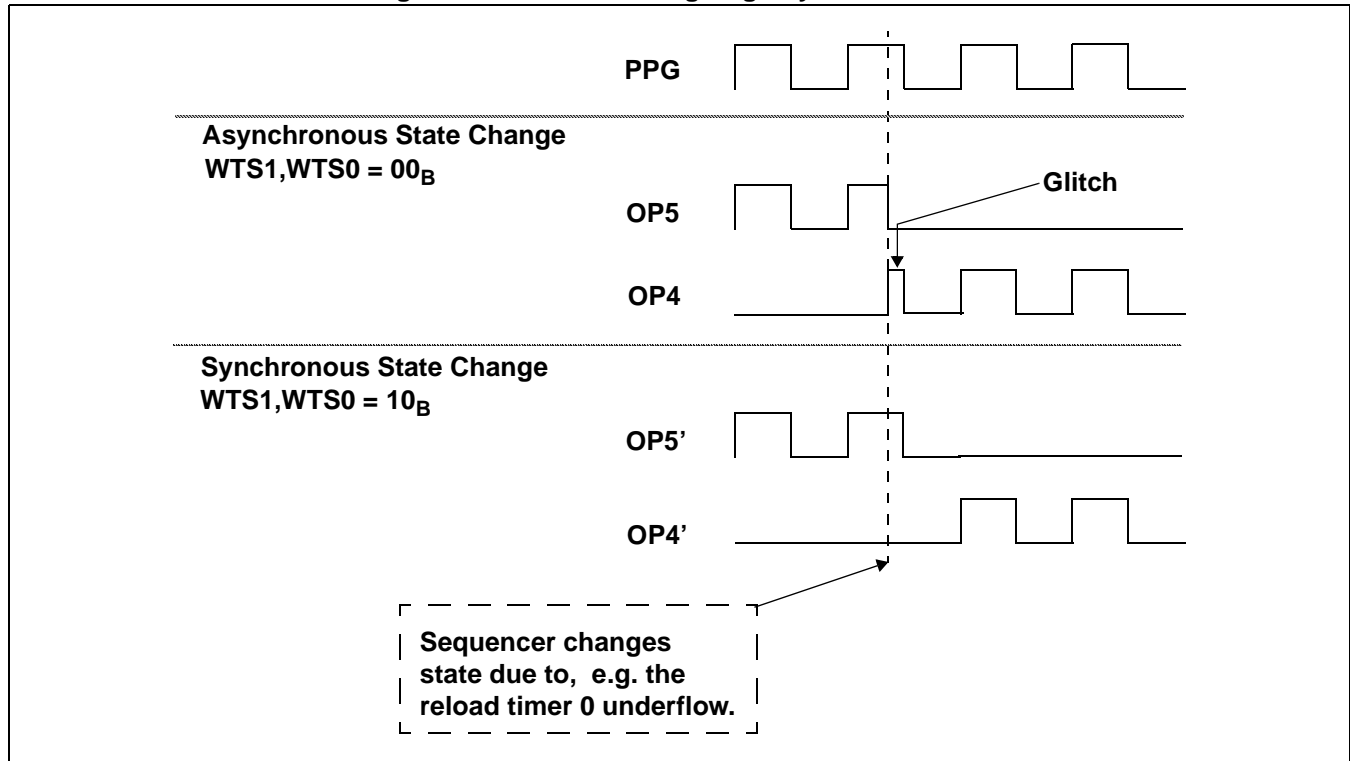
Selection	Noise width for DTTI1 and SNI2 to SNI0 pins
1	Cancel 4-cycle noise.
2	Cancel 8-cycle noise.
3	Cancel 16-cycle noise.
4	Cancel 32-cycle noise.

### ● PPG Synchronization for Output Signal

In order to avoid short pulse (or glitch) during sequencer state changes, the write timing (WTO) needs to be delayed and synchronized with the next coming edge of PPG output waveform. See Figure 15.1-1 and Figure 15.1-2 for details. This function can be enabled or disabled by software. WTS1 and WTS0 bits of the Input Control Register (IPCR) are used to disable this function and to select the polarity of the PPG edge to synchronize with.

**Figure 15.1-1 PPG Rising Edge Synchronization**



**Figure 15.1-2 PPG Falling Edge Synchronization**

Note:

Switch from one PPG synchronization mode to another PPG synchronization mode (e.g. from rising-edge synchronization to falling-edge synchronization or vice versa) is inhibited, no synchronization mode must be the transit for such switch.

### ● Input Position Detect Control

The input signal at the Multi-pulse Generator input pins (SNI2 to SNI0) is used to detect the rotor position of the DC motor. There is a Noise Filter for all SNI2 to SNI0 input and Table 15.1-2 shows the noise width for noise filter of SNI2 to SNI0 pins. The followings are conditions for the input position detect circuit:

- 3 edge selection for all SNI2 to SNI0; Rising edge, falling edge and both edges.
- Compare the levels of SNI2 to SNI0 inputs with RDA2 to RDA0 bits of Output Data Register (OPDR: RDA2 to RDA0).

After above condition met, the writing timing signal will be generated for the data transfer between OPDBR registers and OPDR register.

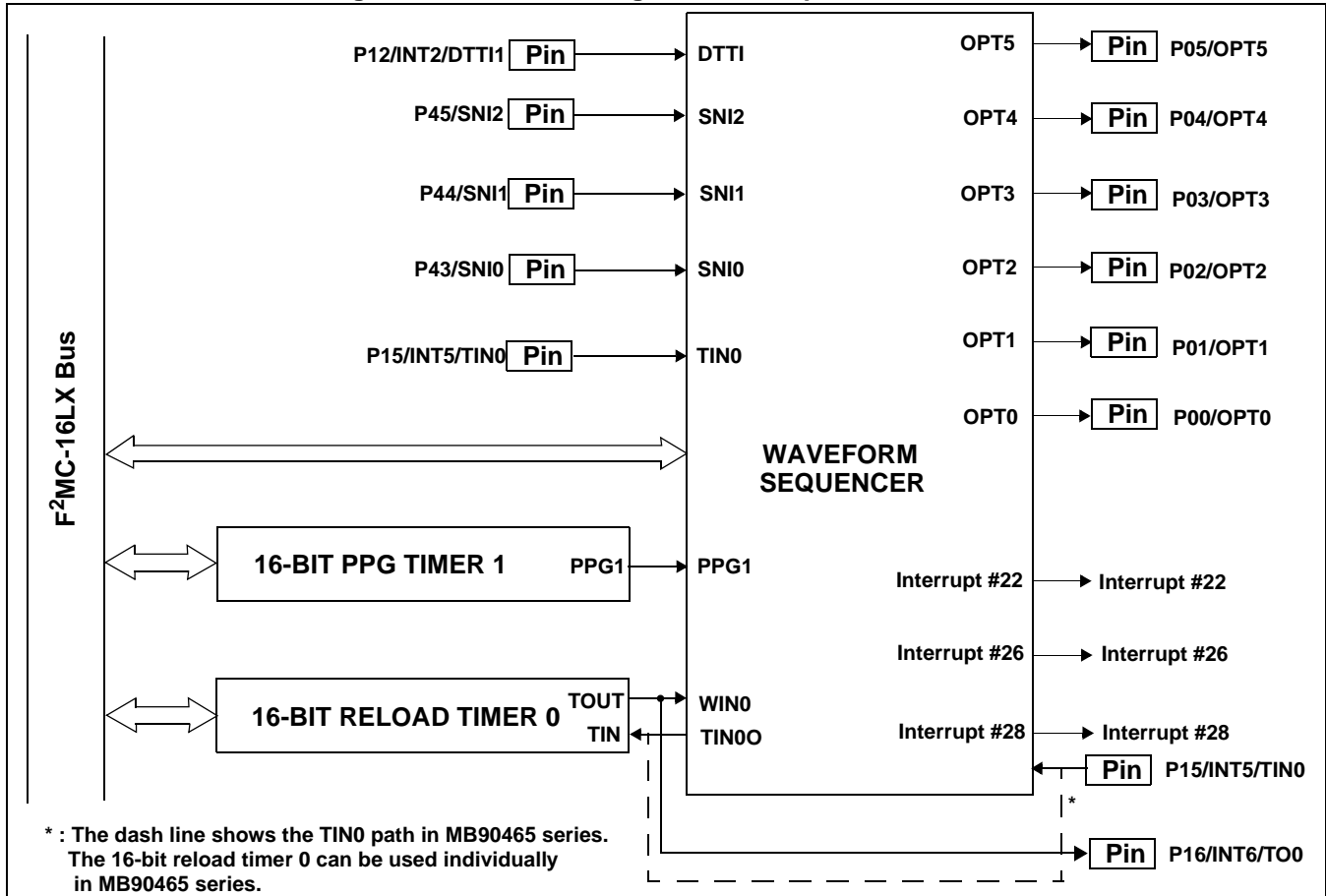
Furthermore, the edge detection for individual input (SNI2 to SNI0) can be disable/enable.

## 15.2 Block Diagram of Multi-pulse Generator

Figure 15.2-1 shows the block diagram of the Multi-pulse Generator and Figure 15.2-2 shows the block diagram of the Waveform Sequencer.

### ■ Block Diagram of Multi-pulse Generator

Figure 15.2-1 Block Diagram of Multi-pulse Generator



#### ● 16-bit PPG Timer 1

The 16-bit PPG Timer 1 is used to provide a PPG signal for Waveform Sequencer. The detail of 16-bit PPG Timer 1 is described in Chapter 13, 16-bit PPG Timer.

#### ● 16-bit Reload Timer 0

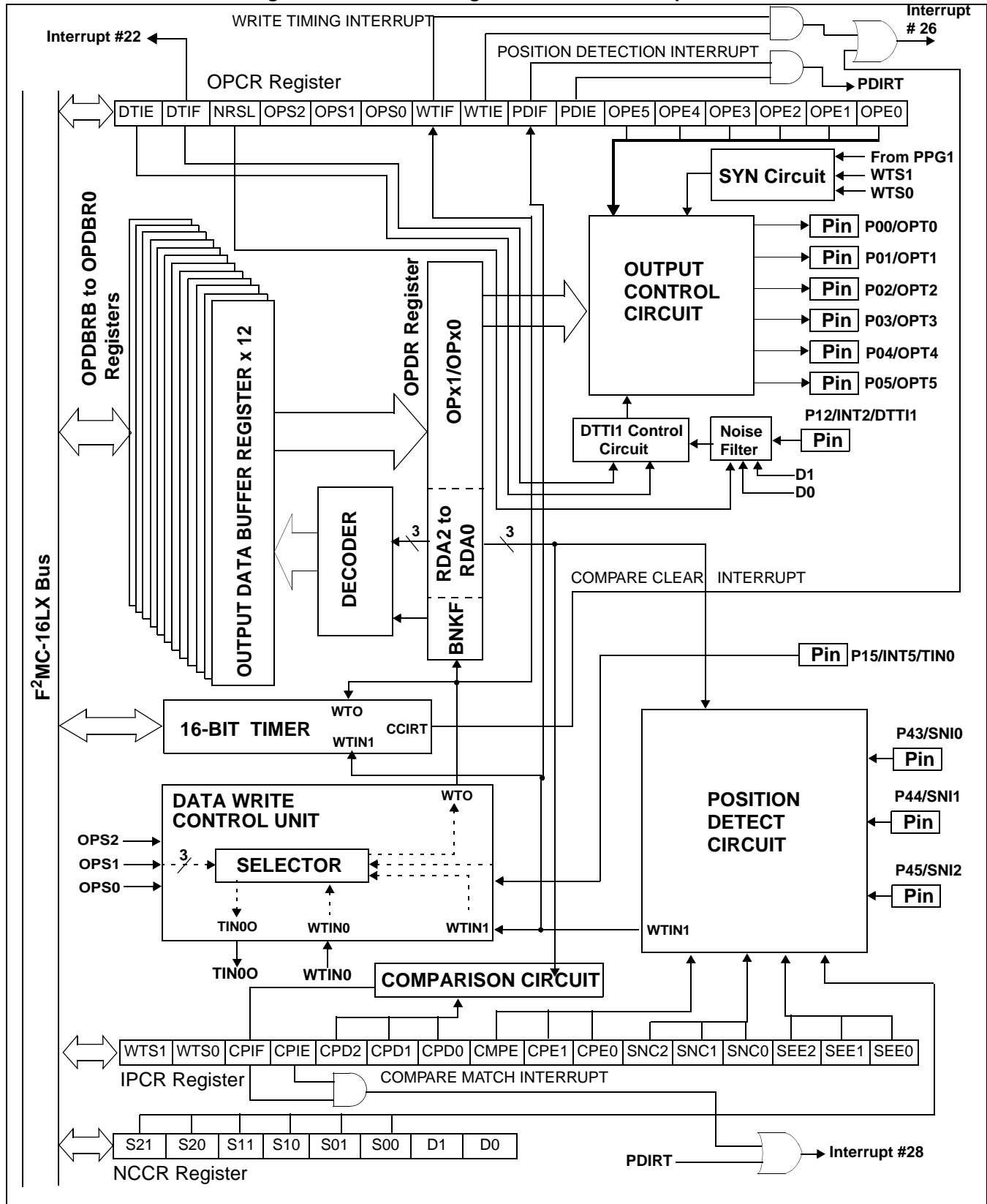
The 16-bit Reload Timer 0 is used to act as interval timer for Waveform Sequencer. The detail of 16-bit Reload Timer 0 is described in Chapter 12, 16-bit Reload Timer.

#### ● Waveform Sequencer

The Waveform Sequencer is the heart of Multi-pulse Generator, that can generate various waveforms. The block diagram is shown in Figure 15.2-2.

## ■ Block Diagram of Waveform Sequencer

Figure 15.2-2 Block Diagram of Waveform Sequencer



### ● 16-bit Timer

The 16-bit timer is used to act as an interval timer for motor speed checking and abnormal detection timer when control DC sensorless motor. The detail is shown in Figure 15.2-3.

### ● Comparison Circuit

The Comparison Circuit is used to compare the RDA2 to RDA0 bits of the Output Data Register (OPDR: RDA2 to RDA0) with the CPD2 to CPD0 bits of the Input Control Register (IPCR: CPD2 to CPD0) for motor direction change. A compare match interrupt is generated when a match is happened.

### ● Data Write Control Unit

The Data Write Control Unit is used to generate the write signal (WTO) for transferring data from the Output Data Buffer Register (OPDBR) to Output Data Register (OPDR). The detail is shown in Figure 15.2-4.

### ● Decoder

The Decoder is used to decode the bit15 to bit12 of Output Data Register (OPDR: BNKF, RDA2 to RDA0) to select which Output Data Buffer Register (OPDBRB to OPDBR0) is loaded into Output Data Register.

### ● DTTI1 Control

The DTTI1 Control is used to stop the Multi-pulse Generator output in case of emergency, that is triggered by level "0" of DTTI1 input.

### ● Noise Filter

The Noise Filter is used to filter out the noise of the input signal in which there are 4 kind of sampling clock for selection.

### ● Output Control Unit

The Output Control Unit is used to enable/disable PPG signal to the Multi-pulse Generator Outputs (OPT5 to OPT0).

### ● Position Detect Circuit

The Position Detect Circuit is used to detect the edge/level of the position input (SNI2 to SNI0). The detail is shown in Figure 15.2-5.

### ● SYN Circuit

The SYN Circuit is used to synchronize the OPT5 to OPT0 outputs with the PPG signal.

### ● Noise Cancellation Control Register (NCCR)

The Noise Cancellation Control Register (NCCR) is used to select one of four sampling clock for the Noise Filter.

### ● Output Control Register (OPCR)

The Output Control Register (OPCR) is a register which enables the write timing interrupt and flag, position detect interrupt and flag, sets the data transfer method, and sets the control of the OPT5 to OPT0 and DTTI1 pins.

### ● Output Data Buffer Register (OPDBRB to OPDBR0)

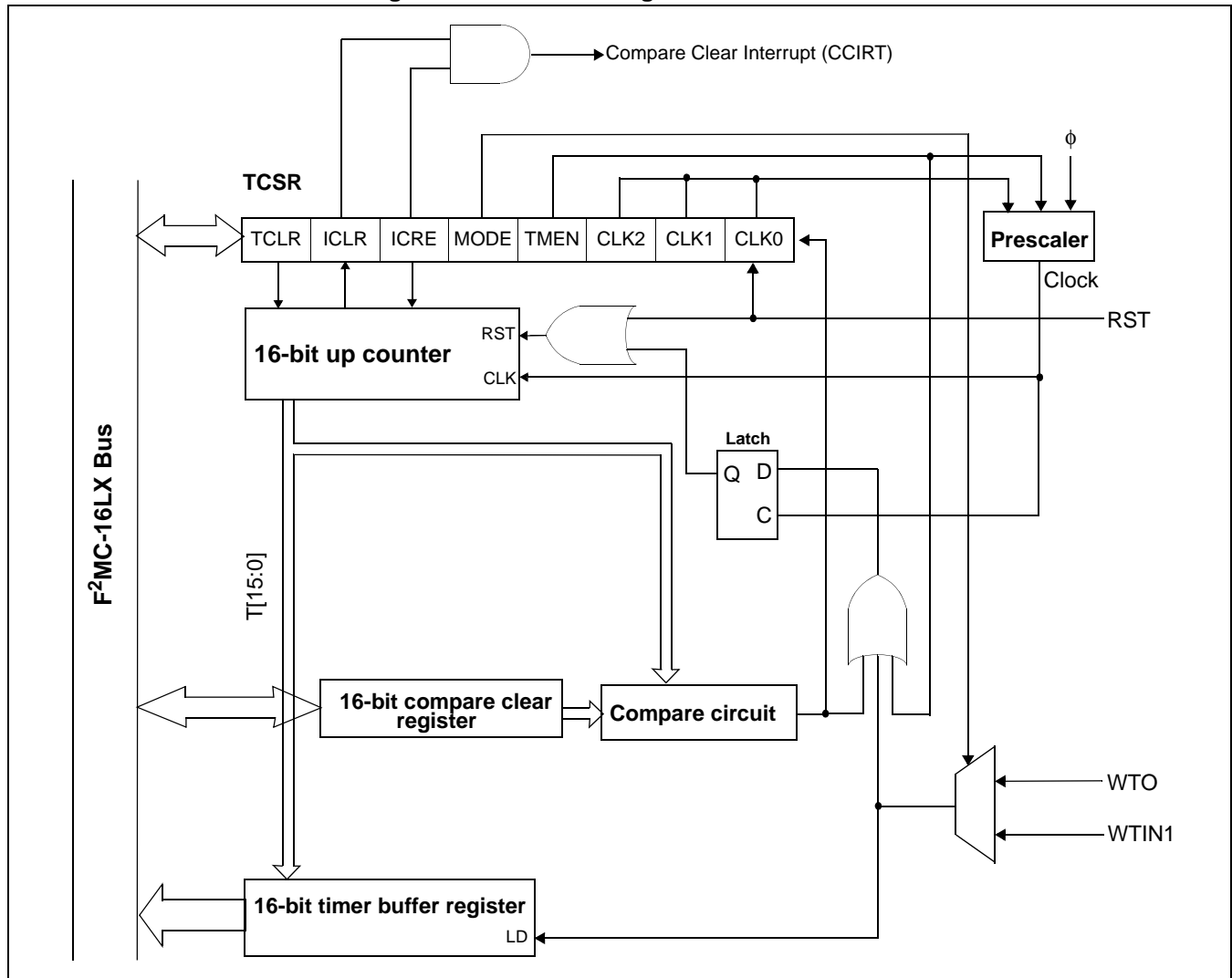
The Output Data Buffer Register is composed of twelve registers (OPDBRB to OPDBR0). The value of the OPDBRx register specified by the BNKF, RDA2 to RDA0 bits is loaded into the OPDR register at the rising edge of the write signal generated by the Data Write Control Unit.

### ● Output Data Register (OPDR)

The Output Data Register (OPDR) is used to store the output data to the OPT5 to OPT0 pins.

## ■ Block Diagram of 16-bit Timer

Figure 15.2-3 Block Diagram of 16-bit Timer



### ● 16-bit Up Counter

The 16-bit Up Counter will be cleared when the match is happened between the count value and the Compare Clear register.

### ● Compare Circuit

The Compare Circuit is used to compare the count value of the 16-bit Up Counter and the Compare Clear register.

### ● Compare Clear Register (CPCR)

The Compare Clear Register (CPCR) is used to store the 16-bit value which is used to compare the value of the 16-bit Up Counter.

### ● Timer Buffer Register (TMBR)

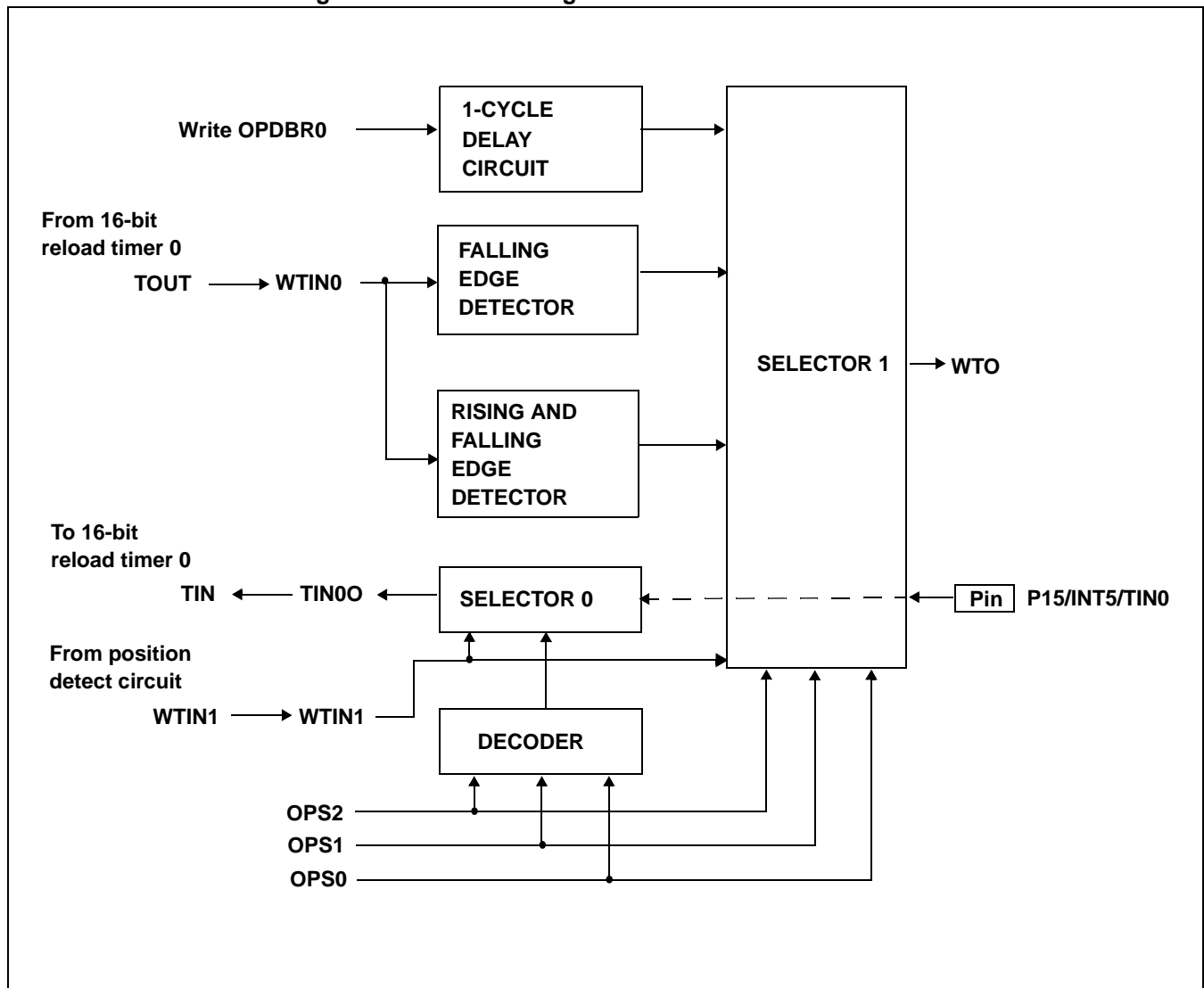
The Timer Buffer Register (TMBR) is used to store the value of the 16-bit Up Counter when a write timing interrupt or position detect interrupt occurs.

### ● Timer Control Register (TCSR)

The Timer Control Status Register (TCSR) is used to control the operation of the 16-bit timer such as the clock frequency, enable/disable the interrupt.

## ■ Block Diagram of Data Write Control Unit

Figure 15.2-4 Block Diagram of Data Write Control Unit



### ● 1-Cycle Delay Circuit

The 1-Cycle Delay Circuit is used to delay one CPU clock cycle of the trigger signal when the Output Data Buffer Register 0 (OPDBR0) is written.

### ● Selector 0

The Selector 0 is used to select from either WTIN1 of the Position Detect Circuit or external pin (P15/INT5/TIN0) to enable the count of the 16-bit Reload Timer 0.

### ● Selector 1

The Selector 1 is used to select from among Write OPDBR or TOUT of 16-bit Reload timer 0 or WTIN1 of Position Detect Circuit to generate the Write Timing signal (WTO).

### ● Falling Edge Detector

The Falling Edge Detector is used to detect the falling edge of the 16-bit Reload Timer 0 output (TOUT).

### ● Rising and Falling Edge Detector

The Rising and Falling Edge Detector is used to detect the rising and falling edge of the 16-bit Reload Timer 0 output (TOUT).

When timer underflow trigger is used in following modes, the WTIN0 signal is generated by the trigger edge selected by OPS2 to OPS0 bits:

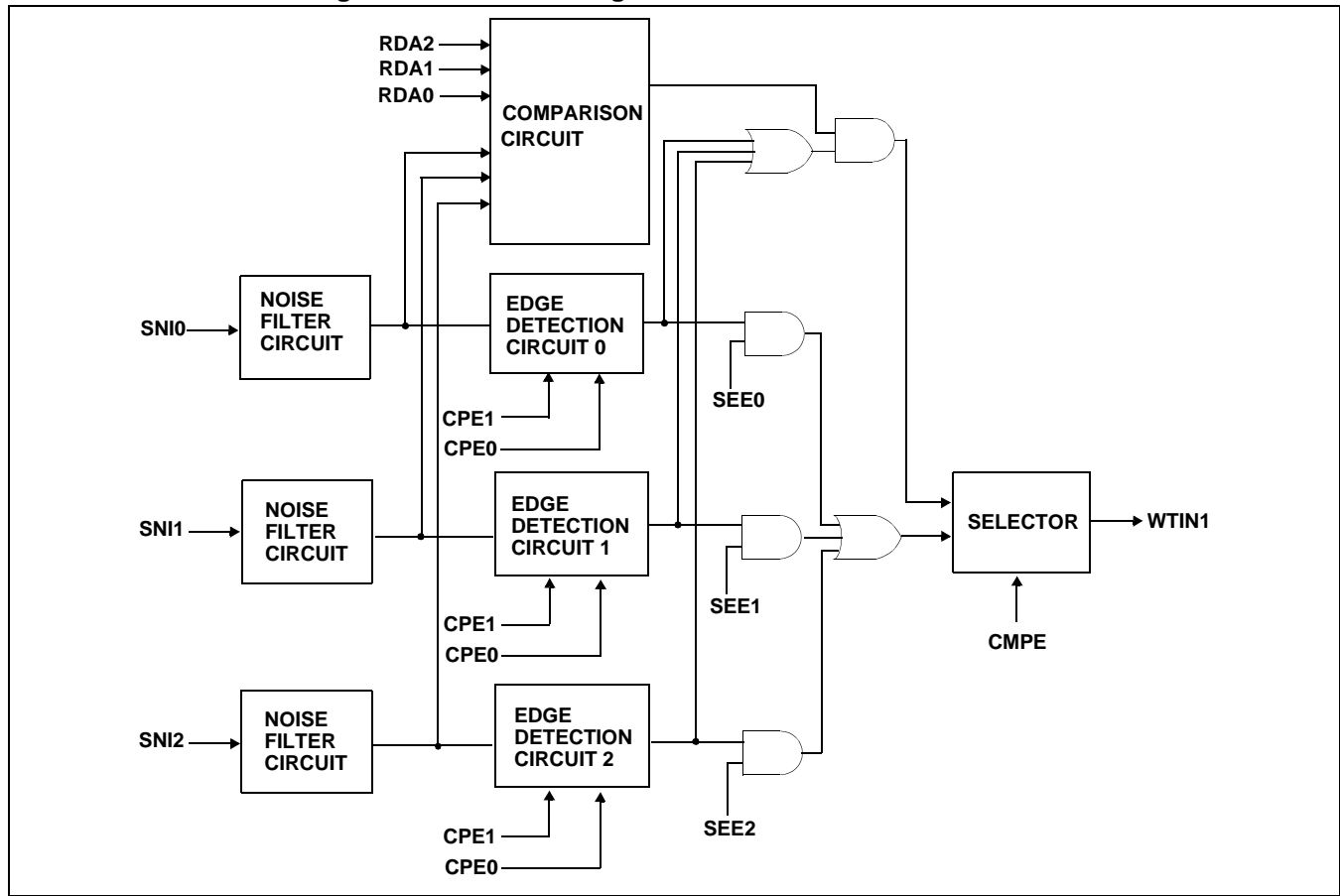
**Table 15.2-1 TOUT Trigger Edge Selection for WTIN0**

OPS2	OPS1	OPS0	TOUT Trigger Edge for WTIN0
0	0	0	-
0	1	0	Rise and Fall
0	1	0	-
0	1	1	Fall
1	0	0	Rise and Fall
1	0	1	Rise and Fall
1	1	0	-
1	1	1	Fall



## ■ Block Diagram of Position Detection Circuit

Figure 15.2-5 Block Diagram of Position Detection Circuit



### ● Comparison Circuit

The Comparison Circuit is used to compare the level of the position detection input (SNI2 to SNI0) with RDA2 to RDA0 bits of the Output Data Register (OPDR: RDA2 to RDA0). If the selector is selected, a data write time output signal is generated when a match is detected.

### ● Edge Detect Circuit 0, 1, 2

Edge Detect Circuit 0, 1 and 2 are identical.

The Edge Detect Circuit is used to compare the edge of the position input (SNI2 to SNI0) with 3 different kind of edge setting. If the selector is selected, a data write time output signal is generated when an effective edge is detected at the one of SNI2 to SNI0 inputs.

### ● Noise Filter

The Noise Filter is used to filter out the noise of the input signal in which there are 4 kind of sampling clock for selection.

### ● Selector

The Selector is used to select from either Edge Detect Circuit or Comparison Circuit to generate data write time output signal to the Data Write Control Unit.

## 15.3 Multi-pulse Generator Pins

---

**This section describes the pins of the Multi-pulse Generator and provides a pin block diagram.**

---

### ■ Pins of Multi-pulse Generator

Multi-pulse Generator uses P00/OPT0 to P05/OPT5, P43/SNI0 to P45/SNI2, P12/INT2/DTTI1 and P15/INT5/TIN0.

#### ● P00/OPT0 to P05/OPT5 Pins

P00/OPT0 to P05/OPT5 pins can function either as a general-purpose I/O port (P00 to P05) or as waveform output for Multi-pulse Generator.

Enabling waveform output bit (OPCLR: OPE5 to OPE0 = 111111<sub>B</sub>) automatically sets the P00/OPT0 to P05/OPT5 pin as an output pin, regardless of the port data direction register (DDR0: bit5 to bit0) value, and sets the pin to function as the OPT5 to OPT0 pin.

#### ● P43/SNI0 to P45/SNI2 Pins

P43/SNI0 to P45/SNI2 pins can function either as a general-purpose I/O port (P43 to P45) or as the position detect input for Multi-pulse Generator.

Set P43/SNI0 to P45/SNI2 pins as an input port in the data direction register (DDR4: bit5 to bit3 = 000<sub>B</sub>) when using as SNI2 to SNI0 pins.

#### ● P12/INT2/DTTI1 Pin

P12/INT2/DTTI1 pin can function either as a general-purpose I/O port (P12) or external interrupt INT2, or as the DTTI1 input for Multi-pulse Generator.

Set P12/INT2/DTTI1 pin as an input port in the data direction register (DDR1: bit1 = 0) when using as DTTI1 pins.

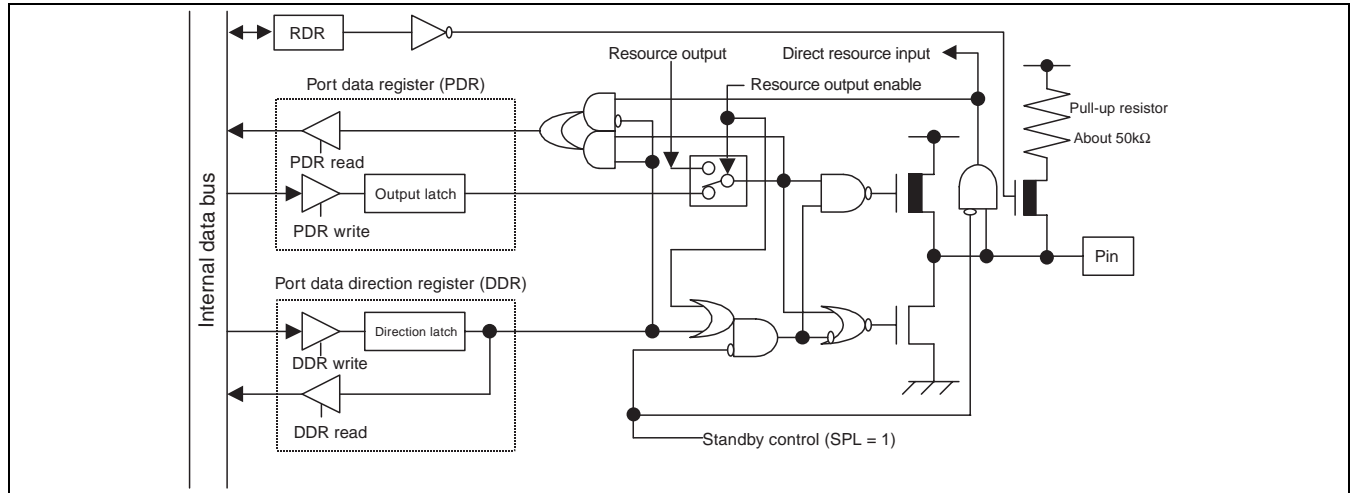
#### ● P15/INT5/TIN0 Pin

P15/INT5/TIN0 pin can function either as a general-purpose I/O port (P15) external interrupt INT5, or as the input of 16-bit Reload Timer 0 for Multi-pulse Generator.

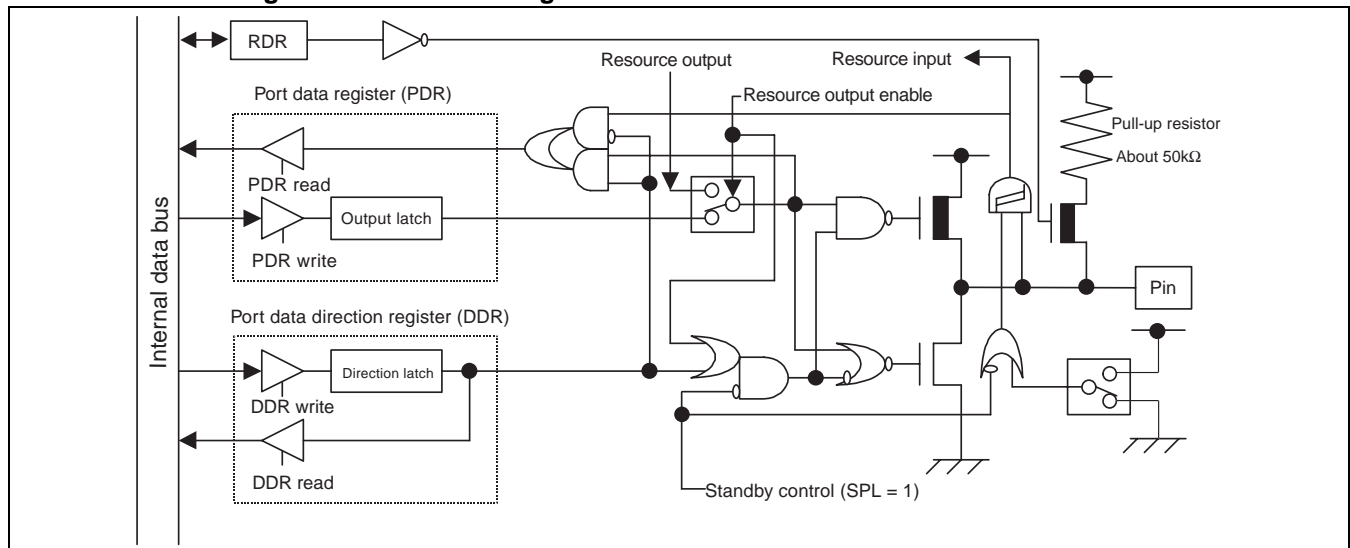
Set P15/INT5/TIN0 pin as an input port in the data direction register (DDR1: bit5 = 0) when using as TIN0 pins.

## ■ Block Diagram of Multi-pulse Generator Pins

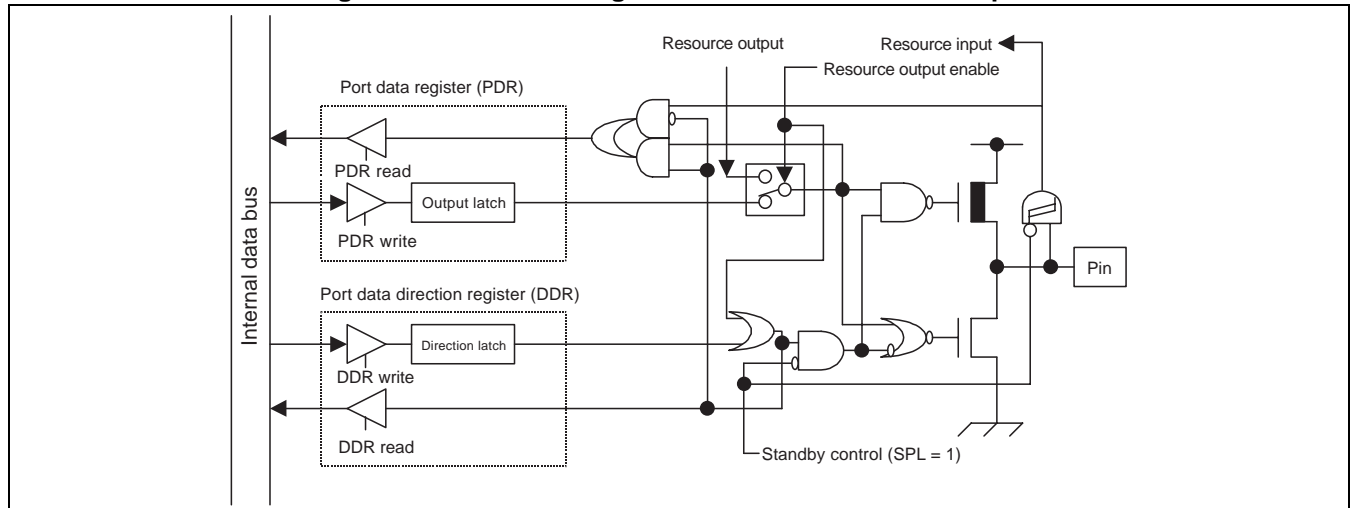
**Figure 15.3-1 Block Diagram of P00/OPT0 to P05/OPT5 Pins**



**Figure 15.3-2 Block Diagram of P12/INT2/DTT1 to P15/INT5/TIN0 Pins**



**Figure 15.3-3 Block diagram of P43/SNI0 to P45/SNI2 pins**



## 15.4 Registers of Multi-pulse Generator

This section describes the registers of the Multi-pulse Generator.

### ■ Registers of Multi-pulse Generator

Figure 15.4-1 Registers of Multi-pulse Generator

Output Control Register (Upper)

	bit 15	14	13	12	11	10	9	8	
Address: 00008B <sub>H</sub>	DTIE	DTIF	NRSL	OPS2	OPS1	OPS0	WTIF	WTIE	OPCUR
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	

Output Control Register (Lower)

	bit 7	6	5	4	3	2	1	0	
Address: 00008A <sub>H</sub>	PDIF	PDIE	OPE5	OPE4	OPE3	OPE2	OPE1	OPE0	OPCLR
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	

Output Data Register (Upper)

	bit 15	14	13	12	11	10	9	8	
Address: 003FF9 <sub>H</sub>	BNKF	RDA2	RDA1	RDA0	OP51	OP50	OP41	OP40	OPDR
Read/Write ⇨	R	R	R	R	R	R	R	R	
Initial Value ⇨	0	0	0	0	X	X	X	X	

Output Data Register (Lower)

	bit 7	6	5	4	3	2	1	0	
Address: 003FF8 <sub>H</sub>	OP31	OP30	OP21	OP20	OP11	OP10	OP01	OP00	OPDR
Read/Write ⇨	R	R	R	R	R	R	R	R	
Initial Value ⇨	X	X	X	X	X	X	X	X	

(Continued)

(Continued)

## Output Data Buffer Registers (Upper)

	bit	15	14	13	12	11	10	9	8	
Addresses: 003FF7 <sub>H</sub> to E1 <sub>H</sub> (Odd Addresses)		BNKF	RDA2	RDA1	RDA0	OP51	OP50	OP41	OP40	OPDBRB to OPDBR0
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	

## Output Data Buffer Registers (Lower)

	bit	7	6	5	4	3	2	1	0	
Addresses: 003FF6 <sub>H</sub> to E0 <sub>H</sub> (Even Addresses)		OP31	OP30	OP21	OP20	OP11	OP10	OP01	OP00	OPDBRB to OPDBR0
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	

## Input Control Register (Upper)

	bit	15	14	13	12	11	10	9	8	
Address: 00008D <sub>H</sub>		WTS1	WTS0	CPIF	CPIE	CPD2	CPD1	CPD0	CMPE	IPCUR
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	

## Input Control Register (Lower)

	bit	7	6	5	4	3	2	1	0	
Address: 00008C <sub>H</sub>		CPE1	CPE0	SNC2	SNC1	SNC0	SEE2	SEE1	SEE0	IPCLR
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	

(Continued)

(Continued)

## Compare Clear Register (Upper)

	bit	15	14	13	12	11	10	9	8	
Address: 003FFB <sub>H</sub>		CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	CPCR

Read/Write ⇨ R/W R/W R/W R/W R/W R/W R/W R/W

Initial Value ⇨ X X X X X X X X

## Compare Clear Register (Lower)

	bit	7	6	5	4	3	2	1	0	
Address: 003FFA <sub>H</sub>		CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	CPCR

Read/Write ⇨ R/W R/W R/W R/W R/W R/W R/W R/W

Initial Value ⇨ X X X X X X X X

## Timer Buffer Register (Upper)

	bit	15	14	13	12	11	10	9	8	
Address: 003FFD <sub>H</sub>		T15	T14	T13	T12	T11	T10	T09	T08	TMBR

Read/Write ⇨ R R R R R R R R

Initial Value ⇨ 0 0 0 0 0 0 0 0

## Timer Buffer Register (Lower)

	bit	7	6	5	4	3	2	1	0	
Address: 003FFC <sub>H</sub>		T07	T06	T05	T04	T03	T02	T01	T00	TMBR

Read/Write ⇨ R R R R R R R R

Initial Value ⇨ 0 0 0 0 0 0 0 0

## Timer Control Status Register

	bit	15	14	13	12	11	10	9	8	
Address: 00008F <sub>H</sub>		S21	S20	S11	S10	S01	S00	D1	D0	NCCR

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

## Noise Cancellation Control Register

	bit	7	6	5	4	3	2	1	0	
Address: 00008E <sub>H</sub>		TCLR	MODE	ICLR	ICRE	TMEN	CLK2	CLK1	CLK0	TCSR

Read/Write ⇨ R/W R/W R/W R/W R/W R/W R/W R/W

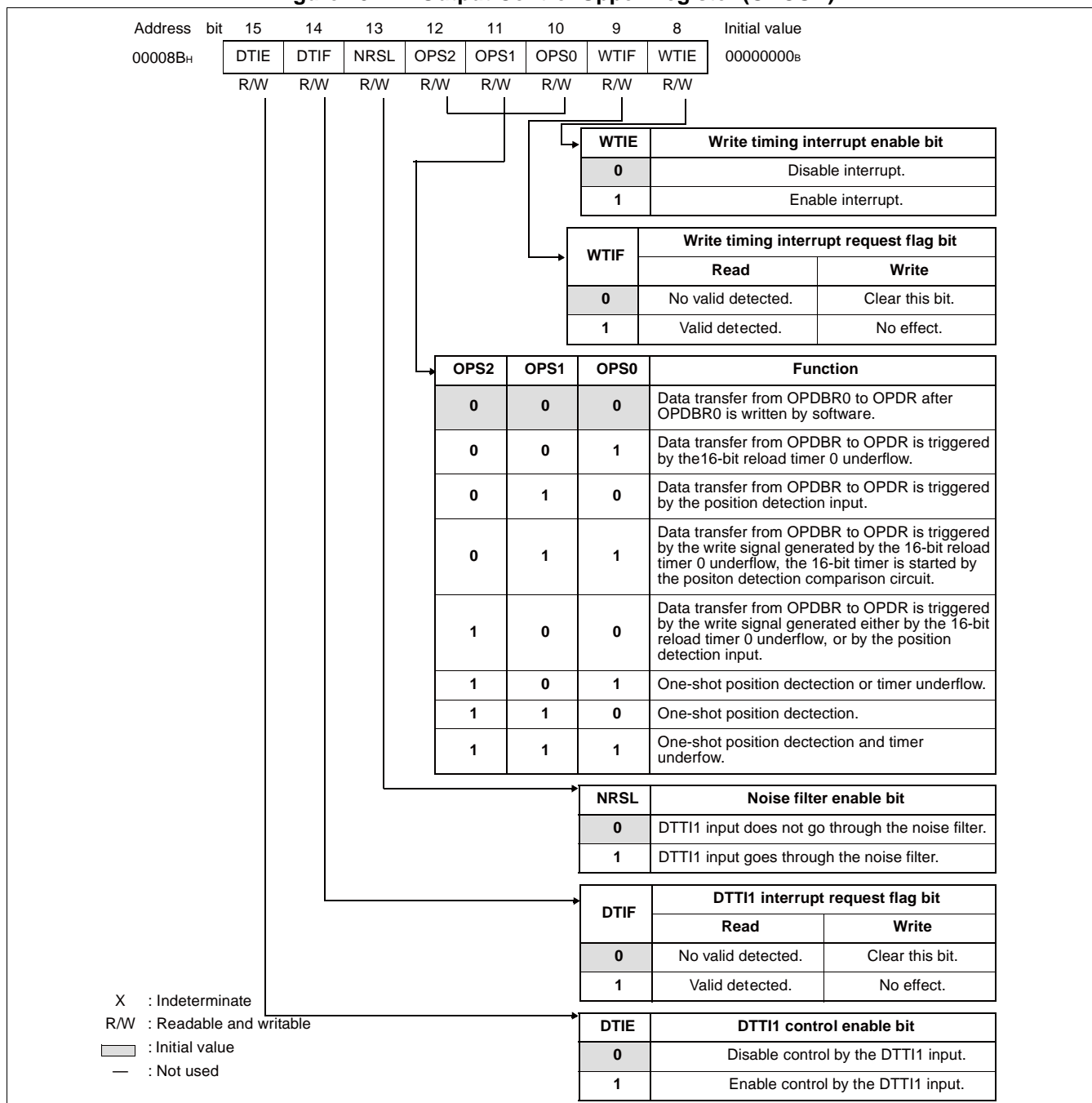
Initial Value ⇨ 0 0 0 0 0 0 0 0

## 15.4.1 Output Control Register (OPCR)

The Output Control Register (OPCR) is a register which enables the write timing interrupt and flag, position detect interrupt and flag, sets the data transfer method, and sets the control of the OPT5 to OPT0 and DTTI1 pins.

### ■ Output Control Upper Register (OPCUR)

Figure 15.4-2 Output Control Upper Register (OPCUR)



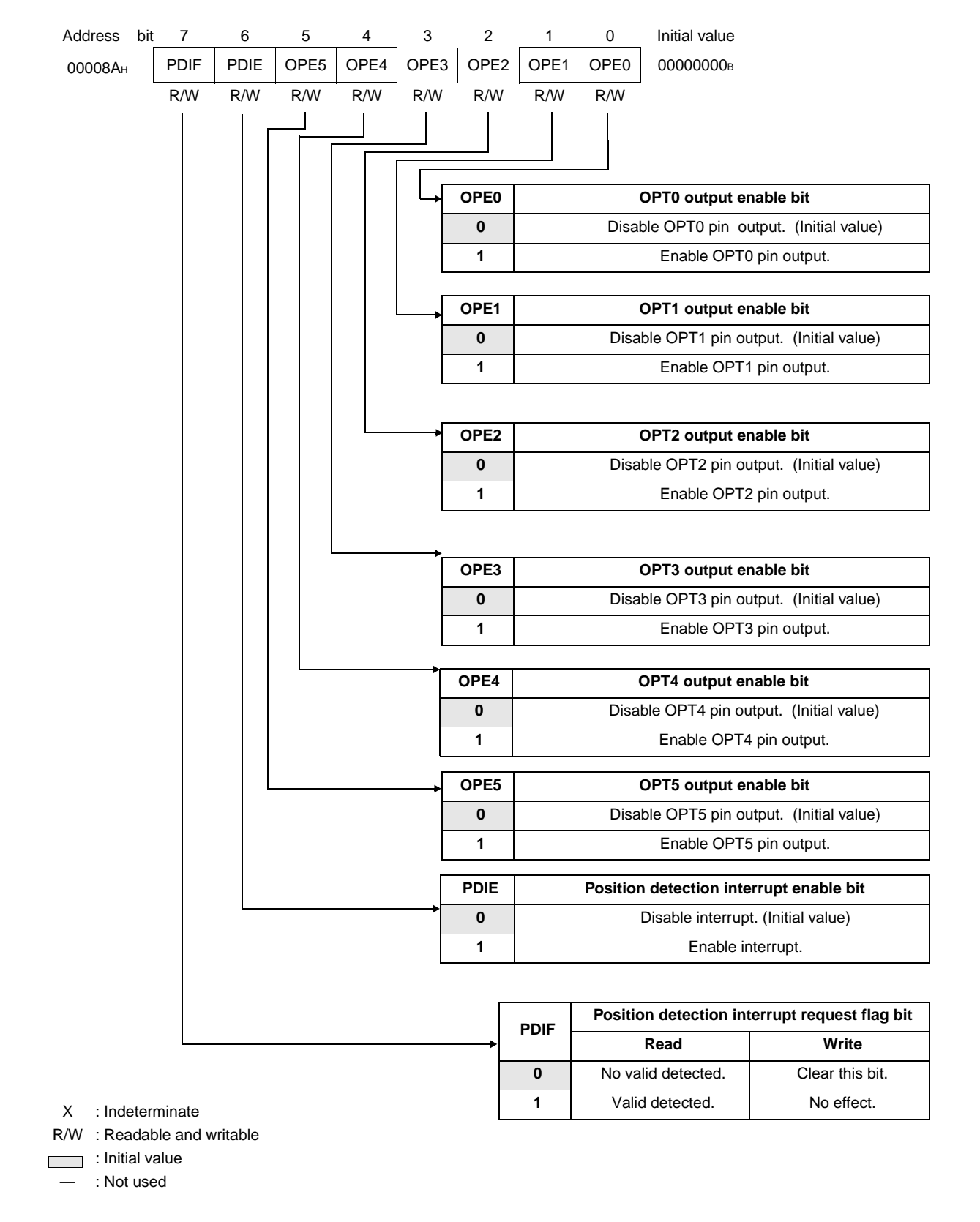
**Table 15.4-1 Output Control Upper Register (OPCUR) Bits**

Bit name		Function
bit15	DTIE: DTTI1 control enable bit	<ul style="list-style-type: none"> <li>• DTTI1 pin input enable bit.</li> <li>• This bit is used to enable the DTTI1 pin to control the output levels of the OPT5 to OPT0 pins. The software can set the inactive level for each OPTx pin in PDRx of PORTx.</li> </ul>
bit14	DTIF: DTTI1 interrupt flag bit	<ul style="list-style-type: none"> <li>• DTTI1 interrupt request flag.</li> <li>• It is an interrupt request flag of the DTTI1 input, which is set whenever a falling edge of DTTI1 is detected and the DTTI1 control enable bit is set to “1”.</li> <li>• When this bit is set to “1”, the interrupt is generated. This bit is cleared by writing “0”. Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit13	NRSL: Noise filter enable bit	<ul style="list-style-type: none"> <li>• This bit is used to select the noise cancellation function when DTTI1 pin input is enabled.</li> <li>• The noise cancellation circuit starts the internal n-bit counter when an active level is input (the value of n can be 2, 3, 4, 5, which depends on the setting of D1,D0 bits in the Noise Cancellation Register). If the active level is held until the counter overflows, the circuit accepts input from the DTTI1 pin. Therefore, the pulse width of noise that can be cancelled is about 2<sup>n</sup> machine cycles.</li> </ul> <p>(Note) When the noise cancellation circuit is enable, the input becomes invalid in a mode such as STOP mode in which the internal clock is stopped.</p>
bit12 to bit10	OPS2 to OPS0: Data transfer method selection bits	<ul style="list-style-type: none"> <li>• OPTx pin output timing control selection bits.</li> <li>• These bits are used to select the OPDR register write timing control operation mode. Data is transferred from the Output Data Buffer Register to the Output Data Register at the write timing controlled by the selected operation mode.</li> </ul>
bit9	WTIF: Write timing interrupt flag bit	<ul style="list-style-type: none"> <li>• Write timing interrupt request flag.</li> <li>• It is an interrupt request flag of the output timing switch, which is set by the write signal. Data in the OPDBRx register which specified by the BNKF, RDA2 to RDA0 bits in Output Data Register (OPDR) is transferred to the OPDR at the rising edge of the write signal and the WTIF bit is set to “1”.</li> <li>• When this bit is set to “1”, the interrupt is generated if the write timing interrupt enable bit (WTIE) is also set to “1”. This bit is cleared by writing “0”. Writing “1” has no effect.</li> <li>• In read-modify-write operation, “1” is always read.</li> </ul>
bit8	WTIE: Write timing interrupt enable bit	<ul style="list-style-type: none"> <li>• Write timing interrupt enable bit.</li> <li>• When this bit is set to “1”, the interrupt is generated if write timing interrupt request flag (WTIF) is also set to “1”.</li> </ul>



■ Output Control Lower Register (OPCLR)

Figure 15.4-3 Output Control Lower Register (OPCLR)



**Table 15.4-2 Output Control Lower Register (OPCLR) Bits**

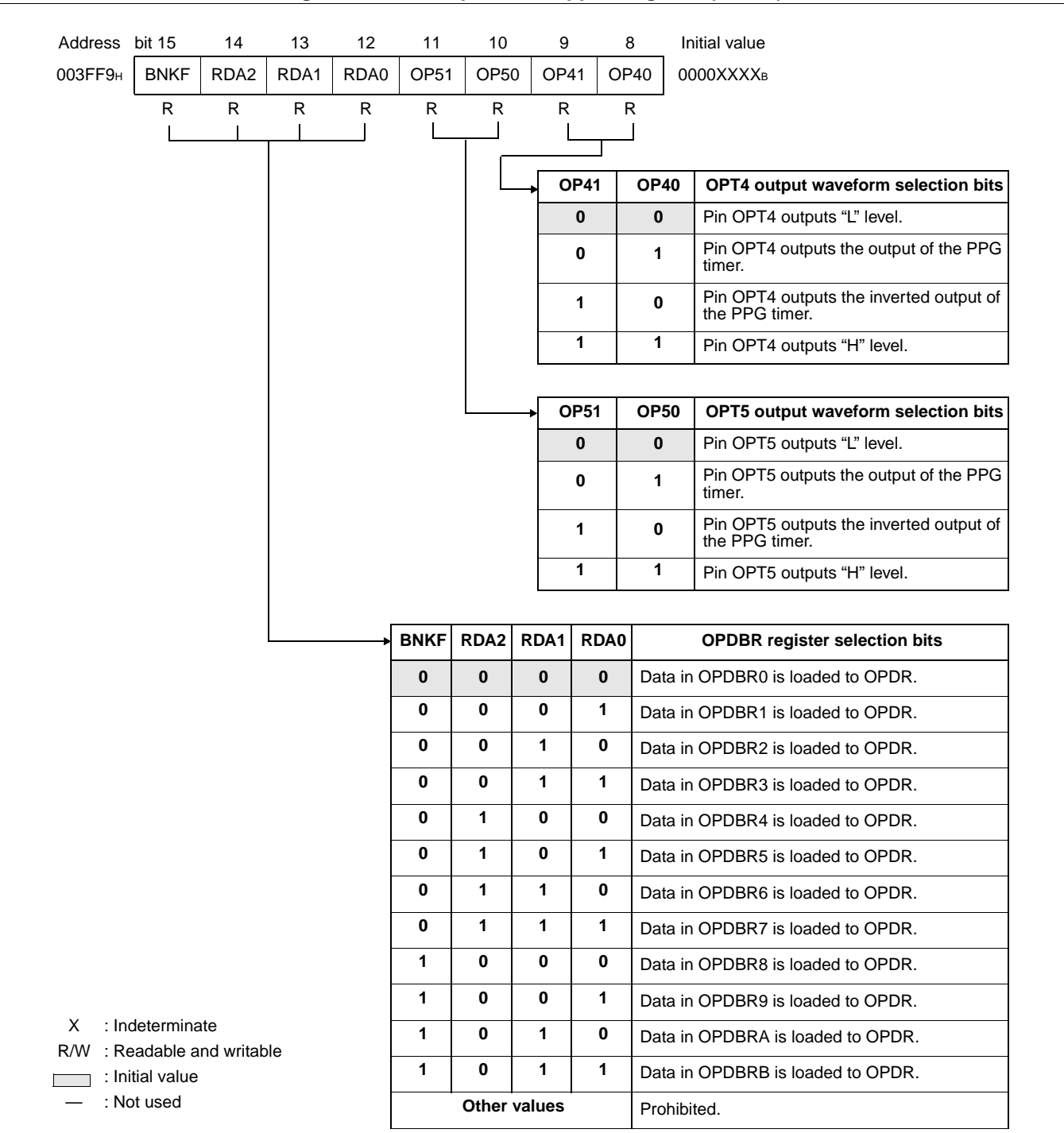
Bit name		Function
bit7	PDIF: Position detect interrupt flag bit	<ul style="list-style-type: none"> <li>Position detection interrupt request flag.</li> <li>It is an interrupt request flag for the position detection. When CMPE is set to "0" and the SNI2 to SNI0 bits are compared and matched with the RDA2 to RDA0 bit, or when CMPE is set to "1" and any effective edge is detected at SNI2 to SNI0 pins, this bit is set to "1".</li> <li>When this bit is set to "1", the interrupt is generated if the position detection interrupt enable bit (PDIE) is also set to "1". This bit is cleared by writing "0". Writing "1" has no effect.</li> <li>In read-modify-write operation, "1" is always read.</li> </ul>
bit6	PDIE: Position detect interrupt enable bit	<ul style="list-style-type: none"> <li>Position detection interrupt enable bit.</li> <li>When this bit is set to "1", the interrupt is generated if position detection interrupt request flag (PDIF) is also set to "1".</li> </ul>
bit5 to bit0	OPE5 to OPE0: OPT5 to OPE0 output enable bits	<ul style="list-style-type: none"> <li>Output enable bits of OPT5 to OPE0 pins.</li> <li>When these bits are set, the outputs to the OPT5 to OPE0 pins are enable.</li> </ul>

# 15.4.2 Output Data Register (OPDR)

This is a register which stores the output data to the OPT5 to OPT0 pins.

## ■ Output Data Upper Register (OPDR)

Figure 15.4-4 Output Data Upper Register (OPDR)

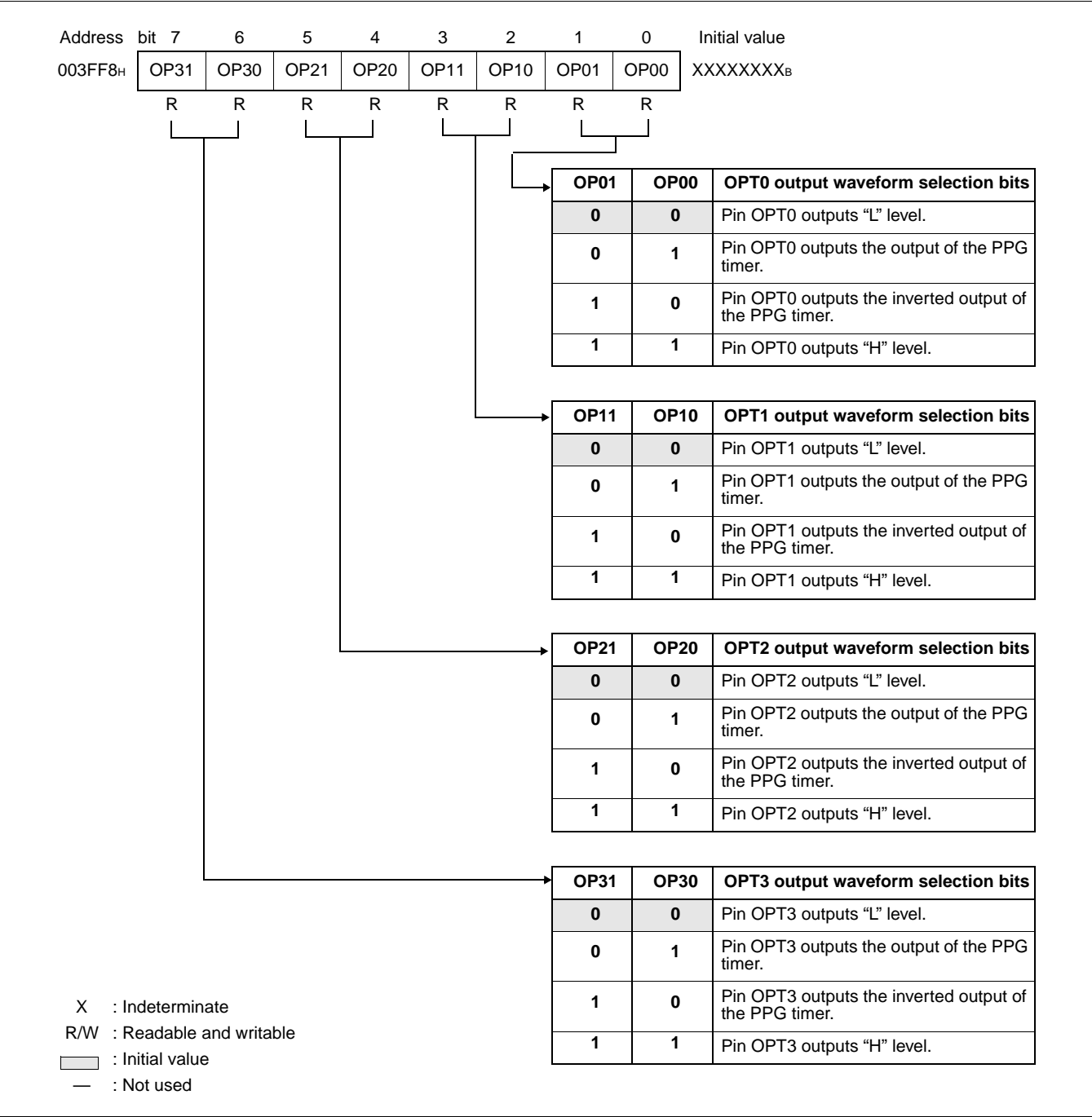


**Table 15.4-3 Output Data Upper Register (OPDR) Bits**

Bit name		Function
bit15 to bit12	BNKF, RDA2 to RDA0: OPDBR register selection bits	<ul style="list-style-type: none"> <li>These bits indicate the addresses of the OPDBR registers and decide which Output Data Buffer Register value is loaded into the OPDR register.</li> </ul>
bit11, bit10	OP51, OP50: OPT5 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT5 pin.</li> </ul>
bit9, bit8	OP41, OP40: OPT4 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT4 pin.</li> </ul>

■ Output Data Lower Register (OPDR)

Figure 15.4-5 Output Data Lower Register (OPDR)



**Table 15.4-4 Output Data Lower Register (OPDR) Bits**

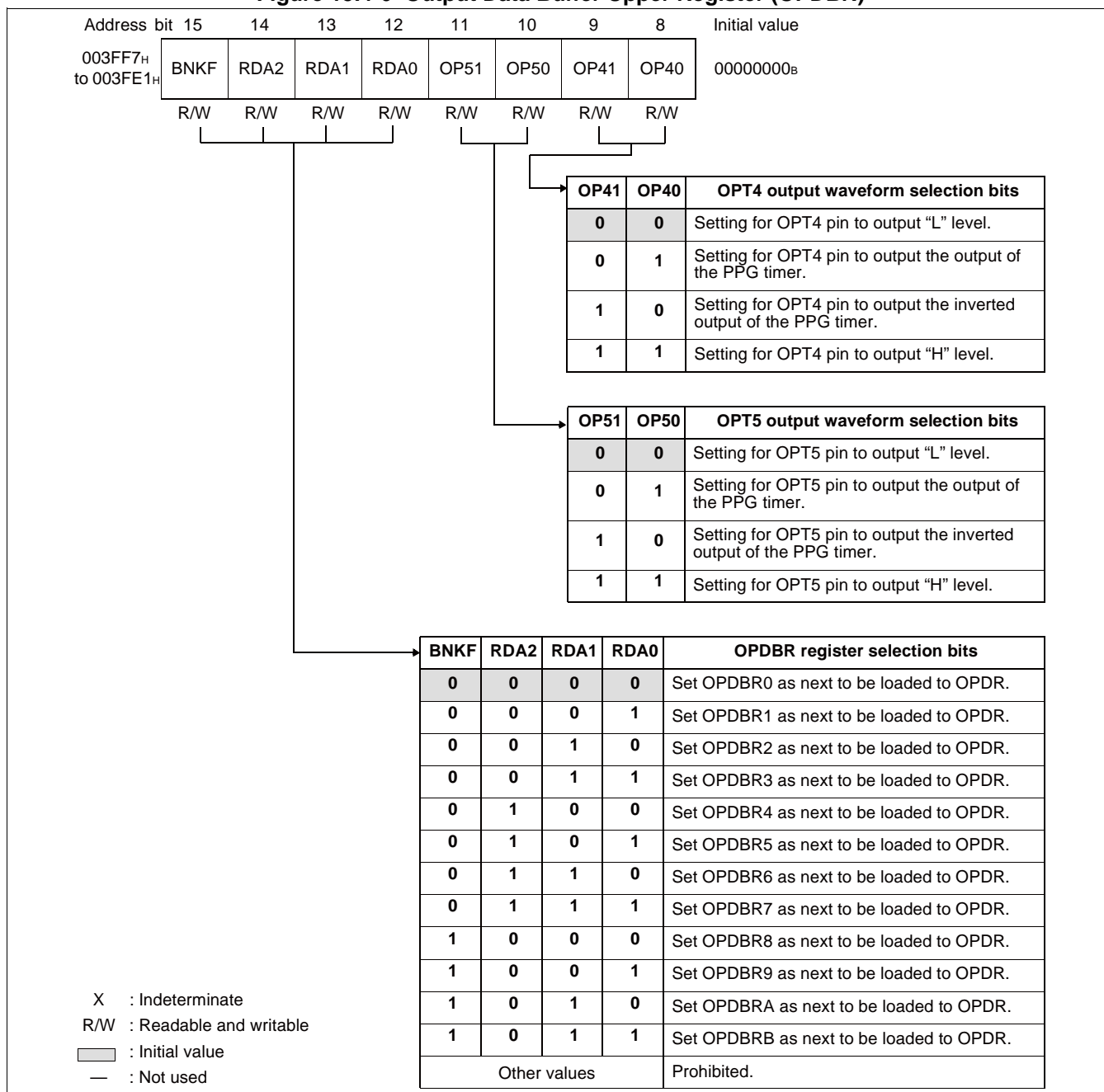
Bit name		Function
bit7, bit6	OP31, OP30: OPT3 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT3 pin.</li> </ul>
bit5, bit4	OP21, OP20: OPT2 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT2 pin.</li> </ul>
bit3, bit2	OP11, OP10: OPT1 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT1 pin.</li> </ul>
bit1, bit0	OP01, OP00: OPT0 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT0 pin.</li> </ul>

### 15.4.3 Output Data Buffer Register (OPDBR)

The Output Data Buffer Register is composed of twelve registers (OPDBRB to OPDBR0). The value of the OPDBRx register specified by the BNKF, RDA2 to RDA0 bits is loaded into the OPDR register at the rising edge of the write signal generated by the Data Write Control Unit.

#### ■ Output Data Buffer Upper Register (OPDBR)

Figure 15.4-6 Output Data Buffer Upper Register (OPDBR)



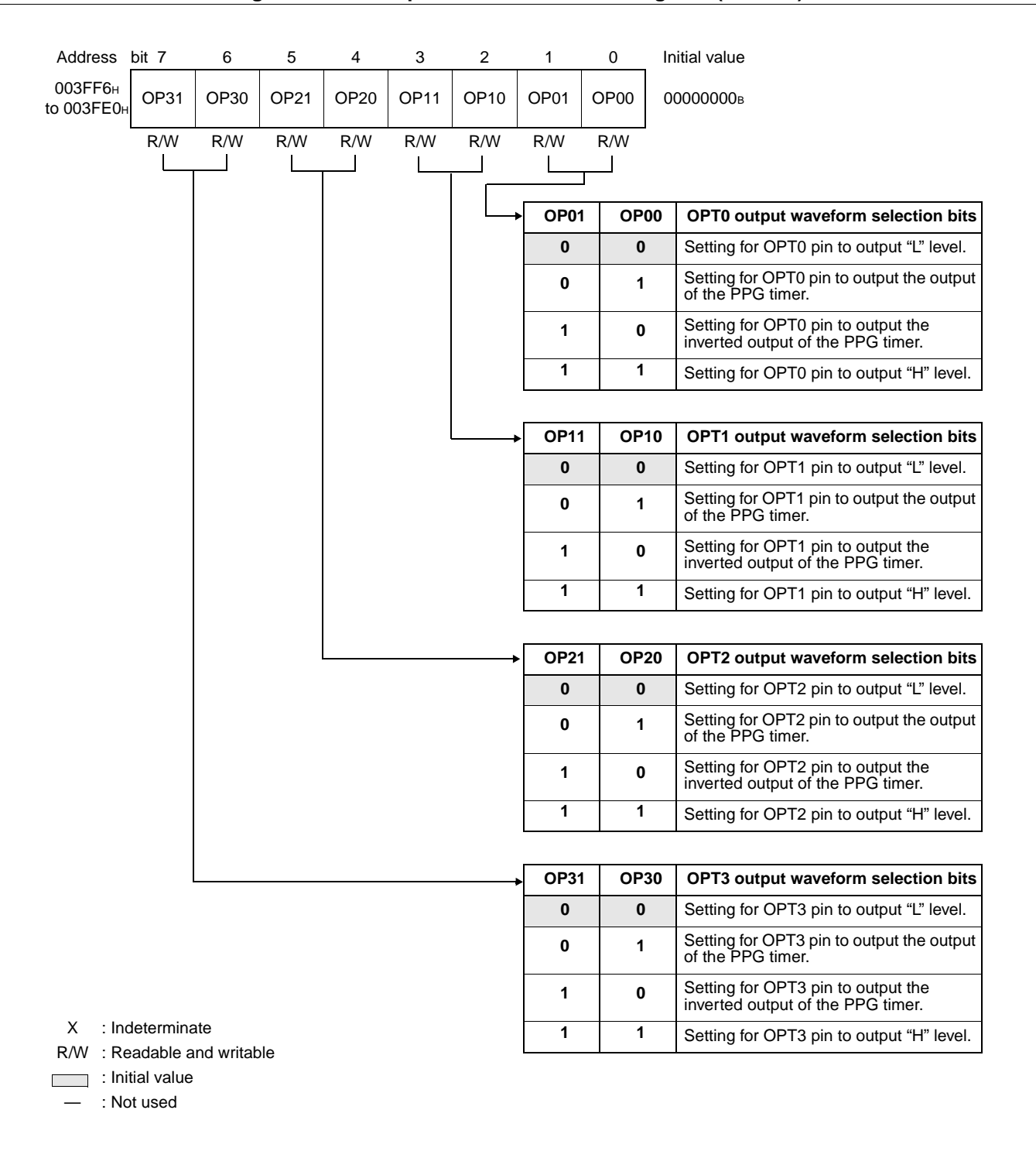
**Table 15.4-5 Output Data Buffer Upper Register (OPDBR) Bits**

Bit name		Function
bit15 to bit12	BNKF, RDA2 to RDA0: OPDBR register selection bits	<ul style="list-style-type: none"> <li>These bits indicate the addresses of the OPDBR registers and decide which Output Data Buffer Register value is loaded into the OPDR register after it is loaded into the OPDR register.</li> </ul>
bit11, bit10	OP51, OP50: OPT5 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT5 pin after it is loaded into the OPDR register.</li> </ul>
bit9, bit8	OP41, OP40: OPT4 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT4 pin after it is loaded into the OPDR register.</li> </ul>



■ Output Data Buffer Lower Register (OPDBR)

Figure 15.4-7 Output Data Buffer Lower Register (OPDBR)



**Table 15.4-6 Output Data Buffer Lower Register (OPDBR) Bits**

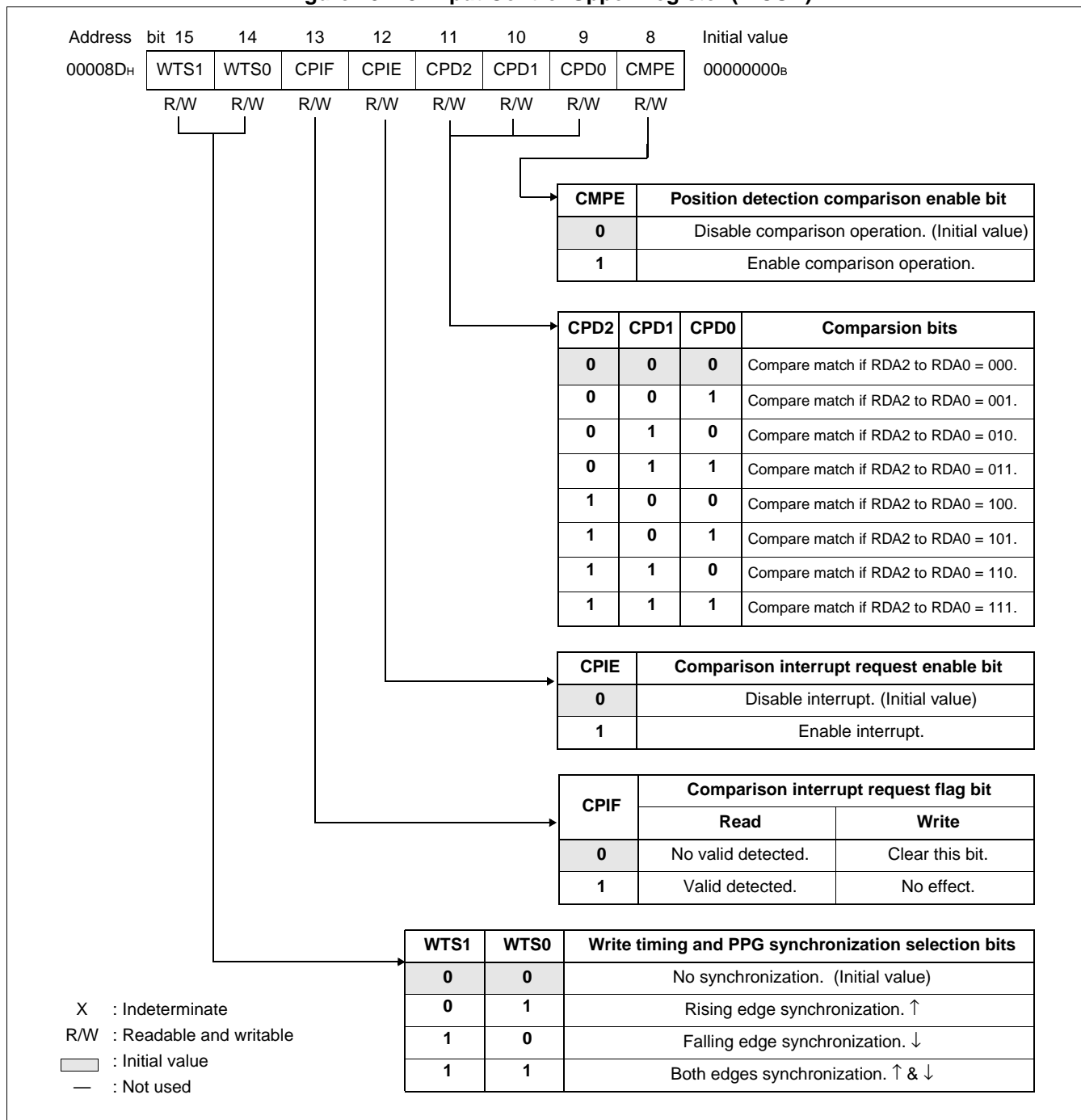
Bit name		Function
bit7, bit6	OP31, OP30: OPT3 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT3 pin after it is loaded into the OPDR register.</li> </ul>
bit5, bit4	OP21, OP20: OPT2 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT2 pin after it is loaded into the OPDR register.</li> </ul>
bit3, bit2	OP11, OP10: OPT1 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT1 pin after it is loaded into the OPDR register.</li> </ul>
bit1, bit0	OP01, OP00: OPT0 output waveform selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the kind of the output waveform to the OPT0 pin after it is loaded into the OPDR register.</li> </ul>

## 15.4.4 Input Control Register (IPCR)

The Input Control Register (IPCR) is a register which sets the control of the position detection inputs.

### Input Control Upper Register (IPCUR)

Figure 15.4-8 Input Control Upper Register (IPCUR)

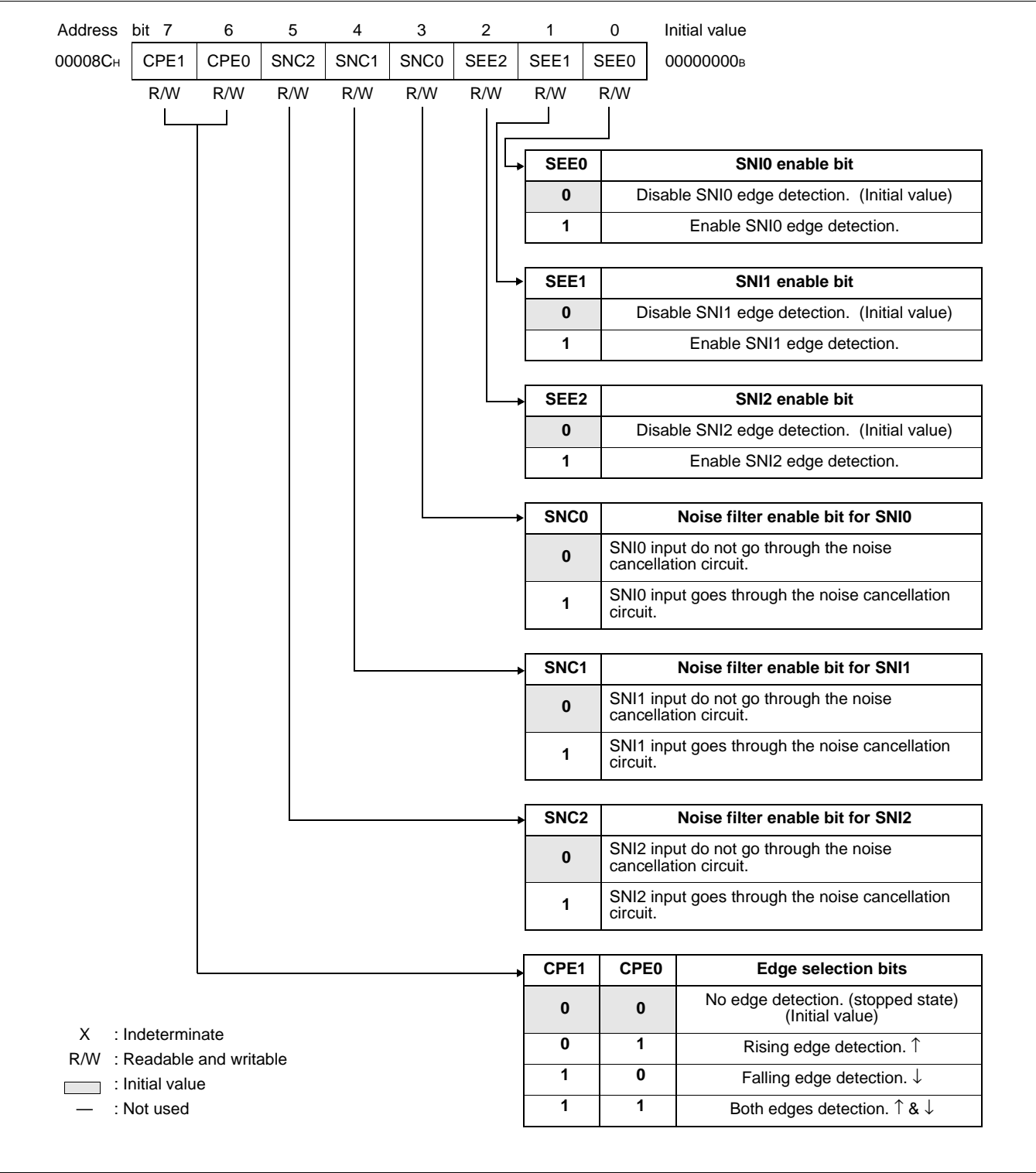


**Table 15.4-7 Input Control Upper Register (IPCUR) Bits**

Bit name		Function
bit15, bit14	WTS1, WTS0: PPG edge synchronization selection bits	<ul style="list-style-type: none"> <li>These bits are used to select the synchronization edge of the next coming of PPG signal with the write timing.</li> </ul>
bit13	CPIF: Comparison interrupt request flag bit	<ul style="list-style-type: none"> <li>Comparison interrupt request flag.</li> <li>It is a comparison interrupt request flag for the comparison circuit. When the SNI2 to SNI0 bits are compared and matched with the CPD2 to CPD0 bits, this bit is set to “1”.</li> <li>When comparison interrupt enable bit (CPIE) is also set to “1”, the interrupt is generated.</li> <li>This bit is cleared by writing “0”. Writing “1” has no effect.</li> <li>In read-modify-write operation, “1” is always read.</li> </ul>
bit12	CPIE: Comparison interrupt request enable bit	<ul style="list-style-type: none"> <li>Comparison interrupt enable bit.</li> <li>When this bit is set to “1” and the comparison interrupt request flag (CPIF) is also set to “1”, the interrupt is generated.</li> </ul>
bit11 to bit9	CPD2 to CPD0: Comparison bits	<ul style="list-style-type: none"> <li>These bits are used to compare with the RDA2 to RDA0 bits of the Output Data Register, when the value of these bits are matched with the value of RDA2 to RDA0 bits, the compare interrupt flag (CPIF) is set to “1”.</li> </ul>
bit8	CMPE: Position detection comparison enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable the comparison operation for the position detection.</li> </ul>

■ Input Control Lower Register (IPCLR)

Figure 15.4-9 Input Control Lower Register (IPCLR)



**Table 15.4-8 Input Control Lower Register (IPCLR) Bits**

Bit name		Function
bit7, bit6	CPE1, CPE0: Input polarity selection bits	<ul style="list-style-type: none"> <li>• Input polarity selection bits.</li> <li>• These bits are used to select the polarity of the input edge for the position detection, the position detection operates according to the input edge polarity set to these bits.</li> </ul>
bit5 to bit3	SNC2 to SNC0: Noise filter enable bits for SNI2 to SNI0	<ul style="list-style-type: none"> <li>• These bits are used to select the noise cancellation function when the inputs of the pins SNI2 to SNI0 are enable.</li> <li>• The noise cancellation circuit starts the internal n-bit counter when an active level is inputted (the value of n can be 2, 3, 4, 5, which depends on the setting of S21,S20, S11,S10 and S01,S00 bits in the Noise Cancellation Register). If the active level is held until the counter overflows, the circuit accepts input from the SNI2 to SNI0 pins. Therefore, the pulse width of noise that can be cancelled is about <math>2^n</math> machine cycles.</li> </ul> <p>(Note) When the noise cancellation circuit is enable, the input becomes invalid in a mode such as STOP mode in which the internal clock is stopped.</p>
bit2 to bit0	SEE2 to SEE0: SNI2 to SNI0 enable bits	<ul style="list-style-type: none"> <li>• Pins SNI2 to SNI0 edge detection enable bits.</li> <li>• When they are set to “1”, the edge detection of the pins SNI2 to SNI0 are enable.</li> <li>• Please set these bits before setting CMPE to “1”.</li> </ul>

## 15.4.5 Compare Clear Register (CPCR)

The Compare Clear Register (CPCR) is 16-bit register. When this register is matched with the count value of 16-bit timer, the 16-bit timer is reset to "0000<sub>H</sub>".

### ■ Compare Clear Register (CPCR)

Compare Clear Register is a 16-bit register and is used to compare the count value of the 16-bit timer. The initial value of this register is undetermined, so that the register must be set a value before starting an operation.

#### Notes:

Word access instruction to this register must be used.

When this register is matched with the count value of 16-bit timer, 16-bit timer is reset to "0000<sub>H</sub>" and the compare clear interrupt flag is set. Furthermore, when the interrupt operation is enabled, interrupt request is sent to the CPU.

If the Compare Clear Register (CPCR) is loaded a value same as the Timer Counter value at that moment, the comparison operation will NOT be performed until next same counter value.

**Figure 15.4-10 Compare Clear Register (CPCR)**

Compare Clear Register (Upper)										
	bit	15	14	13	12	11	10	9	8	
Address: 003FFB <sub>H</sub>		CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	CPCR
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		X	X	X	X	X	X	X	X	

Compare Clear Register (Lower)										
	bit	7	6	5	4	3	2	1	0	
Address: 003FFA <sub>H</sub>		CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	CPCR
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨		X	X	X	X	X	X	X	X	

## 15.4.6 Timer Buffer Register (TMBR)

The Timer Buffer Register (TMBR) is used to read the count value of 16-bit timer.

### ■ Timer Buffer Register (TMBR)

The timer buffer register is used to store the count value of the 16-bit timer at the moment when a write timing or position detection trigger is generated, and the counter is then cleared to "0000<sub>H</sub>".

Note: Word access instruction to the Timer Buffer Register must be used.

**Figure 15.4-11 Timer Buffer Register (TMBR)**

Timer Buffer Register (Upper)									
	bit	15	14	13	12	11	10	9	8
Address: 003FFD <sub>H</sub>		T15	T14	T13	T12	T11	T10	T09	T08
Read/write ⇨		R	R	R	R	R	R	R	R
Initial value ⇨		0	0	0	0	0	0	0	0
Timer Buffer Register (Lower)									
	bit	7	6	5	4	3	2	1	0
Address: 003FFC <sub>H</sub>		T07	T06	T05	T04	T03	T02	T01	T00
Read/write ⇨		R	R	R	R	R	R	R	R
Initial value ⇨		0	0	0	0	0	0	0	0

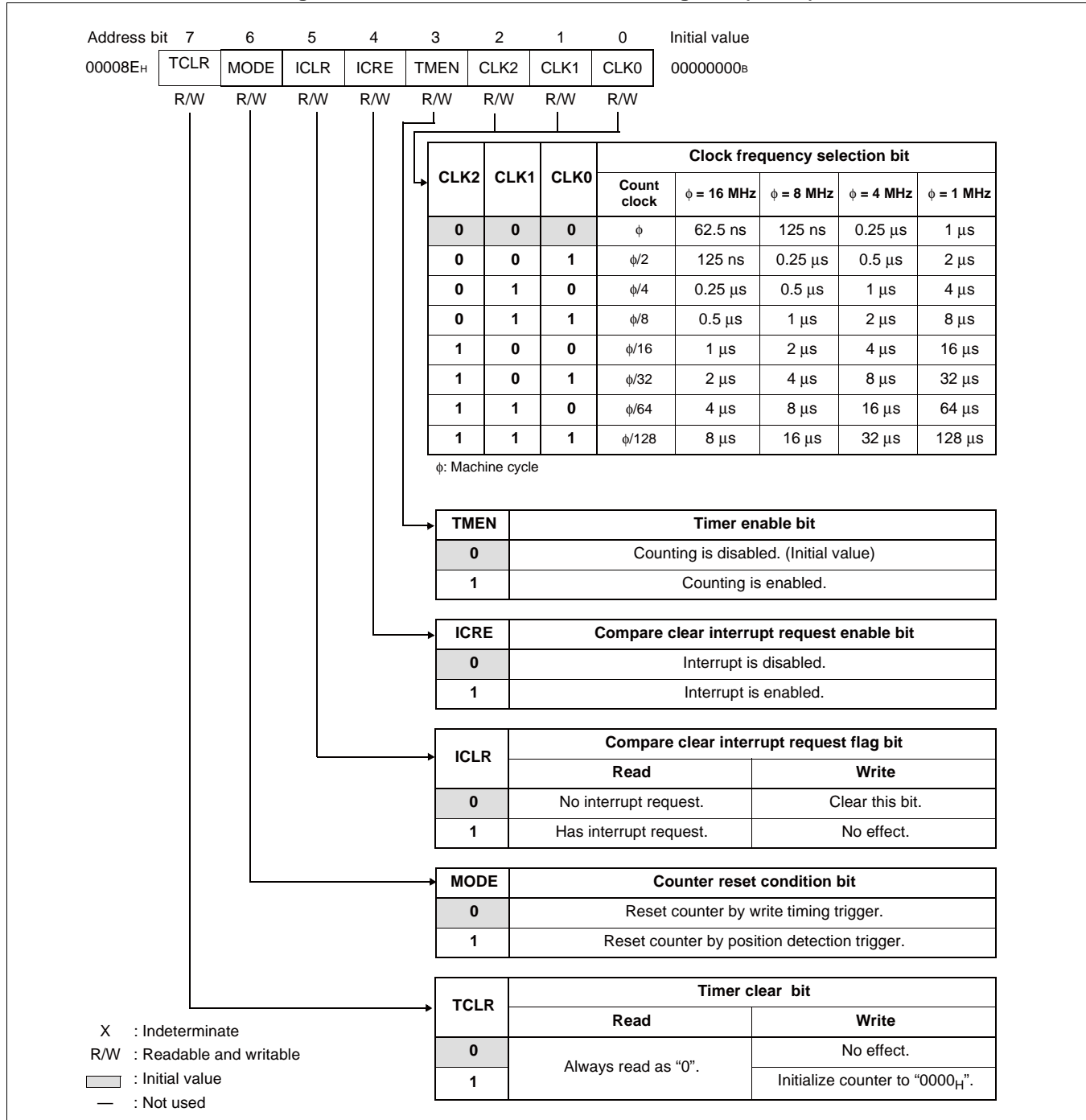


## 15.4.7 Timer Control Status Register (TCSR)

The Timer Control Status Register (TCSR) is used to control the operation of the 16-bit timer.

### ■ Timer Control Status Register (TCSR)

Figure 15.4-12 Timer Control Status Register (TCSR)



**Table 15.4-9 Timer Control Status Register (TCSR)**

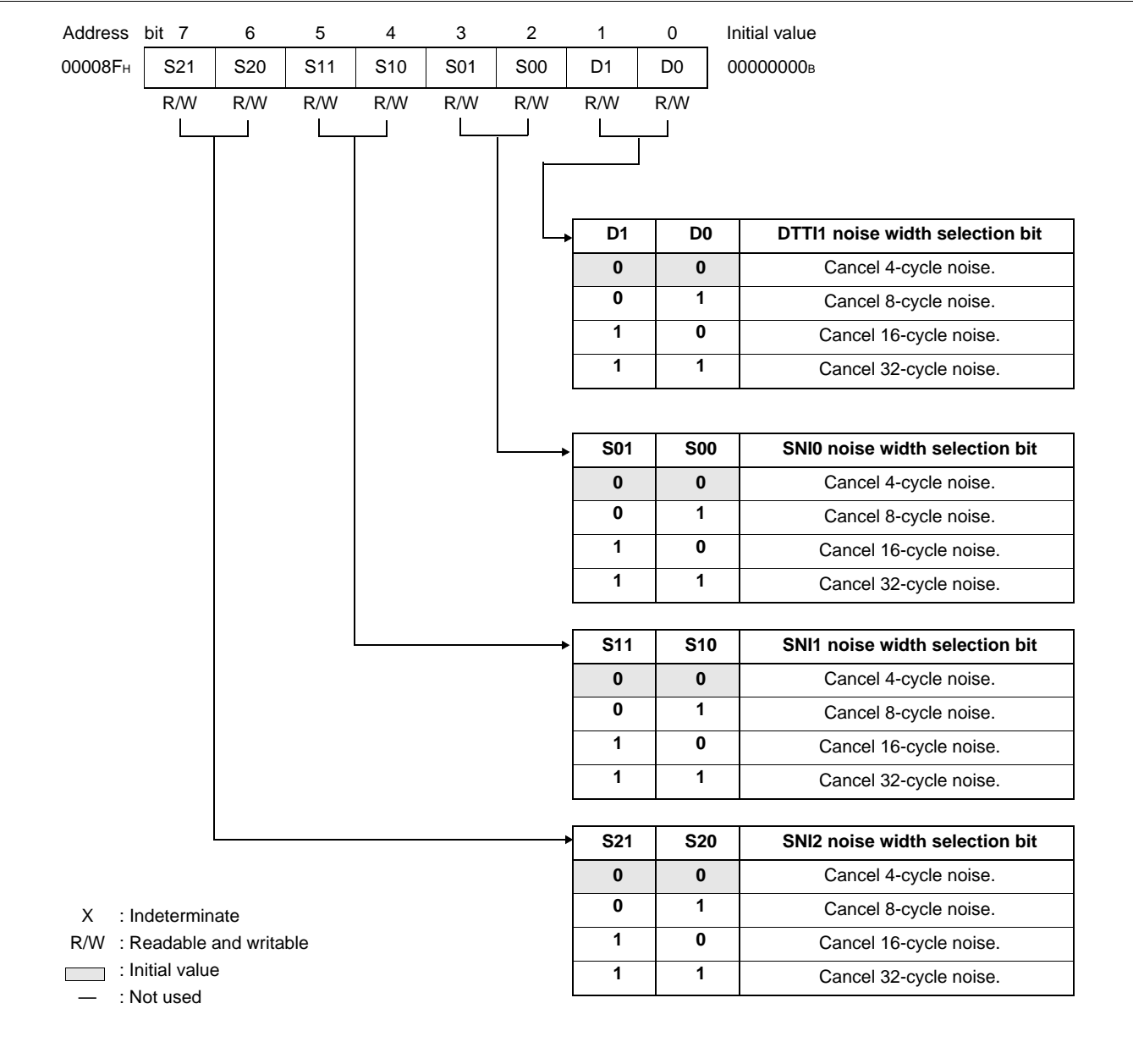
Bit name		Function
bit7	TCLR: Timer clear bit	<ul style="list-style-type: none"> <li>The read value is always “0”.</li> <li>Writing “1” to this bit initialize the counter to “0000<sub>H</sub>”.</li> <li>Writing “0” has no effect.</li> </ul>
bit6	MODE: Timer reset condition bit	<ul style="list-style-type: none"> <li>This bit is used to set the reset condition for the 16-bit timer.</li> <li>When it is “0”, 16-bit timer is reset by the write timing signal.</li> <li>When it is “1”, 16-bit timer is reset by the position detection signal.</li> </ul> <p>(Note) Reset of the counter value is done at the changing point of the count value.</p>
bit5	ICLR: Compare clear interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is an interrupt request flag for compare clear.</li> <li>When the compare clear register and 16-bit free-run timer value are matched, the counter is cleared and this bit becomes “1”.</li> <li>Interrupt is generated when the interrupt request enable bit (bit12: ICRE) is set to “1”.</li> <li>Writing “0” clears this bit.</li> <li>Writing “1” has no effect.</li> <li>In read-modify-write operation, “1” is always read.</li> </ul>
bit4	ICRE: Compare clear interrupt request enable bit	<ul style="list-style-type: none"> <li>This is the interrupt request enable bit for the compare clear.</li> <li>When this bit is “1” and the interrupt flag (bit13: ICLR) is set to “1”, an interrupt is generated.</li> </ul>
bit3	TMEN: Timer enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable/disable the counting of the 16-bit timer.</li> <li>Writing “1” to this bit enables the counting of the 16-bit timer.</li> <li>Writing “0” to this bit disables the counting of the 16-bit timer.</li> </ul> <p>(Note) When the 16-bit timer is disable, the output compare operation is also disabled.</p>
bit2 to bit0	CLK2 to CLK0: Clock frequency selection bit	<ul style="list-style-type: none"> <li>These bits are used to select count clock for the 16-bit free-run timer.</li> </ul> <p>(Note) It is recommend to change these bits when the timer is in stop state because the clock is changed as soon as these bits are updated.</p>

### 15.4.8 Noise Cancellation Control Register (NCCR)

The Noise Cancellation Control Register (NCCR) is used to control the noise pulse width to be cancelled for DTTI1 and SNIx pins.

■ Noise Cancellation Control Register (NCCR)

Figure 15.4-13 Noise Cancellation Control Register (NCCR)



**Table 15.4-10 Noise Cancellation Control Register (NCCR) Bits**

Bit name		Function
bit7, bit6	S21,S20: Noise width selection bits	<ul style="list-style-type: none"> <li>These bits are used to specify the noise pulse width to be removed for SNI2 pin.</li> </ul>
bit5, bit4	S11,S10: Noise width selection bits	<ul style="list-style-type: none"> <li>These bits are used to specify the noise pulse width to be removed for SNI1 pin.</li> </ul>
bit3, bit2	S01,S00: Noise width selection bits	<ul style="list-style-type: none"> <li>These bits are used to specify the noise pulse width to be removed for SNI0 pin.</li> </ul>
bit1, bit0	D1,D0: Noise width selection bits	<ul style="list-style-type: none"> <li>These bits are used to specify the noise pulse width to be removed for DTTI1 pin.</li> </ul>

## 15.5 Multi-pulse Generator Interrupts

---

The Multi-pulse Generator can generate an interrupt request when in the following causes:

- Write timing output is generated by the Data Write Control Unit
  - Any valid position detection input is detected
  - Comparison match between CPD2 to CPD0 of Input Control Register (IPCR: CPD2 to CPD0) and RDA2 to RDA0 bit of Output Data Register (OPDR: RDA2 to RDA0)
  - Compare Clear is generated by the 16-bit Timer
  - DTTI1 is changed to low signal level
- 

### ■ Multi-pulse Interrupts

There are five interrupt causes generated from Multi-pulse Generator, that are as follows:

- Write Timing Interrupt
- Compare Clear Interrupt
- Position Detect Interrupt
- Compare Match Interrupt
- DTTI1 Interrupt

Write Timing Interrupt is multiplexed with Compare Clear Interrupt and Position Detect Interrupt is multiplexed with Compare Match Interrupt.

#### ● Write Timing Interrupt

If the WTIE bit of the Output Control Register (OPCR: WTIE) is set to "1", this Write Timing Interrupt is generated when the write timing is generated by the Data Write Control Circuit to make data transfer from one of 12 Output Data Buffer Registers (OPDBRB to OPDBR0) to the Output Data Register (OPDR).

When this interrupt is generated, the write timing interrupt flag bit of the Output Control Register (OPCR: WTIF) is set to "1".

#### ● Compare Clear Interrupt

If the ICRE bit of the Timer Control Register (TCSR: ICRE) is set to "1", this Compare Clear Interrupt is generated when the 16-bit Timer is underflow.

When this interrupt is generated, the Compare Clear interrupt flag bit of the Timer Control Register (TCSR: ICLR) is set to "1".

### ● Position Detect Timing Interrupt

If the PDIE bit of the Output Control Register (OPCR: PDIE) is set to "1", this Position Detect Interrupt is generated when the write timing is output by Position Detect Circuit to make data transfer from one of 12 Output Data Buffer Registers (OPDBRB to OPDBR0) to the Output Data Register (OPDR). This write timing output can be generated by either the compare match of the level of the position input (SNI2 to SNI0) with RDA2 to RDA0 bits of the Output Data Register (OPDR: RDA2 to RDA0), or a edge detected of the position input (SNI2 to SNI0) with one of 3 different kinds of edge setting.

When this interrupt is generated, the position detect interrupt flag bit of the Output Control Register (OPCR: PDIF) is set to "1".

### ● Compare Match Interrupt

If the CPIE bit of the Input Control Register (IPCR: CPIE) is set to "1", this Compare Match Interrupt is generated when the RDA2 to RDA0 bits of the Output Data Register (OPDR: RDA2 to RDA0) are matched with the CPD2 to CPD0 bits of the Input Control Register (IPCR: CPD2 to CPD0).

When this interrupt is generated, the Compare match interrupt flag bit of the Input Control Register (IPCR: CPIF) is set to "1".

### ● DTTI1 Interrupt

If the DTIE bit of the Output Control Register (OPCR: DTIE) is set to "1", this DTTI1 Interrupt is generated whenever a low input is detected at the DTTI1 pin.

When this interrupt is generated, the DTTI1 interrupt flag bit of the Output Control Register (OPCR: DTIF) is set to "1".

## ■ Multi-pulse Generator Interrupt Source

INTERRUPT #22: This interrupt is generated when a DTTI1 interrupt is happened.

DTTI1 interrupt is generated if OPCR: DTIE is set to "1" when a low level input is detected at the DTTI1 pin.

INTERRUPT #26: This interrupt is generated when either a Write Timing interrupt or Compare Clear interrupt is happened.

Write timing interrupt is generated if OPCR: WTIE is set to "1" when a write timing signal is generated from the Data Write Control Circuit.

Compare clear interrupt is generated if TCSR: ICRE is set to "1" when the count value of 16-bit timer matches with the Compare clear register (CPCR).

INTERRUPT #28: This interrupt is generated when either a Position Detect interrupt or Compare Match interrupt is happened.

Position detect interrupt is generated if OPCR: PDIE is set to "1" when an effective edge at SNI2 to SNI0 is detected.

Compare match interrupt is generated if IPCR: CPIE is set to "1" when the SNI2 to SNI0 inputs match with the RDA2 to RDA0 bits of the Output Data Register (OPDR).

## ■ Multi-pulse Generator Interrupts and EI<sup>2</sup>OS

**Table 15.5-1 Multi-pulse Generator Interrupts and EI<sup>2</sup>OS**

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Upper	Bank	
DTTI1	#22 (16 <sub>H</sub> )	ICR05	0000B5 <sub>H</sub>	FFFFAC <sub>H</sub>	FFFFAD <sub>H</sub>	FFFFAE <sub>H</sub>	×
Write Timing or Compare Clear	#26(1A <sub>H</sub> )	ICR07	0000B7 <sub>H</sub>	FFFF94 <sub>H</sub>	FFFF95 <sub>H</sub>	FFFF96 <sub>H</sub>	○
Position Detect or Compare Match	#28 (1C <sub>H</sub> )	ICR08	0000B8 <sub>H</sub>	FFFF8C <sub>H</sub>	FFFF8D <sub>H</sub>	FFFF8E <sub>H</sub>	○

○ : Support EI<sup>2</sup>OS

## ■ Multi-pulse Generator EI<sup>2</sup>OS Functions

Multi-pulse Generator has a circuit for operating EI<sup>2</sup>OS, which can be started up when either Write Timing/Compare Clear interrupt or Position Detect/Compare Match interrupt is generated.

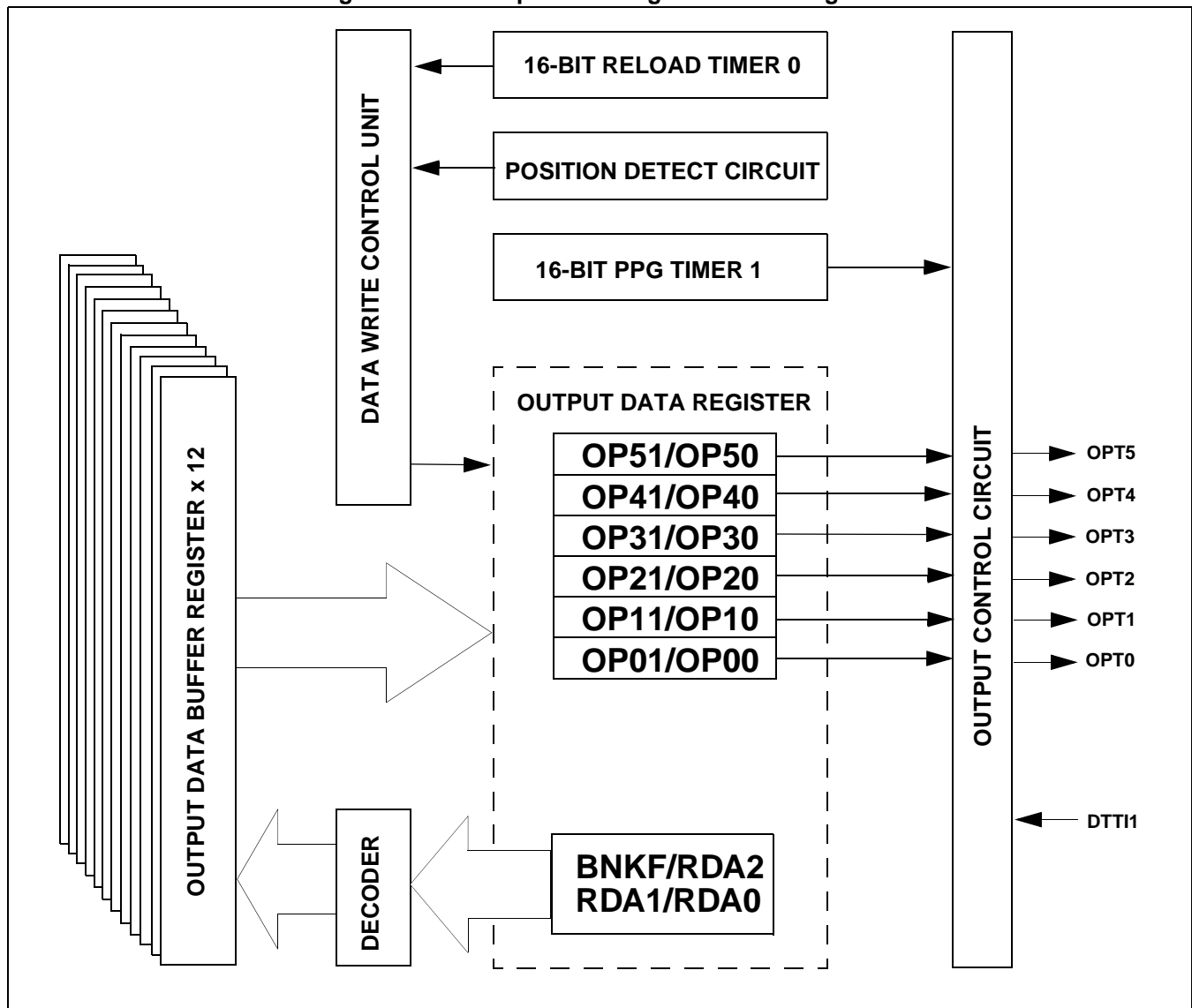
However, EI<sup>2</sup>OS is available only when other peripheral functions sharing the interrupt control register (ICR) do not use interrupts. For example, when Write Timing/Compare Clear interrupt is used to trigger EI<sup>2</sup>OS, DTP/external interrupt channels 4/5 detection must be disabled.

## 15.6 Operation of Multi-pulse Generator

The operation of the Multi-pulse Generator will be described in the following sections. According to the setting of (OPx1/OPx0) bits in the Output Data Register (OPDR), the OPTx pin outputs the corresponding kind of waveforms ("H" or "L" or PPG output). See Table 15.6-1.

### ■ Output Data Register Block Diagram

Figure 15.6-1 Output Data Register Block Diagram





## ■ Output Data Register (OPDR)

The content of the Output Data Register (OPDR) is received from the Output Data Buffer Register (OPDBRB to OPDBR0) according to the write timing signal (WTO) generated by the Data Write Control Unit, and the OPTx output waveform is updated. Moreover, the output level can be compulsorily fixed by the DTTI1 pin input.

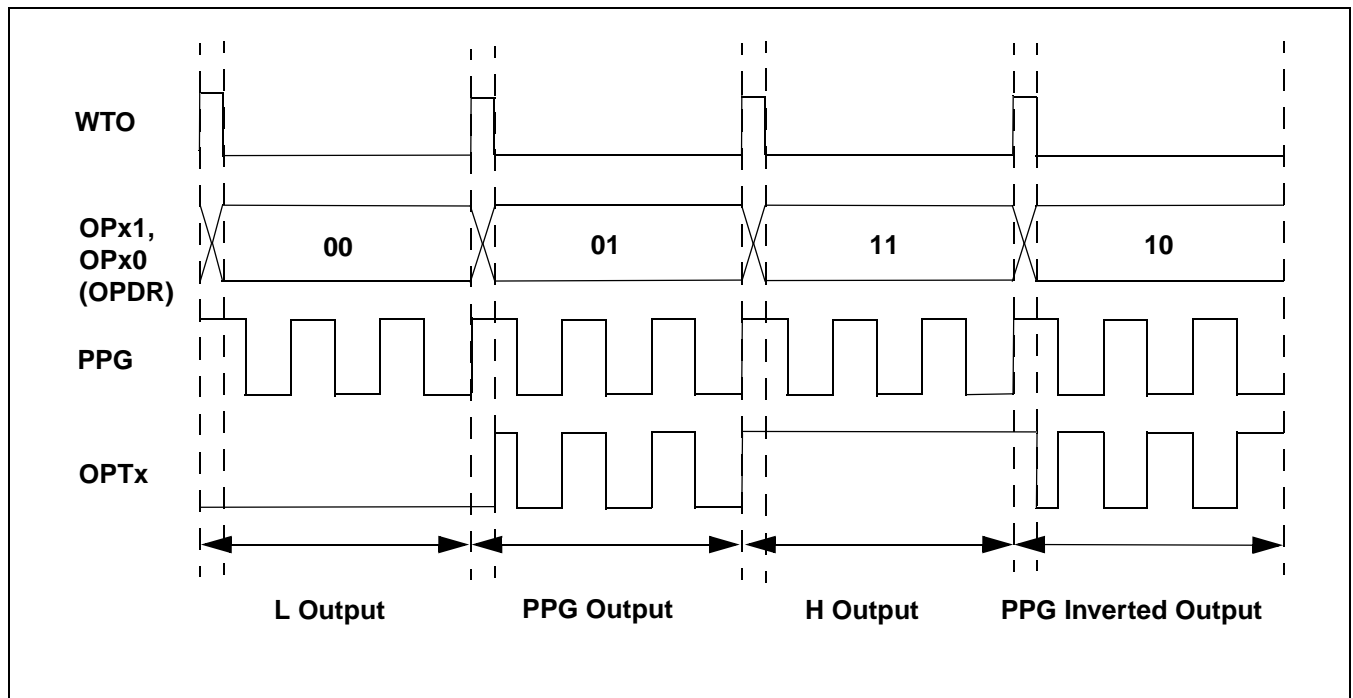
**Table 15.6-1 Output Data Register (OPDR)**

OPx1,OPx0 Setting	OPTx Output
OPx1,OPx0 = 0,0	Low Level
OPx1,OPx0 = 0,1	16-bit PPG Timer Output
OPx1,OPx0 = 1,0	16-bit PPG Timer Inverted Output
OPx1,OPx0 = 1,1	High Level

The OPTx output waveform timing diagram is shown in Figure 15.6-2 and the operation is explained in following paragraphs.

## ■ OPTx Output Waveform Timing Diagram (WTS1,WTS0 = 00<sub>B</sub>)

**Figure 15.6-2 OPTx Output Waveform Timing Diagram (WTS1,WTS0 = 00<sub>B</sub>)**



## 15.6.1 Operation of Position Detection

This section describes the operation of the Position Detection Circuit. When the effective position is detected, a Data Write Timing Output (WTIN1) will be generated to the Data Write Control Unit and a Position Detect Interrupt is generated if the OPCR: PDIE is set to "1".

### ■ Operation of Position Detection

The WTIN1 signal is generated by the Position Detection Circuit under the following conditions:

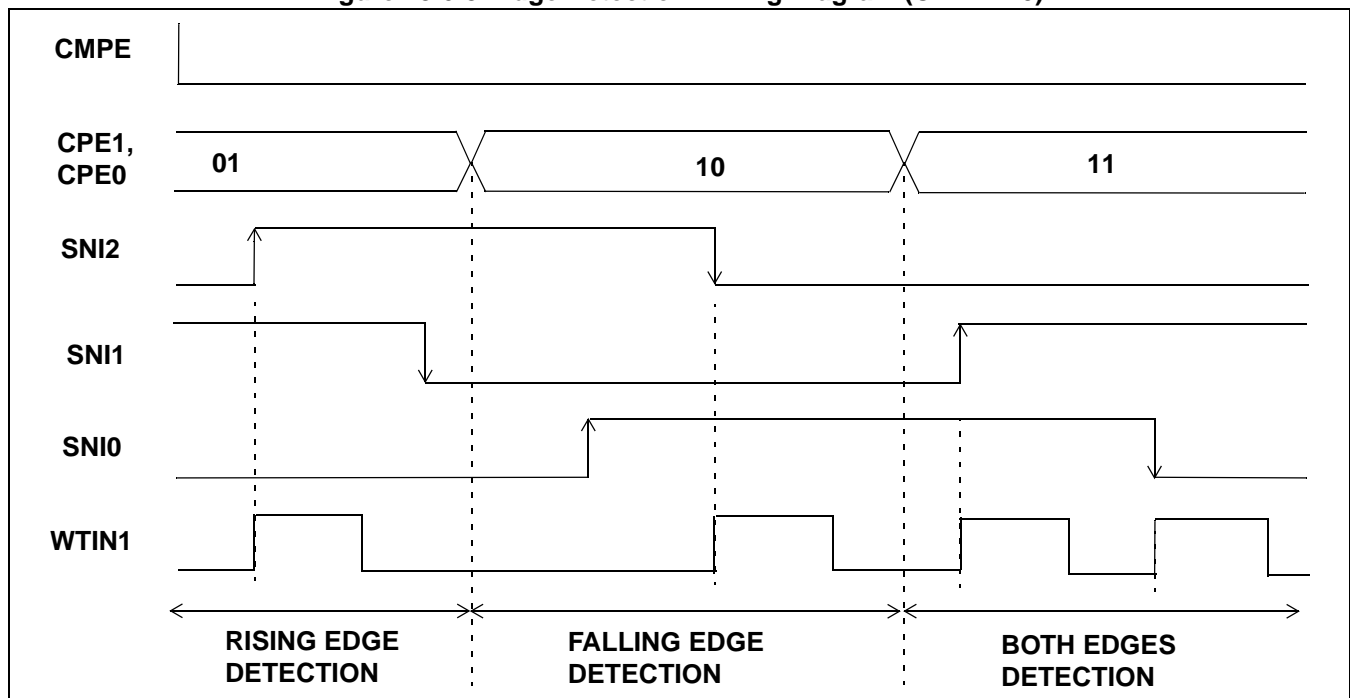
- A comparison match between SNI2 to SNI0 and RDA2 to RDA0, which is triggered by any effective edge of SNI2 to SNI0.
- A detection of effective edge at SNIx which is enabled by the corresponding SEEx bit.

When the CMPE bit (bit8) of the Input Control Register (IPCR) is set to "0", only the edge detection of SINx pins enabled by the SEE2 to SEE0 bits will engage in the edge detection operation for the position detection. For instance, when only the SEE0 bit is set to "1", the input edge to the pin SNI0 is in effect, the data write output signal is generated only when an effective edge is detected at the SIN0 pin. See Figure 15.6-3 for the timing diagram of the edge detection when CMPE = 0.

When the CMPE bit (bit8) of the Input Control Register (IPCR) is set to "1", the SNI2 to SNI0 will be engaged in the comparison operation with the RDA2 to RDA0 bits. The comparison is triggered by any edge change at SNI2 to SNI0 pins. See Figure 15.6-4 for the timing diagram of the edge detection when CMPE = 1.

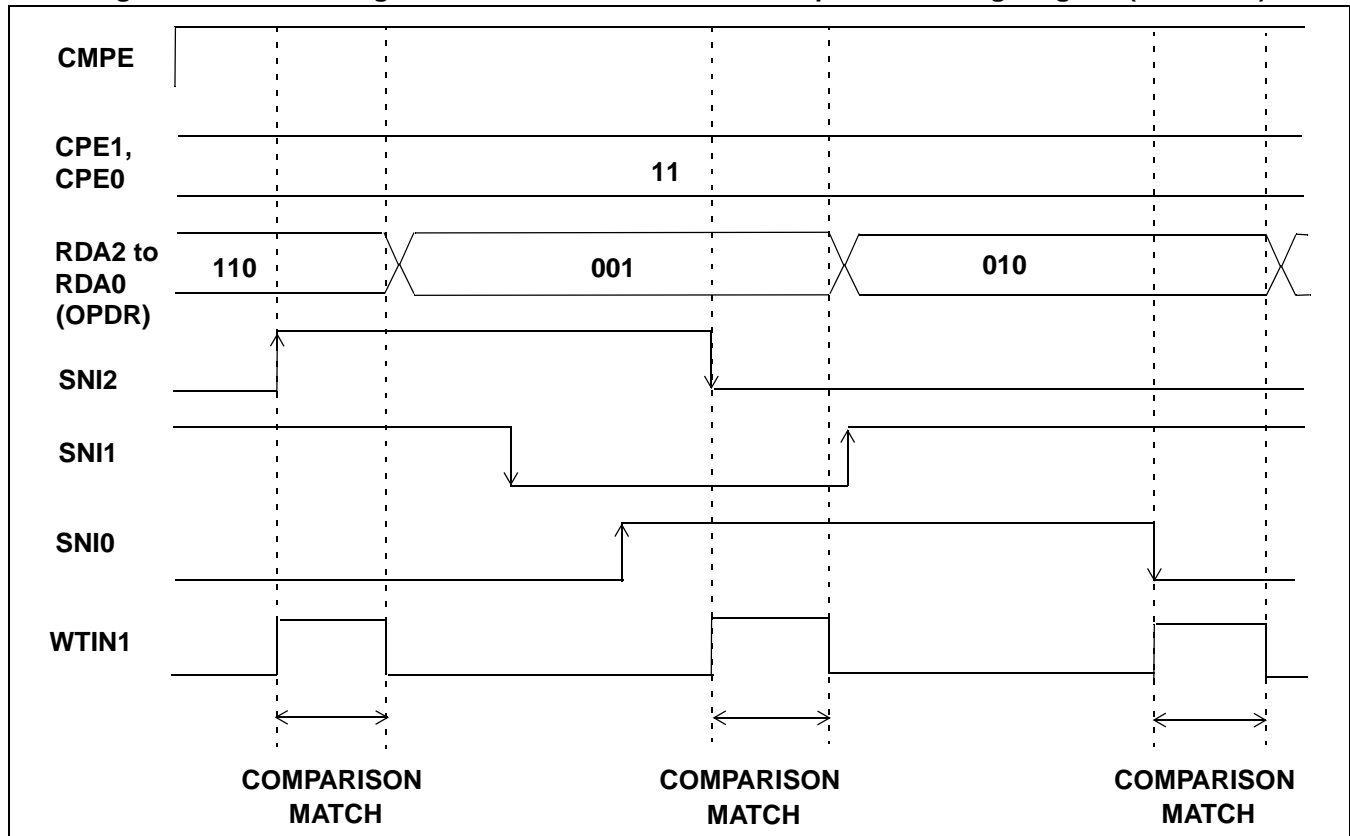
### ■ Edge Detection Timing Diagram (CMPE = 0)

Figure 15.6-3 Edge Detection Timing Diagram (CMPE = 0)



### ■ Both Edges Detection and SNIx/RDAx Comparison Timing Diagram (CMPE = 1)

Figure 15.6-4 Both Edges Detection and SNIx/RDAx Comparison Timing Diagram (CMPE = 1)



### ■ WTIN1 Output Condition and Register Setting

Table 15.6-2 WTIN1 Output Condition and Register Setting

CMPE	CPE1	CPE0	SEEx	WTIN1 Output Condition
0	0	0	0	No output. (Initial value)
0	X	X	0	No output.
0	0	0	1	No output.
0	0	1	1	Detect SNIx rising edge.
0	1	0	1	Detect SNIx falling edge.
0	1	1	1	Detect SNIx both edges.
1	0	0	X	Prohibited.
1	0	1	X	Detect SNIx rising edge and SNIx/RDAx comparison match.
1	1	0	X	Detect SNIx falling edge and SNIx/RDAx comparison match.
1	1	1	X	Detect SNIx both edges and SNIx/RDAx comparison match.

Note:

When CMPE = 1, SEEx should be set to "0", setting SEEx = 1 is not recommended.

## 15.6.2 Operation of Data Write Control Unit

The Data Write Control Unit is used to generate the write timing output (WTO) for transferring data from the Output Data Buffer Register (OPDBR) to Output Data Register (OPDR).

### ■ Operation of Data Write Control Unit

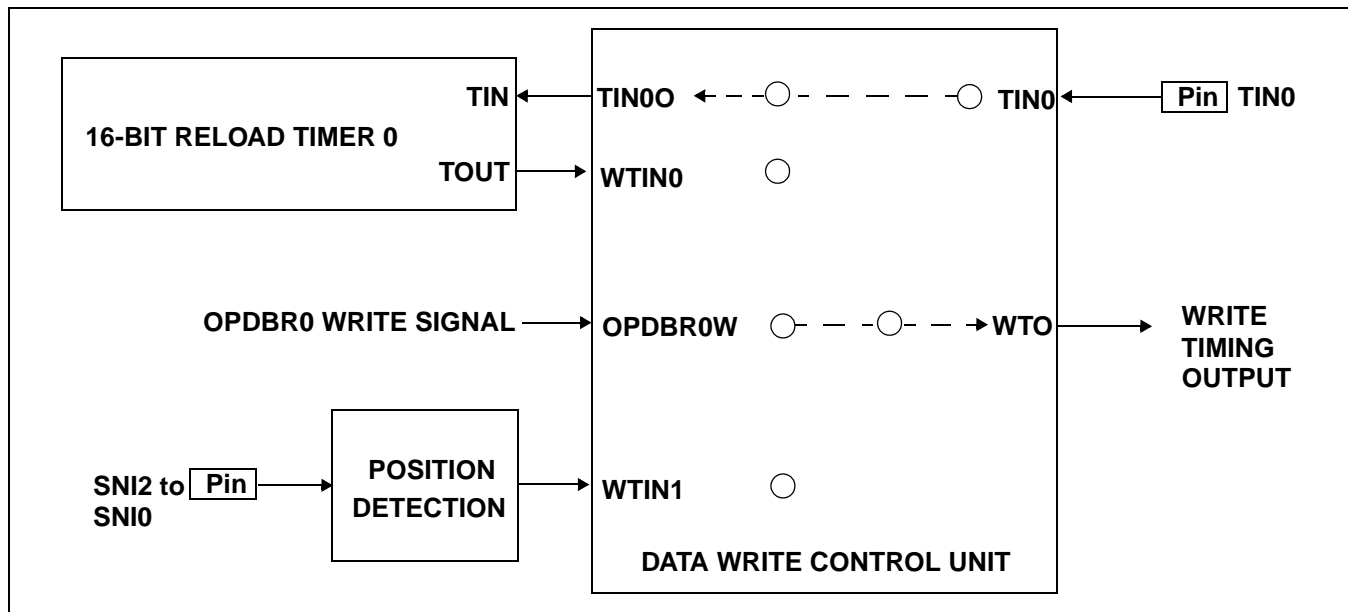
The Write Timing Output (WTO) can be generated by the following condition:

- After OPDBR0 is written by software.
- Triggered by the 16-bit reload timer 0 underflow.
- Triggered by the 16-bit reload timer 0 underflow. The 16-bit timer is started by the position detection comparison circuit.
- Triggered by the position detection input (SNI2 to SNI0) (16-bit Reload timer 0 acts as a delay).
- Triggered either by the 16-bit reload timer 0 underflow, or by the position detection input.

At the mean time the cause of generation of WTO will be defined by setting different value of OPS2 to OPS0 bit of the Output Control Register (OPCR: OPS2 to OPS0).

### ■ Signal Flow Diagram for OPDBR0 by Setting OPS2 to OPS0 = 000<sub>B</sub>

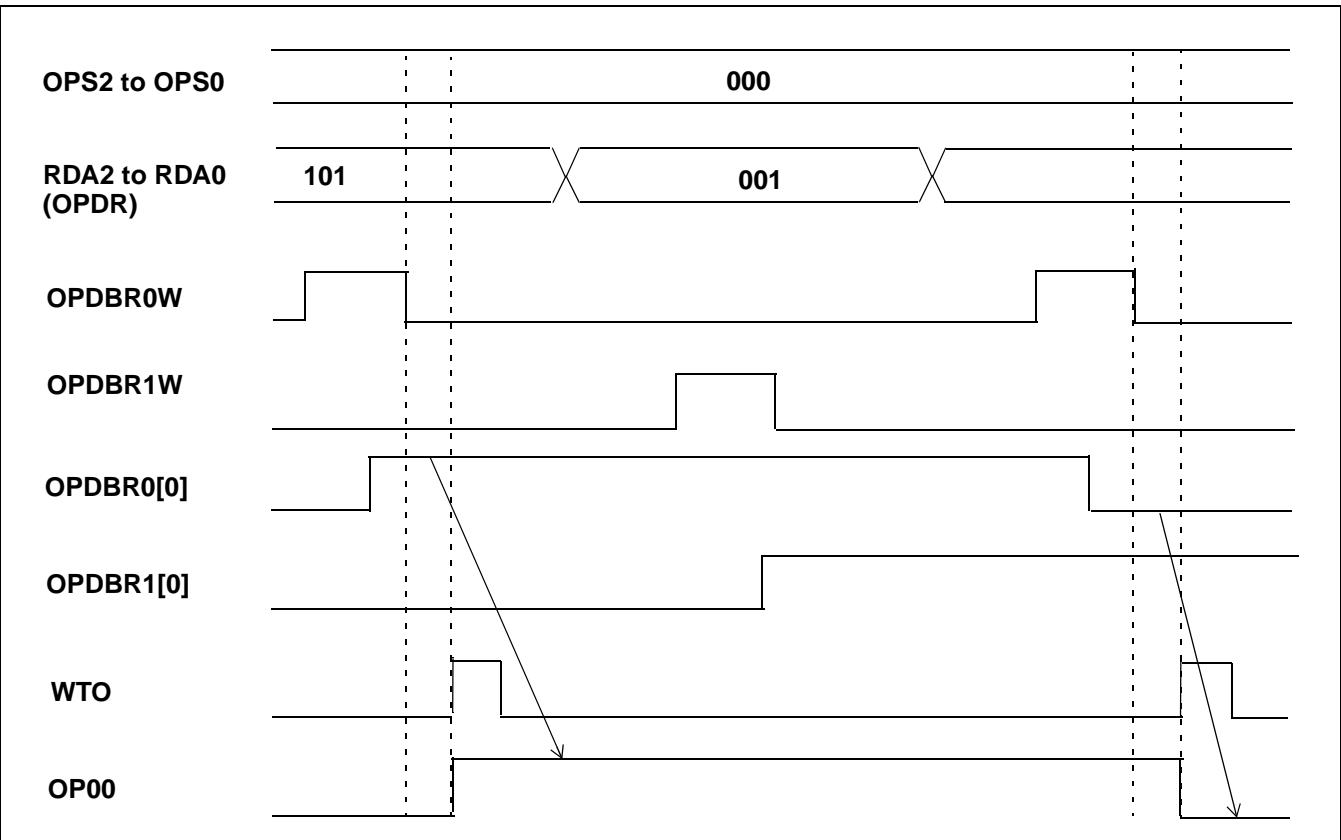
Figure 15.6-5 Signal Flow Diagram for OPDBR0 (OPS2 to OPS0 = 000<sub>B</sub>)



The write timing output signal is generated from the Data Write Control Unit whenever a value is written to the OPDBR0 register, and the data in OPDBR0 is transferred to OPDR register one cycle later.

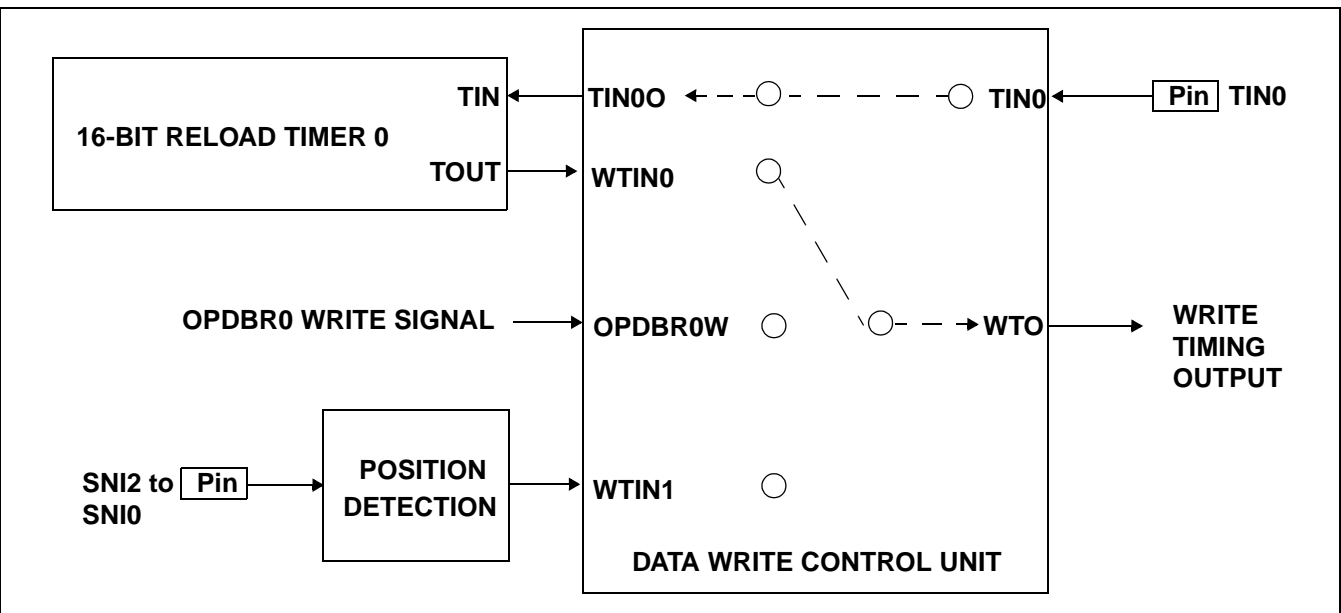
■ OPDR Register Write Timing Diagram (OPS2 to OPS0 = 000<sub>B</sub>)

Figure 15.6-6 OPDR Register Write Timing Diagram (OPS2 to OPS0 = 000<sub>B</sub>)



■ Signal Flow Diagram for Reload Timer 0 Underflow by Setting OPS2 to OPS0 = 001<sub>B</sub>

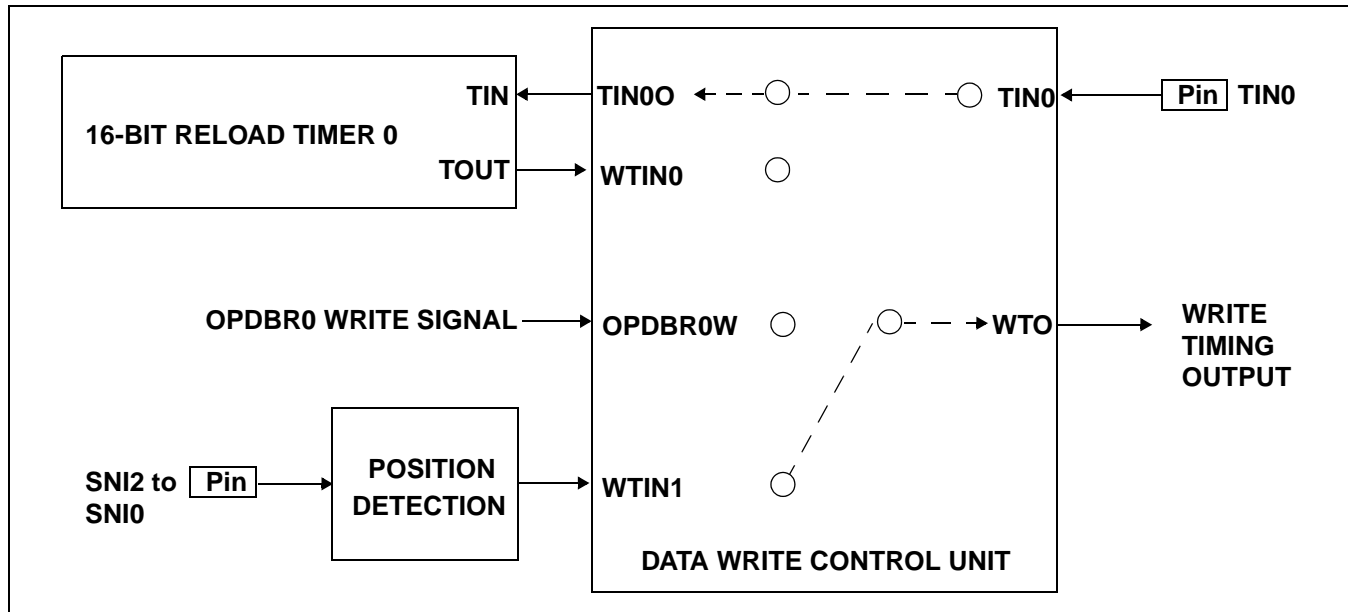
Figure 15.6-7 Signal Flow Diagram for Reload Timer 0 Underflow (OPS2 to OPS0 = 001<sub>B</sub>)



The 16-bit reload timer 0 can be triggered by both TIN input and software to generate the write signal at this setting. The write signal is controlled by the 16-bit reload timer 0 underflow.

### ■ Signal Flow Diagram for Position Detection by Setting OPS2 to OPS0 = 010<sub>B</sub> or 110<sub>B</sub>

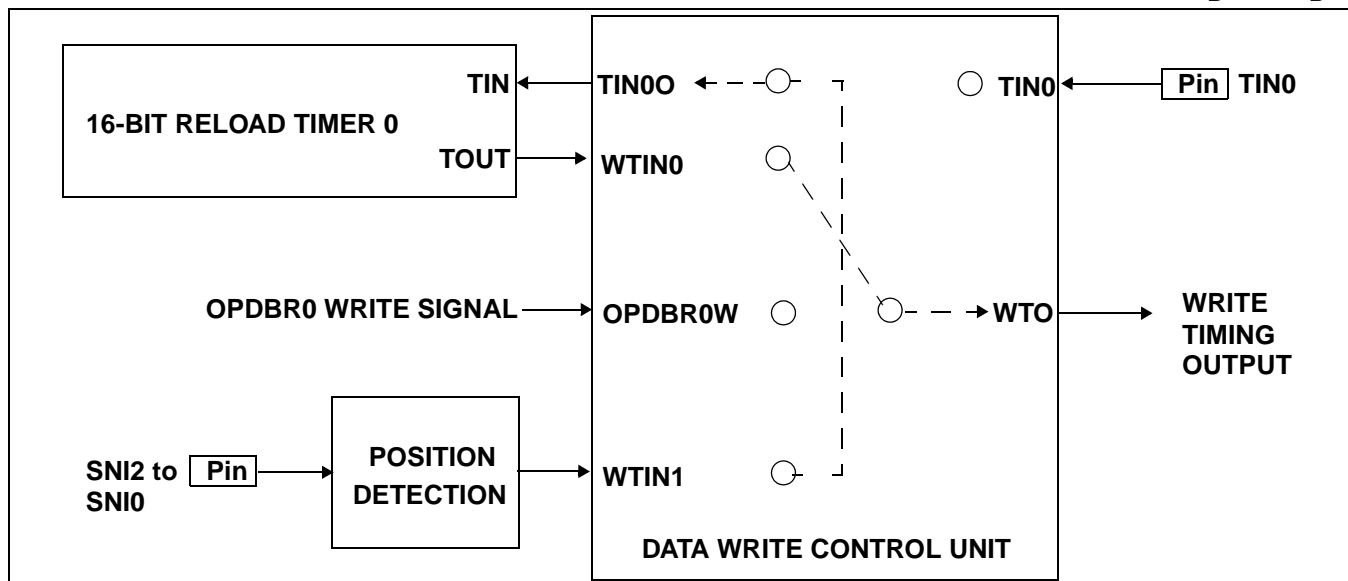
Figure 15.6-8 Signal Flow Diagram for Position Detection (OPS2 to OPS0 = 010<sub>B</sub> or 110<sub>B</sub>)



The write signal is generated by a comparison match or effective edge input of position detection.

### ■ Signal Flow Diagram for Reload Timer 0 and Position Detection by Setting OPS2 to OPS0 = 011<sub>B</sub> or 111<sub>B</sub>

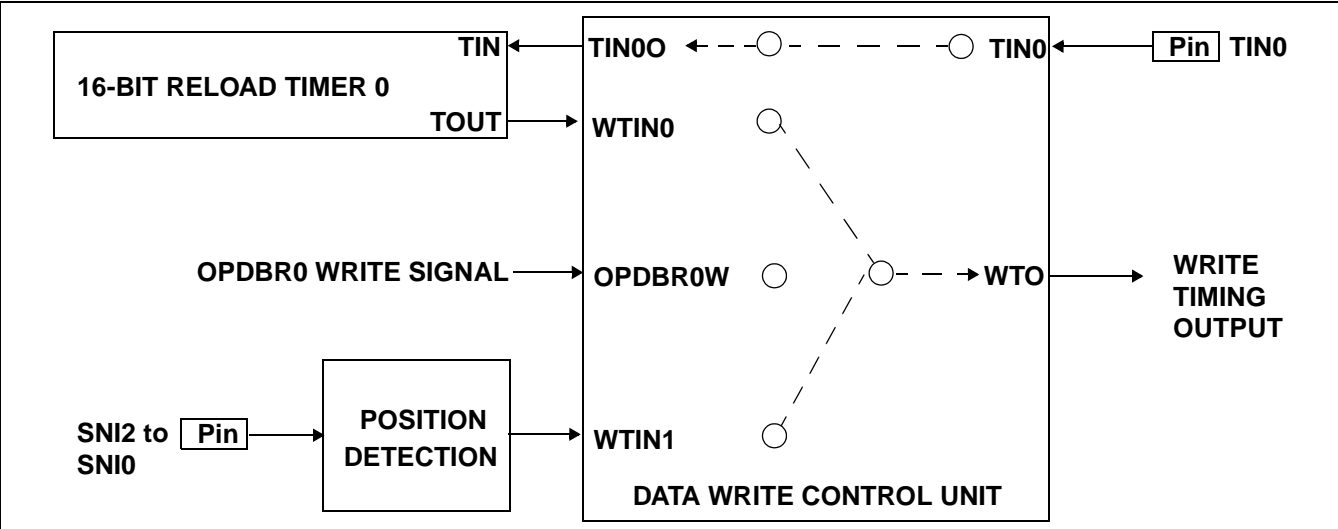
Figure 15.6-9 Signal Flow Diagram for Reload Timer 0 & Position Detect (OPS2 to OPS0 = 011<sub>B</sub> or 111<sub>B</sub>)



At this setting the 16-bit reload timer 0 is started by the compare match or effective edge input of the position detection circuit, write signal is then generated whenever the 16-bit reload timer 0 is underflow. The compare match is triggered by any effective edge change in SNI2 to SNI0 pins.

■ Signal Flow Diagram for Reload Timer 0 or Position Detection by Setting OPS2 to OPS0 = 100<sub>B</sub> or 101<sub>B</sub>

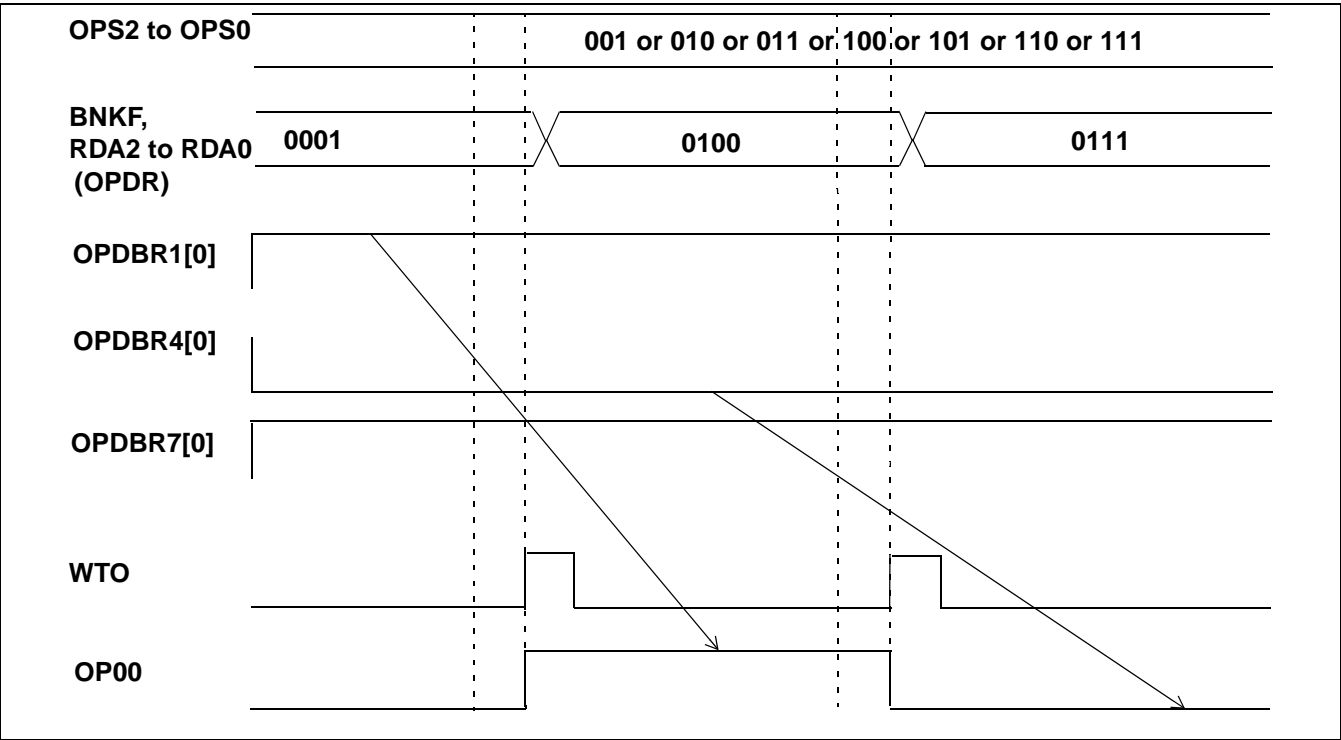
Figure 15.6-10 Signal Flow Diagram for Reload Timer 0 or Position Detect (OPS2 to OPS0 = 100<sub>B</sub> or 101<sub>B</sub>)



At this setting the write signal is generated by the compare match or effective edge input of the position detection or whenever the 16-bit reload timer 0 is underflow. The compare match is triggered by any effective edge change in SNI2 to SNI0 pins.

■ OPDR Register Write Timing Diagram  
(OPS2 to OPS0 = 001<sub>B</sub>, 010<sub>B</sub>, 011<sub>B</sub>, 100<sub>B</sub>, 101<sub>B</sub>, 110<sub>B</sub>, 111<sub>B</sub>)

Figure 15.6-11 OPDR Register Write Timing Diagram  
(OPS2 to OPS0 = 001<sub>B</sub>, 010<sub>B</sub>, 011<sub>B</sub>, 100<sub>B</sub>, 101<sub>B</sub>, 110<sub>B</sub>, 111<sub>B</sub>)



### 15.6.3 Operation of Output Data Buffer Register

The Output Data Buffer Register (OPDBR) is composed of twelve registers. By loading different OPDBR register into the Output Data Register (OPDR), various kind of waveform is output at the Multi-pulse Generator Output (OPT5 to OPT0).

#### ■ Operation of Output Data Buffer Register

The data in the Output Data Buffer Register (OPDBR) whose address specified by the BNKF, RDA2 to RDA0 bits is transferred to the Output Data Register (OPDR) at the write timing generated by the Data Write Control Unit.

The BNKF, RDA2 to RDA0 bits of the Output Data Buffer Register (OPDBR) decide the order of data transfer to the Output Data Register (OPDR), and the OPx1/OPx0 bits decide the shape of the output waveform. The output waveform is updated automatically as long as the write timing (WTO) is generated.

An example of setting the Output Data Buffer Register (OPDBR) is shown in Table 15.6-3.

**Table 15.6-3 Output Data Buffer Register (OPDBR)**

No.	0	1	2	3	4	5	6	7	8	9	A
BNKF	0	0	0	0	0	1	0	X	X	0	1
RDA2	1	1	0	0	1	0	0	X	X	1	0
RDA1	0	0	1	0	1	1	1	X	X	0	1
RDA0	0	1	1	1	0	0	0	X	X	0	1
OP51	0	0	0	1	0	0	0	X	X	0	0
OP50	0	0	1	1	0	0	0	X	X	0	1
OP41	1	0	0	0	0	1	0	X	X	0	0
OP40	1	1	0	0	0	1	0	X	X	1	0
OP31	0	0	0	0	0	0	1	X	X	0	0
OP30	0	0	0	0	1	0	1	X	X	0	0
OP21	0	0	0	0	1	0	0	X	X	0	0
OP20	1	0	0	0	1	1	0	X	X	0	0
OP11	0	0	1	0	0	0	0	X	X	0	1
OP10	0	0	1	0	0	0	1	X	X	0	1
OP01	0	1	0	0	0	0	0	X	X	1	0
OP00	0	1	0	1	0	0	0	X	X	1	0
OPBDR No. Sequence	4	5	3	1	6	A	2	X	X	4	B
OPT5 Output	L	L	PPG	H	L	L	L	X	X	L	PPG
OPT4 Output	H	PPG	L	L	L	H	L	X	X	PPG	L
OPT3 Output	L	L	L	L	PPG	L	H	X	X	L	L
OPT2 Output	PPG	L	L	L	H	PPG	L	X	X	L	L
OPT1 Output	L	L	H	L	L	L	PPG	X	X	L	H
OPT0 Output	L	H	L	PPG	L	L	L	X	X	H	L



Setting the Output Data Buffer Register 0 (OPDBR0) (No. 0) as shown in Table 15.6-3 initializes the value of the Output Data Register (OPDR). The following sequence begins to operate according to the write timing generated:

No. 4 -> No. 6 -> No. 2 -> No. 3 -> No. 1 -> No. 5 -> No. A -> No. B -> No. 9 -> No. 4 and recycle.

The data is transferred to the Output Data Register (OPDR) sequentially. The Output Data Buffer Register (OPDBR) is not used if it is not set, e.g. No. 7 and No. 8 in Table 15.6-3.

## 15.6.4 Operation of Data Transfer of Output Data Register

Eight methods can be used to transfer data from the Output Data Buffer Register (OPDBR) to the Output Data Register (OPDR) automatically, which are described in the following paragraphs. Each method is selected by setting the OPS2 to OPS0 bits in the Output Control Register (OPCR).

### ■ Operation of Data Transfer of Output Data Register

There are eight methods of data transfer from Output Data Buffer Registers (OPDBRB to OPDBR0) to the Output Data Register (OPDR):

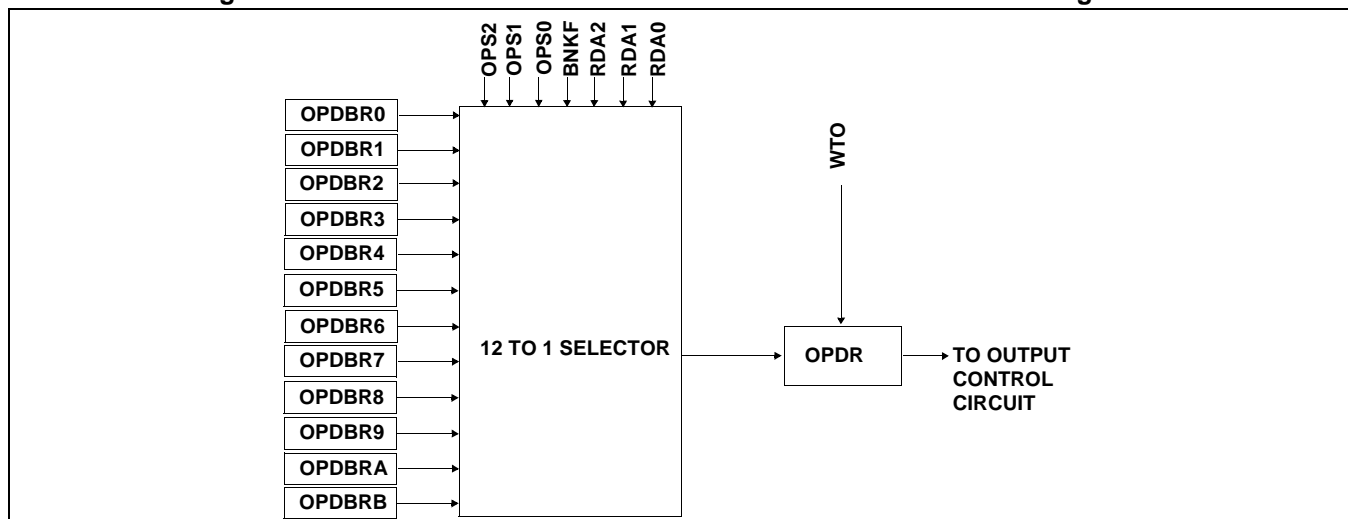
- OPDBR0 Write
- 16-bit Reload Timer Underflow
- Position Detection
- Position Detection and 16-bit Reload Timer Underflow
- Position Detection or 16-bit Reload Timer Underflow
- One-shot Position Detection
- One-shot Position Detection and 16-bit Reload Timer Underflow
- One-shot Position Detection or 16-bit Reload Timer Underflow

The value of the Output Data Buffer Register (OPDBR) which is selected by the BNKF, RDA2 to RDA0 bits in Output Data Register (OPDR), is transferred to the Output Data Register (OPDR) when the write signal is generated from the Data Write Control Circuit. However, at the time when OPS2 to OPS0 = 000<sub>B</sub>, the value of OPDBR0 is always transferred to the Output Data Register (OPDR) in spite of the value of BNKF, RDA2 to RDA0 bits. Figure 15.6-2 shows structure between OPDBRB to OPDBR0 registers and OPDR register.

Note:

When the data transfer method is changed, the next Data Buffer Register to be selected is always specified by the BNKF, RDA2 to RDA0 bits in the Data Output Register. This does not apply to the OPDBR0 Write method, in OPDBR0 Write method BNKF, RDA2 to RDA0 bits are ignored. Word access to Output Data Register must be used.

**Figure 15.6-12 Structure between OPDBRB to OPDBR0 and OPDR Registers**



### 15.6.4.1 When OPDBR0 Write

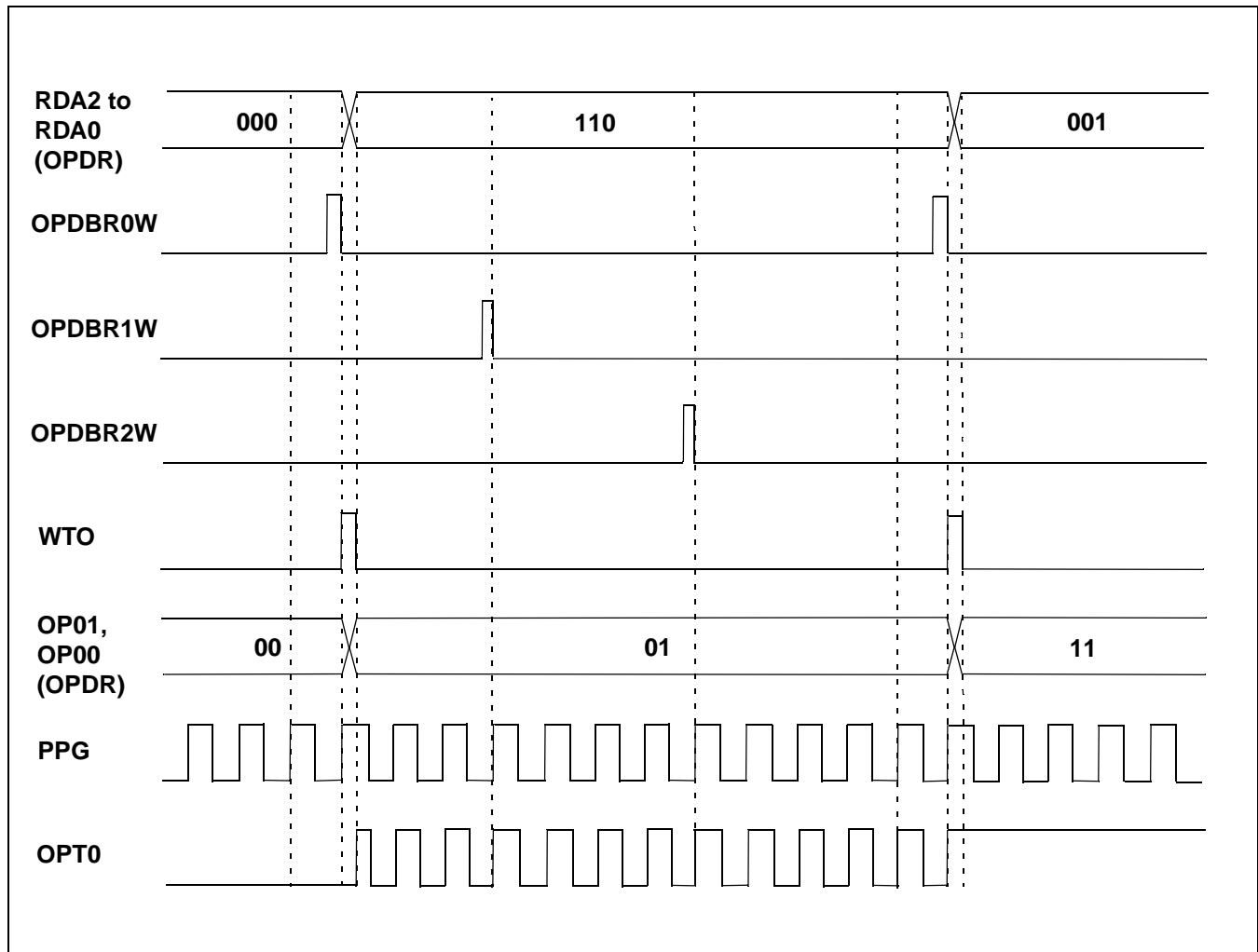
The timing change of the output pin OPTx, which is triggered by the OPDBR0 write, is shown in Figure 15.6-13.

**Note:**

Word access to Output Data Buffer Register 0 must be used in this operation, byte access to either lower register or upper register does not start any transfer operation. The reload timer 0 is free to be used in this operation mode.

#### ■ Timing Generated by OPDBR0 Write (OPS2 to OPS0 = 000<sub>B</sub>)

Figure 15.6-13 Timing Generated by OPDBR0 Write (OPS2 to OPS0 = 000<sub>B</sub>)

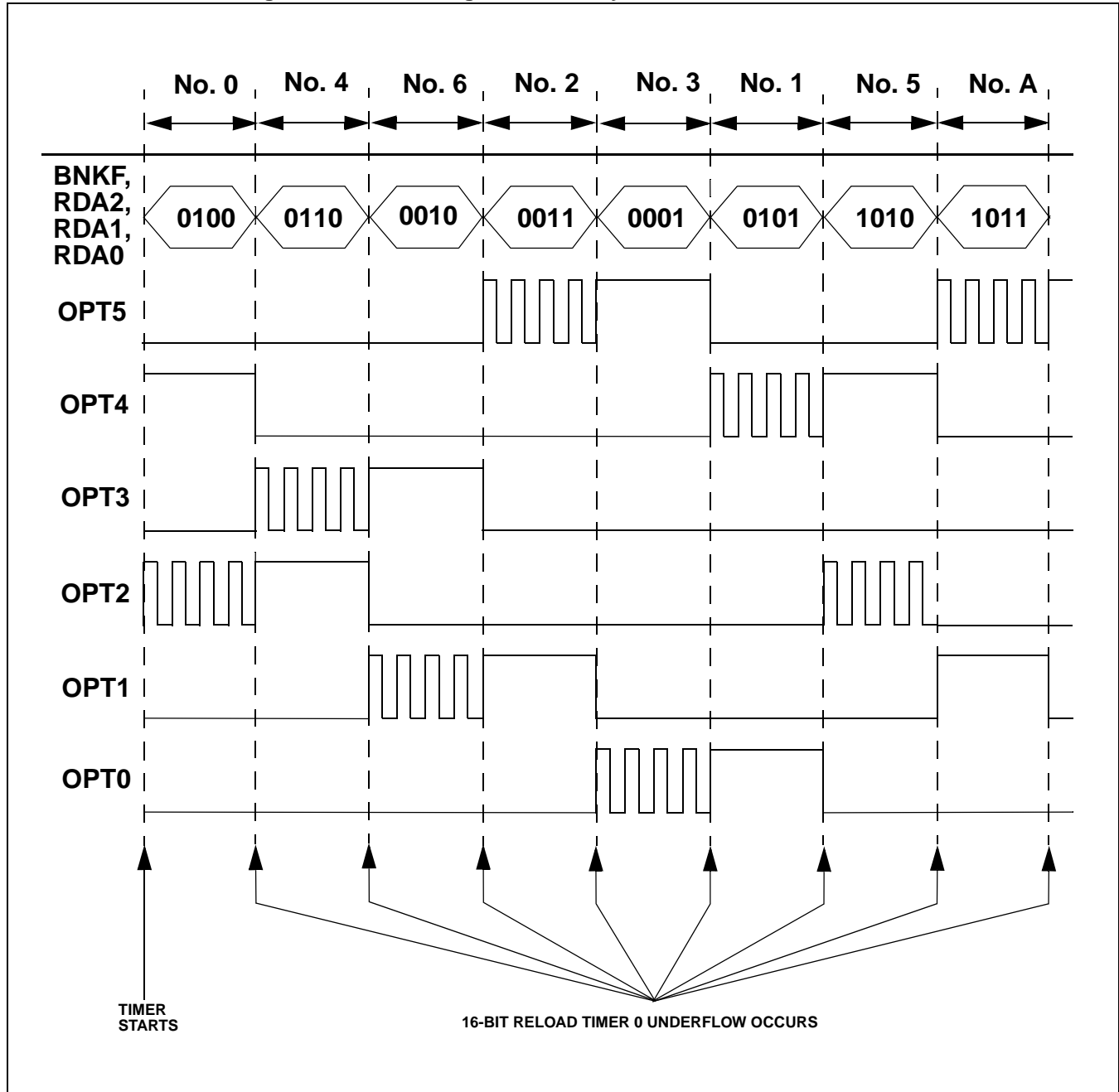


### 15.6.4.2 When 16-bit Reload Timer Underflow

The timing change of the output pin OPTx, which is triggered by the 16-bit reload timer 0 underflow, is shown in Figure 15.6-14 and Figure 15.6-15.

#### ■ Timing Generated by Reload Timer Underflow

Figure 15.6-14 Timing Generated by Reload Timer Underflow

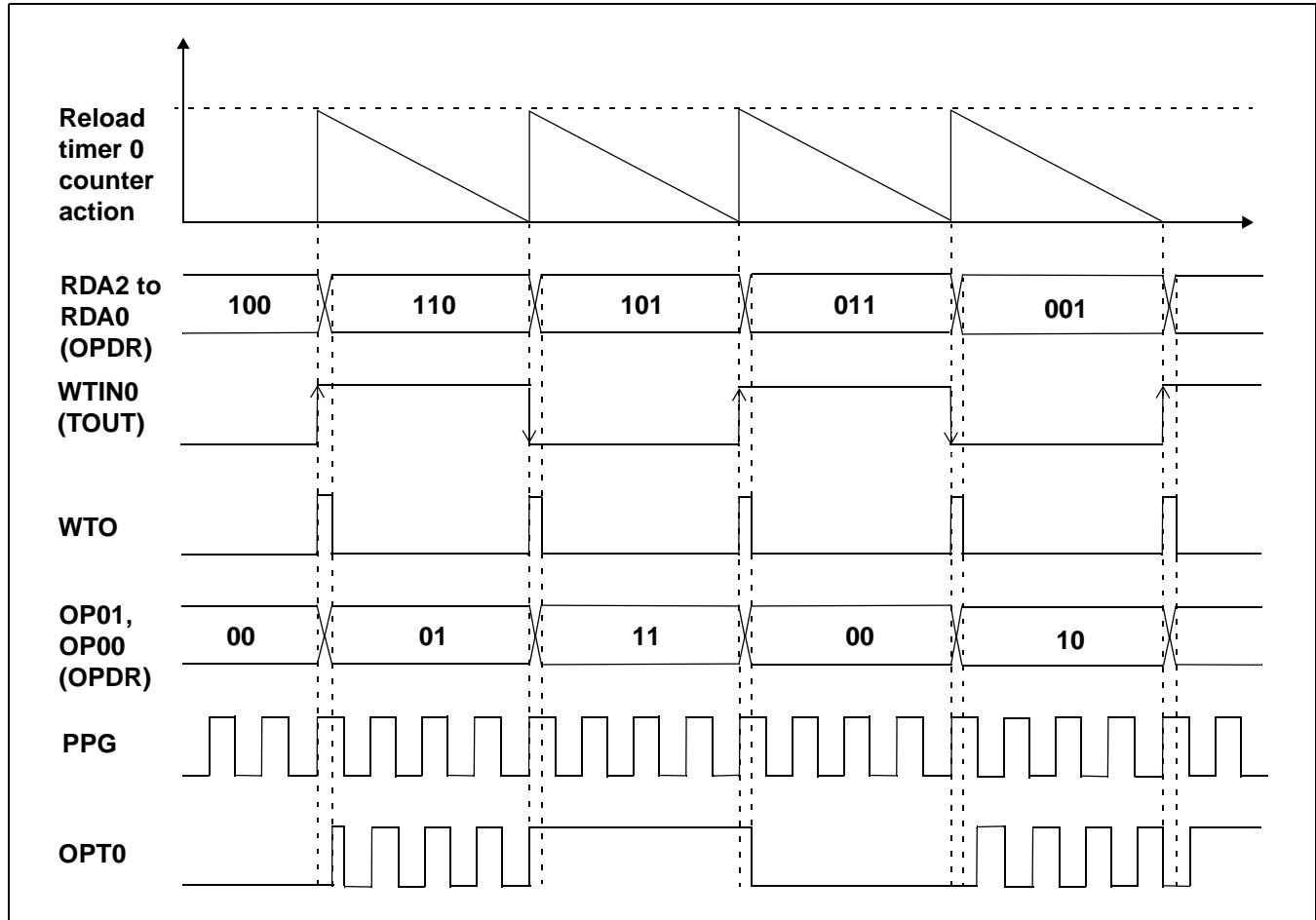


The data transfer from the Output Data Buffer Register (OPDBR) specified by the BNKF, RDA2 to RDA0 bits to the Output Data Register (OPDR) is updated automatically whenever a 16-bit reload timer 0 underflow is generated as shown in Figure 15.6-15.

In order to use this method, the reload timer should be used in "Reload Mode". Software trigger is needed to be used for the startup of the reload timer. The 16-bit reload timer 0 is needed for setting the update time in advance and executing the continuous control action.

### ■ Timing Generated by Reload Timer Underflow (OPS2 to OPS0 = 001<sub>B</sub>)

Figure 15.6-15 Timing Generated by Reload Timer Underflow (OPS2 to OPS0 = 001<sub>B</sub>)

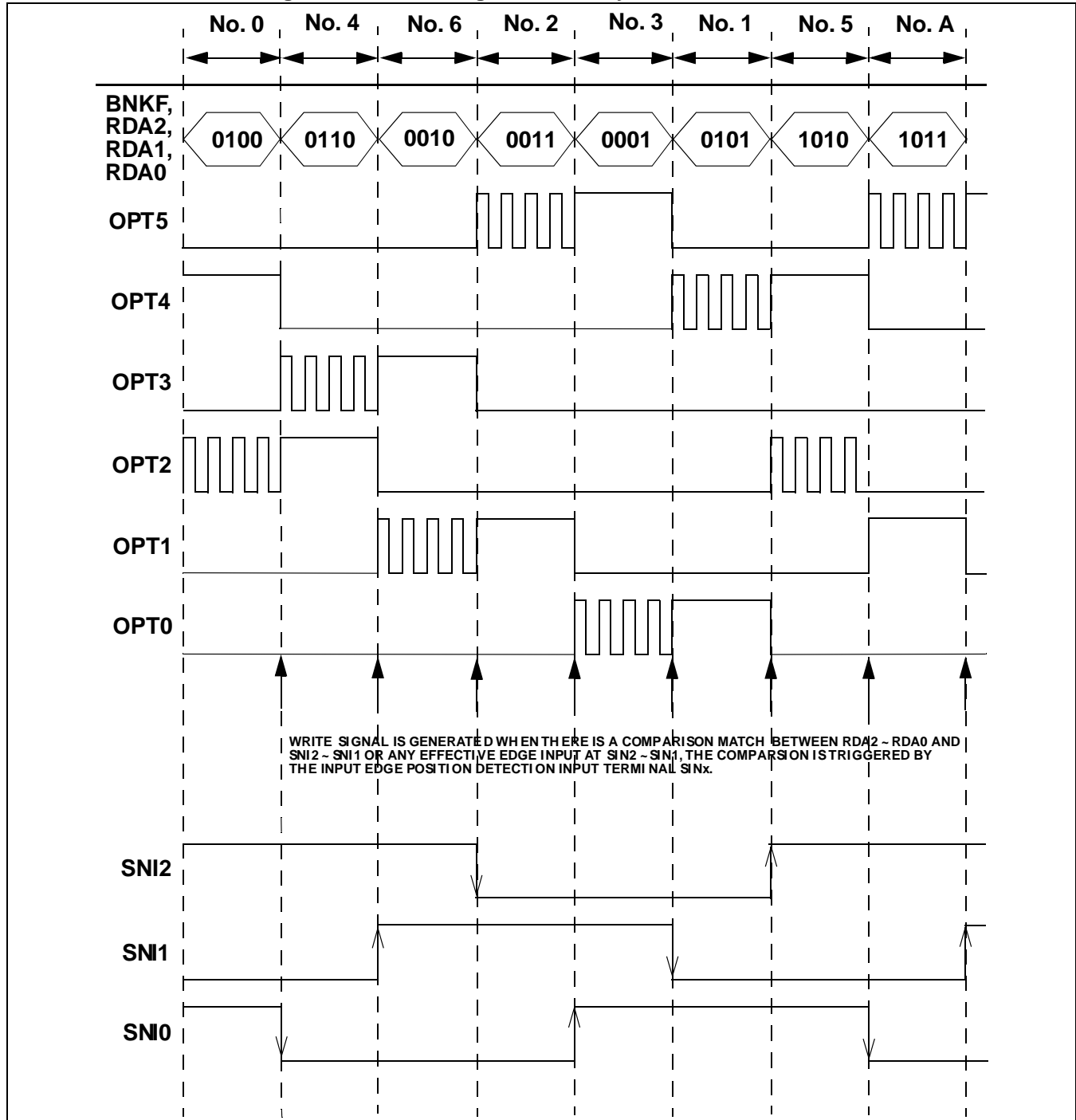


### 15.6.4.3 When Position Detection

The output timing change, which is triggered by the input pin SN<sub>ix</sub> for the position detection, is shown in Figure 15.6-16 and Figure 15.6-17.

#### ■ Timing Generated by Position Detection

Figure 15.6-16 Timing Generated by Position Detection



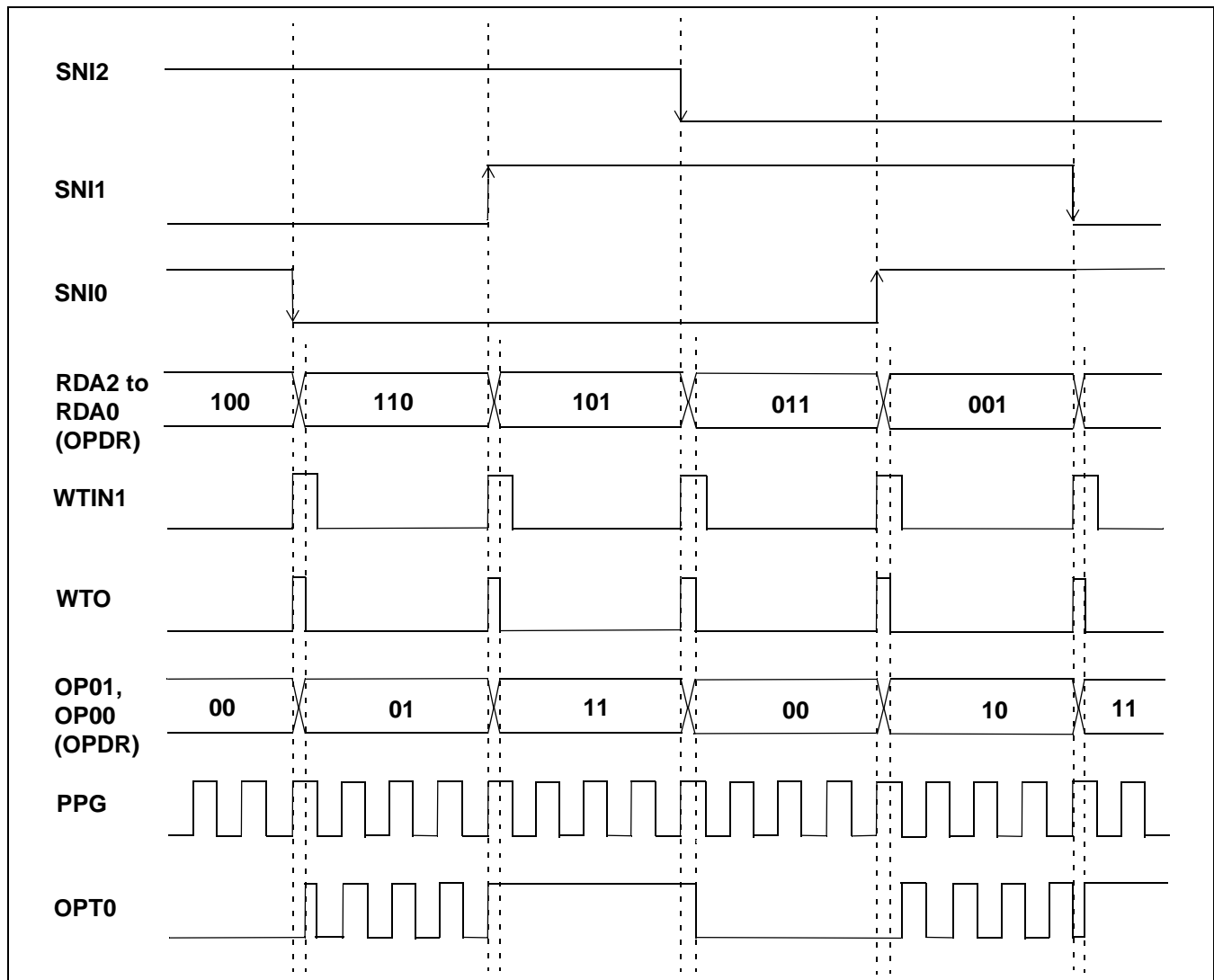
The comparisons between pin SNI2 and RDA2 bit, pin SNI1 and RDA1 bit, pin SNI0 and RDA0 bit are done for each position detection.

The OPTx output waveform is updated according to the effective edge input to pin SNIx as shown in Figure 15.6-17. The data of the Output Data Buffer Register (OPDBR) specified by the BNKF, RDA2 to RDA0 bits is transferred to the Output Data Register (OPDR), and the output data is renewed automatically when pins SNI2 to SNI0 are compared with the value of the RDA2 to RDA0 bits and matches.

The reload timer 0 is free to be used in this operation mode.

### ■ Timing Generated by Position Detection (OPS2 to OPS0 = 010<sub>B</sub>)

Figure 15.6-17 Timing Generated by Position Detection (OPS2 to OPS0 = 010<sub>B</sub>)

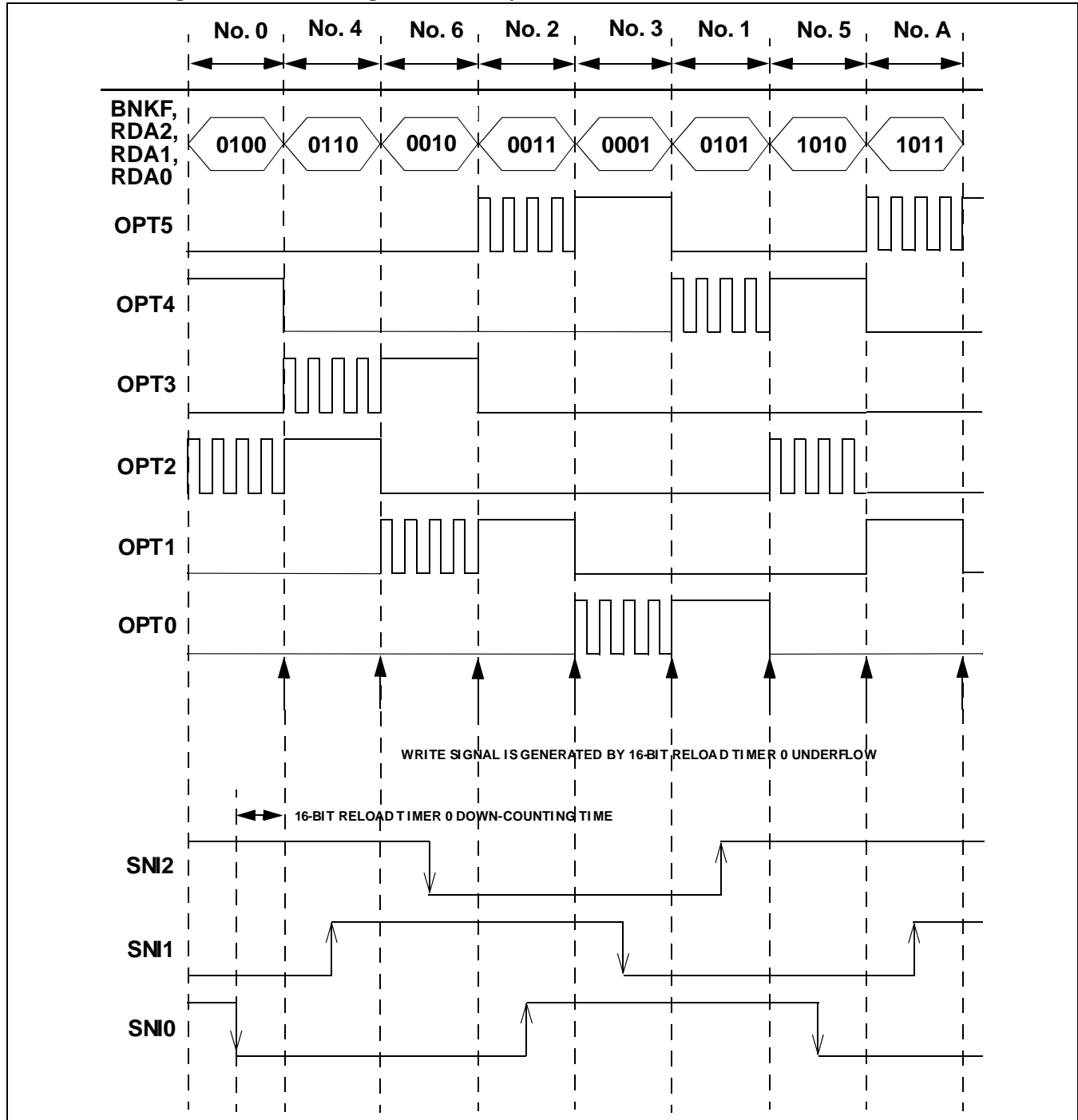


### 15.6.4.4 When Position Detection and Timer Underflow

The output timing change of the operation of the Position Detection and Reload Timer underflow is shown in Figure 15.6-18 and Figure 15.6-19.

#### ■ Timing Generated by Position Detection and Timer Underflow

Figure 15.6-18 Timing Generated by Position Detection and Timer Underflow





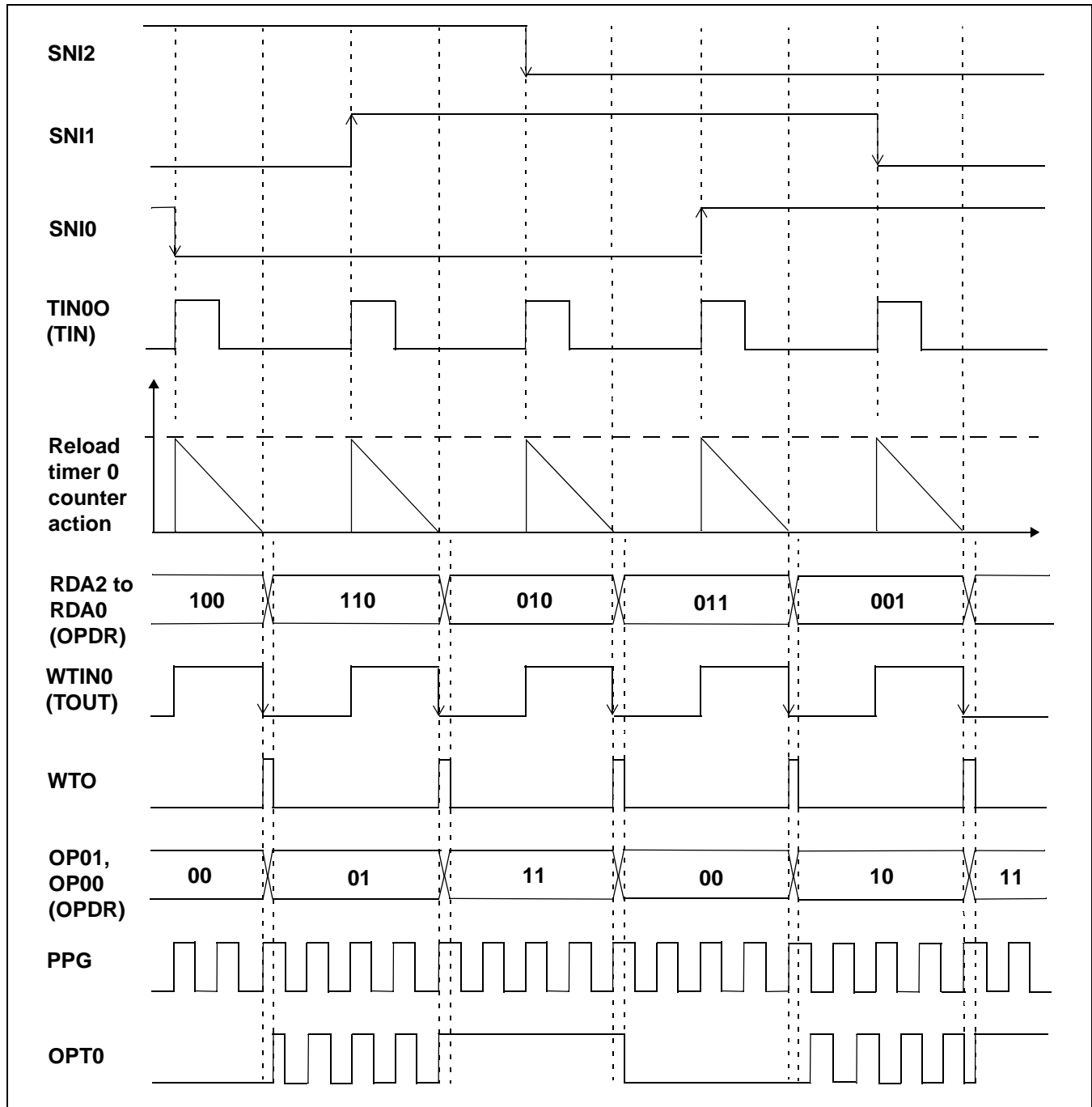
The comparison for the position detection is done in pair for each SNIx pin and RDAx bit (SNI2 and RDA2, SNI1 and RDA1, SNI0 and RDA0), a comparison match starts the 16-bit reload timer 0. The write signal is generated by the 16-bit reload timer 0 underflow.

Pin OPTx output waveform according to the effective edge input to pin SNIx is shown as in Figure 15.6-19. The 16-bit reload timer 0 is started when the pins SNI2 to SNI0 are compared with the value of the RDA2 to RDA0 bits and matches. Data transfer from the Output Data Buffer register (OPDBR) specified by the RDA2 to RDA0 bits to the Output Data Register (OPDR) is triggered by the underflow of the 16-bit reload timer 0. The operation of output data is renewed automatically.

In order to use this method, the reload timer should be used in "Single Shot Mode". TIN00 must be longer than two machine cycles.

# ■ Timing Generated by Position Detection and Timer Underflow (OPS2 to OPS0 = 011<sub>B</sub>)

Figure 15.6-19 Timing Generated by Position Detection and Timer Underflow (OPS2 to OPS0 = 011<sub>B</sub>)

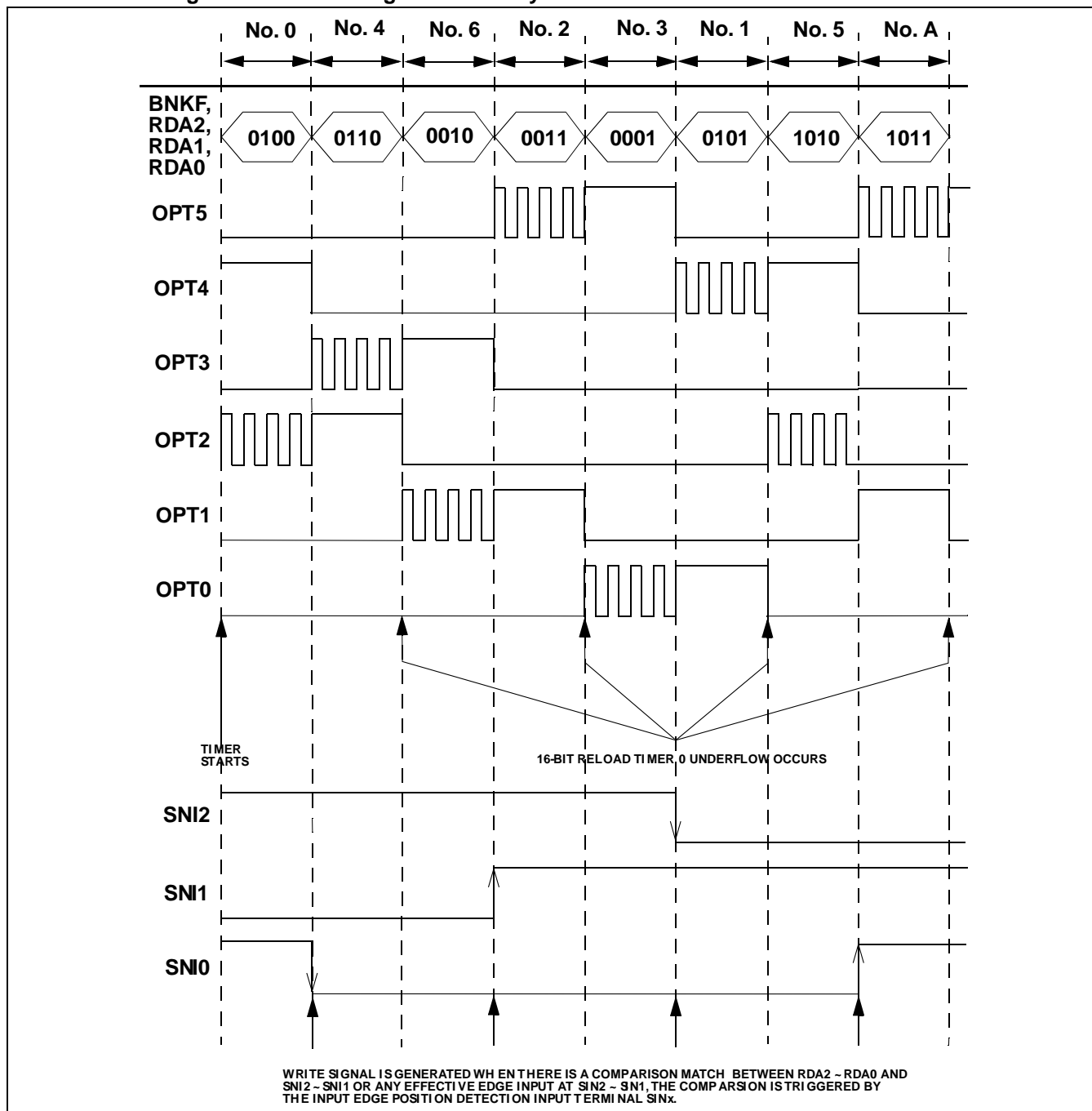


### 15.6.4.5 When Position Detection or Timer Underflow

The output timing changes of the operation of the Position Detection or Reload Timer underflow are shown in Figure 15.6-20 and Figure 15.6-21. This operation mode is selected by setting the OPS2 to OPS0 = 100<sub>B</sub>.

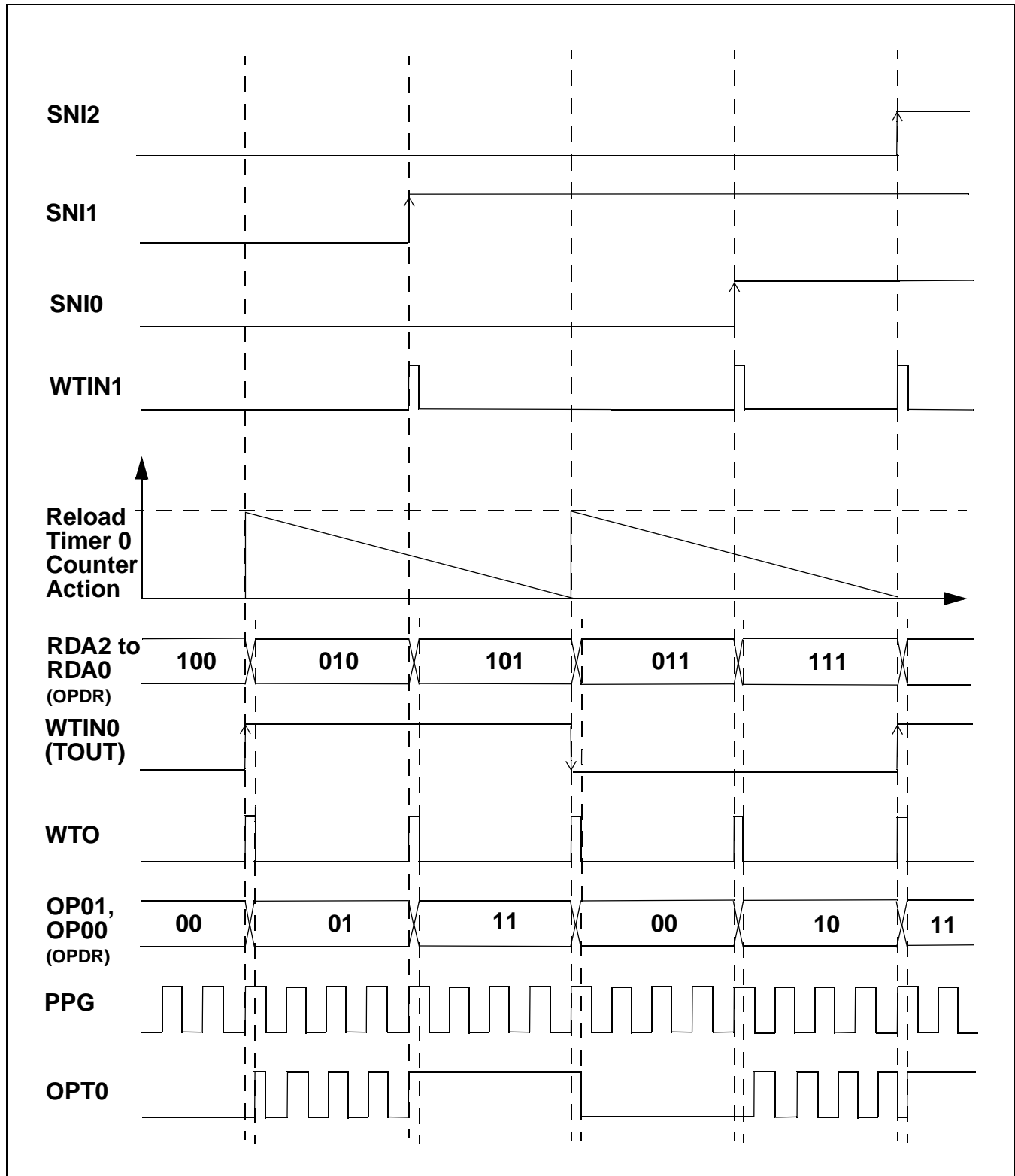
#### ■ Timing Generated by Position Detection or Timer Underflow

Figure 15.6-20 Timing Generated by Position Detection or Timer Underflow



■ Timing Generated by Position Detection or Timer Underflow (OPS2 to OPS0 = 100<sub>B</sub>)

Figure 15.6-21 Timing Generated by Position Detection or Timer Underflow (OPS2 to OPS0 = 100<sub>B</sub>)



### 15.6.4.6 When One-shot Position Detection

The output timing change, which is triggered by the input pin SNIx for the one-shot position detection, is shown in Figure 15.6-22.

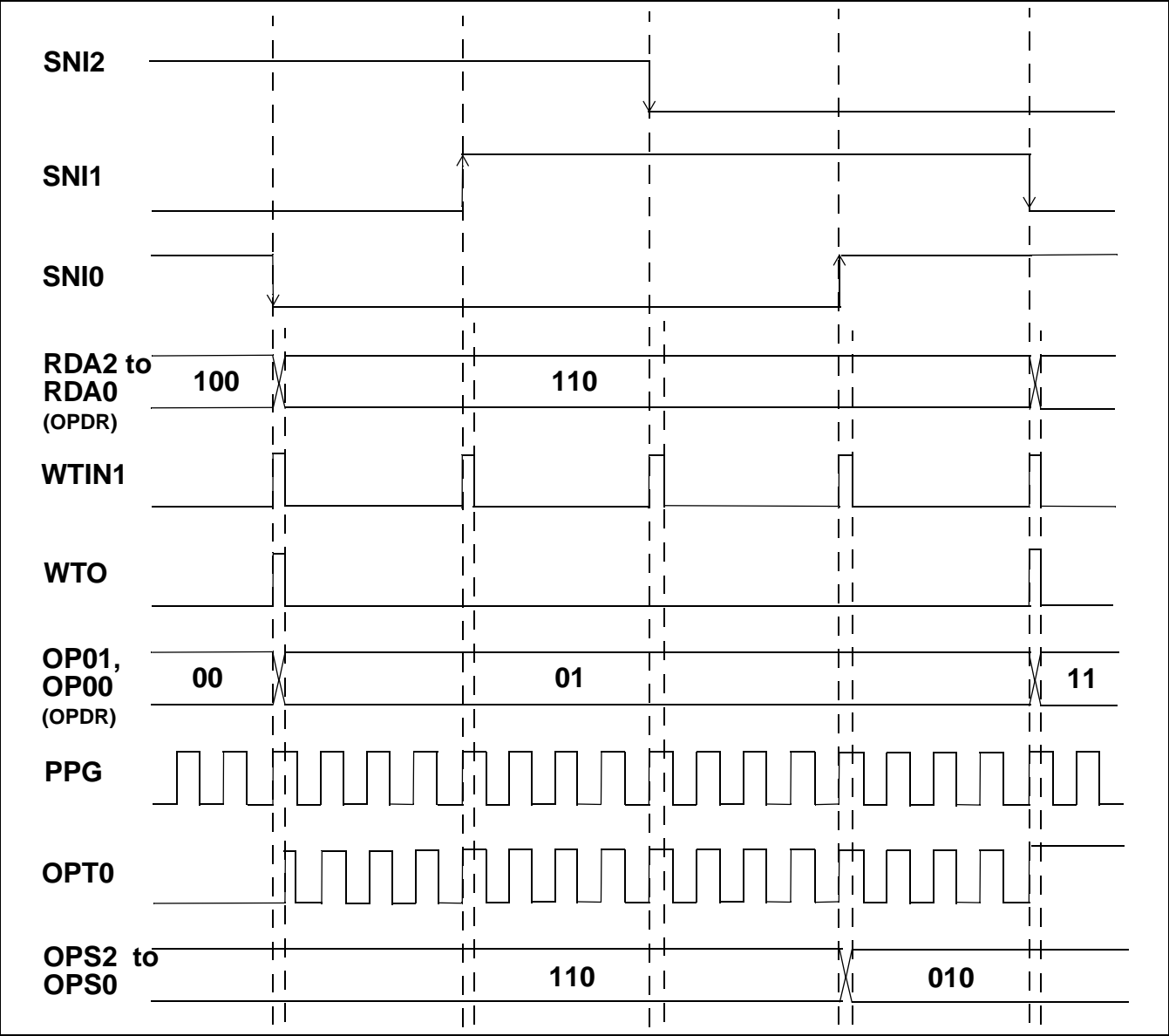
#### ■ When One-shot Position Detection

Same as operation of position detection except that no further position detection will be recognized after the first valid detection until it is changed to ANY OTHER operation mode. The OPTx output waveform is shown in Figure 15.6-22.

The reload timer 0 is free to be used in this operation mode.

#### ■ Timing Generated by One-shot Position Detection (OPS2 to OPS0 = 110<sub>B</sub>)

Figure 15.6-22 Timing Generated by One-shot Position Detection (OPS2 to OPS0 = 110<sub>B</sub>)



### 15.6.4.7 When One-shot Position Detection and Timer Underflow

The output timing change of the operation of the One-shot Position Detection and Reload Timer underflow is shown in Figure 15.6-23.

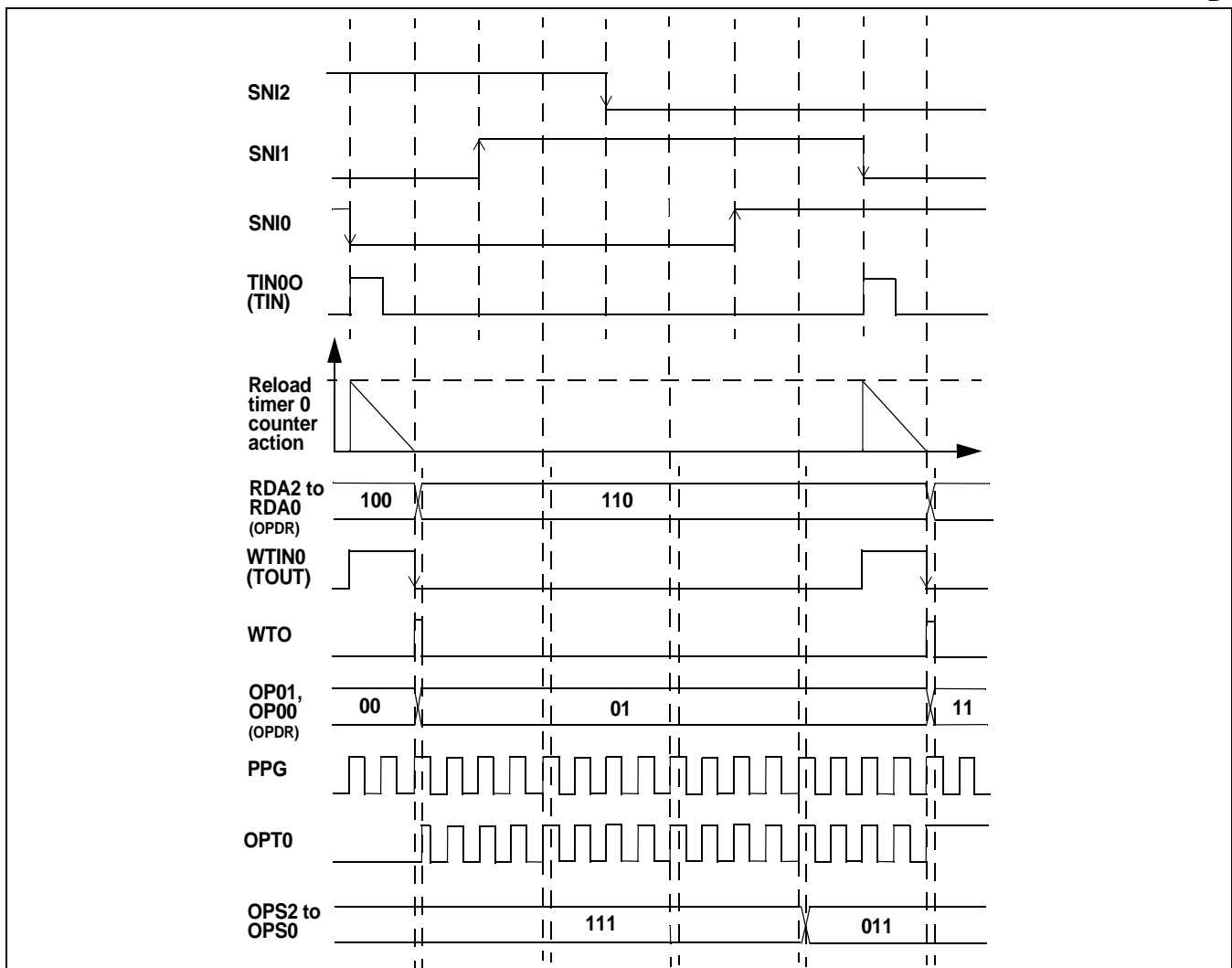
#### ■ When One-shot Position Detection and Timer Underflow

Same as operation of position detection and timer underflow except that no further position detection will be recognized after first valid position detection until it is changed to ANY OTHER operation mode. Pin OPTx output waveform is shown as in Figure 15.6-23.

In order to use this method, the reload timer should be used in "Single Shot Mode". TIN00 must be longer than two machine cycles.

#### ■ Timing Generated by One-shot Position Detection and Timer Underflow (OPS2 to OPS0 = 111<sub>B</sub>)

Figure 15.6-23 Timing Generated by One-shot Position Detection and Timer Underflow (OPS2 to OPS0 = 111<sub>B</sub>)



### 15.6.4.8 When One-shot Position Detection or Timer Underflow

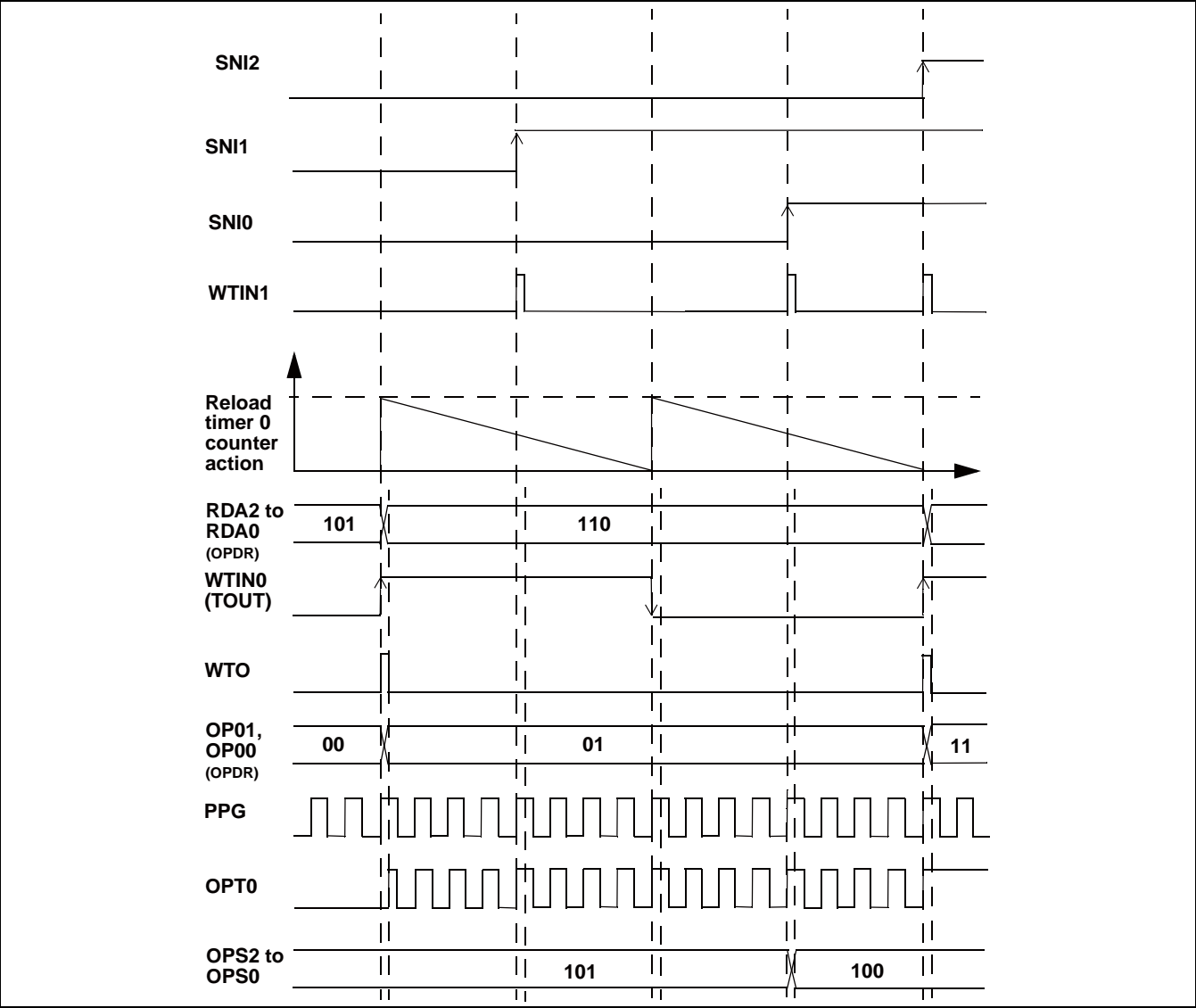
The output timing change of the operation of the One-shot Position Detection or Reload Timer underflow is shown in Figure 15.6-24. This operation mode is selected by setting the OPS2 to OPS0 = 101<sub>B</sub>.

#### ■ When One-shot Position Detection or Timer Underflow

Same as operation of position detection or timer underflow except that no further position detection will be recognized after first valid position detection until it is changed to ANY OTHER operation mode. Pin OPTx output waveform is shown as in Figure 15.6-24.

#### ■ Timing Generated by One-shot Position Detection or Timer Underflow (OPS2 to OPS0 = 101<sub>B</sub>)

Figure 15.6-24 Timing Generated by One-shot Position Detection or Timer Underflow (OPS2 to OPS0 = 101<sub>B</sub>)



## 15.6.5 Operation of DTTI1 Input Control

This section describes the operation of the DTTI1 Input Control Circuit.

### ■ Operation of DTTI1 Input Control

The DTTI1 circuit controls the output of the value of PDRx (PORTx Data Register) to the pin OPTx which is multiplexed with the PORTx where OPTx is enable by setting OPEx = 1. The operation mode is enabled by the DTIE bit (bit15) of the Output Control Register (OPCR).

Note:

Before the DTTI1 circuit is in effect, make sure that the PORTx which is multiplexed with the OPTx is configured as an output port by setting its Data Direction Register.

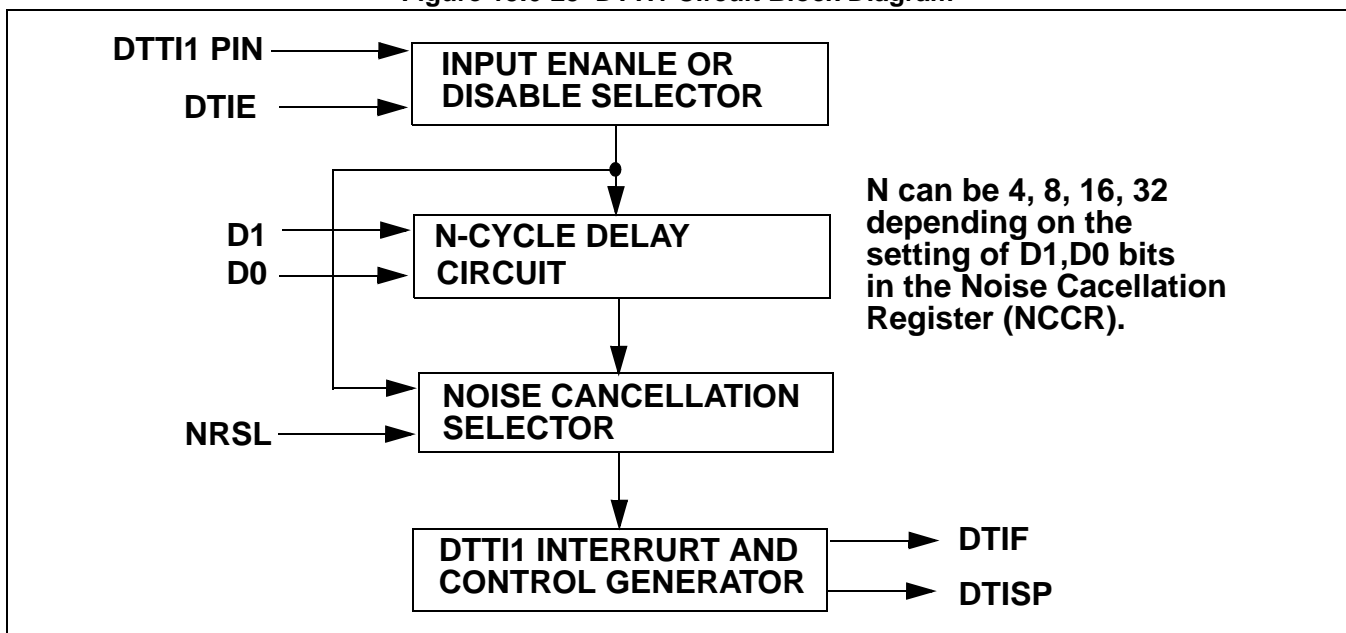
When the DTIE bit (bit14) of the Output Control Register (OPCR) is set to "1", the waveform output at OPT5 to OPT0 pins are enabled by the valid level of the DTTI1 pin. When the low input level is placed at the DTTI1 pin, the output of OPTx is fixed at the inactive level. The software can set the inactive level for each OPTx pin in PDRx of PORTx, the OPTx pin is then driven by the data written in the PDRx of PORTx.

Even while the output is fixed at the inactive level by the input of the DTTI1 pin, the timer keeps running, the position detection function does not stop and the data transfer from the Output Data Buffer Register (OPDBR) to the Output Data Register (OPDR) is continued for waveform generation, but no waveform is outputted to the OPT5 to OPT0 pins.

Figure 15.6-25 shows the DTTI1 circuit block diagram and Figure 15.6-26 shows the DTTI1 circuit timing diagram when D1,D0 is set to "00<sub>B</sub>".

### ■ DTTI1 Circuit Block Diagram

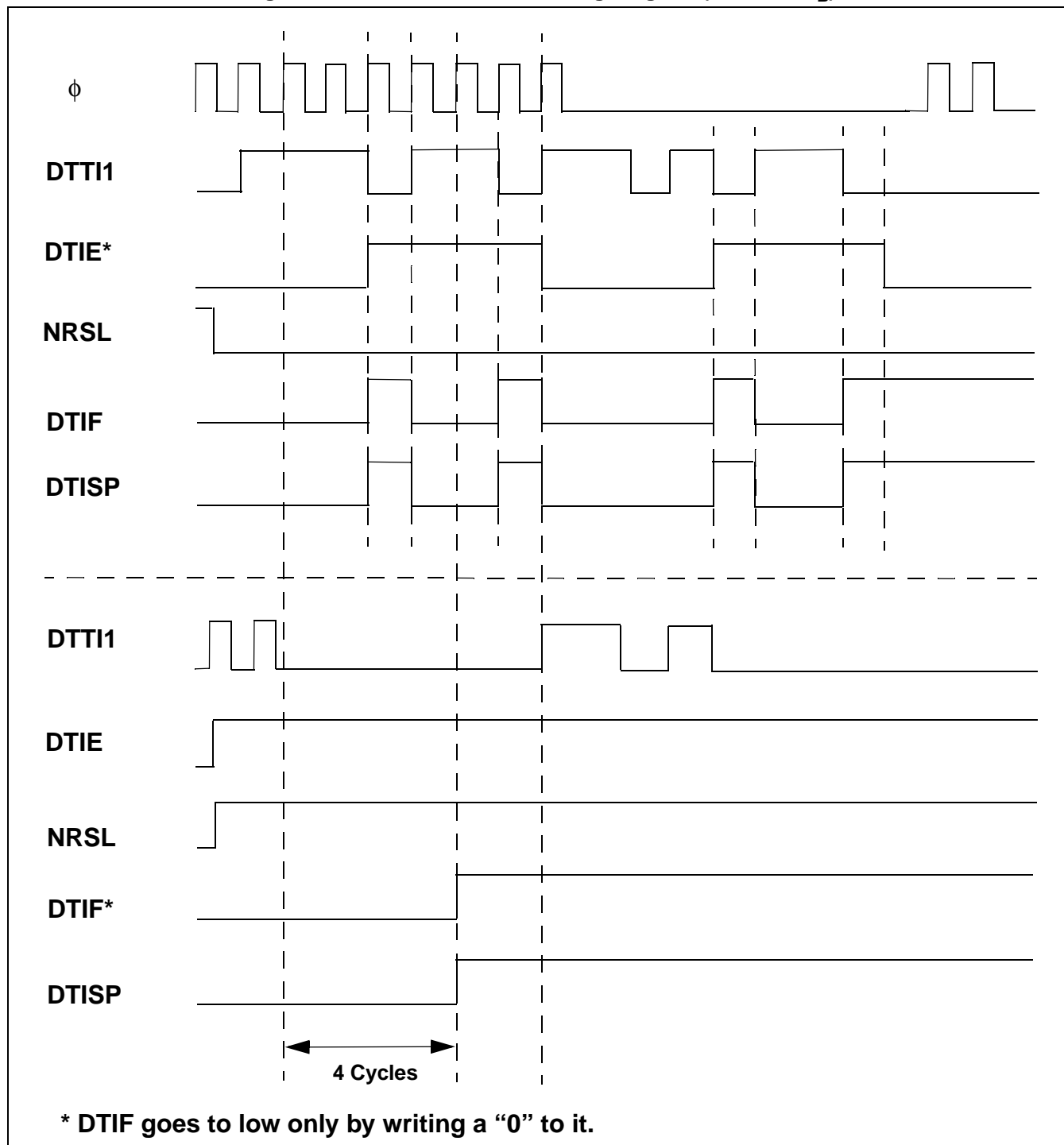
Figure 15.6-25 DTTI1 Circuit Block Diagram





# **■ DTTI1 Circuit Timing Diagram (D1,D0 = 00<sub>B</sub>)**

**Figure 15.6-26 DTTI1 Circuit Timing Diagram (D1,D0 = 00<sub>B</sub>)**



Note:

In worst case the time from DTTI1 being recognized (after noise cancellation) to DTISP in effect takes 2 cycles, in best case it takes 1 cycle.

## ■ Relationship between DTTI1 and OPTx Output

**Table 15.6-4 Relationship between DTTI1 and OPTx Output**

NRSL	DTIE	DTTI1	Function
X	0	X	DTTI1 has no effect on OPTx. (Initial value)
0	1	0	DTTI1 takes effect. Noise filter is not enable. An “L” input at DTTI1 pin triggers the output of the inactive level set in PDRx. The DTTI1 interrupt is generated.
0	1	1	DTTI1 has no effect on OPTx.
1	1	0	DTTI1 takes effect. Noise filter is enable. An “L” input at DTTI1 pin triggers the output of the inactive level set in PDRx. The DTTI1 interrupt is generated.
1	1	1	DTTI1 has no effect on OPTx.

## 15.6.6 Operation of Noise Cancellation Function

---

**This section describes the noise cancellation function for the SNIx and DTTI1 pins.**

---

### ■ Operation of Noise Cancellation Function

#### ● DTTI1 Pin Noise Cancellation Function

When NRSL bit (bit12) of the Output Control Register (OPCR) is set to "1", the noise cancellation function for DTTI1 pin input can be used. When the noise cancellation function is selected, the time for fixing an output pin at the inactive level is delayed for about 4, 8, 16 or 32 machine clocks by the noise cancellation circuit.

---

**Note:**

Since the DTTI1 Input Control Circuit uses a peripheral clock, input is invalidated even if the DTTI1 input is enabled in a mode such as STOP mode in which the oscillator stops.

---

#### ● SNI2 to SNI0 Pins Noise Cancellation Function

When SNC2 to SNC0 bits (bit5 to bit3) of the Input Control Register (IPCR) are set to "1", the noise cancellation function for SNI2 to SNI0 pins input can be used. When the noise cancellation function is selected, the input is delayed for about four machine clocks by the noise cancellation circuit. Since the noise cancellation circuit uses a peripheral clock, input is invalidated in a mode such as STOP mode in which the oscillator stops even if the SNIx input is enabled.

#### ● Programmable Noise Cancellation Circuit

Noise to be cancelled is programmable to have pulse width less than 4, 8, 16 and 32 machine cycles, i.e. for 16 MHz machine clock, the circuit can filter 0.25  $\mu$ s to 2  $\mu$ s width pulses. The control for the programming of the noise cancellation circuit of the SNIx and DTTI1's pins are separated. Figure 15.4-13 shows the noise cancellation control register.

## 15.6.7 Operation of 16-bit Timer

The 16-bit timer has buffer and compare clear function, which is used for motor speed checking and abnormal detection timeout. The 16-bit timer starts counting up from counter value "0000<sub>H</sub>" after a reset has been completed and counting enable bit is set.

### ■ 16-bit Timer Operation

The counter value is cleared in the following conditions:

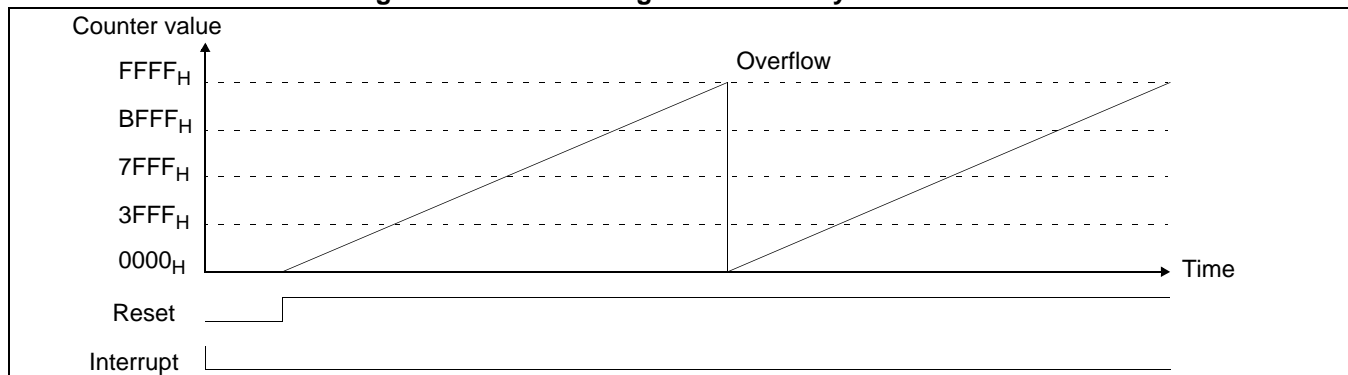
- When an overflow has occurred.
- When a match with Compare Clear Register (CPCR) is detected.
- When "1" is written to the TCLR bit of the TCSR register during operation.
- When a write timing signal is generated and MODE bit of the TCSR is "0".
- When a position detection signal is generated and MODE bit of the TCSR is "1".
- Reset

An interrupt can be generated when the counter is cleared due to a match with Compare Clear Register. There is no interrupt when an overflow occurs.

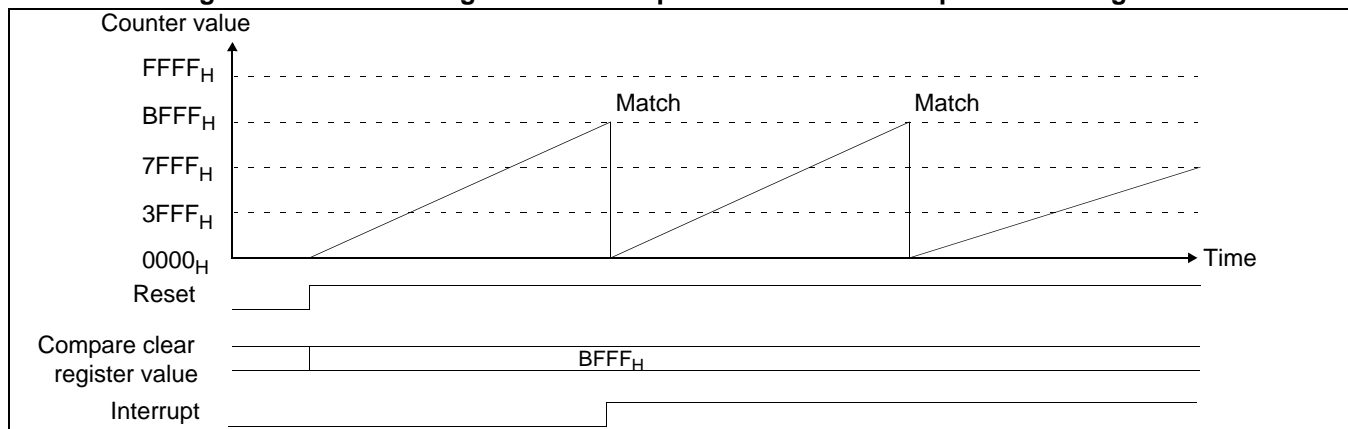
Note:

Word access to Compare Clear Register and Timer Buffer Register must be used.

**Figure 15.6-27 Clearing the Counter by an Overflow**



**Figure 15.6-28 Clearing the Counter upon a Match with Compare Clear Register**

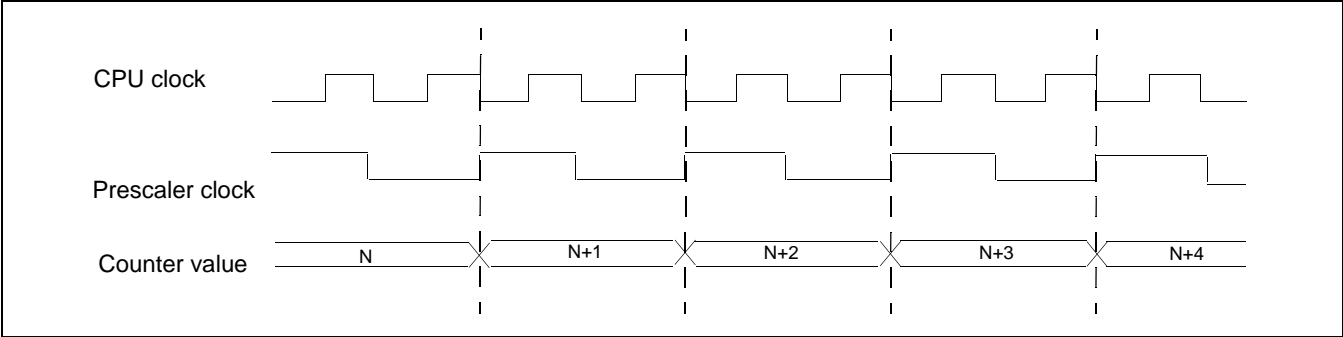


■ 16-bit Timer Timing

The 16-bit timer is incremented based on the prescaler clock and counts up at a rising edge.

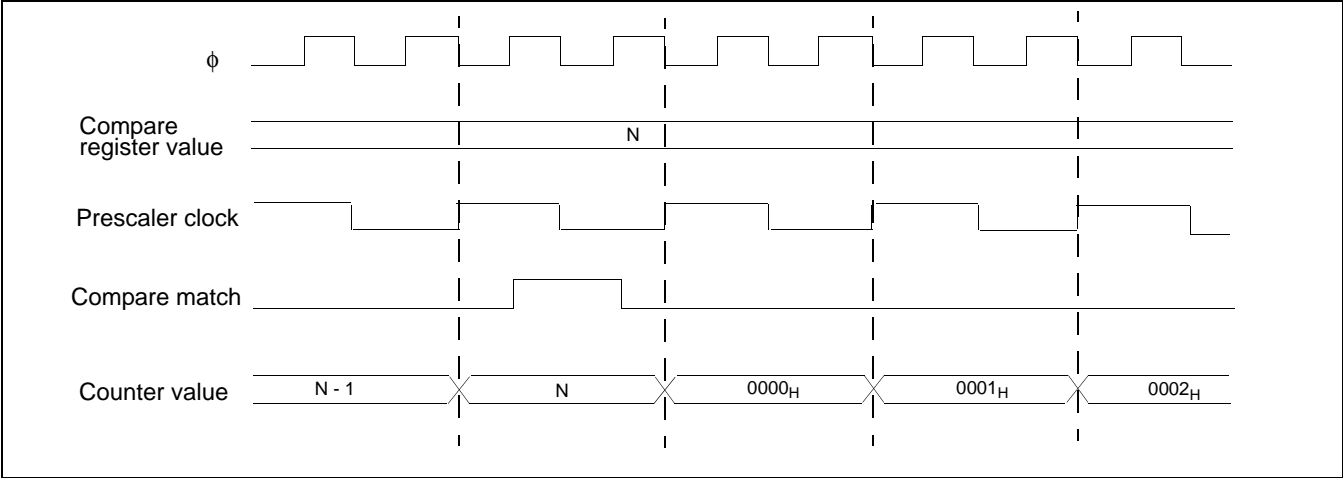
Note: Before the prescaler clock is changed, the Timer Counter should be disabled first by setting the TMEN bit to "0".

Figure 15.6-29 16-bit Timer Count Timing



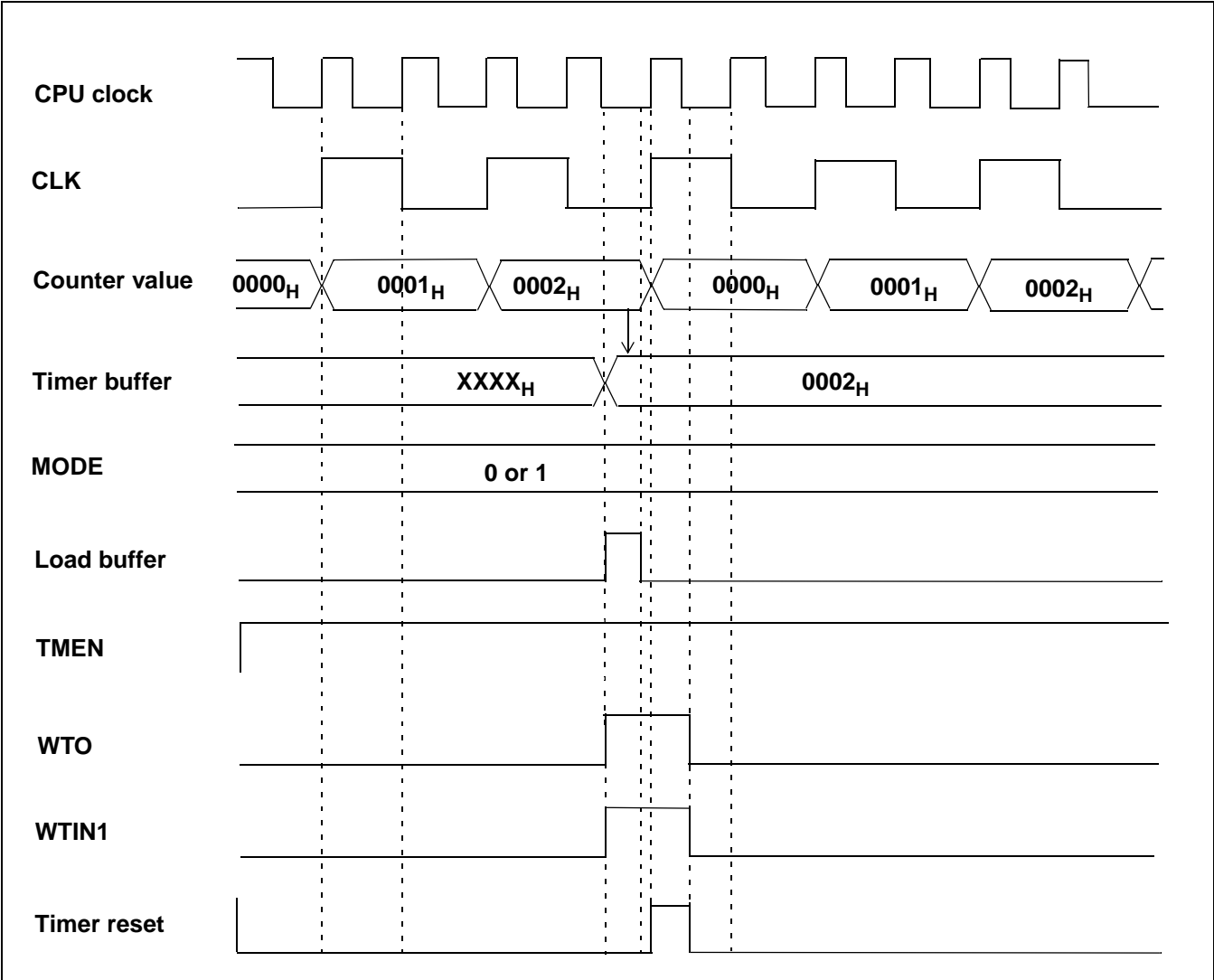
The counter can be cleared upon a reset, software clear (TCLR), a match with Compare Clear Register, the Write Timing signal or the Position Detection signal. By a reset, the counter is immediately cleared. By a match with Compare Clear Register, software clear (TCLR), the Write Timing signal or the Position Detection signal, the counter is cleared in synchronization with the count timing.

Figure 15.6-30 16-bit Timer Clear Timing



■ 16-bit Timer Buffer Operation Timing Diagram

Figure 15.6-31 16-bit Timer Buffer Operation Timing Diagram



## ■ The Use of the 16-bit Timer in Multi-pulse Generator

The timer is reset when write timing or position detection interrupt flag is set, which is selectable by the MODE bit in the Timer Control Status Register (TCSR).

The timer can be started or stopped by setting the TMEN bit in the Timer Control Status Register (TCSR). There is no timer overflow interrupt. Whenever the timer is restarted, the current counter value is latched to a buffer for speed calculation.

If the counter value is matched with Compare Clear Register (CPCR), it interrupts the CPU and the timer is reset.

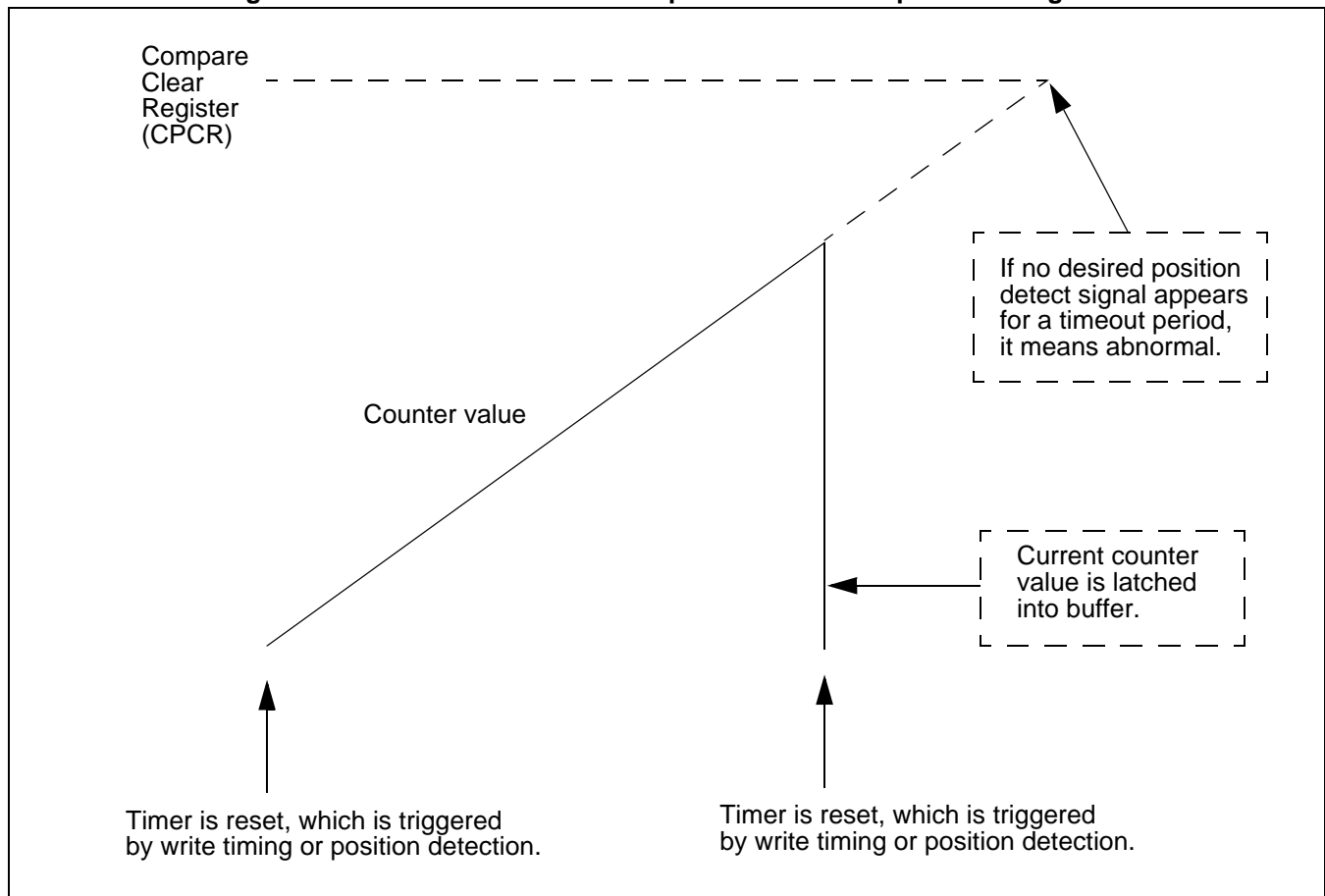
Note:

If the Compare Clear Register (CPCR) is loaded a value same as the Timer Counter value at that moment, the comparison operation will NOT be performed until next same counter value.

The Compare Clear interrupt shares the same interrupt vector with the Write Timing interrupt while Compare Match interrupt shares the same vector as that of the Position Detect interrupt.

## ■ 16-bit Timer in Multi-pulse Generator Operation Diagram

Figure 15.6-32 16-bit Timer in Multi-pulse Generator Operation Diagram



## 15.7 Usage Notes on the Multi-pulse Generator

---

Notes on using the Multi-pulse Generator are given below.

---

### ■ Usage Notes on the Waveform Sequencer

#### ● Notes on using a program for setting

- Switch from one PPG synchronization mode to another PPG synchronization mode (e.g. from rising-edge synchronization (IPCUR: WTS1,WTS0 = 01<sub>B</sub>) to falling-edge synchronization (IPCUR: WTS1,WTS0 = 10<sub>B</sub>) or vice versa) is inhibited, no synchronization mode (IPCUR: WTS1,WTS0 = 00<sub>B</sub>) must be the transit for such switch.
- When the data transfer method is changed, the next data buffer register to be selected is always specified by the BNKF, RDA2 to RDA0 bits in the data output register (OPDR). This does not apply to the OPDBR0 write method (OPCUR: OPS2 to OPS0 = 000<sub>B</sub>), in OPDBR0 write method BNKF, RDA2 to RDA0 bits are ignored.
- Word access to output data register (OPDR) must be used.
- When using OPDBR0 write method for data transfer (OPCUR: OPS2 to OPS0 = 000<sub>B</sub>), word access to output data buffer register 0 must be used, byte access to either lower register or upper register does not start any transfer operation.
- In order to use the 16-bit reload timer underflow transfer method (OPCUR: OPS2 to OPS0 = 010<sub>B</sub>), the reload timer should be used in "Reload Mode". Software trigger is needed to be used for the startup of the reload timer. The 16-bit reload timer is needed for setting the update time in advance and executing the continuous control action.
- In order to use the position detection and timer underflow transfer method (OPCUR: OPS2 to OPS0 = 011<sub>B</sub> or 111<sub>B</sub>), the reload timer should be used in "Single Shot Mode". TIN00 must be longer than two machine cycles.
- Before DTTI1 circuit is in effect (OPCUR: DTIE = 1), make sure that the PORTx which is multiplexed with the OPTx is configured as an output port by setting its data direction register (DDR<sub>x</sub>).
- Since the DTTI1 input control circuit uses a peripheral clock, input is invalidated even if the DTTI1 input is enabled (OPCUR: DTIE = 1) in a mode such as STOP mode in which the oscillator stops.
- In worst case the time from DTTI1 being recognized (after noise cancellation) to DTISP in effect takes 2 cycles, in best case it takes 1 cycle.
- Always change the D1 and D0 bits of noise cancellation control register (NCCR) when the noise cancellation function is disabled (OPCUR: NRSL = 0).
- Always change the S21, S20, S11, S10, S01 and S00 bits of noise cancellation control register (NCCR) when the noise cancellation function is disabled (IPCLR: SNC2 to SNC0 = 000<sub>B</sub>).



### ● Notes about interrupts

- When the DTIF bit of the output control upper register (OPCUR) is set to "1", control cannot be returned from interrupt processing. Always clear the DTIF bit.
- When the WTIF bit of the output control upper register (OPCUR) is set to "1", control cannot be returned from interrupt processing. Always clear the WTIF bit.
- When the PDIF bit of the output control lower register (OPCLR) is set to "1", control cannot be returned from interrupt processing. Always clear the PDIF bit.
- When the CPIF bit of the input control upper register (IPCUR) is set to "1", control cannot be returned from interrupt processing. Always clear the CPIF bit.
- Since the above interrupts share an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by these interrupts, shared resource interrupts must be disabled.

### ■ Usage Notes on the 16-bit Timer

#### ● Notes about using a program for setting

- Word access to compare clear register (CPCR) and timer buffer register (TMBR) must be used.
- Before the prescaler clock is changed, the timer counter should be disabled first by setting the TMEN bit to "0". Change the CLK2, CLK1 and CLK0 bits of the timer control status register (TCSR) only when the timer is not counting.
- If the compare clear register (CPCR) is loaded a value same as the timer counter value at that moment, the comparison operation will NOT be performed until next same counter value.

#### ● Notes about interrupts

- When the ICLR bit of the timer control status register (TCSR) is set to "1" and an interrupt request is enabled (TCSR: ICRE = 1), control cannot be returned from interrupt processing. Always clear the ICLR bit.
- Since the 16-bit timer shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the 16-bit timer, shared resource interrupts must be disabled.

## 15.8 Sample Programs for the Multi-pulse Generator

This section contains sample programs for the Multi-pulse Generator.

### ■ Sample Program for the Multi-pulse Generator

#### ● Processing

- An output in PPG is directed to OPT0 and an inverted output in PPG is directed to OPT1 when write timing interrupt is generated.
- OPDBR0 write method is used for data transfer to output data register OPDR.
- The 16-bit PPG timer is used in PWM and is started with a software trigger.
- EI<sup>2</sup>OS is not used.
- 16 MHz is used for the machine clock, and 62.5 ns is used for the count clock of 16-bit PPG timer.

#### ● Coding example

ICR07	EQU	0000B7H	;Interrupt control register for the waveform sequencer
PCSR1	EQU	000042H	;PPG period setting register
PDUT1	EQU	000044H	;PPG duty setting register
PCNT1	EQU	000046H	;PPG control status register
OPCR	EQU	00008AH	;Output control register
OPDBR0	EQU	003FE0H	;Output data buffer register
WTIF	EQU	OPCR:9	;Interrupt request flag bit

;-----Main program-----

CODE CSEG

START:

;	:		;Assumes that stack pointer (SP) has already been initialized
	AND	CCR,#0BFH	;Interrupt disable
	MOV	I:ICR01,#00H	;Interrupt level 0 (strongest)
	MOVW	I:PCSR0,#0064H	;Sets the period of the PPG output
	MOVW	I:PDUT0,#003CH	;Sets the duty ratio of the PPG output
	MOVW	I:PCNT0,#0110000000000110B	
			;Enables PPG output in normal polarity
			;Enables 16-bit PPG timer, and 62.5 ns clock
			;Software triggers PPG
			;Select PWM mode
			;Clears interrupt flag, and starts counter
	MOVW	I:OPCR,#0103H	;Enable OPT0 and OPT1 output
			;Sets OPDBR0 write method for data transfer

```

; Enable write timing interrupt
; Clears interrupt flag
MOVW      I:OPDBR0,#0009H      ;Sets OPT0 pin as PPG output
;Sets OPT1 pin as inverted PPG output
;Starts data transfer
MOV        ILM,#07H             ;Sets ILM in PS to level 7
OR         CCR,#40H             ;Interrupt enable
LOOP:      MOV        A,#00H     ;Endless loop
MOV        A,#01H               ;
BRA        LOOP                 ;
;-----Interrupt program-----
WARI:
        CLRB          I:WTIF    ;Clears interrupt request flag
;
;
;      User processing
;
;      RETI              ;Returns from interrupt

CODE     ENDS
;-----Vector setting-----
VECT     CSEG          ABS=0FFH
        ORG           0FF94H    ;Sets vector for interrupt #26 (1AH)
        DSL           WARI
        ORG           0FFDCH    ;Sets reset vector
        DSL           START
        DB            00H       ;Sets single-chip mode
VECT     ENDS
        END            START

```

# **CHAPTER 16**

---

## ***PWC Timer***

**This chapter explains the functions and operations of the PWC timer.**

- 16.1 Overview of the PWC Timer
- 16.2 Block Diagram of the PWC Timer
- 16.3 PWC Timer Pins
- 16.4 PWC Timer Registers
- 16.5 PWC Timer Interrupts
- 16.6 Operation of the PWC Timer
- 16.7 Usage Notes on the PWC Timer
- 16.8 Sample Programs for the PWC Timer

## 16.1 Overview of the PWC Timer

---

The PWC timer (pulse-width measurement) is the multi-functional 16-bit up counter with the reload function and also has a function that calculates the pulse width of the input signal.

The PWC timer consists of a 16-bit counter, an input pulse divider, a division rate control register, a count input pin, a pulse output pin, and a 16-bit control register.

---

### ■ PWC Timer (× 2, PWC Timer 0 is not present in MB90465 Series)

The MB90460 series contain two PWC timer channels, while MB90465 series contains only PWC timer 1. The PWC timer has the following characteristics:

#### ● Timer function

- Generates an interrupt request at the specified time interval.
- Outputs the pulse signal that is synchronized with the timer period.
- Selects the counter clock from three internal clocks.

#### ● Pulse-width measurement function

- Generates an interrupt request at the specified time interval.
- Outputs the pulse signal that is synchronized with the timer period.
- Selects the counter clock from three internal clocks.

#### ● Pulse-width measurement function

- Measures the time between external pulse input events.
- Selects the counter clock from three internal clocks.
- Count mode
  - H pulse width (rising edge to falling edge) / L pulse width (falling edge to rising edge)
  - Rising edge period (rising edge to falling edge) / falling edge period (falling edge to rising edge)
  - Intermediate edge count (rising or falling edge to falling or rising edge)
- Uses the 16-bit input divider to divide the input pulse by  $2^2$ ,  $2^4$ ,  $2^6$ , and  $2^8$  to enable period measurement.
- Generates an interrupt request at completion of count.
- Selects single count or continuous count.

#### ● PWC timer operation

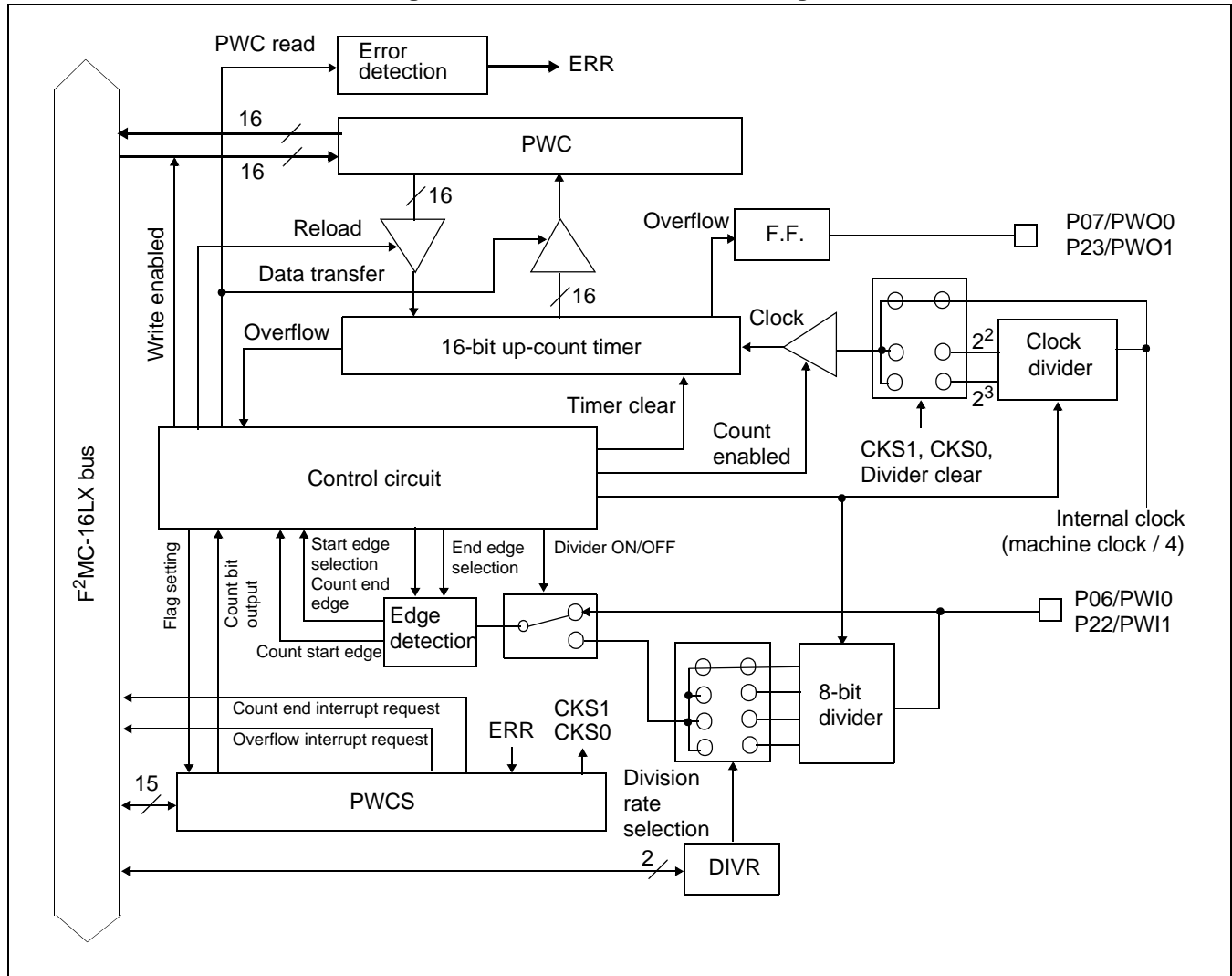
This block is a multi-functional timer that is based on the 16-bit up-count timer and contains a count input pin and an 8-bit input divider. The block has two main functions, a timer function and a pulse-width measurement function, both of which enable the selection of two types of count clocks.

## 16.2 Block Diagram of the PWC Timer

Figure 16.2-1 PWC timer block diagram.

### ■ PWC Timer Block Diagram

Figure 16.2-1 PWC Timer Block Diagram



# 16.3 PWC Timer Pins

This section describes the pins of the PWC timer and provides a pin block diagram.

## ■ PWC Timer Pins

The pins of the PWC timer are shared with the general-purpose ports. Table 16.3-1 lists the functions of the pins, I/O format, and settings required to use the 16-bit reload timer.

Table 16.3-1 16-bit PWC Timer Pins

Pin name	Pin function	I/O format	Pull-up option	Standby control	Settings required for pins
P06/PWI0	Port 0 input-output / timer input	CMOS output / CMOS input	Selection allowed	Available	Setting for the input port (DDR0: bit6 = 0)
P07/PWO0	Port 0 input-output / timer output				Setting for timer enable (PWCSL0: MOD2 to MOD0 not equal 0)
P22/PWI1	Port 2 input-output / timer input	CMOS output / CMOS hysteresis input	Not provided		Setting for the input port (DDR2: bit2 = 0)
P23/PWO1	Port 2 input-output / timer output				Setting for timer enable (PWCSL1: MOD2 to MOD0 not equal 0)

## ■ Block Diagram of the PWC Timer Pins

Figure 16.3-1 shows the block diagram of the PWC timer 0 pins.

Figure 16.3-1 Block Diagram of the PWC Timer 0 Pins

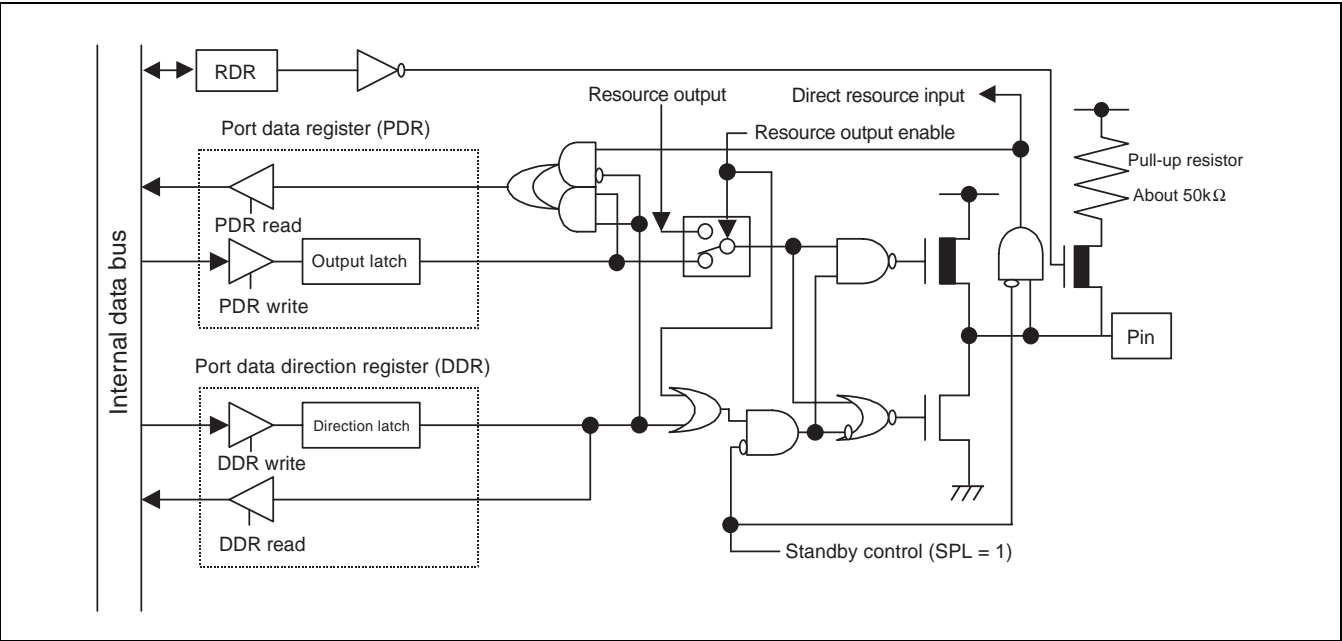
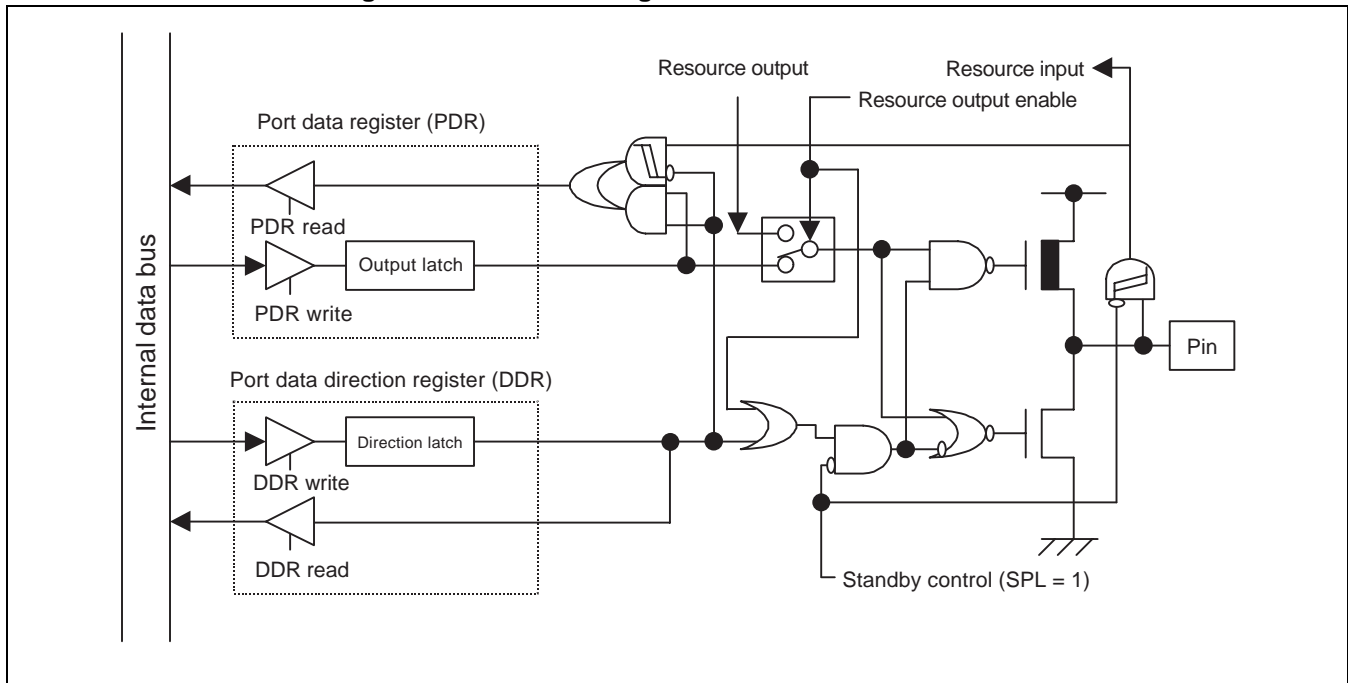


Figure 16.3-2 shows the block diagram of the PWC timer 1 pins.

**Figure 16.3-2 Block Diagram of the PWC Timer 1 Pins**





## 16.4 PWC Timer Registers

Following are the PWC timer registers.

### ■ PWC Timer Registers

Figure 16.4-1 PWC Timer Registers

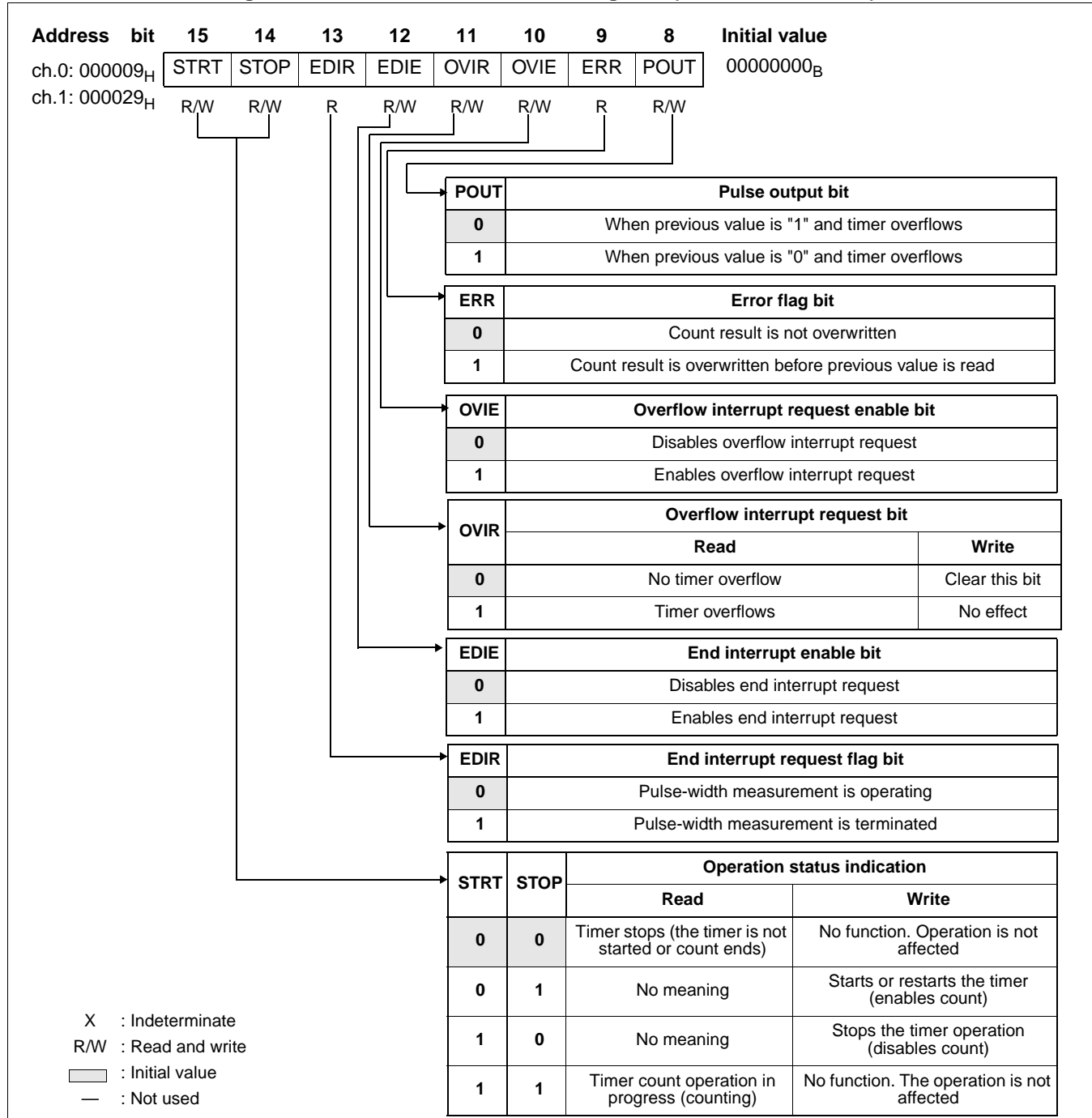
PWC control status register (Upper)									
Address: ch.0 000009 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
ch.1 000029 <sub>H</sub>	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	POUT	PWCSH0, PWCSH1
Read/write ⇨	R/W	R/W	R	R/W	R/W	R/W	R	R/W	
Initial value ⇨	0	0	0	0	0	0	0	0	
PWC control status register (Lower)									
Address: ch.0 000008 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	
ch.1 000028 <sub>H</sub>	CKS1	CKS0	Reserved	Reserved	S/C	MOD2	MOD1	MOD0	PWCSL0, PWCSL1
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	
PWC data buffer register (Upper)									
Address: ch.0 00000B <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
ch.1 00002B <sub>H</sub>	PW15	PW14	PW13	PW12	PW11	PW10	PW09	PW08	PWC0,PWC1
Read/write ⇨	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨	X	X	X	X	X	X	X	X	
PWC data buffer register (Lower)									
Address: ch.0 00000A <sub>H</sub>	bit 7	6	5	4	3	2	1	0	
ch.1 00002A <sub>H</sub>	PW07	PW06	PW05	PW04	PW03	PW02	PW01	PW00	PWC0,PWC1
Read/write ⇨	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨	X	X	X	X	X	X	X	X	
Division rate control register									
Address: ch.0 00000C <sub>H</sub>	bit 7	6	5	4	3	2	1	0	
ch.1 00002C <sub>H</sub>	—	—	—	—	—	—	DIV1	DIV0	DIV0,DIV1
Read/write ⇨	—	—	—	—	—	—	R/W	R/W	
Initial value ⇨	—	—	—	—	—	—	0	0	

## 16.4.1 PWC Control Status Register (PWCSH0/PWCSH1, PWCSL0/PWCSL1)

The PWC control status register (PWCSH0/PWCSH1, PWCSL0/PWCSL1) controls the PWC timer operation and reads the PWC timer state.

### ■ PWC Control Status Register, Upper Byte (PWCSH0/PWCSH1)

Figure 16.4-2 PWC Control Status Register (PWCSH0/PWCSH1)

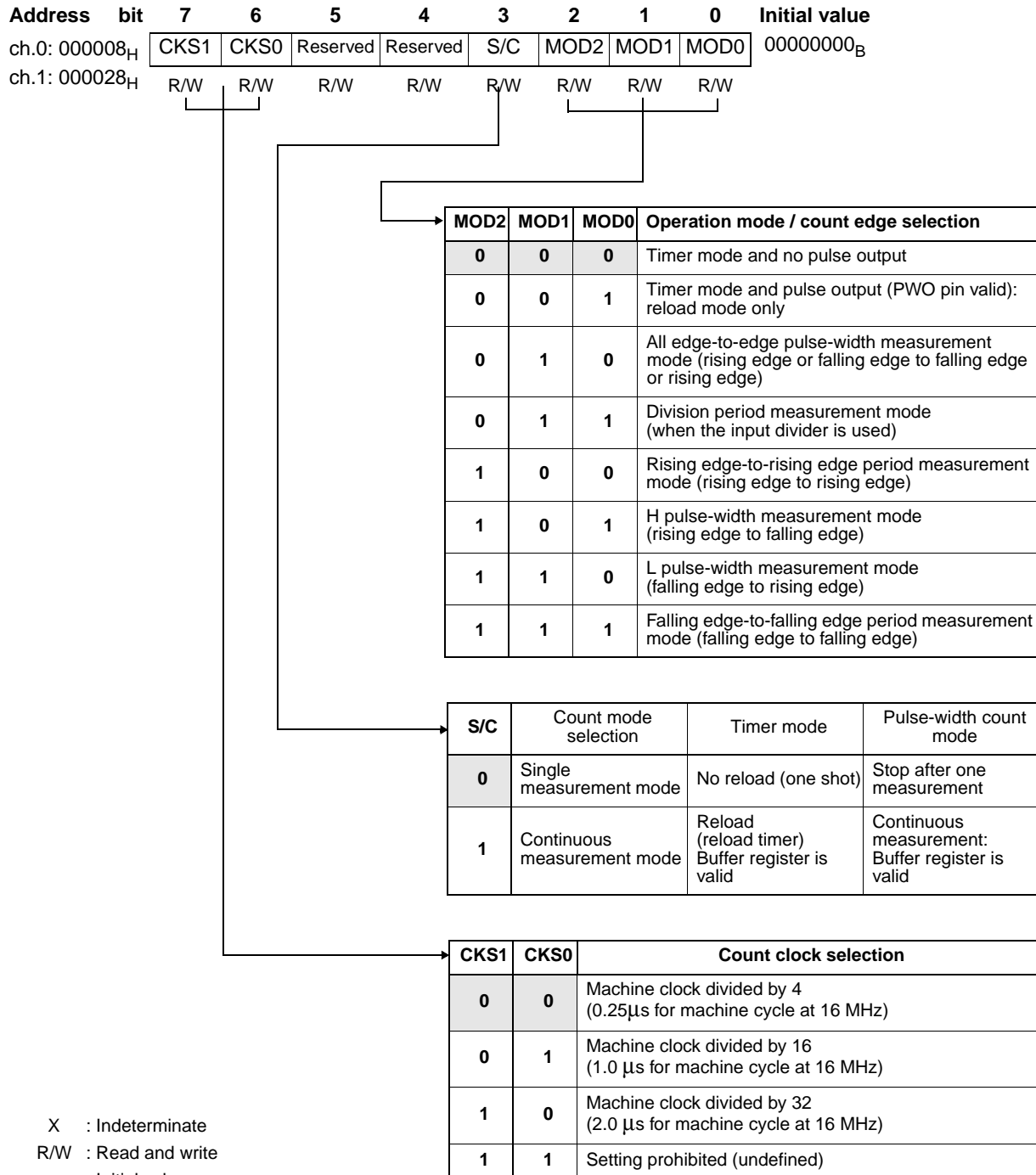


**Table 16.4-1 PWC Control Status Register (PWCSH0/PWCSH1)**

Bit name		Function
bit15, bit14	STRT, STOP: Start and Stop bits	<ul style="list-style-type: none"> <li>These bits are used to start, restart, and stop the 16-bit up-count timer.</li> <li>When these bits are read, the timer operation status is returned.</li> <li>These bits can be read and written. The meaning of bits depend on whether they are read or written.</li> <li>In read-modify-write operation, “11<sub>B</sub>” is always read.</li> <li>When the STRT and STOP bits are written to start and stop the timer, a bit manipulation instruction (such as bit clear instruction) can be used. However, when the operation status (which always indicates that the timer is operating, for example) is read, a bit manipulation instruction cannot be used.</li> </ul>
bit13	EDIR: End interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit indicates that measurement terminated in pulse-width measurement mode.</li> <li>When pulse-width measurement terminates, the bit is set (PWC0/PWC1 contains the measurement result).</li> <li>This bit is cleared automatically when the measurement result in PWC data buffer register, PWC0/PWC1, is read.</li> <li>In timer mode, this bit is meaningless.</li> <li>This bit is read-only, writing this bit is meaningless.</li> </ul>
bit12	EDIE: End interrupt enable bit	<ul style="list-style-type: none"> <li>This bit is used to control a measurement termination interrupt request in pulse-width count mode.</li> <li>When this bit is “1” and EDIR is set to “1”, the end interrupt request will be generated to CPU.</li> <li>Always set “0” in timer mode.</li> </ul>
bit11	OVIR: Overflow interrupt request bit	<ul style="list-style-type: none"> <li>This bit is used to specify when the 16-bit up-count timer overflows. The operation affects all modes.</li> <li>When timer overflow occurs (FFFF<sub>H</sub> to 0000<sub>H</sub>), the bit is set.</li> <li>Writing “0” will clear the bit.</li> <li>Writing “1” has no effect.</li> <li>In read-modify-write operation, “1” is always read.</li> </ul> <p>(Note) In H/L pulse-width count mode, do not use this bit for pulse-width time measurement.</p>
bit10	OVIE: Overflow interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable timer overflow interrupt request.</li> <li>When this bit is “1” and OVIR is set to “1”, the overflow interrupt request will be generated to CPU.</li> </ul> <p>(Note) In the H/L pulse-width count mode, set this bit to “0”.</p>
bit9	ERR: Error flag bit	<ul style="list-style-type: none"> <li>This bit is used to execute a continuous count in the pulse-width count mode. This flag indicates that the next count has been completed before the previous count result is read from PWC0/PWC1. If this state occurs, PWC0/PWC1 is overwritten by new count result and the previous result is lost. The count operation continues regardless of the value of this bit.</li> <li>The bit is read-only. Writing to this bit is meaningless.</li> <li>When the count result that has not been read is overwritten by the next result, the bit is set.</li> <li>This bit is cleared automatically when the measurement result in PWC data buffer register, PWC0/PWC1, is read.</li> </ul>
bit8	POUT: Pulse output bit	<ul style="list-style-type: none"> <li>When the 16-bit up-count timer overflows in timer mode, this bit is reversed.</li> <li>In the pulse-width count mode, this bit is meaningless.</li> <li>The bit can be read and written. However, the bit can be written only if the timer stops (both bit15: STRT and bit14: STOP are set to “0”). If the bit is written during timer operation (both bit15: STRT and bit14: STOP are set to “1”), the bit value remains unchanged.</li> <li>When the POUT value is “0” and the timer overflows in the range from FFFF<sub>H</sub> to 0000<sub>H</sub> or the timer stops and “1” is written, the bit is set.</li> <li>When the POUT value is “1” and the timer overflows in the range from FFFF<sub>H</sub> to 0000<sub>H</sub> or the timer stops and “0” is written, the bit is cleared. The bit is also cleared by reset.</li> </ul>

## ■ PWC control Status Register, Lower Byte (PWCSL0/PWCSL1)

Figure 16.4-3 PWC Control Status Register (PWCSL0/PWCSL1)



**Table 16.4-2 PWC Control Status Register (PWCSL0/PWCSL1)**

Bit name		Function
bit7, bit6	CLK1,CLK0: Clock select bits	<ul style="list-style-type: none"> <li>CKS1 and CKS0 bits are used to select the internal count clock. These bits are used to select the internal count clock.</li> <li>After reset, the bits are initialized to “00<sub>B</sub>”. The bits can be read and written. However, “11<sub>B</sub>” cannot be set.</li> </ul> <p>(Note) After the timer is started, changing the setting is prohibited. Write these bits before the timer is started or after the timer is stopped.</p>
bit5, bit4	Reserved bits	<ul style="list-style-type: none"> <li>These bits are reserved. Always write “00<sub>B</sub>” to these bits.</li> </ul>
bit3	S/C: Single/continuous bit	<ul style="list-style-type: none"> <li>The S/C bit is used to select the count mode.</li> <li>After reset, the bit is initialized to “0”. The bit can be read and written.</li> </ul> <p>(Note) After the timer is started, changing the setting is prohibited. Write this bit before the timer is started or after the timer is stopped.</p>
bit2 to bit0	MOD2 to MOD0: Operation mode bits	<ul style="list-style-type: none"> <li>Setting these bits enables selection of the operating mode and the pulse edge that fits the pulse-width count.</li> <li>After reset, these bits are initialized to “000<sub>B</sub>”. These bits can be read and written.</li> </ul> <p>(Note) After the timer is started, changing the setting is prohibited. Write these bits before the timer is started or after the timer is stopped. If the continuous measurement mode is set for the setting marked *, the number of edges are totaled and the divider for the internal count clock is not cleared at the end of count. In all other modes, the divider for the internal count clock is cleared at the end of the count.</p>

## 16.4.2 PWC Data Buffer Register (PWC0/PWC1)

The PWC data buffer register (PWC0/PWC1) has functions that depend on the operation mode of the PWC timer.

### ■ PWC Data Buffer Register (PWC0/PWC1)

Figure 16.4-4 PWC Data Buffer Register (PWC0/PWC1)

PWC data buffer register (Upper)									
Address: ch.0 00000B <sub>H</sub> ch.1 00002B <sub>H</sub>	bit 15	14	13	12	11	10	9	8	PWC0,PWC1
	PW15	PW14	PW13	PW12	PW11	PW10	PW09	PW08	
Read/write ⇨	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨	X	X	X	X	X	X	X	X	
PWC data buffer register (Lower)									
Address: ch.0 00000A <sub>H</sub> ch.1 00002A <sub>H</sub>	bit 7	6	5	4	3	2	1	0	PWC0,PWC1
	PW07	PW06	PW05	PW04	PW03	PW02	PW01	PW00	
Read/write ⇨	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨	X	X	X	X	X	X	X	X	

#### ● Timer mode

In the reload timer operation mode (PWCSL0/PWCSL1:S/C = 1), this register contains the reload value. The register can be read or written.

In the single timer operation mode (PWCSL0/PWCSL1:S/C = 0), direct access to this register accesses the up-count timer. In this mode, this register can be read or written. However, the register is written only when the timer stops. The register can always be read and the current timer value is read.

#### ● Pulse-width measurement mode (read-only)

In the continuous measurement mode (PWCSL0/PWCSL1:S/C = 1), this register functions as the buffer register and contains the previous count result. This register is read-only. Writing to this register has no effect.

In the single measurement mode (PWCSL0/PWCSL1:S/C = 0), direct access to this register accesses the up-count timer. In this mode, the register is also read-only. Writing to this register has no effect. The register can always be read and the current timer value is read. After the count, the register contains the count results.

#### Notes:

- To access this register, always use the word transfer instruction.
- After reset, this register is initialized to "0000<sub>H</sub>".

### 16.4.3 Division Rate Control Register (DIV0/DIV1)

The division rate control register (DIV0/DIV1) is used in the division period measurement mode (PWCSL:MOD2, 1, and 0 = 011<sub>B</sub>). This register has no meaning in other modes.

#### ■ Division Rate Control Register (DIV0/DIV1)

Figure 16.4-5 Division Rate Control Register (DIV0/DIV1)

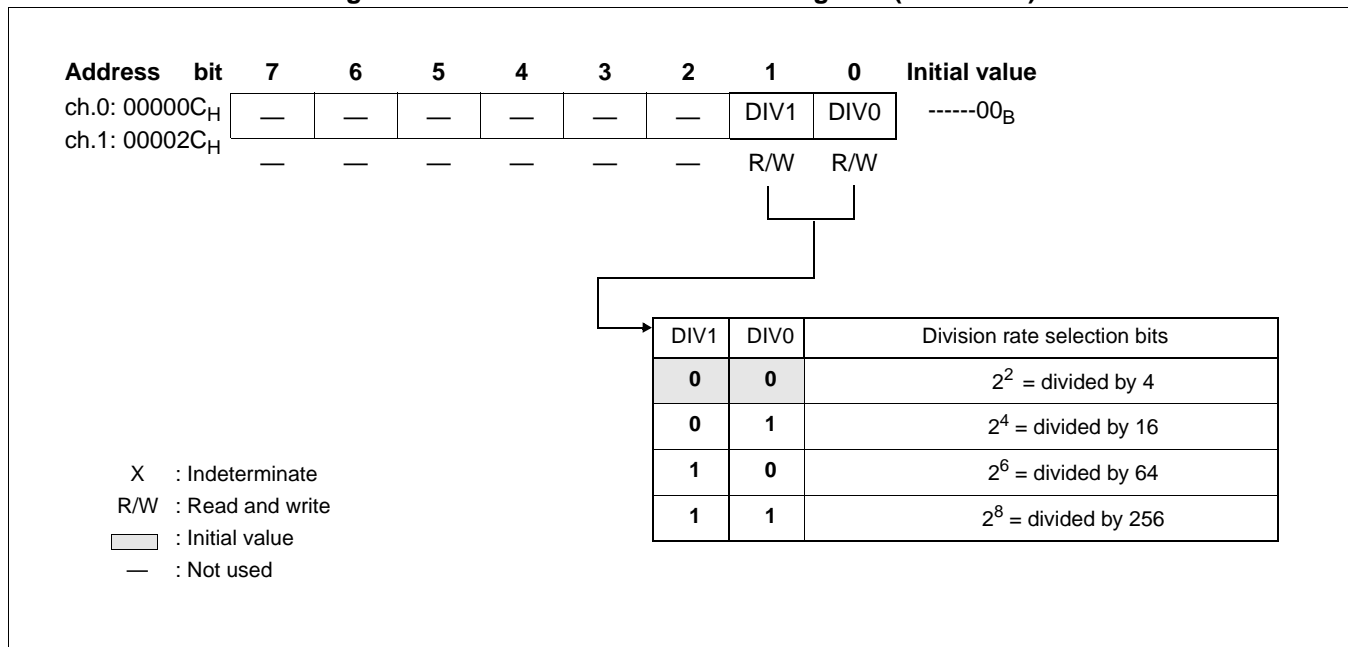


Table 16.4-3 Division Rate Control Register (DIV0/DIV1)

Bit name		Function
bit7 to bit2	Unused bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on the operation.</li> </ul>
bit1, bit0	DIV1,DIV0: Division rate selection bits	<ul style="list-style-type: none"> <li>In the division range measurement mode, this register is used to divide the pulse input from the measurement pin and measure the one-period width after division.</li> <li>After reset, these bits are initialized to “00<sub>B</sub>”. These bits can be read and written.</li> </ul> <p>(Note) After the timer starts, the setting cannot be changed. Write these bits before the timer has started or after the timer has stopped.</p>

## 16.5 PWC Timer Interrupts

The PWC timer is enabled to generate an interrupt request in an overflow of the counter or measurement terminated in pulse-width measurement mode. It is also coordinated with the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ PWC Timer Interrupts

Table 16.5-1 lists the interrupt control bits and interrupt causes of the PWC timer.

**Table 16.5-1 Interrupt Control Bits and Interrupt Causes of the PWC Time**

	PWC timer 0		PWC timer 1	
Interrupt request flag bit	PWCSL0: OVIR	PWCSL0: EDIR	PWCSL1: OVIR	PWCSL1: EDIR
Interrupt request enable bit	PWCSL0: OVIE	PWCSL0: EDIE	PWCSL1: OVIE	PWCSL1: EDIE
Interrupt cause	Overflow of the 16-bit up counter	Measurement terminated in pulse-width measurement mode	Overflow of the 16-bit up counter	Measurement terminated in pulse-width measurement mode

In the PWC timer, the OVIR bit of the PWC control status register (PWCSL) is set to "1" by an overflow (from FFFF<sub>H</sub> to 0000<sub>H</sub>) of the up counter. If an interrupt request is enabled (PWCSL:OVIE = 1) in this operation, the interrupt request is output to the interrupt controller.

The EDIR bit of the PWC control status register (PWCSL) is set to "1" by measurement terminated in pulse-width measurement mode. If an interrupt request is enabled (PWCSL:EDIE = 1) in this operation, the interrupt request is output to the interrupt controller.

### ■ PWC Timer Interrupts and EI<sup>2</sup>OS

Table 16.5-2 lists the PWC timer interrupts and EI<sup>2</sup>OS.

**Table 16.5-2 16-bit PWC Timer Interrupts and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
PWC timer 0 <sup>*1</sup>	#13 (0D <sub>H</sub> )	ICR01	0000B1 <sub>H</sub>	FFFC8 <sub>H</sub>	FFFC9 <sub>H</sub>	FFFC A <sub>H</sub>	O
PWC timer 1 <sup>*2</sup>	#24 (18 <sub>H</sub> )	ICR06	0000B6 <sub>H</sub>	FFF9C <sub>H</sub>	FFF9D <sub>H</sub>	FFF9E <sub>H</sub>	

\*1: The same interrupt number as that for 16-bit PPG timer 0 is assigned to PWC timer 0.

\*2: The same interrupt number as that for output compare channel 5 match is assigned to PWC timer 1.



## ■ EI<sup>2</sup>OS Function of the PWC Timer

Since the PWC timer has a circuit that coordinates with EI<sup>2</sup>OS, the counter can start EI<sup>2</sup>OS when an overflow or measurement termination occurs.

However, EI<sup>2</sup>OS is available only when other peripheral functions sharing the interrupt control register (ICR) do not use interrupts. For example, when PWC timer 0 uses EI<sup>2</sup>OS, interrupts of the 16-bit PPG timer 0 must be disabled.

## 16.6 Operation of the PWC Timer

The PWC timer is the multi-functional timer based on the 16-bit up-count timer and contains the count input pin and 8-bit input divider. The block has two main functions: timer function and pulse-width count function. Both the timer function and the pulse-width count function enable the selection of two types of count clocks.

### ■ Timer Function

The timer function is the up-count timer that enables selection of the operation in single mode or reload mode.

When the timer is started, a timer count is performed at each count clock.

When an overflow occurs in the range from  $FFFF_H$  to  $0000_H$ , an interrupt request is issued.

If an overflow occurs, the following occurs:

During single mode, count is discontinued (see Figure 16.6-1).

During reload mode, the reload register contents are reloaded to the timer and the count is restarted (see Figure 16.6-2).

**Figure 16.6-1 Timer Operation (Single Mode)**

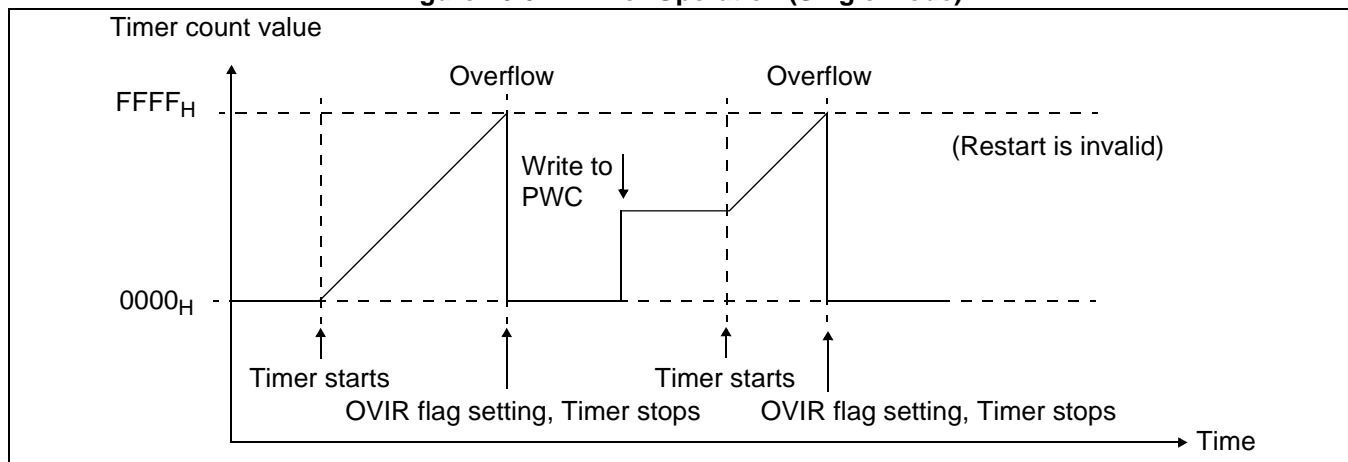
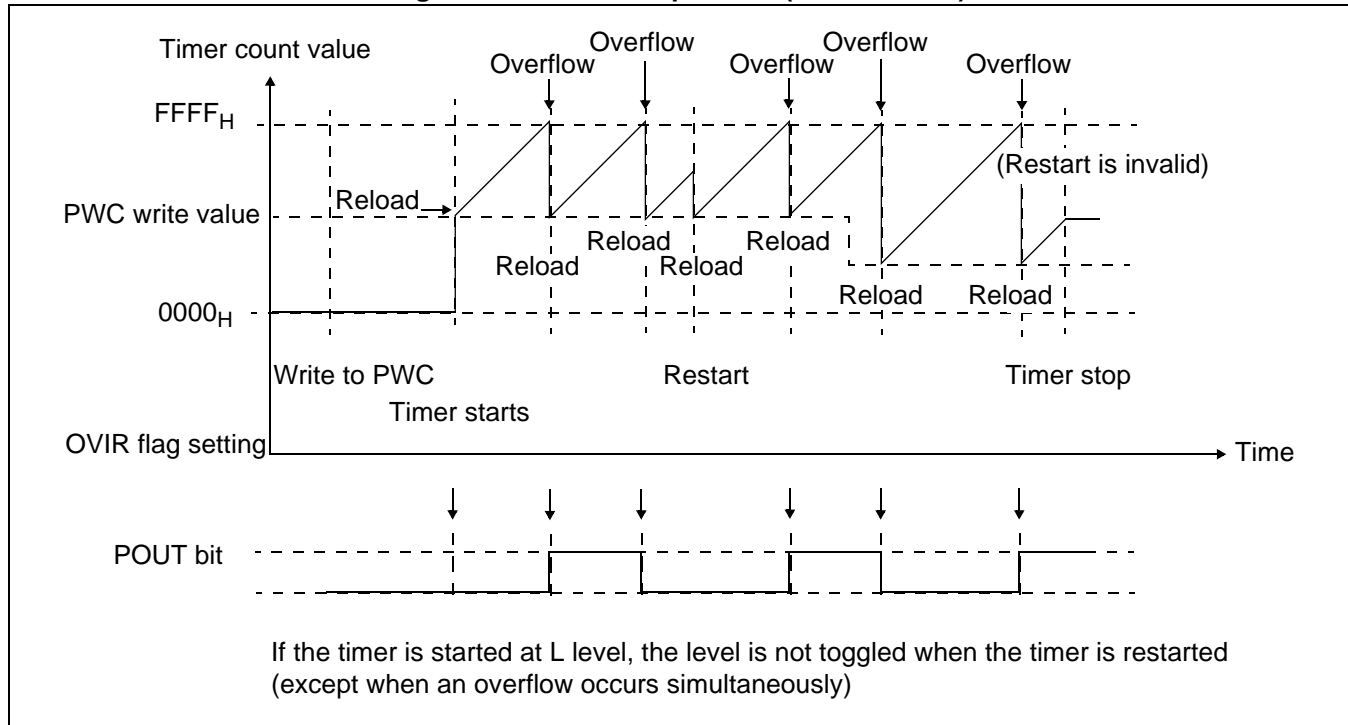


Figure 16.6-2 Timer Operation (Reload Mode)



## ■ Pulse-width Measurement Function

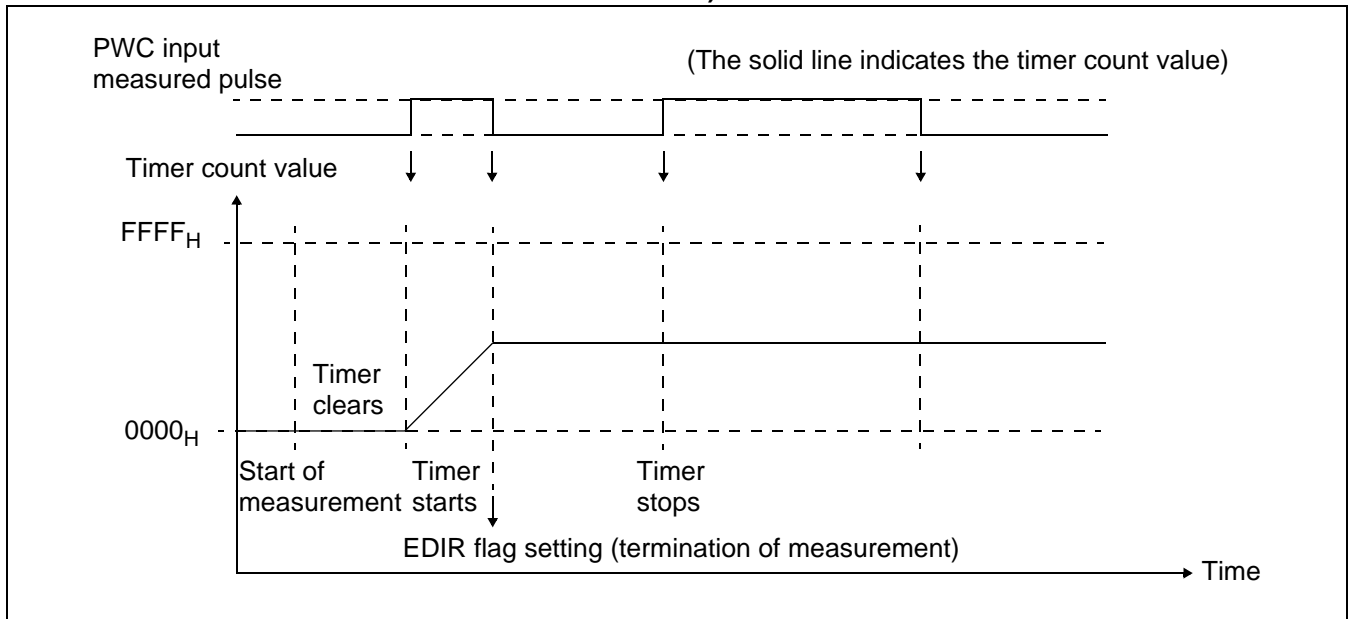
The pulse-width measurement function calculates the time between the specified events related to the input pulse.

When this function is activated, a count is started after the specified count start edge is input. If the counter is cleared to "0000<sub>H</sub>", a count is started when the start edge is detected, then the stop edge is detected. The count value during this period is held in the register as the pulse width.

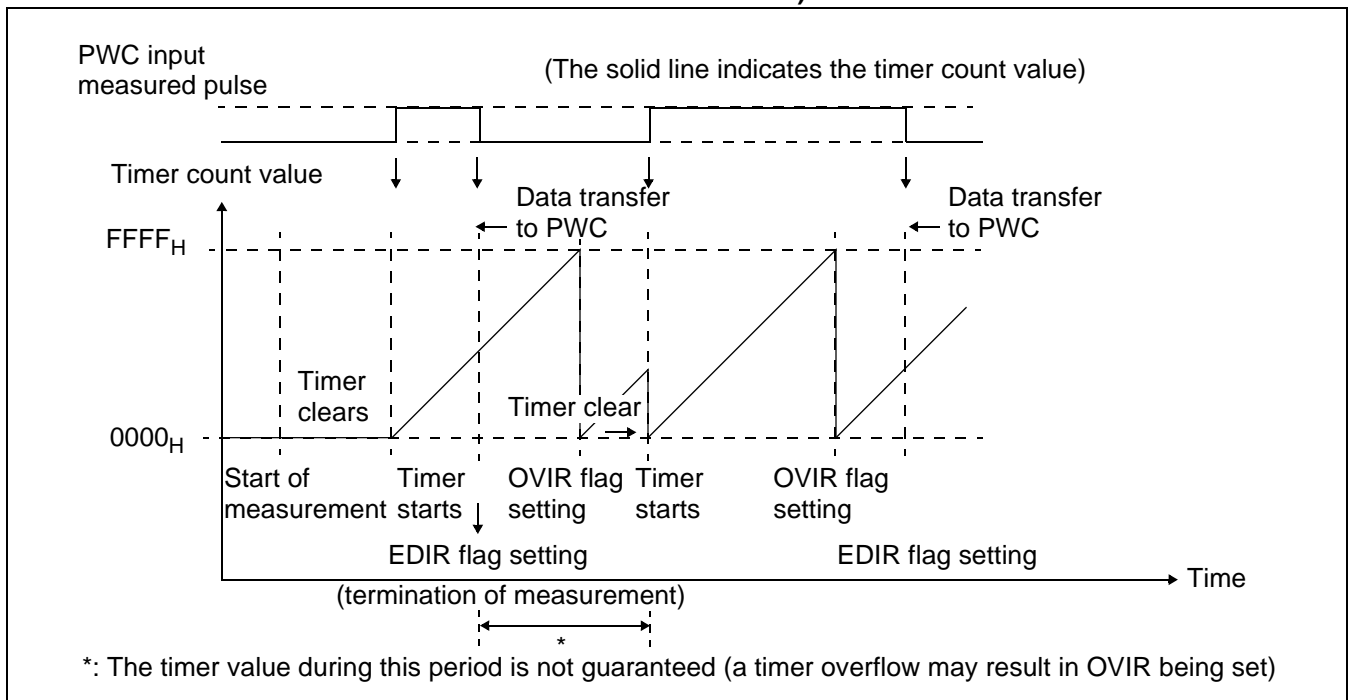
When the measurement terminates or an overflow occurs, an interrupt request can be generated. When the measurement is completed, the following occurs:

- Single measurement mode  
The operation is discontinued (see Figure 16.6-3).
- Continuous measurement mode  
The timer value is transferred to the buffer register and the timer is in free-run state until the next edge is input (see Figure 16.6-4).

**Figure 16.6-3 Pulse-width Measurement Operation (Single Measurement Mode, H-width Measurement Mode)**



**Figure 16.6-4 Pulse-width Measurement Operation (Continuous Measurement Mode, H-width Measurement Mode)**



## 16.6.1 Operation Mode Selection

Operation modes and count modes are selected according to the setting of PWCS.

### ■ Operation Mode Selection

The following registers are used to set the selection of operation modes and count modes:

- Operation mode setting: PWCSL:MOD2, MOD1, and MOD0 bits

Select the timer mode or pulse-width measurement mode to specify control of the count operation.

- Count mode setting: PWCSL:S/C bit

Select single measurement or continuous measurement or reload operation or one-shot operation.

Table 16.6-1 lists the operation modes selected using the mode setting bits.

**Table 16.6-1 Operation Mode Selection**

Operation mode			S/C	MOD2	MOD1	MOD0
Timer	One-shot timer		0	0	0	0
	Reload timer		1	0	0	0 / 1
	Setting prohibited		0	0	0	1
Pulse-width measurement	Rising edge or falling edge to falling edge or rising edge: All edge-to-edge measurement	Single measurement: Buffer invalid	0	0	1	0
		Continuous measurement: Buffer valid	1	0	1	0
	Division count: Divide by 4 to 256	Single measurement: Buffer invalid	0	0	1	1
		Continuous measurement: Buffer valid	1	0	1	1
	Rising edge to rising edge: Rising edge to rising edge period measurement	Single measurement: Buffer invalid	0	1	0	0
		Continuous measurement: Buffer valid	1	1	0	0
	Rising edge to falling edge: H pulse-width measurement	Single measurement: Buffer invalid	0	1	0	1
		Continuous measurement: Buffer valid	1	1	0	1
Pulse-width measurement	Falling edge to rising edge: L pulse-width measurement	Single measurement: Buffer invalid	0	1	1	0
		Continuous measurement: Buffer valid	1	1	1	0
	Falling edge to falling edge: Falling edge-to-falling edge period measurement	Single measurement: Buffer invalid	0	1	1	1
		Continuous measurement: Buffer valid	1	1	1	1

After reset, the one-shot timer is selected as an initial value.

Note:

Before the timer starts, always selects the operation mode.

## 16.6.2 Starting and Stopping the Timer and Pulse-width Measurement and Clearing the Timer

To start, restart, and forcibly stop the timer and pulse-width measurement, use the PWCSH0/PWCSH1:STRT and PWCSH0/PWCSH1:STOP.

The 16-bit up-count timer is cleared to "0000<sub>H</sub>" at reset and when the measurement start edge is detected and the count is started in the pulse-width measurement mode.

### ■ Starting and Stopping Timer and Pulse-width Measurement

Writing "0" to the PWCSH0/PWCSH1:STRT bit starts or restarts the operation, and writing "0" to the PWCSH0/PWCSH1:STOP bit stops the operation. However, unless the value is written to these two bits are different, none of the bits executes operations. If an instruction (byte or word instruction) other than the bit manipulation instruction is being used, a value is written to the following bit combinations only.

**Table 16.6-2 Pulse-width Measurement Operation  
(Single Measurement Mode, H-width Measurement Mode)**

Function	STRT	STOP
Starts and restarts the timer or pulse-width measurement	0	1
Stops the timer or pulse-width measurement	1	0

If a bit manipulation instruction (clear bit instruction) is being used, the hardware automatically writes the above combination of values. The user need not know which value is to be written.

#### ● Operation after start

Timer mode: The count operation is started immediately.

Pulse-width measurement mode: Measurement is started after the measurement start edge is input. After the measurement start edge is detected, the 16-bit up-count timer is cleared to "0000<sub>H</sub>" and the count is started.

#### ● Restarting the timer

While the timer operation continues after the timer is started in the timer mode or pulse-width measurement mode, starting the start (writing "0" to the PWCSH0/PWCSH1:STRT bit) is called timer restart. The operations to be executed during restart are dependent on the following modes:

One-shot mode: The operation is not affected.

Reload timer mode: Reload is executed and the operation is continued. If the timer is restarted when an overflow occurs, the overflow flag (PWCSH0/PWCSH1:OVIR) is set and the POUT bit is reversed.

Pulse-width measurement mode: In the measurement start edge wait state, the operation is not affected. During measurement, the count stops and the timer state returns to the "measurement start edge wait" state. When the timer is restarted on termination of measurement, the measurement termination flag (PWCSH0/PWCSH1:EDIR) is set and the measurement results are transferred to PWC0/PWC1 in continuous measurement mode.

### ● Stopping the timer

In one-shot timer mode or single measurement mode, measurement is automatically discontinued when the timer overflows or at the end of a count. The user need not know if the timer has stopped. However, in other modes, the timer must be stopped. This is also true when the timer is to be stopped before the timer automatically stops.

### ● Checking operation state

The previously described STRT and STOP bits function as bits that indicate the operation state of the timer during a read operation. Table 16.6-3 lists the contents of the indicated values

**Table 16.6-3 Functions of Operation State Indication Bits**

STRT	STOP	Operation state
0	0	Timer is stopping (except measurement start edge wait state). The bits indicate that the timer has not started or a measurement has terminated.
1	1	Measurement start edge wait state or timer count operation

During a read operation, both the STRT bit and the STOP bit have the same value. However, during a read operation using the read modify write instruction (such as bit manipulation instruction), the values of the bits are always 11<sub>B</sub>. Do not use this instruction to read the values of the bits.

### ■ Clearing the Timer

In the following cases, the 16-bit up-count timer is cleared to "0000<sub>H</sub>":

- During reset
- When a count has started after the count start edge is detected in the pulse-width measurement mode

## 16.6.3 Timer Mode Operation

---

**The timer mode includes the one-shot operation mode and reload operation mode.**

---

### ■ One-shot Operation Mode

When the timer is started in this mode, a count is incremented at each count clock. The timer automatically stops when an overflow occurs from  $FFFF_H$  to  $0000_H$ .

If PWC0/PWC1 is set before the timer has started, the count is started from this set value. After overflow, the set value is deleted and the current count value remains in PWC0/PWC1.

PWCSH0/PWCSH1:POUT is reversed if an overflow occurs.

### ■ Reload Operation Mode

When the timer is started in this mode, the reload value in PWC0/PWC1 is set in the timer and the count is incremented at each count clock. If an overflow occurs when the timer counts  $FFFF_H$  to  $0000_H$ , the reload value in PWC0/PWC1 is set in the timer again, the PWCSH0/PWCSH1:POUT bit is reversed, and the count operation is repeated. The timer does not stop until a value is written to the PWCSH0/PWCSH1:STOP to stop the timer or it is reset. The port bit will output to pin PWO0/PWO1 if pulse output mode is specified.

The reload value (set in PWC0/PWC1 before the timer is started) is stored during a count. When the timer is started or restarted and an overflow occurs, the reload value is always set in the timer. If the value that is set during a count is to be changed, a new reload value becomes valid when the next overflow occurs or the timer is restarted.

### ■ Timer Value and Reload Value

In one-shot operation mode, direct access to PWC0/PWC1 accesses the up-count timer. When a value is written to PWC0/PWC1, the value is written directly to the timer. When PWC0/PWC1 is read during a count operation, the current timer value is read. If the value is set in PWC0/PWC1 before the timer is started, the timer starts a count from the specified value.

In reload operation mode, the up-count timer cannot be accessed and PWC0/PWC1 functions as a reload register (stores the reload value). When the timer is started or restarted and an overflow occurs, the value written to PWC0/PWC1 is always set in the timer. When PWC0/PWC1 is read, the stored reload value is read.

The PWC0/PWC1 value and timer value are undefined if the timer is set in one-shot mode after the operation is discontinued in reload mode. Therefore, always set the values before the timer is used.

The PWC0/PWC1 value is undefined if the timer is set in reload mode after the operation is forcibly discontinued in one-shot mode. Therefore, always set the value before the timer is used.

### ■ Interrupt Request Generation

During operation in timer mode, an overflow enables the generation of an interrupt request. If the increment of a timer count causes an overflow, the overflow flag is set, an overflow interrupt request is enabled, and an interrupt request is generated.



## ■ Timer Period

If the timer is started in one-shot mode after "0000<sub>H</sub>" is set in PWC0/PWC1, a timer overflow occurs and the count is discontinued if the count exceeds 65536. The following formula is used to calculate the time from start to stop of the timer.

$$T_1 = (65536 - n_1) \times t$$

{

$T_1$  ..... Time from start to stop ( $\mu$ s)  
 $n_1$  ..... Timer value set in PWC0/PWC1 when the timer is started  
 $t$  ..... Count clock period ( $\mu$ s)

If the timer is started after "0000<sub>H</sub>" is set in PWC0/PWC1, a timer overflow occurs every time the count exceeds 65536. The following formula is used to calculate the reload period and the PWO pin output pulse period.

$$T_R = (65536 - N_R) \times t$$

{

$T_R$  ..... Reload period (overflow period) ( $\mu$ s)  
 $T_{\text{PWO}}$  ..... PWO0/PWO1 pin output pulse period ( $\mu$ s)  
 $N_R$  ..... Reload value stored in PWC0/PWC1  
 $t$  ..... Time from start to stop ( $\mu$ s)

## ■ Count Clock and Maximum Period

In timer mode, when "0000<sub>H</sub>" is set in PWC0/PWC1, the maximum period results.

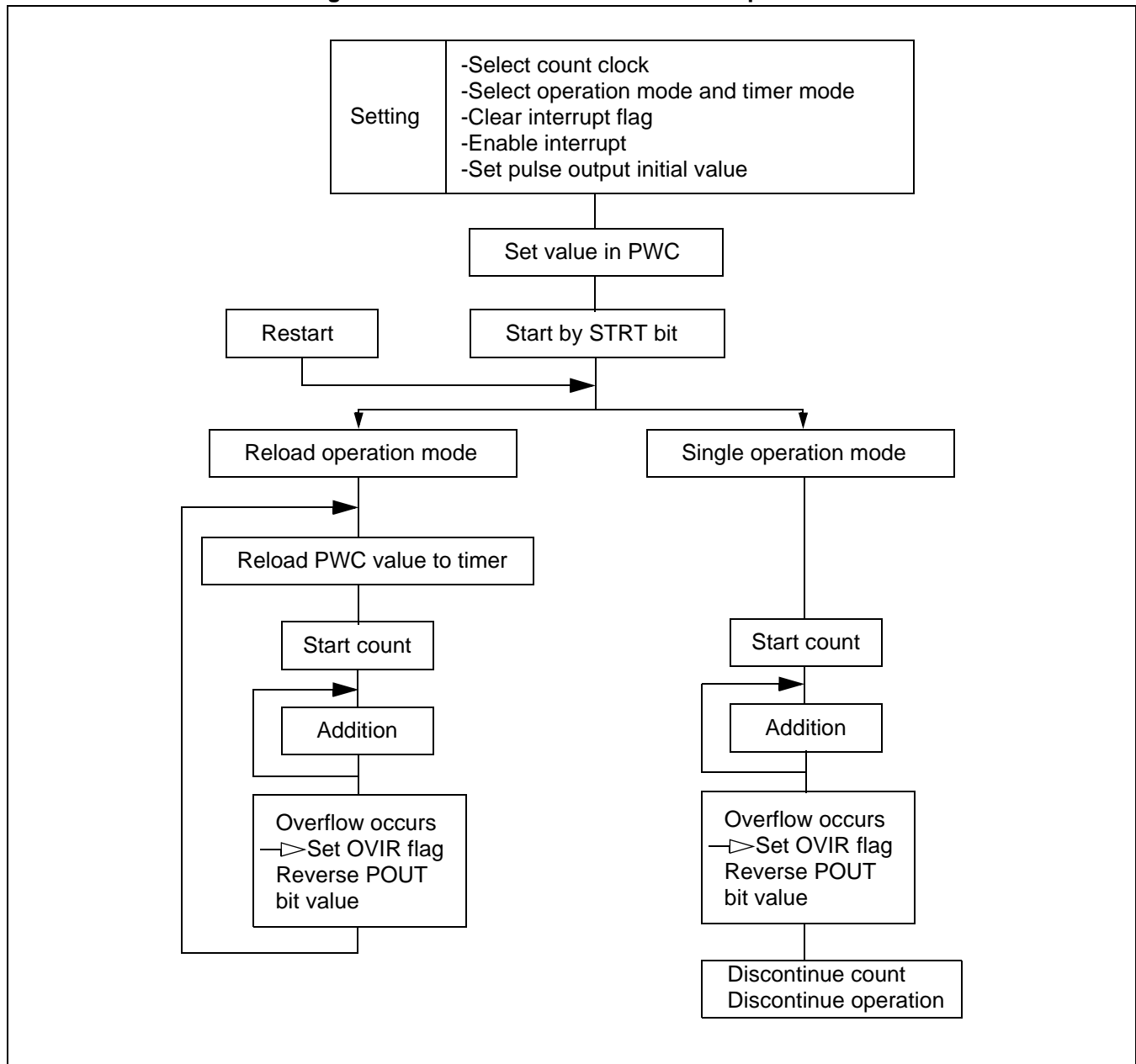
Table 16.6-4 lists the count clock period and maximum timer period corresponding to the machine cycle (indicated by  $f$  in the table) at 16 MHz

**Table 16.6-4 Count Clock and Period**

Count clock selection	When CKS1,CKS0=00 <sub>B</sub> ( $\phi/4$ )	When CKS1,CKS0=01 <sub>B</sub> ( $\phi/16$ )	When CKS1,CKS0=10 <sub>B</sub> ( $\phi/32$ )
Count clock period	0.25 $\mu$ s	1 $\mu$ s	2 $\mu$ s
Maximum timer period	16.38 ms	65.5 ms	131.1 ms

## ■ Flowchart of Timer Mode Operation

Figure 16.6-5 Flowchart of Timer Mode Operation



## 16.6.4 Pulse Width Measurement Mode Operation

The signal for pulse-width measurement is input from the PWI pin.

The pulse-width measurement mode includes the single measurement mode in which the count is performed only once and continuous measurement mode in which the pulse width is continuously measured.

### ■ Single Measurement Mode and Continuous Measurement Mode

The differences between the single measurement mode and continuous measurement mode are as follows:

#### ● Single measurement mode

When the first count end edge is input, the timer discontinues the count, the count end flag (EDIR) of PWCSH0/PWCSH1 is set, and the subsequent measurement is not performed. However, if a timer restart is also specified, the timer state changes to measurement start edge wait state.

#### ● Continuous measurement mode [H/L pulse-width measurement mode]

When the count end edge is input, the count end flag (EDIR) of PWCSH0/PWCSH1 is set, the timer count result is transferred to PWC0/PWC1, and the timer may continue incrementing the count in a free-run state. When the next count start edge is input, the timer is cleared to "0000<sub>H</sub>" and the pulse-width count is started.

Notes:

When the count end edge is input and the timer enters a free-run state, the timer may overflow and the OVIR flag may be set. In the H/L pulse-width measurement mode, do not use the OVIR flag to measure the pulse-width time.

[All edge-to-edge pulse-width measurement mode, division period measurement mode, rising edge-to-rising edge measurement mode, and falling edge-to-falling edge measurement mode]

When the count end edge (count start edge) is input, the count end flag (EDIR) of PWCSH0/PWCSH1 is set, the timer count result is transferred to PWC0/PWC1, the timer is cleared to "0000<sub>H</sub>" and the count is restarted.

### ■ Measurement Result Data

Handling of the measurement result, timer value, and PWC0/PWC1 function varies with the single measurement mode and continuous measurement mode as follows:

#### ● Single measurement mode

When PWC0/PWC1 is read during timer operation, the current timer value is read.

When PWC0/PWC1 is read after termination of measurement, the measurement results are read.

### ● Continuous measurement mode

At termination of measurement, the timer measurement results are transferred to PWC0/ PWC1.

When PWC0/ PWC1 is read, the previous measurement results are read. While measurement is in progress, the previous measurement results are stored in PWC0/ PWC1. During measurement, the timer value cannot be read.

In continuous measurement mode, unless the previous measurement results are read before completion of the next measurement, a new measurement result overwrites the existing value. The error flag (ERR) of PWCSH0/PWCSH1 is set. When PWC0/ PWC1 is read, the error flag (ERR) is cleared automatically.

### ■ Minimum Input Pulse Width

The pulse must be input to the pulse-width count input pin (PWIO/PWII) longer than the following minimum input pulse width.

Pulse width: 2 machine cycles (0.125 μs or more for the machine clock at 16 MHz)

However, the input pulse that is shorter than the above specification may also be recognized as a valid pulse.

### ■ Calculating Pulse Width/period

The pulse width or pulse period of the measurement object is calculated based on the count result read from PWC0/PWC1 at the end of a count as follows.

$$T_W = n \times t / \text{Div} (\mu\text{s})$$

{

$T_W$  ..... Measured pulse width or pulse period (μs)

$n$  ..... Measurement result contained in PWC0/PWC1

$t$  ..... Count clock period (μs)

$\text{Div}$  ..... Division rate set in the division rate register (DIV0/DIV1)  
(a value of "1" is used in a mode other than the division count mode)

### ■ Pulse Width/period Measurement Range

The range of the pulse width/period that can be measured depends on the count clock and division rate of an input divider.

Table 16.6-5 lists the measurement range for the machine cycle (indicated by  $\phi$ ) at 16 MHz

**Table 16.6-5 Pulse Width Measurement Range**

Division rate	DIV1, DIV0	CKS1,CKS0=00 <sub>B</sub> ( $\phi/4$ )	CKS1,CKS0=01 <sub>B</sub> ( $\phi/16$ )	CKS1,CKS0=10 <sub>B</sub> ( $\phi/32$ )
No division	-	0.125 μs to 16.38 ms [0.25 μs]	0.125 μs to 65.5 ms [1.0 μs]	0.125 μs to 131 ms [2 μs]
Divide-by 4	00 <sub>B</sub>	0.125 μs to 4.10 ms [62.5 ns]	0.125 μs to 16.38 ms [0.25 μs]	0.125 μs to 32.75 ms [500 ns]
Divide-by 16	01 <sub>B</sub>	0.125 μs to 1024 μs [15.6 ns]	0.125 μs to 4.10 ms [62.5 ns]	0.125 μs to 8.19 ms [125 ns]
Divide-by 64	00 <sub>B</sub>	0.125 μs to 256 μs [3.91 ns]	0.125 μs to 1024 μs [15.6 ns]	0.125 μs to 2.048 ms [31.25 ns]
Divide-by 256	11 <sub>B</sub>	0.125 μs to 64 μs [0.98 ns]	0.125 μs to 256 μs [3.91 ns]	0.125 μs to 512 μs [7.81 ns]

(Note) The number in [ ] indicates the resolution per bit.

## ■ Interrupt Request Generation

In the pulse-width measurement mode, the following two interrupt requests can be generated:

### ● Timer overflow interrupt request

If an overflow occurs during a count, the overflow flag is set. When the overflow interrupt request is enabled, an interrupt request is generated.

### ● Measurement termination interrupt request

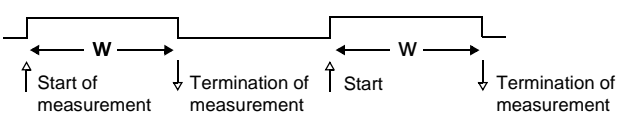
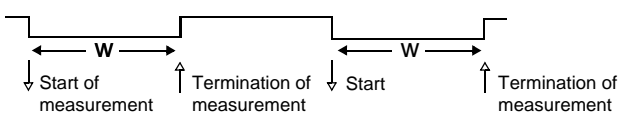
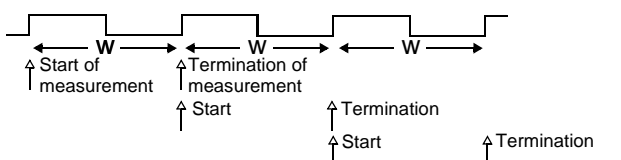
When the measurement termination edge is detected, the count end flag (EDIR) of PWCSH0/PWCSH1 is set. If the measurement termination interrupt is enabled, an interrupt request is generated.

The measurement termination flag (EDIR) is automatically cleared when PWC0/ PWC1 is read.

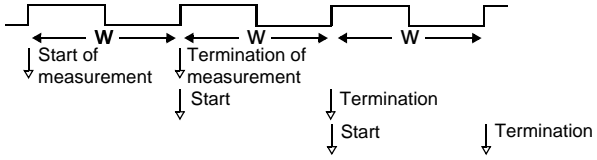
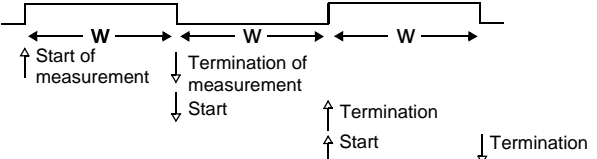
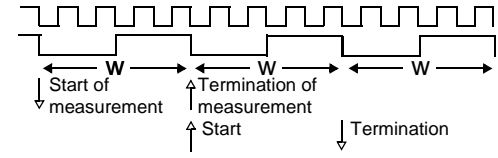
## ■ Measurement Mode and Measurement Operation

Table 16.6-6 lists measurement mode operations.

**Table 16.6-6 Measurement Mode Operation (1/2)**

Measurement mode	MOD2	MOD1	MOD0	Measurement operation
H pulse-width measurement	1	0	1	 <p>The H period width is measured.</p> <p>Start of measurement: When the rising edge is detected</p> <p>Termination of measurement: When the falling edge is detected</p>
L pulse-width measurement	1	1	0	 <p>The L period width is measured.</p> <p>Start of measurement: When the falling edge is detected</p> <p>End of measurement: When the rising edge is detected</p>
Rising edge-to-rising edge period measurement	1	0	0	 <p>The rising edge-to-rising edge time is measured.</p> <p>Start of measurement: When the rising edge is detected</p> <p>Termination of measurement: When the rising edge is detected</p>

**Table 16.6-6 Measurement Mode Operation (2/2)**

Measurement mode	MOD2	MOD1	MOD0	Measurement operation
Falling edge-to-falling edge period measurement	1	1	1	 <p>The falling edge-to-falling edge time is measured.</p> <p>Start of measurement: When the falling edge is detected</p> <p>Termination of measurement: When the falling edge is detected</p>
All edge pulse-width measurement	0	1	0	 <p>The width between continuous input edges is measured.</p> <p>Start of measurement: When the edge is detected</p> <p>Termination of measurement: When the edge is detected</p>
Division measurement	0	1	1	 <p>(Divided by 4 in the above example.)</p> <p>The input pulse is divided by the division rate set in the division rate register (DIVR), and the measurement period is obtained as a result.</p> <p>Start of measurement: The falling edge is detected after the operation is started.</p> <p>Termination of measurement: One period of division signal ends.</p>

W: Pulse width being measured

In all modes, the timer does not start count during the period from the start of measurement to input of measurement start edge. After the measurement start edge is input, the timer is cleared to "0000<sub>H</sub>", and the count is incremented at each count clock until the measurement termination edge is input.

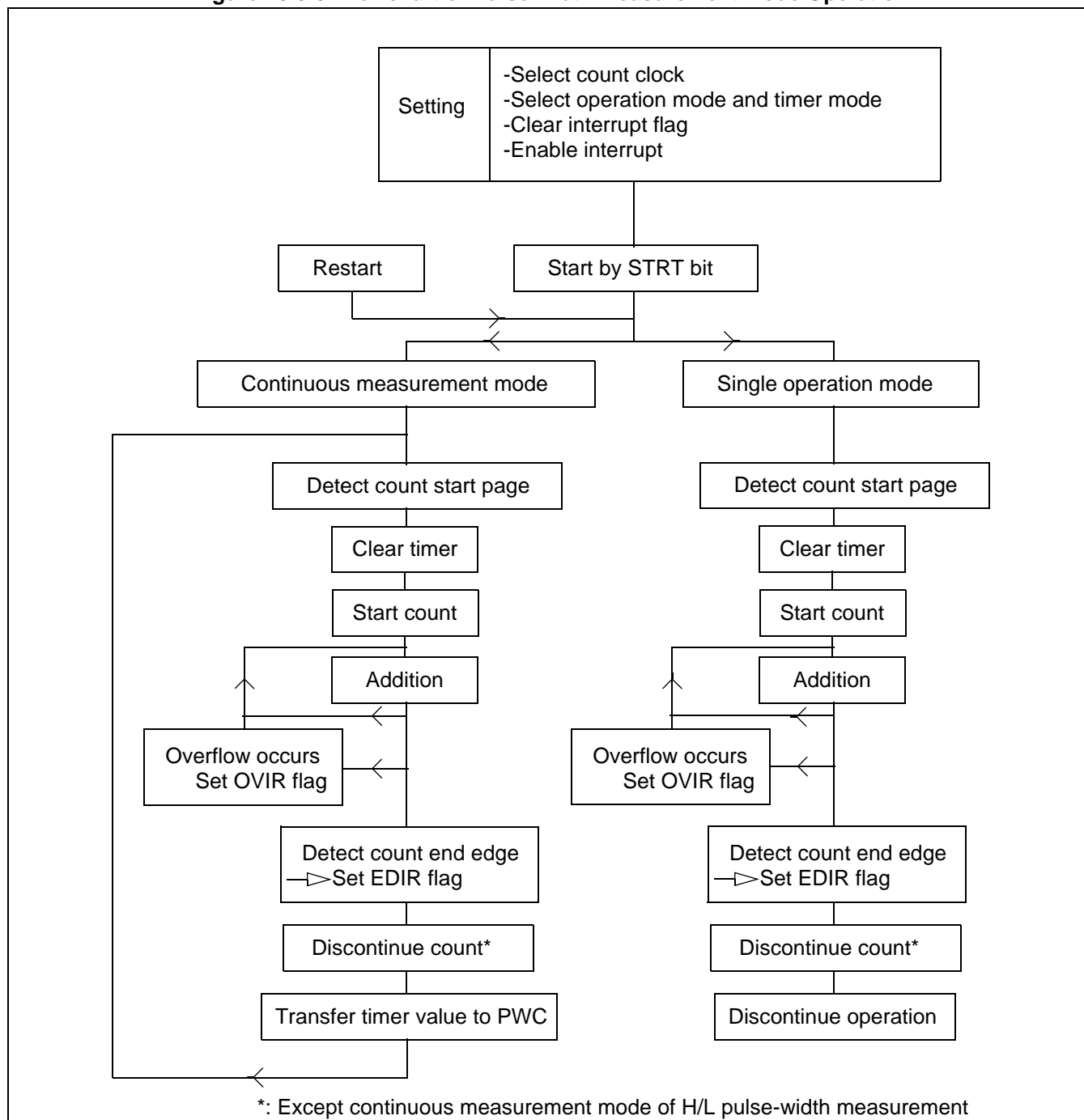
When the measurement termination edge is input, the following operations are executed

- (1) The count end flag (EDIR) of PWCSH0/PWCSH1 is set.
- (2) The timer stops count operation (except if the timer is restarted at the same time or continuous measurement mode of the H/L pulse-width measurement is used).
- (3) Continuous measurement mode: The timer value (measurement result) is transferred to PWC0/PWC1.
- (4) Single measurement mode: Measurement is terminated (except if the timer is restarted at the same time).

If all edge-to-edge pulse-width measurement, period measurement, falling edge-to-falling edge period measurement, or rising edge-to-rising edge period measurement is done in continuous measurement mode, the termination edge becomes the next measurement start edge.

## ■ Flowchart of Pulse-width Measurement Operation

Figure 16.6-6 Flowchart of Pulse-width Measurement Mode Operation



## 16.7 Usage Notes on the PWC Timer

---

Notes on using the PWC timer are given below.

---

### ■ Usage Notes on the PWC Timer

#### ● Notes about using a program for setting

- Changing the following PWCS0/PWCS1 bit values is prohibited during timer operation. The bit values are changed only before the timer is started or after the operation is discontinued.

[bit7, bit6] CKS1 and CKS0: Clock selection bits

[bit3] S/C: Measurement mode (single or continuous) selection bit

[bit2 to bit0] MOD2, MOD1, and MOD0: Operation mode and measurement edge selection bits

Note that the value of pulse output level indication bit (POUT: bit8) remains unchanged even if the bit is written during timer operation.

- Changing the DIV0/DIV1 value is prohibited during timer operation. Change the DIV0/DIV1 value before the timer is started or after the operation has stopped.
- Setting the clock selection bits CKS1 and CKS0 of PWC control status register (PWCSL0/PWCSL1) to 11<sub>B</sub> is prohibited.
- The PWC0/PWC1 and timer values are determined when the timer is set in the one-shot mode or after the operation is terminated in reload timer mode. Therefore, always set the values after the timer is used.
- The PWC0/PWC1 value is undefined if the timer is set in reload timer mode after the operation is discontinued in the one-shot mode. Therefore, always set the value before the timer is used.
- To change the mode from pulse-width measurement mode to timer mode, always set the value in PWC0/PWC1 before the timer has started.
- When division period measurement mode is used in pulse-width measurement mode, the input pulse is divided. Note that the pulse width calculated from the count result becomes a mean.
- During continuous measurement in pulse-width measurement mode, the division circuit for an internal count clock is not cleared, and the number of edges smaller than the count clock is added to the count result.



● Notes about using a program for status checking

- In timer mode, the value of the measurement termination interrupt request flag (EDIR) of PWCSH0/PWCSH1 is insignificant. Therefore, always set "0" in the count end interrupt request (EDIE) enable bit of PWCSH0/PWCSH1.
- The STRT and STOP bits of PWC control status register (PWCSH0/PWCSH1) are dependent on whether they are read or written (see the details of registers). Read modify write instruction always reads the bits as 11<sub>B</sub>. So bit manipulation instruction cannot be used to read the operation state.

However, a bit manipulation instruction (bit clear instruction) can be used to start or stop the timer by writing the STRT or STOP bit.

- In the pulse-width measurement mode, the measurement start edge causes the timer to be cleared, and the previous timer data is insignificant.

● Notes about pulse input to the pulse width measurement input pin

- Minimum pulse width is divide-by 2 of machine cycle (0.125  $\mu$ s or more for the machine cycle at 16 MHz)
- Maximum input frequency is divide-by 4 of machine cycle (4 MHz or less for the machine cycle at 16 MHz)  
If a pulse width smaller than the above or a frequency larger than the above is input, the timer operation is not guaranteed. A noise violating the above constraint and appearing in the input signal must be reduced.
- If a pulse width smaller than the above or a frequency larger than the above is input, the timer operation is not guaranteed. A noise violating the above constraint

● Notes about restart the timer during operation

- If the timer is restarted when an overflow occurs in reload timer mode, the timer is restarted but the overflow flag (OVIR) is set and the POUT bit is reversed (that is, the same operation as the normal overflow is executed).
- If the timer is restarted when the measurement termination edge is detected in one-shot pulse-width measurement mode, the timer is restarted and enters measurement start edge wait state but the measurement termination flag (EDIR) is also set.
- If the timer is restarted when the measurement termination edge is detected in continuous pulse-width measurement mode, the timer is restarted and enters the measurement start edge wait state, the count termination flag (EDIR) is set, and the measurement results are transferred to PWC0/PWC1.
- To restart the timer during operation, note the flag (OVIR, EDIR) operations to generate interrupts and exercise other controls.

● Notes about interrupts

- When the OVIR bit of the PWC control status register (PWCSH0/PWCSH1) is set to "1" and an interrupt request is enabled (PWCSH0/PWCSH1:OVIE = 1), control cannot be returned from interrupt processing. Always clear the OVIR bit.
- When the EDIR bit of the PWC control status register (PWCSH0/PWCSH1) is set to "1" and an interrupt request is enabled (PWCSH0/PWCSH1:EDIE = 1), control cannot be returned from interrupt processing. Always clear the OVIR bit.
- Since the PWC timer shares an interrupt vector with other resource, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI<sup>2</sup>OS is used by the PWC timer, shared resource interrupts must be disabled.

## 16.8 Sample Programs for the PWC Timer

---

This section contains sample programs for the PWC timer.

---

### ■ Sample Program for the PWC Timer

#### ● Processing

- An output PWO0 of 30.6 Hz is generated with PWC timer 0.
- The PWC is used in reload timer mode to repeatedly generate an overflow interrupt.
- EI<sup>2</sup>OS is not used.
- 16 MHz is used for the machine clock, and 0.25  $\mu$ s is used for the count clock.

#### ● Coding example

```

ICR01      EQU      0000B1H      ;Interrupt control register for the PWC timer
PWCS0      EQU      000008H      ;PWC control status register
PWC0       EQU      00000AH      ;PWC data buffer register
OVIR       EQU      PWCS0:11     ;Interrupt request flag bit
;-----Main program-----
CODE                      CSEG
START:
;           :                      ;Assumes that stack pointer (SP) has already been
;                               ;initialized
;           AND      CCR,#0BFH    ;Interrupt disable
;           MOV      I:ICR01,#00H ;Interrupt level 0 (strongest)
;           MOVW     I:PSC0,#0FF00H ;Sets the reload value
;           MOVW     I:PWCS0,#4409H ;Sets reload timer mode, 0.25 $\mu$ s clock
;                               ;Enable PWC output
;                               ;Sets overflow interrupt
;                               ;Clears interrupt flag and starts PWC timer
;           MOV      ILM,#07H     ;Sets ILM in PS to level 7
;           OR       CCR,#40H     ;Interrupt enable
LOOP:      MOV      A,#00H        ;Endless loop
;           MOV      A,#01H        ;
;           BRA      LOOP          ;
;-----Interrupt program-----
WARI:
;           CLRB     I:OVIR       ;Clears interrupt request flag
;           :

```

```

;      User processing
;      :
;      RETI                      ;Returns from interrupt

CODE      ENDS

;-----Vector setting-----
VECT      CSEGABS=0FFH
          ORG      0FFC8H        ;Sets vector for interrupt #13 (0DH)
          DSL      WARI
          ORG      0FFDCH        ;Sets reset vector
          DSL      START
          DB        00H          ;Sets single-chip mode
VECT      ENDS
          END      START

```



# **CHAPTER 17**

---

## **UART**

**This chapter explains the functions and operation of UART.**

- 17.1 Overview of UART
- 17.2 Block Diagram of UART
- 17.3 UART Pins
- 17.4 UART Registers
- 17.5 UART Interrupts
- 17.6 UART Baud Rates
- 17.7 Operation of UART
- 17.8 Usage Notes on UART
- 17.9 Sample Program for UART

## 17.1 Overview of UART

UART is a general-purpose serial data communication interface for performing synchronous or asynchronous (start-stop synchronization) communication with external devices. The UART has a normal bidirectional communication function (normal mode), additionally the master-slave communication function (multiprocessor mode) is only available for the master system.

### ■ UART Functions (× 2)

#### ● UART functions

UART is a general-purpose serial data communication interface for transmitting serial data to and receiving data from another CPU and peripheral devices. It has the functions listed in Table 17.1-1.

**Table 17.1-1 UART Functions**

	Function
Data buffer	Full-duplex, double buffering
Transfer mode	<ul style="list-style-type: none"> <li>• Clock synchronous</li> <li>• Clock asynchronous (start-stop synchronization)</li> </ul>
Baud rate	<ul style="list-style-type: none"> <li>• A dedicated baud rate generator is provided. Eight settings can be selected</li> <li>• An external clock can be input</li> <li>• Internal clock (internal clocks supplied from 16-bit reload timer 0 can be used)</li> </ul>
Data length	<ul style="list-style-type: none"> <li>• 7 bits (in asynchronous normal mode only)</li> <li>• 8 bits</li> </ul>
Signal mode	Non-return to zero (NRZ)
Reception error detection	<ul style="list-style-type: none"> <li>• Framing error</li> <li>• Overrun error</li> <li>• Parity error (cannot be detected in multiprocessor mode)</li> </ul>
Interrupt request	<ul style="list-style-type: none"> <li>• Reception interrupt (reception completion and reception error detection)</li> <li>• Transmission interrupt (transmission completion)</li> <li>• Extended intelligent I/O service (EI<sup>2</sup>OS) is available for both transmission and reception interrupts</li> </ul>
Master-slave communication function (multiprocessor mode)	One-to-n communication (one master to n slaves) can be performed (this function is supported only for the master system)

Note: During clock synchronous transfer, start and stop bits are not added so only data is transferred in UART.

**Table 17.1-2 UART Operation Mode**

Operation mode		Data length		Synchronization on mode	Stop bit length
		When parity is disabled	When parity is enabled		
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits *2
1	Multiprocessor	8+1*1 bits	–	Asynchronous	
2	Normal mode	8 bits	–	Synchronous	None

– : Setting not possible.

\*1: "+" indicates the address/data selection bit (A/D) for communication control.

\*2: During reception, only one stop bit can be detected.

## ■ UART Interrupt and EI<sup>2</sup>OS

**Table 17.1-3 UART Interrupt and EI<sup>2</sup>OS**

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25 <sub>H</sub> )	ICR13	0000BD <sub>H</sub>	FFFF68 <sub>H</sub>	FFFF69 <sub>H</sub>	FFFF6A <sub>H</sub>	⊙
UART1 transmission interrupt	#38(26 <sub>H</sub> )	ICR13	0000BD <sub>H</sub>	FFFF64 <sub>H</sub>	FFFF65 <sub>H</sub>	FFFF66 <sub>H</sub>	Δ
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>	⊙
UART0 transmission interrupt	#40(28 <sub>H</sub> )	ICR14	0000BE <sub>H</sub>	FFFF5C <sub>H</sub>	FFFF5D <sub>H</sub>	FFFF5E <sub>H</sub>	Δ

⊙ : Provided with a function that detects a UART reception error and stops EI<sup>2</sup>OS

Δ : Usable when ICR13 and ICR14 or interrupt causes that share an interrupt vector are not used

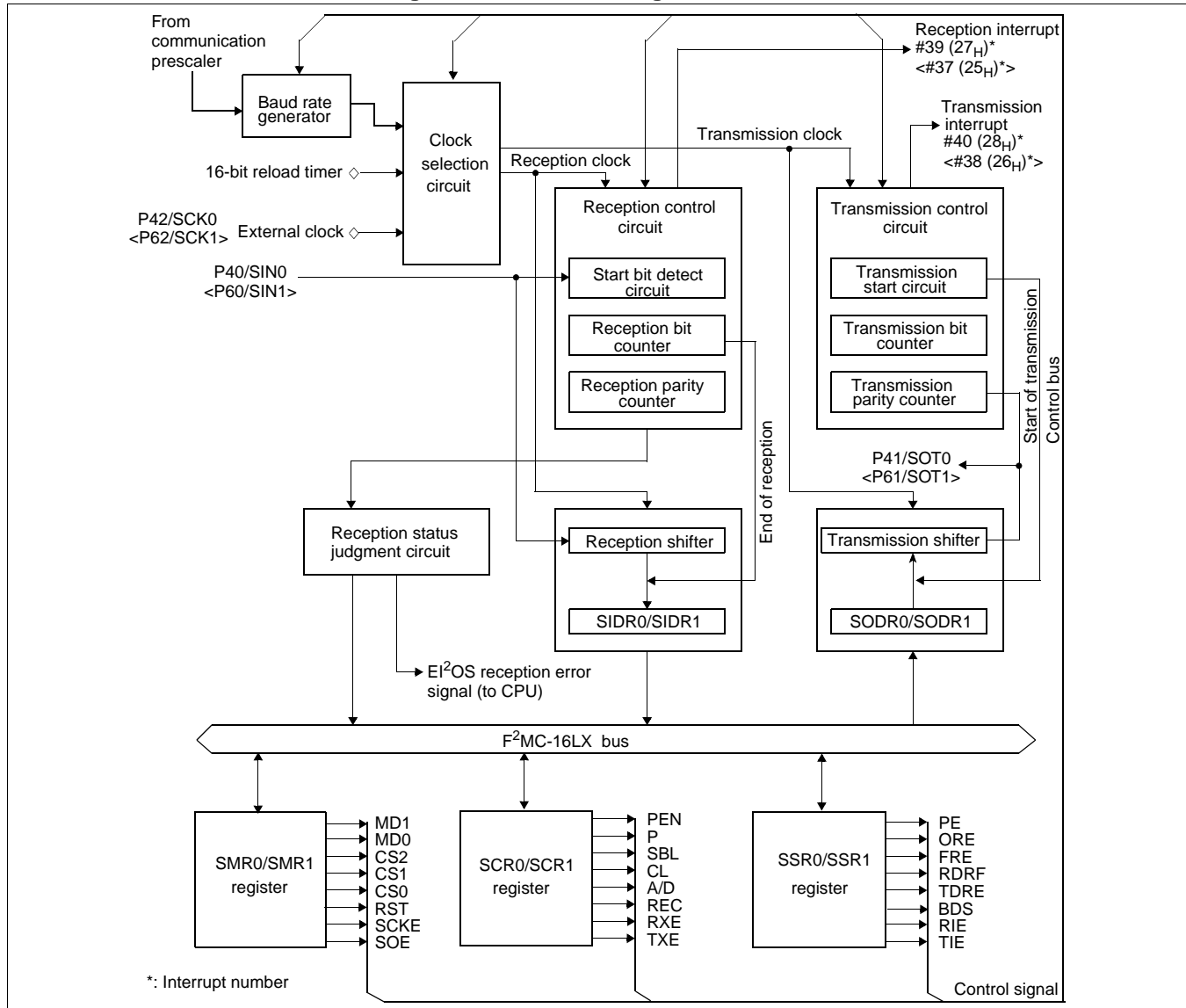


## 17.2 Block Diagram of UART

UART consists of the following 11 blocks, the block diagram is shown in Figure 17.2-1.

### ■ Block Diagram of UART

Figure 17.2-1 Block Diagram of UART



### ● Clock selector

The clock selector selects the dedicated baud rate generator, external input clock, or internal clock (clock supplied from the 16-bit reload timer) as the transmitting and receiving clocks.

### ● Reception control circuit

The reception control circuit consists of a received bit counter, start bit detection circuit, and received parity counter. The received bit counter counts receive data bits. When reception of one data item for the specified data length is complete, the received bit counter generates a reception interrupt request. The start bit detection circuit detects start bits from the serial input signal. When the circuit detects a start bit, it writes data in the SIDR1 register by shifting at the specified transfer rate. The received parity counter calculates the parity of the receive data.

### ● Transmission control circuit

The transmission control circuit consists of a transmission bit counter, transmission start circuit, and transmission parity counter. The transmission bit counter counts transmission data bits. When transmission of one data item of the specified data length is complete, the transmission bit counter generates a transmission interrupt request. The transmission start circuit starts transmission when data is written to SODR0/SODR1. The transmission parity counter generates a parity bit for data to be transmitted when parity is enabled.

### ● Reception shift register

The reception shift register fetches receive data input from the SIN pin, shifting the data bit by bit. When reception is complete, the reception shift register transfers receive data to the SIDR0/SIDR1 register.

### ● Transmission shift register

The transmission shift register transfers data written to the SODR0/SODR1 register to itself and outputs the data to the SOT pin, shifting the data bit by bit.

### ● Mode control register 1 (SMC0/SMC1)

This register performs the following operations:

- Selecting a UART operation mode
- Selecting a clock input source
- Setting up the dedicated baud rate generator
- Selecting a clock rate (clock division value) when using the dedicated baud rate generator
- Specifying whether to enable serial data output to the corresponding pin
- Specifying whether to enable clock output to the corresponding pin

### ● Control register 1 (SCR0/SCR1)

This register performs the following operations:

- Specifying whether to provide parity bits
- Selecting parity bits
- Specifying a stop bit length
- Specifying a data length
- Selecting a frame data format in mode 1
- Clearing a flag
- Specifying whether to enable transmission
- Specifying whether to enable reception

### ● Status register 1 (SSR0/SSR1)

This register checks the transmission and reception status and error status, and enables and disables transmission and reception interrupt requests.

### ● Input data register 1 (SIDR0/SIDR1)

This register retains receive data. Serial input data is converted and stored in this register.

### ● Output data register 1 (SODR0/SODR1)

This register sets transmission data. Data written to this register is converted to serial data and output.

## 17.3 UART Pins

This section describes the UART pins and provides a pin block diagram.

### ■ UART Pins

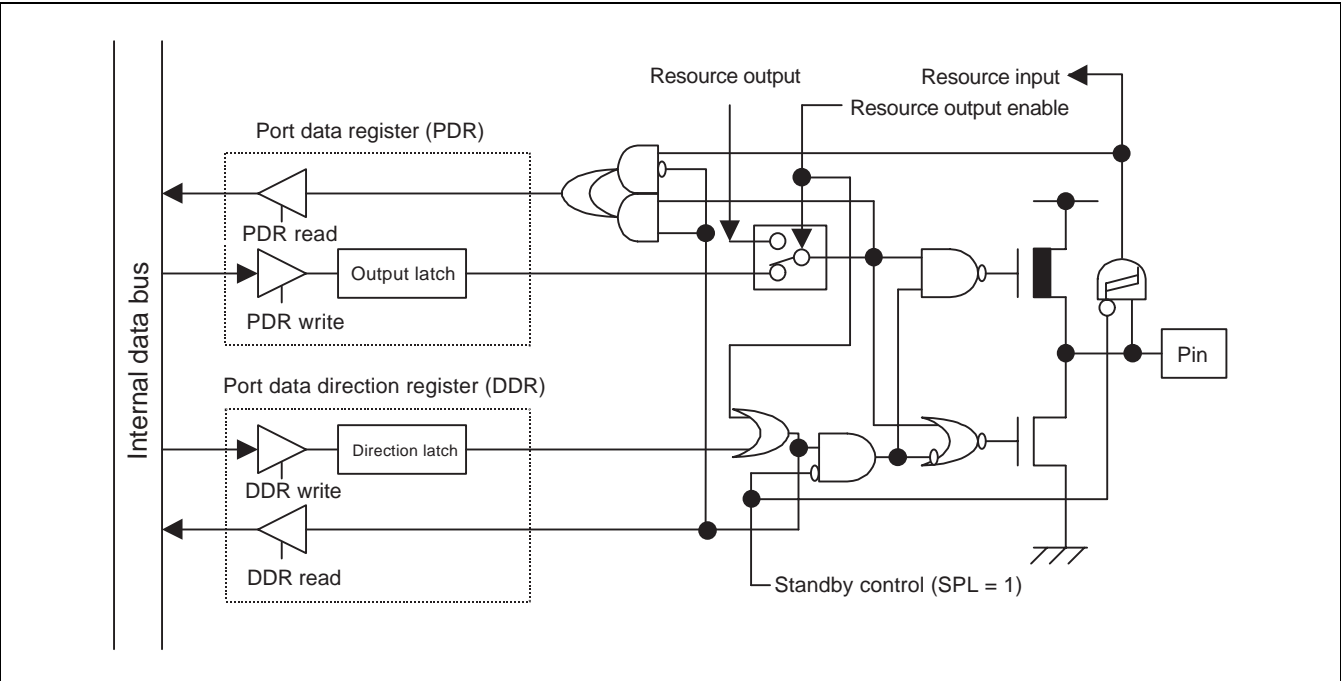
The UART pins also serve as general ports. Table 17.3-1 lists the pin functions, I/O formats and settings required to use UART.

**Table 17.3-1 UART Pins**

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
P40/SIN0	Port 4 I/O or serial data input	CMOS output and CMOS hysteresis input	Not provided	Provided	Set as an input port (DDR4: bit0 = 0)
P41/SOT0	Port 4 I/O or serial data output				Set to output enable mode (SMR0: SOE = 1)
P42/SCK0	Port 4 I/O or serial clock input/output				Set as an input port when a clock is input (DDR4: bit2 = 0)
					Set to output enable mode when a clock is output (SMR0: SCKE = 1)
P60/SIN1	Port 6 I/O or serial data input	CMOS output and CMOS hysteresis input	Not provided	Provided	Set as an input port (DDR6: bit0 = 0)
P61/SOT1	Port 6 I/O or serial data output				Set to output enable mode (SMR1: SOE = 1)
P62/SCK1	Port 6 I/O or serial clock input/output				Set as an input port when a clock is input (DDR6: bit2 = 0)
					Set to output enable mode when a clock is output (SMR1:SCKE = 1)

■ Block Diagram of UART Pins

Figure 17.3-1 Block Diagram of UART Pins

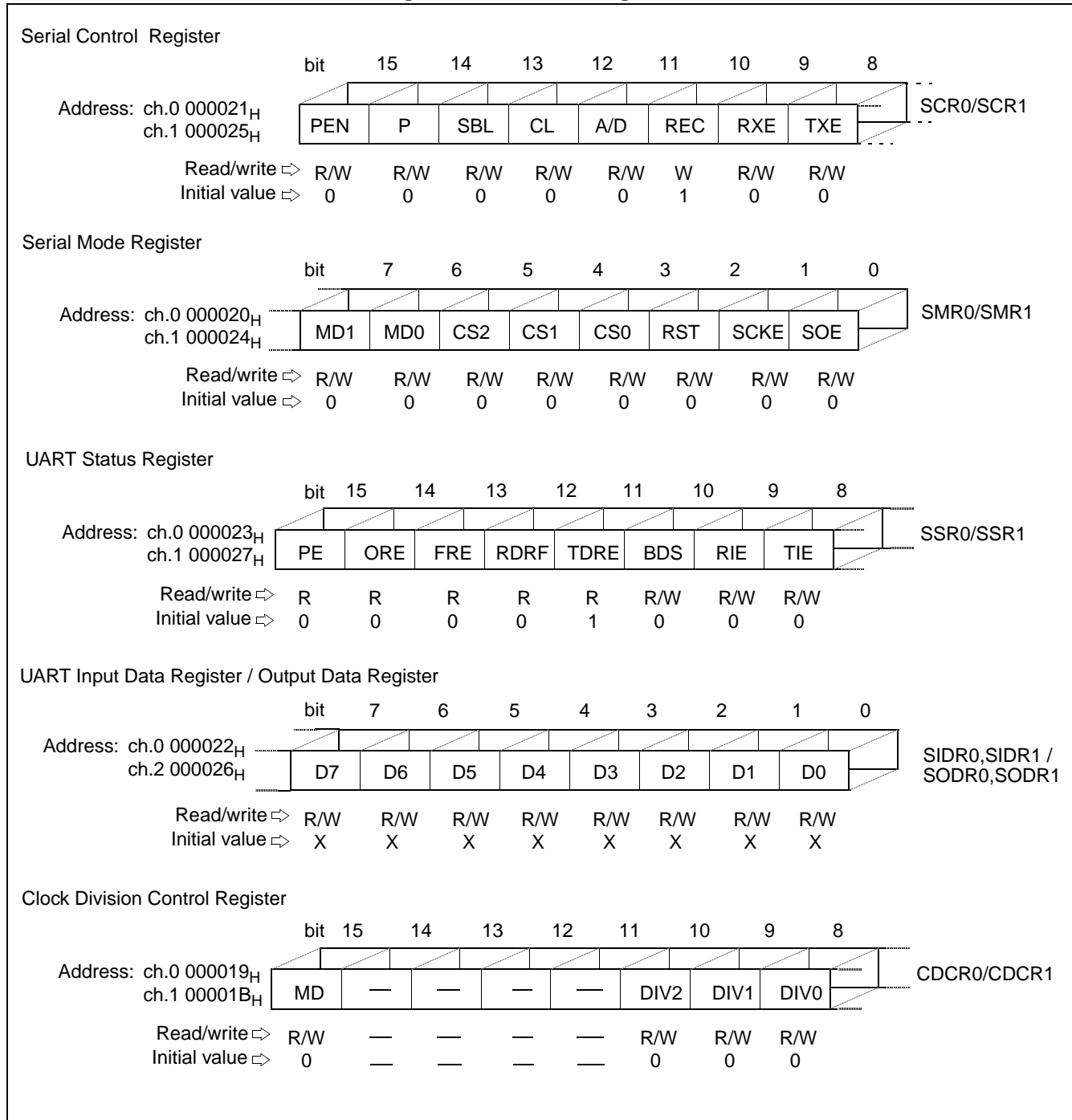


## 17.4 UART Registers

The following figure shows the UART registers.

### ■ UART Registers

Figure 17.4-1 UART Registers

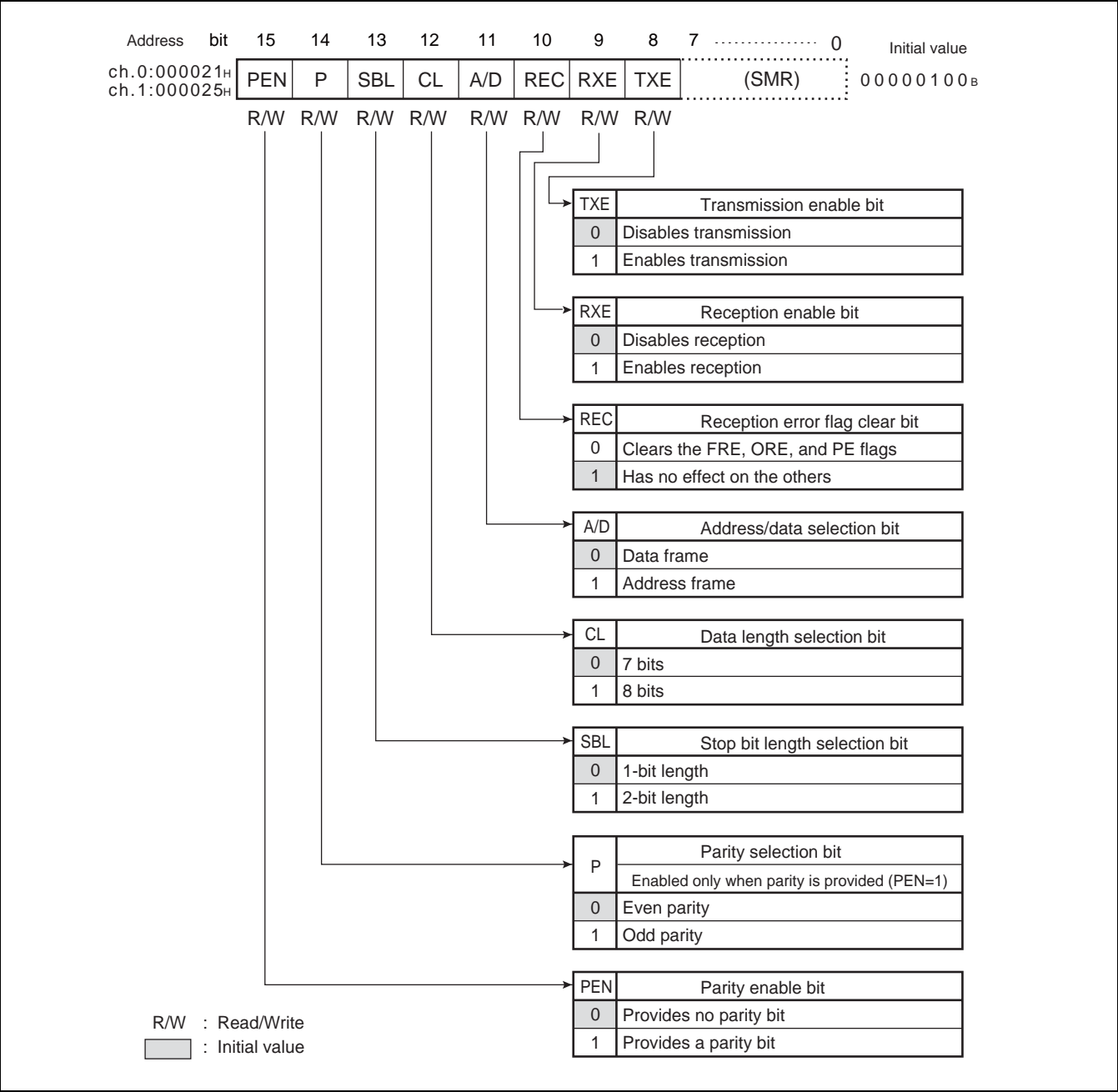


### 17.4.1 Serial Control Register (SCR0/SCR1)

This register specifies parity bits, selects the stop bit and data lengths, selects a frame data format in mode 1, clears the reception error flag, and specifies whether to enable transmission and reception.

■ Serial Control Register (SCR0/SCR1)

Figure 17.4-2 Serial Control Register (SCR0/SCR1)



**Table 17.4-1 Serial Control Register (SCR0/SCR1)**

Bit name		Function
bit15	PEN: Parity enable bit	<ul style="list-style-type: none"> <li>This bit selects whether to add a parity bit during transmission in serial data input-output mode or to detect it during reception.</li> </ul> <p>(Note) No parity can be used in operation modes 1 and 2. Therefore, fix this bit to "0".</p>
bit14	P: Parity selection bit	<ul style="list-style-type: none"> <li>When parity is provided (PEN = 1), this bit selects an even or odd parity.</li> </ul>
bit13	SBL: Stop bit length selection bit	<ul style="list-style-type: none"> <li>This bit selects the length of the stop bits or the frame end mark of send data in asynchronous transfer mode.</li> </ul> <p>(Note) During reception, only the first bit of the stop bits is detected.</p>
bit12	CL: Data length selection bit	<ul style="list-style-type: none"> <li>This bit specifies the length of send and receive data.</li> </ul> <p>(Note) Seven bits can be selected in operation mode 0 (asynchronous) only. Be sure to select eight bits (CL = 1) in operation mode 1 (multiprocessor mode) and operation mode 2 (synchronous).</p>
bit11	A/D: Address/data selection bit	<ul style="list-style-type: none"> <li>Specify the data format of a frame to be sent or received in multiprocessor mode (mode 1).</li> <li>Select usual data when this bit is "0", and select address data when the bit is "1".</li> </ul>
bit10	REC: Reception error flag clear bit	<ul style="list-style-type: none"> <li>This bit clears the FRE, ORE and PE flags of the status register (SSR).</li> <li>Write "0" to this bit to clear the FRE, ORE and PE flag. Writing "1" to this bit has no effect on the others.</li> </ul> <p>(Note) If UART is active and a reception interrupt is enabled, clear the REC bit only when the FRE, DRE or PE flag indicates 1.</p>
bit9	RXE: Reception enable bit	<ul style="list-style-type: none"> <li>This bit controls UART reception.</li> <li>When this bit is "0", reception is disabled. When it is "1", reception is enabled.</li> </ul> <p>(Note) If this bit is cleared during reception, reception can only be disabled until the reception of current frame is completed and the reception data is stored in the reception data is stored in the input data register SIDR0/SIDR1.</p>
bit8	TXE: Transmission enable bit	<ul style="list-style-type: none"> <li>This bit controls UART transmission.</li> <li>When this bit is "0", transmission is disabled. When the bit is "1", transmission is enabled.</li> </ul> <p>(Note) If this bit is cleared during transmission, transmission can only be disabled until all data in the output data register SOR0/SOR1 has been transmitted.</p>

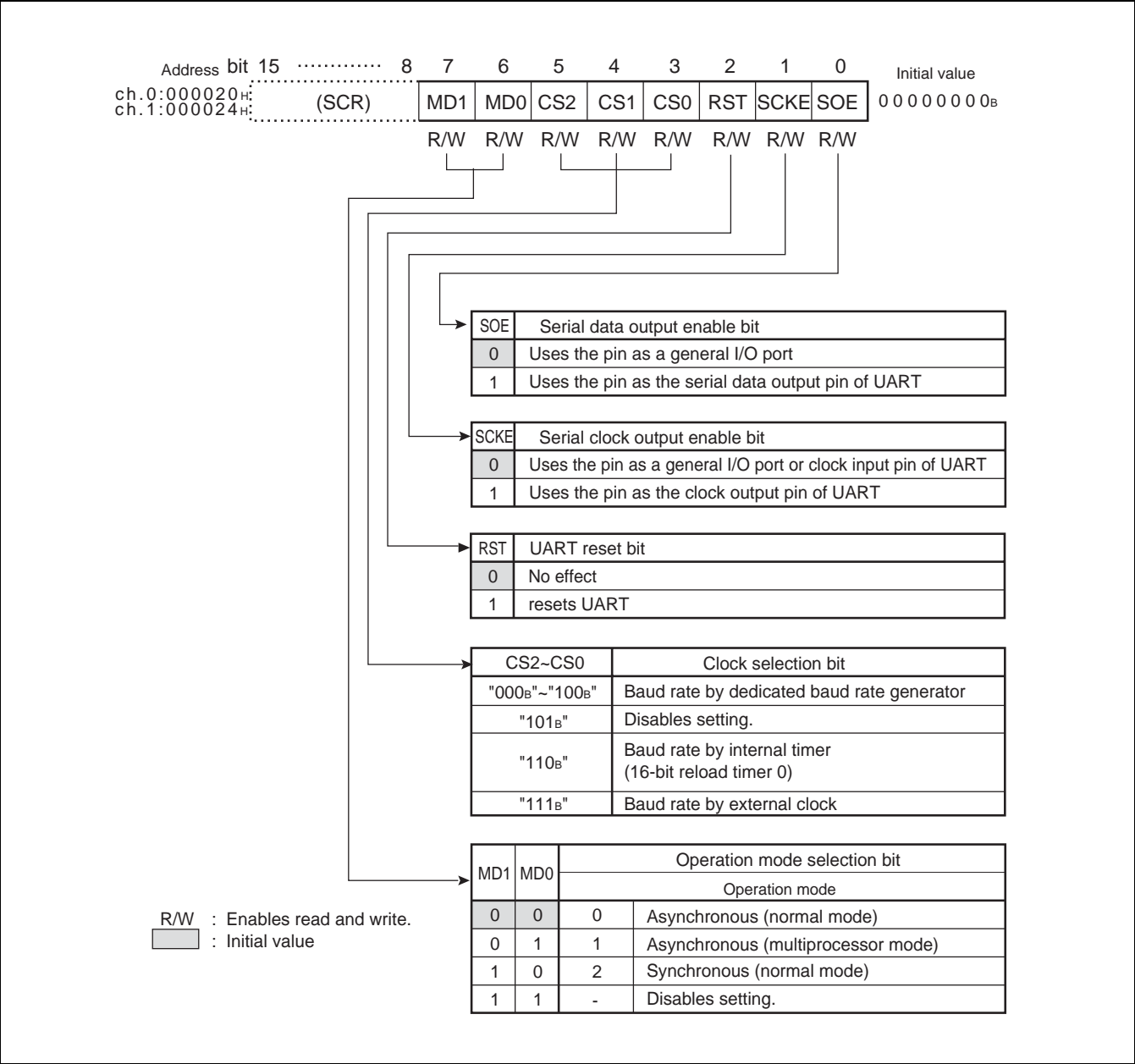


## 17.4.2 Serial Mode Register (SMR0/SMR1)

This register selects an operation mode and baud rate clock and specifies whether to enable output of serial data and clocks to the corresponding pin.

### Serial Mode Register (SMR0/SMR1)

Figure 17.4-3 Serial Mode Register (SMR0/SMR1)



**Table 17.4-2 Serial Mode Register (SMR0/SMR1)**

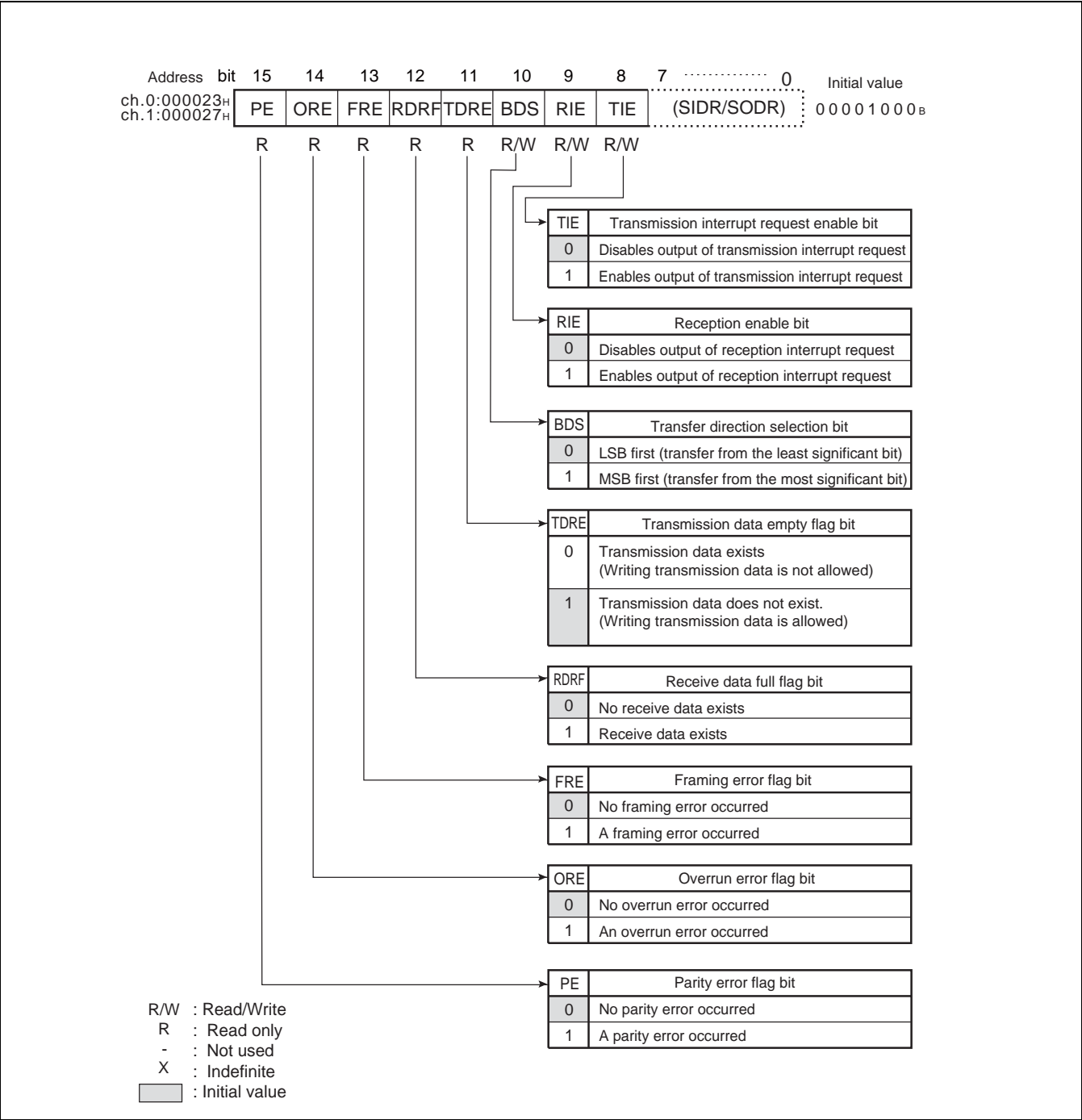
Bit name		Function
bit7, bit6	MD1, MD0: Operation mode selection bits	<ul style="list-style-type: none"> <li>These bits select an operation mode.</li> </ul> <p>(Note)</p> <p>Operation mode 1 (multiprocessor mode) can be used only from the master system during master-slave communication. UART cannot be used from the slave system because it has no address/data detection function during reception.</p>
bit5 to bit3	CS2 to CS0: Clock selection bits	<ul style="list-style-type: none"> <li>These bits select the baud rate clock source. When the dedicated baud rate generator is selected, the baud rate is determined by the value of these bits.</li> <li>When the dedicated baud rate generator is selected, six baud rates can be selected for asynchronous/synchronous transfer mode. With the baud rate generated by internal and external timer, there are totally eight baud rate selections.</li> <li>Input clocks can be selected from external clocks (SCK0/SCK1 pin), 16-bit reload timer 0, and the dedicated baud rate generator.</li> </ul>
bit2	RST: UART reset bit	<ul style="list-style-type: none"> <li>Writing "0" to this bit has no effect.</li> <li>Writing "1" to this bit resets the UART.</li> <li>Always read as "0".</li> </ul>
bit1	SCKE: Serial clock output enable bit	<ul style="list-style-type: none"> <li>This bit controls the serial clock input-output ports.</li> <li>When this bit is "0", the P42/SCK0 and P62/SCK1 pins operate as general input-output ports (P42 and P62) or serial clock input pins. When this bit is "1", the pins operate as serial clock output pins.</li> </ul> <p>(Note)</p> <ul style="list-style-type: none"> <li>When using the P42/SCK0 and P62/SCK1 pins as serial clock input (SCKE = 0) pins, set the P40 and P62 as input ports. Also, select external clocks (SMR0/SMR1: CS2 to CS0 = 111<sub>B</sub>) using the clock selection bits.</li> <li>When using the pins as serial clock output (SCKE = 1) pins, select clocks other than external clocks (other than SMR0/SMR1: CS2 to CS0 = 111<sub>B</sub>).</li> </ul> <p>(Reference)</p> <p>When the SCK0/SCK1 pin is assigned to serial clock output (SCKE = 1), it functions as the serial clock output pin regardless of the status of the general input-output ports.</p>
bit0	SOE: Serial data output enable bit	<ul style="list-style-type: none"> <li>This bit enables or disables the output of serial data.</li> <li>When this bit is "0", the P41/SOT0 and P61/SOT1 pins operate as general input-output ports (P41 and P61). When this bit is "1", the P41/SOT0 and P61/SOT1 pins operate as serial data output pins (SOT0/SOT1).</li> </ul> <p>(Reference)</p> <p>When serial data is output (SOE = 1), the enabled, the P41/SOT0 and P61/SOT1 pins function as SOT0/SOT1 pins regardless of the status of general input-output ports (P41 and P61).</p>

### 17.4.3 Serial Status Register (SSR0/SSR1)

This register checks the transmission and reception status and error status, and enables and disables the transmission and reception interrupts.

■ Serial Status Register (SSR0/SSR1)

Figure 17.4-4 Serial Status Register (SSR0/SSR1)



**Table 17.4-3 Functions of Each Bit of Status Register (SSR0/SSR1)**

Bit name		Function
bit15	PE: Parity error flag bit	<ul style="list-style-type: none"> <li>This bit is set to "1" when a parity error occurs during reception and is cleared when "0" is written to the RFC bit of the mode control register (SMR0/SMR1).</li> <li>A reception interrupt request is output when this bit and the RIE bit are "1".</li> <li>Data in input data register (SIDR0/SIDR1) is invalid when this flag is set.</li> </ul>
bit14	ORE: Overrun error flag bit	<ul style="list-style-type: none"> <li>This bit is set to "1" when an overrun error occurs during reception and is cleared when "0" is written to the RFC bit of the mode control register (SMR0/SMR1).</li> <li>A reception interrupt request is output when this bit and the RIE bit are "1".</li> <li>Data in the input data register (SIDR0/SIDR1) is invalid when this flag is set.</li> </ul>
bit13	FRE: Framing error flag bit	<ul style="list-style-type: none"> <li>This bit is set to "1" when a framing error occurs during reception and is cleared when "0" is written to the RFC bit of the mode control register (SMR0/SMR1).</li> <li>A reception interrupt request is output when this bit and the RIE bit are "1".</li> <li>Data in the input data register (SIDR0/SIDR1) is invalid when this flag is set.</li> </ul>
bit12	RDRF: Receive data full flag bit	<ul style="list-style-type: none"> <li>This flag indicates the status of the input data register (SIDR0/SIDR1).</li> <li>This bit is set to "1" when receive data is loaded into SIDR0/SIDR1 and is cleared to "0" when input data register SIDR0/SIDR1 is read.</li> <li>A reception interrupt request is output when this bit and the RIE bit are "1".</li> </ul>
bit11	TDRE: Transmission data empty flag bit	<ul style="list-style-type: none"> <li>This flag indicates the status of output data register (SODR0/SODR1).</li> <li>This bit is cleared to "0" when transmission data is written to SODR0/SODR1 and is set to "1" when data is loaded into the transmission shift register and transmission starts.</li> <li>A transmission interrupt request is output when this bit and the RIE bit are "1".</li> </ul> <p>(Note) This bit is set to "1" (SODR0/SODR1 empty) as its initial value.</p>
bit10	BDS: Transfer direction selection bit	<ul style="list-style-type: none"> <li>This bit selects whether to transfer serial data from the least significant bit (LSB first, BDS = 0) or the most significant bit (MSB first, BDS = 1).</li> </ul> <p>(Note) The high-order and low-order sides of serial data are interchanged with each other during reading from or writing to the serial data register. If this bit is set to another value after the data is written to the SDR register, the data becomes invalid.</p>
bit9	RIE: Reception interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit enables or disables input of a request for transmission interrupt to the CPU.</li> <li>A reception interrupt request is output when this bit and the receive data flag bit (DRRF) are 1 or this bit and one or more error flag bits (PE, ORE and FRE) are "1".</li> </ul>
bit8	TIE: Transmission interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit enables or disables output of a request for transmission interrupt to the CPU.</li> <li>A transmission interrupt request is output when this bit and the TDRE bit are "1".</li> </ul>

## 17.4.4 Input Data Register (SIDR0/SIDR1) and Output Data Register (SOR0/SOR1)

The input data register (SIDR0/SIDR1) is a serial data reception register. The output data register (SODR0/SODR1) is a serial data transmission register. Both SIDR0/SIDR1 and SODR0/SODR1 registers are located in the same address.

### ■ Input Data Register (SIDR0/SIDR1)

Figure 17.4-5 shows the bit configuration of input data register 1.

**Figure 17.4-5 Input Data Register (SIDR0/SIDR1)**

Serial input data register										
Address : 000022 <sub>H</sub> 000026 <sub>H</sub>	bit	7	6	5	4	3	2	1	0	SIDR0 SIDR1
		D7	D6	D5	D4	D3	D2	D1	D0	
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

SIDR0/SIDR1 is a register that contains receive data. The serial data signal transmitted to the SIN0/SIN1 pin is converted in the shift register and stored there. When the data length is 7 bits, the uppermost bit (D7) contains invalid data. When receive data is stored in this register, the receive data full flag bit (SSR0/SSR1: RDRF) is set to "1". If a reception interrupt request is enabled at this point, a reception interrupt occurs.

Read SIDR0/SIDR1 when the RDRF bit of the status register (SSR0/SSR1) is "1". The RDRF bit is cleared automatically to "0" when SIDR0/SIDR1 is read.

Data in SIDR0/SIDR1 is invalid when a reception error occurs (SSR0/SSR1: PE, ORE or FRE = 1).

### ■ Output Data Register (SODR0/SODR1)

Figure 17.4-6 shows the bit configuration of the output data register.

**Figure 17.4-6 Output Data Register (SODR0/SODR1)**

Serial output data register											
		bit	7	6	5	4	3	2	1	0	
Address : 000022 <sub>H</sub> 000026 <sub>H</sub>			D7	D6	D5	D4	D3	D2	D1	D0	SODR0 SODR1
	Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
	Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

When data to be transmitted is written to this register in transmission enable state, it is transferred to the transmission shift register, then converted to serial data, and transmitted from the serial data output terminal (SOT0/SOT1 pin). When the data length is 7 bits, the uppermost bit (D7) contains invalid data.

When transmission data is written to this register, the transmission data empty flag bit (SS0/SS1: TDRE) is cleared to "0". When transfer to the transmission shift register is complete, the bit is set to "1". When the TDRE bit is "1", the next piece of transmission data can be written. If output transmission interrupt requests have been enabled, a transmission interrupt is generated. Write the next piece of transmission data when a transmission interrupt is generated or the TDRE bit is "1".

---

Note:

SODR0/SODR1 is a write-only register and SIDR0/SIDR1 is a read-only register. These registers are located in the same address, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

---

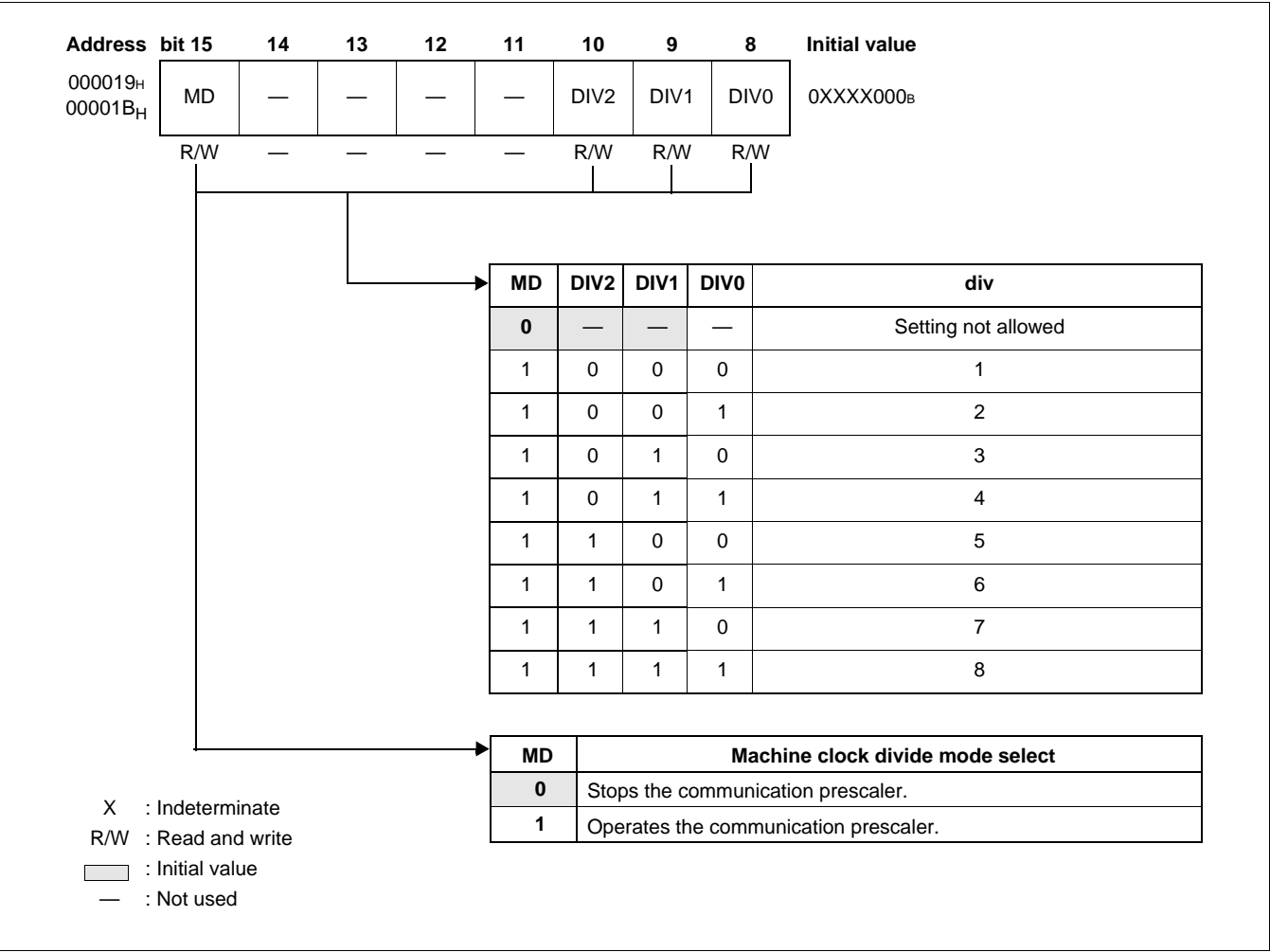
### 17.4.5 Communication Prescaler Control Register (CDCR)

This register controls the division of machine clocks.

■ Communication Prescaler Control Register (CDCR)

The operation clocks of UART can be obtained by dividing machine clocks. UART is designed to obtain certain baud rates for various machine cycles. Output from the communication prescaler is used for the operation clocks of I/O extended serial interfaces. The CDCR bit configuration is shown below.

Figure 17.4-7 Communication Prescaler Control Register



**Table 17.4-4 Communication Prescaler Control Register**

Bit name		Function
bit15	MD: Machine clock divide mode select bit	<ul style="list-style-type: none"> <li>• This bit is the operation enable bit of the communication prescaler.</li> <li>• When "0" is set, the communication prescaler stops.</li> <li>• When "1" is set, the communication prescaler operates.</li> </ul>
bit14 to bit12	Reserved bits	<ul style="list-style-type: none"> <li>• Always read as "0".</li> </ul>
bit10 to bit8	DIV2 to DIV0: Machine clock division bits	<ul style="list-style-type: none"> <li>• These bits determines the machine clock division ratios.</li> <li>• Division ratios can only be set when MD is "1".</li> </ul> (Note) If division ratio is changed, wait 2 cycles as stabilization time before starting communication.



## 17.5 UART Interrupts

UART uses both reception and transmission interrupts. An interrupt request can be generated for either of the receive data is set in the input register (SIDR0/SIDR1), or a reception error occurs and transmission data is transferred from output data register 1 (SODR0/SODR1) to the transmission shift register.

The extended intelligent I-O service (EI<sup>2</sup>OS) is available for these interrupts.

### ■ UART Interrupts

Table 17.5-1 lists the interrupt control bits and causes of UART

**Table 17.5-1 Interrupt Control Bits and Interrupt Causes of UART**

Reception/ transmission	Interrupt request flag bit	Operation mode			Interrupt cause	Interrupt cause enable bit	When interrupt request flag is cleared
		0	1	2			
Reception	RDRF	O	O	O	Loading receive data into buffers (SIDR0/ SIDR1)	SSR0/SSR1:RIE	Receive data is read
	ORE	O	O	O	Overrun error		0 is written to the reception error flag clear bit (SSR0/SSR1: REC)
	FRE	O	O	X	Framing error		
	PE	O	X	X	Parity error		
Transmission	TDRE	O	O	O	Empty transmission buffer (SODR0/ SODR1)	SSR0/SSR1:TIE	Transmission data is written

O: Used

X: Not used

#### ● Reception interrupt

If one of the following events occurs in reception mode, the corresponding flag bit of the status register is set to "1":

- Data reception is complete (SSR0/SSR1: RDRF)
- Overrun error (SSR0/SSR1: ORE)
- Framing error (SSR0/SSR1: FRE)
- Parity error (SSR0/SSR1: PE)

When at least one of the flag bits is "1" and the reception interrupts are enabled (SSR0/SSR1: RIE = 1), a reception interrupt request is output to the interrupt controller.

When the input data register (SIDR0/SIDR1) is read, the receive data full flag (SSR0/SSR1: RDRF) is automatically cleared to "0". When "0" is written to the REC bit of the control register (SCR0/SCR1), all the reception error flags (SSR0/SSR1: PE, ORE and FRE) are cleared to "0".

### ● Transmission interrupt

When transmission data is transferred from the output data register (SODR0/SODR1) to the transfer shift register, the TDRE bit of the status register (SSR0/SSR1) is set to "1". When the transmission interrupts have been enabled (SSR0/SSR1: TIE = 1), a transmission interrupt request is output to the interrupt controller.

## ■ UART Interrupts and EI<sup>2</sup>OS

**Table 17.5-2 UART Interrupts and EI<sup>2</sup>OS**

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25 <sub>H</sub> )	ICR13	0000BD <sub>H</sub>	FFFF68 <sub>H</sub>	FFFF69 <sub>H</sub>	FFFF6A <sub>H</sub>	⊙
UART1 transmission interrupt	#38(26 <sub>H</sub> )	ICR13	0000BD <sub>H</sub>	FFFF64 <sub>H</sub>	FFFF65 <sub>H</sub>	FFFF66 <sub>H</sub>	Δ
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>	⊙
UART0 transmission interrupt	#40(28 <sub>H</sub> )	ICR14	0000BE <sub>H</sub>	FFFF5C <sub>H</sub>	FFFF5D <sub>H</sub>	FFFF5E <sub>H</sub>	Δ

⊙ : Provided with a function that detects a UART reception error and stops EI<sup>2</sup>OS

Δ : Usable when interrupt causes that share the ICR13 and ICR14 or the interrupt vectors are not used

## ■ UART EI<sup>2</sup>OS Functions

UART has a circuit for operating EI<sup>2</sup>OS, which can be started up for either reception or transmission interrupts.

### ● For reception

EI<sup>2</sup>OS can be used regardless of the status of other resources.

### ● For transmission

UART shares the interrupt registers (ICR13 and ICR14) with the UART reception interrupts. Therefore, EI<sup>2</sup>OS can be started up only when no UART reception interrupts are used.

## 17.5.1 Reception Interrupt Generation and Flag Set Timing

The following are the reception interrupt causes: completion of reception (SSR0/SSR1: RDRF) and occurrence of a reception error (SSR0/SSR1: PE, ORE, or FRE).

### ■ Reception Interrupt Generation and Flag Set Timing

Receive data is stored in input data register 1 (SIDR0/SIDR1) if a stop bit is detected (in operation mode 0 or 1) or the last bit of data is detected (in operation mode 2) during reception. If a reception error is detected, the error flags (SSR0/SSR1: PE, ORE and FRE) are set, then the receive data flag (SSR0/SSR1: RDRF) is set to "1". If one of the error flags is "1" in each mode, the SIDR0/SIDR1 register contains invalid data.

#### ● Operation mode 0 (asynchronous, normal mode)

The RDRF bit is set to "1" when a stop bit is detected. If a reception error is detected, the error flags (PE, ORE and FRE) are set.

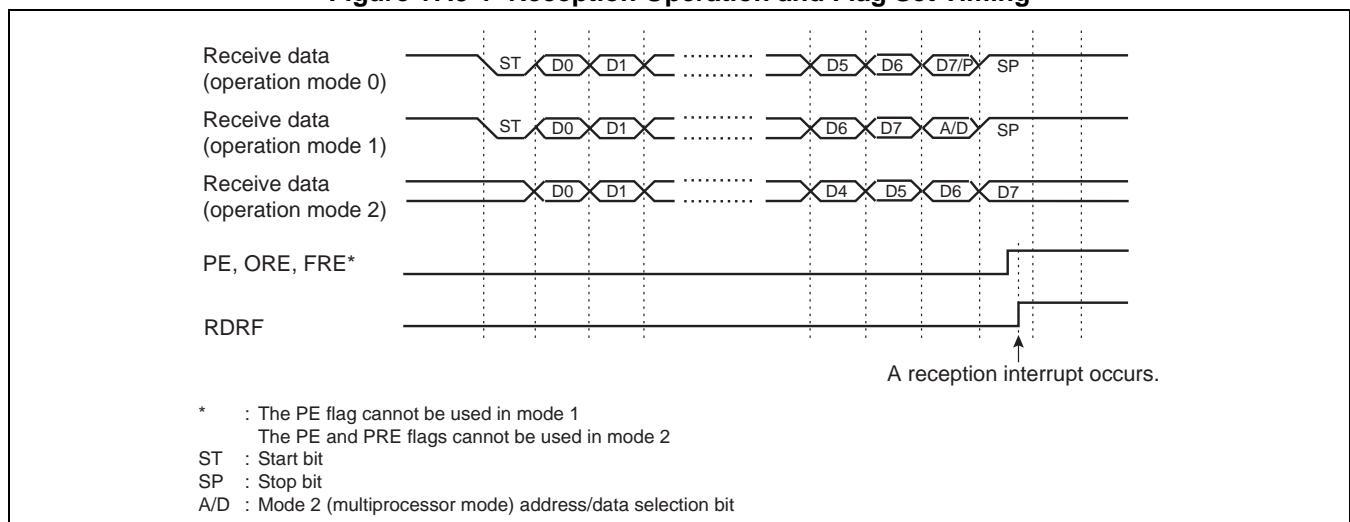
#### ● Operation mode 1 (asynchronous, multiprocessor mode)

The RDRF bit is set to "1" when a stop bit is detected. If a reception error is detected, the error flags (ORE and FRE) are set. Parity errors cannot be detected.

#### ● Operation mode 2 (synchronous, normal mode)

The RDRF bit is set when the last bit of receive data (D7) is detected. If a reception error is detected, the error flag (ORE) is set. Parity and framing errors cannot be detected. Figure 17.5-1 below shows the reception operation and flag set timing.

**Figure 17.5-1 Reception Operation and Flag Set Timing**



#### ● Reception interrupt generation timing

When the RDRF, PE, ORE or FRE flag is set to "1" in the reception interrupt enable state (SSR0/SSR1: RIE = 1), reception interrupt requests (#37 and #39) are generated.

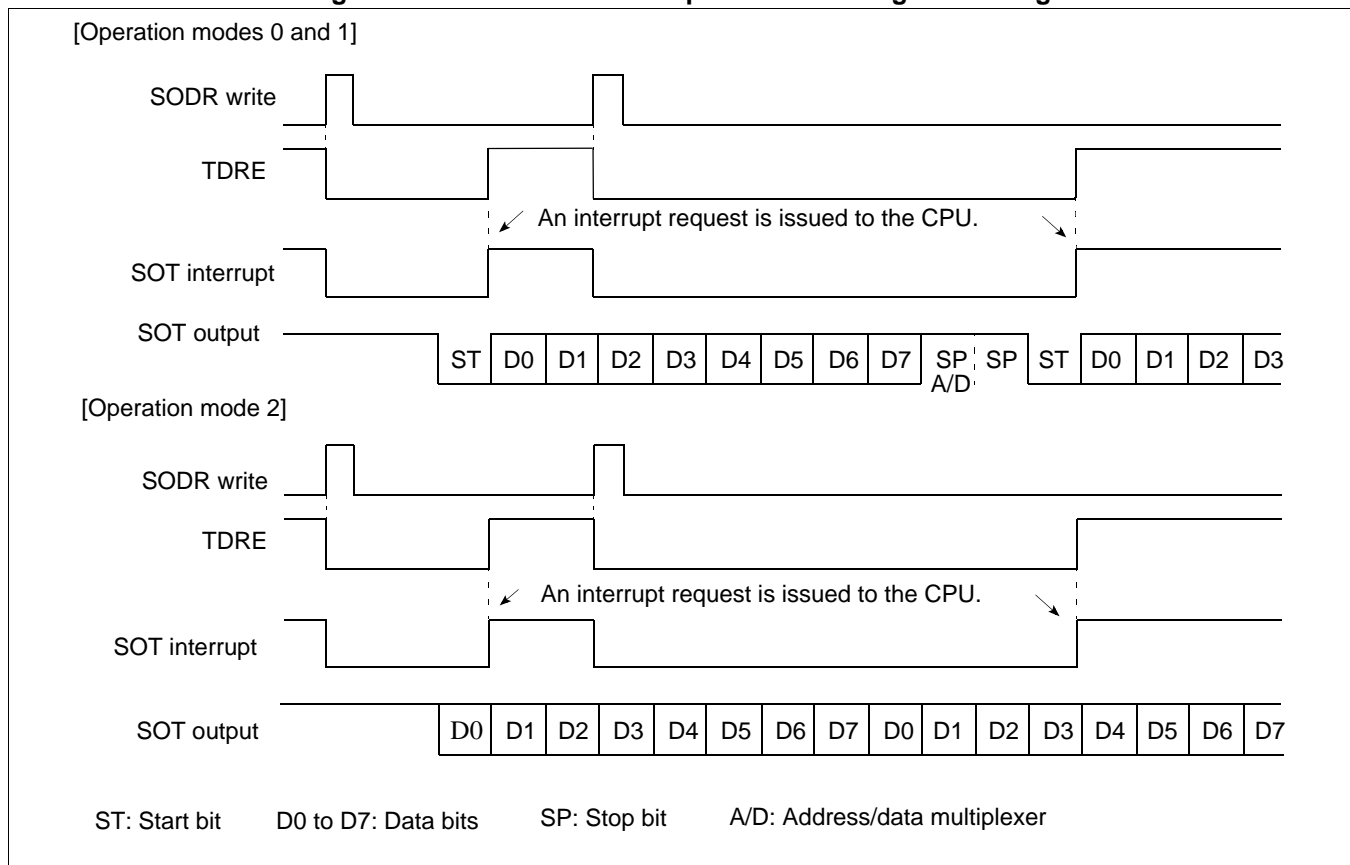
## 17.5.2 Transmission Interrupt Generation and Flag Set Timing

A transmission interrupt is generated when the next piece of data is ready to be written to the output data register (SODR0/SODR1).

### ■ Transmission Interrupt Generation and Flag Set Timing

The transmission data empty flag bit (SSR0/SSR1: TDRE) is set to "1" when data written to the output data register (SODR0/SODR1) is transferred to the transmission shift register, and the next piece of data is ready to be written. TDRE is cleared to "0" when transmission data is written to SODR0/SODR1. Figure 17.5-2 shows the transmission operation and flag set timing.

**Figure 17.5-2 Transmission Operation and Flag Set Timing**



#### ● Transmission interrupt request generation timing

If the TDRE flag is set to "1" when a transmission interrupt is enabled (SSR0/SSR1: TIE = 1), transmission interrupt requests (#38 and #40) are generated.

#### Note:

A transmission completion interrupt is generated immediately after the transmission interrupts are enabled (TIE = 1) because the TDRE bit is set to "1" as its initial value. TDRE is a read-only bit that can be cleared only by writing new data to the output data register (SODR0/SODR1). Carefully specify the transmission interrupt enable timing.

## 17.6 UART Baud Rates

---

**One of the following can be selected as the UART transmitting/receiving block, the block diagram show as below.**

---

### ■ UART Baud Rate Selection

The baud rate selection circuit is designed as shown below. One of the following three types of baud rates can be selected:

- Baud rates determined using the dedicated baud rate generator

UART has an internal dedicated baud rate generator. One of eight baud rates can be selected using the mode control register (SMR0/SMR1).

An asynchronous or synchronous baud rate is selected using the machine clock frequency by setting the CS2 to CS0 bits of the mode control register (SMR0/SMR1).

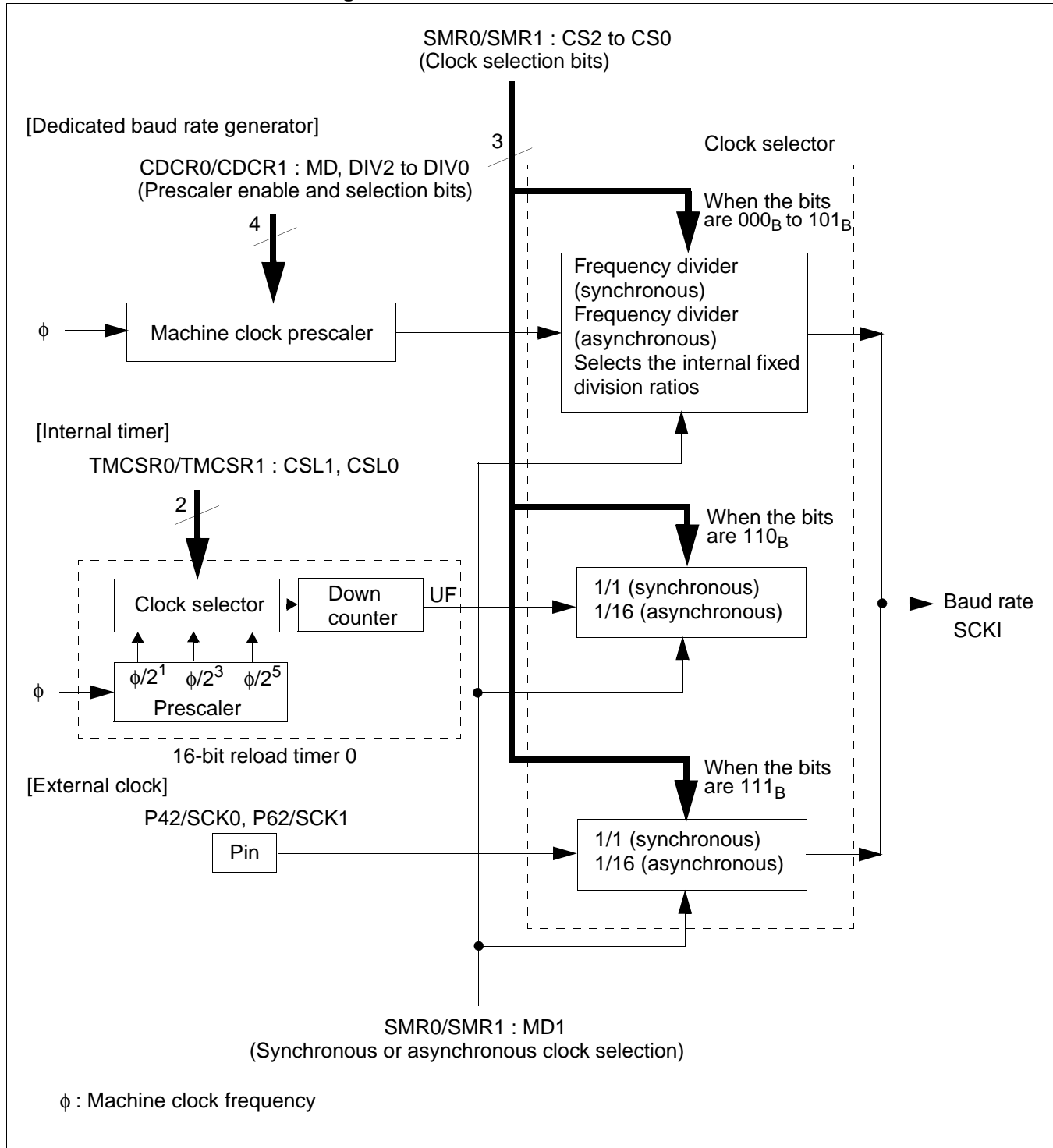
- Baud rates determined using the internal timer

The internal clock supplied from 16-bit reload timer 0 is used as it is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set by setting the reload value.

- Baud rates determined using the external clock

The clock input from the UART clock pulse input pins (P42/SCK0 and P62/SCK1) is used as it is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set externally.

Figure 17.6-1 Baud Rate Selection Circuit



## 17.6.1 Baud Rates Determined Using the Dedicated Baud Rate Generator

This section describes the baud rates that can be set when the clock from the dedicated baud rate generator is selected as the UART transfer clock.

### ■ Baud Rates determined using the Dedicated Baud Rate Generator

When the transfer clock is generated using the dedicated baud rate generator, the machine clock is divided with the machine clock prescaler. The divided machine clock is then divided by the transfer clock division ratio selected with the clock selector again. The machine clock division ratios are common to the asynchronous and synchronous baud rates, but different values set internally are selected as the transfer clock division ratio for the asynchronous and synchronous baud rates.

The actual transfer rate can be calculated using the following formulas:

asynchronous baud rate =  $\phi \times (\text{prescaler division ratio}) \times (\text{asynchronous transfer clock division ratio})$

synchronous baud rate =  $\phi \times (\text{prescaler division ratio}) \times (\text{synchronous transfer clock division ratio})$

$\phi$  : Machine clock frequency

### ● Division ratios for the prescaler (common to asynchronous and synchronous baud rates)

Each machine clock division ratio is selected using the DIV2 to DIV0 bits of the CDCR register as listed in Table 17.6-1

**Table 17.6-1 Selection of Each Division Ratio for the Machine Clock Prescaler**

MD	DIV2	DIV1	DIV0	div
0	–	–	–	Setting not allowed
1	0	0	0	1
1	0	0	1	2
1	0	1	0	3
1	0	1	1	4
1	1	0	0	5
1	1	0	1	6
1	1	1	0	7
1	1	1	1	8

### ● Synchronous transfer clock division ratios

A division ratio for synchronous baud rates is selected using the CS2 to CS0 bits of the mode control register (SMR0/SMR1) as listed in Table 17.6-2.

**Table 17.6-2 Selection of Synchronous Baud Rate Division Ratios**

CS2	CS1	CS0	CLK synchronization	Calculation formula
0	0	0	2 MHz	$(\phi \div \text{div}) / 1$
0	0	1	1 MHz	$(\phi \div \text{div}) / 2$
0	1	0	500 kHz	$(\phi \div \text{div}) / 4$
0	1	1	250 kHz	$(\phi \div \text{div}) / 8$
1	0	0	125 kHz	$(\phi \div \text{div}) / 16$
1	0	1	62.5 kHz	$(\phi \div \text{div}) / 32$

Note that the calculation is supposing that  $\phi$  (machine cycle) = 16 MHz and div (machine clock division ratio) = 8. The maximum baud rate is 1/8 machine clock.

### ● Asynchronous transfer clock division ratios

A division ratio for asynchronous baud rates is selected using the CS2 to CS0 bits of the mode control register (SMR0/SMR1) as listed in Table 17.6-3.

**Table 17.6-3 Selection of Asynchronous Baud Rate Division Ratios**

CS2	CS1	CS0	Asynchronous (start-stop synchronization)	Calculation formula
0	0	0	76923 Hz	$(\phi \div \text{div}) / (8 \times 13 \times 2)$
0	0	1	38461 Hz	$(\phi \div \text{div}) / (8 \times 13 \times 4)$
0	1	0	19230 Hz	$(\phi \div \text{div}) / (8 \times 13 \times 8)$
0	1	1	9615 Hz	$(\phi \div \text{div}) / (8 \times 13 \times 16)$
1	0	0	500 kHz	$(\phi \div \text{div}) / (8 \times 2 \times 2)$
1	0	1	250 kHz	$(\phi \div \text{div}) / (8 \times 2 \times 4)$

Note that the calculation is supposing that  $\phi$  (machine clock) = 16 MHz, div (machine clock division ratio) = 1.



### ● Internal timer

When CS2 to CS0 are set to "110<sub>B</sub>" and the internal timer is selected, the formulas for calculating baud rates (when using the reload timer) are as follows:

Asynchronous (start-stop synchronization):  $(\phi \div N) / (16 \times 2 \times (n + 1))$

CLK synchronization:  $(\phi \div N) / (2 \times (n + 1))$

N: Division ratio for the prescaler of 16-bit re-load timer count clock

n: reload value of the 16-bit reload timer

#### Note:

In mode 2 (CLK synchronization mode), SCK0/SCK1 is up to three clocks later than SCKI. A logically attainable transfer rate is 1/3 of the system clock frequency. However, 1/4 of the system clock frequency is recommended as taken from the actual specifications.

### ● External clock

When CS2 to CS0 are set to "111<sub>B</sub>" and the external clock is selected, note the following:

If the external clock frequency is specified as f, the following baud rates are assumed:

Asynchronous (start-stop synchronization):  $f/16$

CLK synchronization: f

Note that the maximum external clock frequency f is 2 MHz.

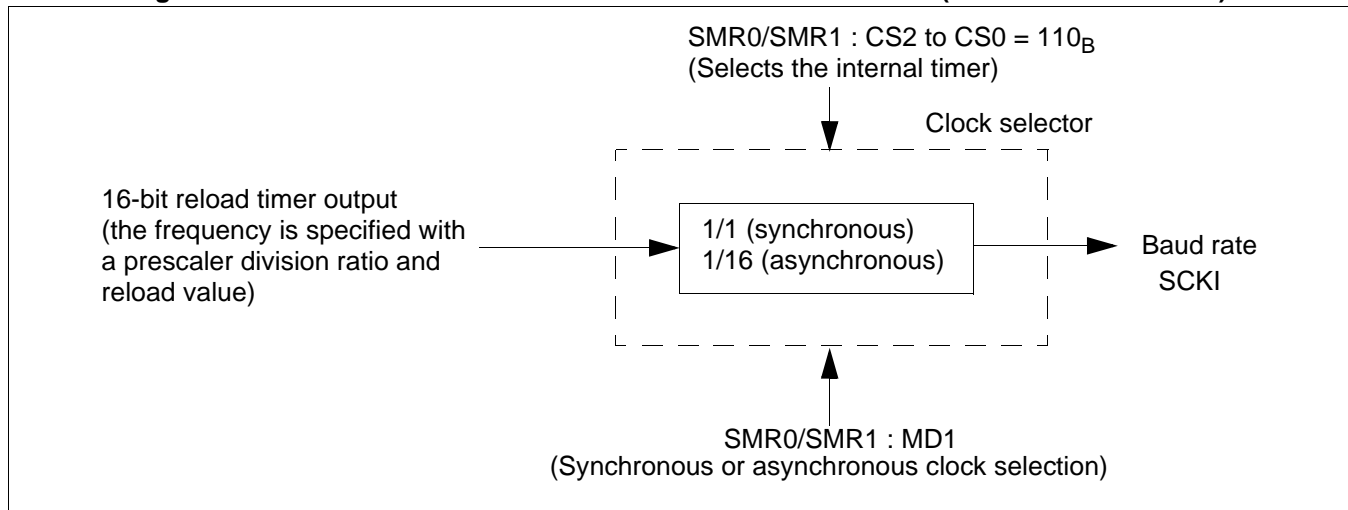
## 17.6.2 Baud Rates Determined Using the Internal Timer (16-bit Reload Timer 0)

This section describes the settings used when the internal clock supplied from 16-bit reload timer 0 is selected as the UART transfer clock. It also shows the baud rate calculation formulas.

### ■ Baud Rates determined using the Internal Timer (16-bit Reload Timer 0)

Writing  $110_B$  to the CS2 to CS0 bits of the mode control register (SMR0/SMR1) selects the baud rate determined using the internal timer. Any baud rate can be set by specifying a prescaler division ratio and reload value for 16-bit reload timer 0. Figure 17.6-2 shows the baud rate selection circuit for the internal timer.

**Figure 17.6-2 Baud Rate Selection Circuit for the Internal Timer (16-bit Reload Timer 0)**



### ● Baud rate calculation formulas

$$\text{Asynchronous baud rate} = \frac{\phi}{X(n+1) \times 2 \times 16} \text{ bps}$$

$$\text{Synchronous baud rate} = \frac{\phi}{X(n+1) \times 2} \text{ bps}$$

$\phi$ : Machine clock frequency

X: Division ratio for the prescaler of 16-bit reload timer 0 ( $2^1, 2^3, 2^5$ )

n: Reload value for 16-bit reload timer 0 (0 to 65535)

- Examples of setting reload values (machine clock: 7.3728 MHz)

**Table 17.6-4 Baud Rates and Reload Values**

Baud rate (bps)	Reload value			
	Clock asynchronous (start-stop synchronization)		Clock synchronous	
	$X=2^1$ (machine cycle divided by 2)	$X=2^3$ (machine cycle divided by 8)	$X=2^1$ (machine cycle divided by 2)	$X=2^3$ (machine cycle divided by 8)
38400	2	–	47	11
19200	5	–	95	23
9600	11	2	191	47
4800	23	5	383	95
2400	47	11	767	191
1200	95	23	1535	383
600	191	47	9071	767
300	383	95	6143	1535

X: Division ratio for the prescaler of 16-bit reload timer 0

- : Setting not allowed

### 17.6.3 Baud Rates Determined Using the External Clock

This section describes the settings used when the external clock is selected as the UART transfer clock. It also shows the baud rate calculation formulas.

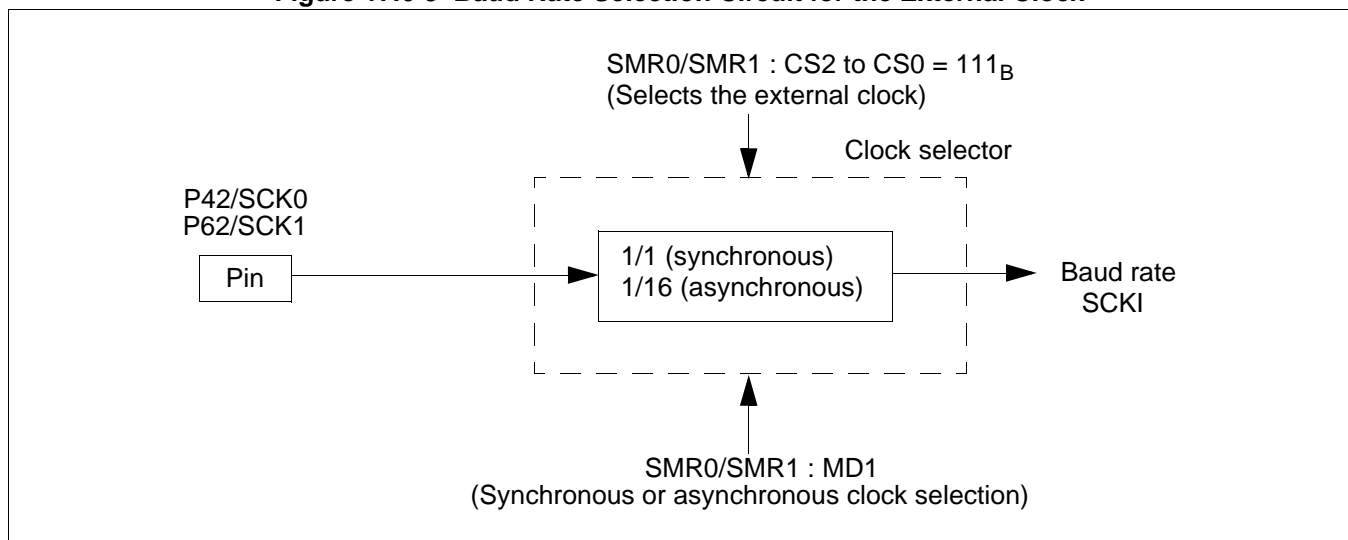
#### ■ Baud Rates determined using the External Clock

The following three settings are required to select the baud rate determined by using the external clock:

- Write  $111_B$  to the CS2 to CS0 bits of the mode control register (SMR0/SMR1) to select the baud rate determined by using the external clock input.
- Set the P42/SCK0 and P62/SCK1 pins as input ports (DDR4: bit2 = 0 and DDR6: bit2 = 0).
- Write "0" to the SCKE bit of the mode control register (SMR0/SMR1) to set the pin as an external clock input pin.

As shown in Figure 17.6-3, a baud rate is selected using the external clock input from the SCK1 pin. To change the baud rate, the external input clock cycle must be changed because the internal division ratio is fixed.

**Figure 17.6-3 Baud Rate Selection Circuit for the External Clock**



#### ● Baud rate calculation formulas

Asynchronous baud rate =  $f/16$

Synchronous baud rate =  $f$

$f$ : External clock frequency (up to 2 MHz)

## 17.7 Operation of UART

---

**UART operates in operation modes 0 and 2 for normal bidirectional serial communication and in operation mode 1 for master-slave communication.**

---

### ■ Operation of UART

#### ● Operation modes

There are three UART operation modes: modes 0 to 2. As listed in Table 17.7-1, an operation mode can be selected according to the inter-CPU connection method and data transfer mode

**Table 17.7-1 UART Operation Mode**

Operation mode		Data length		Synchronization mode	Stop bit length
		When parity is disabled	When parity is enabled		
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits *2
1	Multiprocessor	8+1*1 bits	—	Asynchronous	
2	Normal mode	8 bits	—	Synchronous	None

—: Setting not possible.

\*1: "+1" indicates the address/data selection bit (A/D) for communication control.

\*2: During reception, only one stop bit can be detected.

---

Note:

Operation mode 1 of UART is used only from the master system during master-slave connection.

---

#### ● Inter-CPU connection method

One-to-one connection (normal mode) and master-slave connection (multiprocessor mode) can be selected. For either connection method, the data length, whether to enable parity, and the synchronization method must be common to all CPUs. Select an operation mode as follows:

- In the one-to-one connection method, operation mode 0 or 2 must be used in the two CPUs. Select operation mode 0 for asynchronous transfer mode and operation mode 2 for synchronous transfer mode.
- Select operation mode 1 for the master-slave connection method and use it from the master system. Select "When parity is disabled" for this connection method.

#### ● Synchronization method

Asynchronous mode (start-stop synchronization) or clock synchronous mode can be selected in different operation modes.

#### ● Signal mode

UART can treat data only in NRZ (Non-return to Zero) format.

### ● Operation enable bit

UART controls both transmission and reception using the operation enable bit for TXE (transmission) and that for RXE (reception). If each of the operations is disabled, stop it as follows:

- If reception operation is disabled during reception (data is input to the reception shift register), finish frame reception and store the received data in the input data register (SIDRI). Then stop the reception operation.
- If the transmission operation is disabled during transmission (data is output from the transmission shift register), wait until there is no data in the output data register (SODR0/SODR1) before stopping the transmission operation.

# 17.7.1      Operation in Asynchronous Mode (Operation Modes 0 and 1)

When UART is used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode), the asynchronous transfer mode is selected.

## ■ Operation in Asynchronous Mode

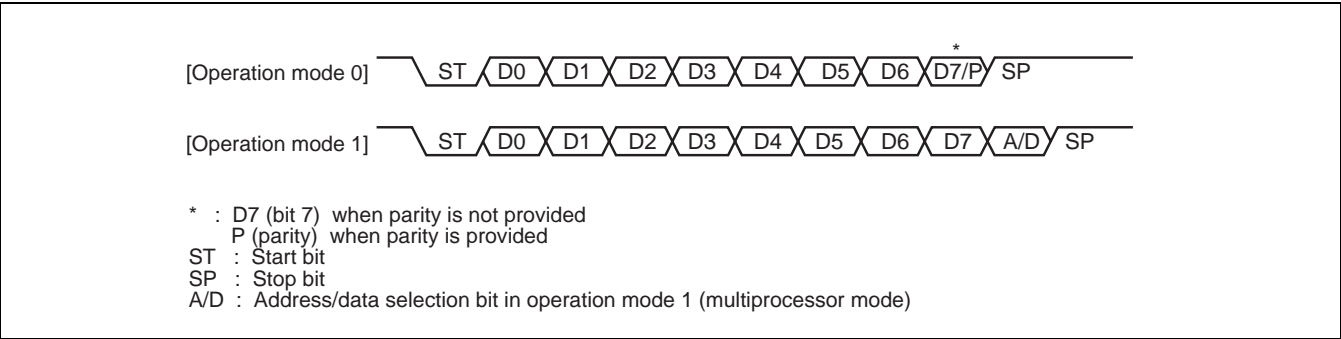
### ● Transfer data format

Transfer data begins with the start bit (L level) and ends with the stop bit (H level). The data of the specified data bit length is transferred in LSB first mode.

- In operation mode 0, the length of data with no parity is fixed to 7 bits, and that of data with parity is fixed to 8 bits.
- In operation mode 1, the length of data is fixed to 8 bits with an address/data (A/D) selection bit added instead of parity.

Figure 17.7-1 shows the data format in asynchronous mode.

**Figure 17.7-1 Transfer Data Format (Operation Modes 0 and 1)**



### ● Transmission operation

Transmission data is written to the output data register (SODR0/SODR1) when the transmission data empty flag bit (SSR0/SSR1: TDRE) is "1". This data is transmitted if the transmission operation is enabled (SCR0/SCR1: TXE = 1).

The TDRE flag is again set to "1" when the transmission data is transferred to the transmission shift register and its transmission starts. Then, the next piece of transmission data gets ready to be set. At this point, a transmission interrupt request is output requesting that the next piece of transmission data be set in the SODR0/SODR1 register if that request is enabled (SSR0/SSR1: TIE = 1). The TDRE flag is cleared to "0" when the transmission data is written to SODR0/SODR1.

### ● Reception operation

Reception operation is performed every time it is enabled (SCR0/SCR1: RXE = 1). When a start bit is detected, a frame of data is received according to the data format specified by the control register (SCR0/SCR1). After the frame has been received, the error flag is set if an error occurs, then the receive data full flag bit (SSR0/SSR1: RDRF) is set to "1". At this point, a reception interrupt request is output if it is enabled (SSR0/1: TIE = 1).

Check each flag of the input data register (SIDR0/SIDR1). If the reception is normal, read the input data register (SIDR0/SIDR1). If an error is found, proceed to error handling. The RDRF flag is cleared to "0" every time receive data is read from SIDR0/SIDR1.

### ● Stop bit

For transmission, 1 or 2 bits can be selected. During reception however, the first bit is the only one that is always checked.

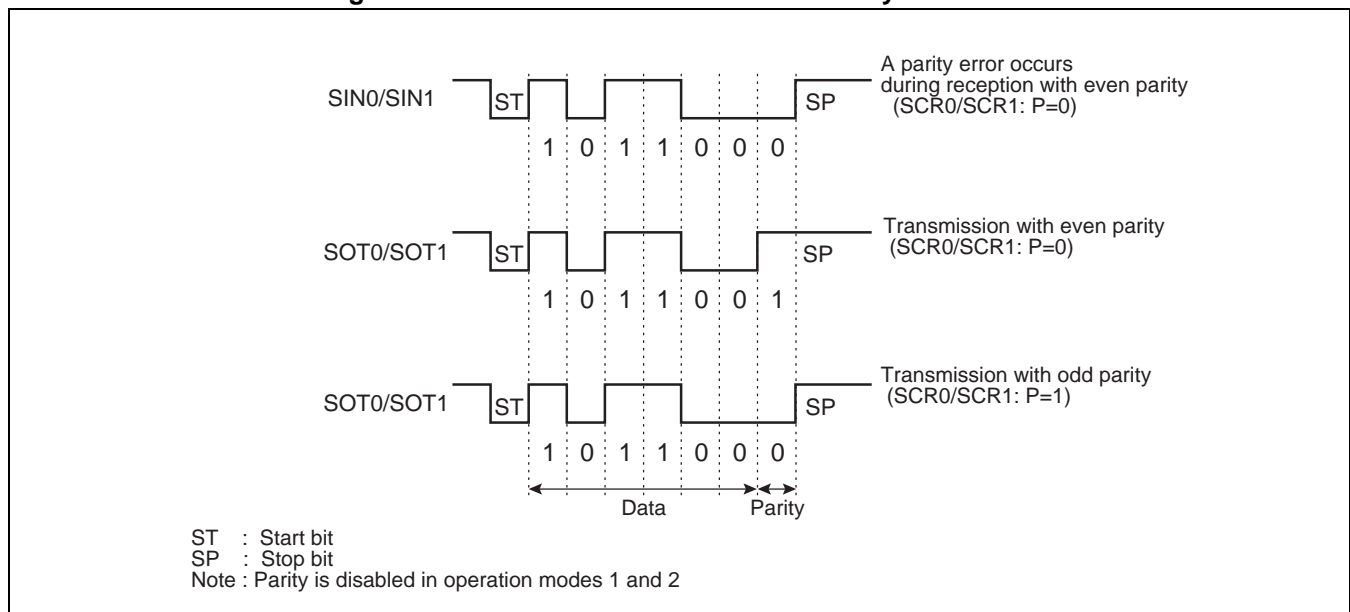
### ● Error detection

- In mode 0, parity, overrun and framing errors can be detected.
- In mode 1, overrun and framing errors can be detected but parity errors cannot be detected.

### ● Parity 0

Parity can only be used in operation mode 0 (asynchronous, normal mode). Whether to provide parity can be specified using the PEN bit of the control register (SCR0/SCR1). Even or odd parity can also be specified using the P bit of the control register (SDR0/SDR1). In operation mode 1 (asynchronous, multiprocessor mode) and operation mode 2 (synchronous, normal mode), parity cannot be used. Figure 17.7-2 shows both transmission and receive data when parity is enabled.

**Figure 17.7-2 Transmission Data when Parity is enabled**





## 17.7.2 Operation in Synchronous Mode (Operation Mode 2)

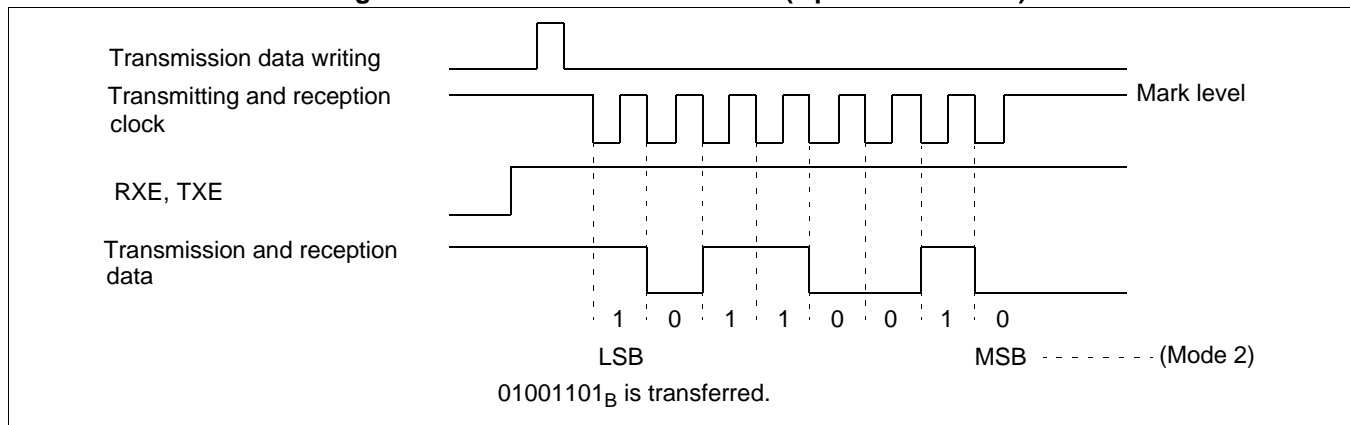
The clock synchronous transfer method is used for UART operation mode 2 (normal mode).

### ■ Operation in Synchronous Mode (Operation Mode 2)

#### ● Transfer data format

In synchronous mode, 8-bit data is transferred using the LSB first method, in which start and stop bits are not added. Figure 17.7-3 shows the data format in clock synchronous mode.

**Figure 17.7-3 Transfer Data Format (Operation Mode 2)**



#### ● Clock supply

In clock synchronous mode (I/O extended serial), as many clocks as the number of transmission and reception bits must be supplied.

- When the internal clock (dedicated baud rate generator or internal timer) is selected, the data receiving synchronous clocks is generated automatically if data is transmitted.
- When the external clock is selected, confirm that the transmission side UART output data register (SODR0/SODR1) contains data (SSR0/SSR1: TDRE = 0). Then, clocks for just 1 byte must be supplied from outside.

The mark level (H) must be retained before transmission starts and after it is complete.

#### ● Error detection

Only overrun errors can be detected; parity and framing errors cannot be detected.

## ● Initialization

The following shows the set values of each control register using the synchronous mode:

[Mode control register (SMR0/SMR1)]

MD1,MD0:"10<sub>B</sub>"

CS2,CS1,CS0:Specify clock input using the clock selector.

SCKE:1 for dedicated baud rate generator or internal timer

0 for clock output and external clock (clock input)

SOE:1 for transmission; 0 for reception only

[Control register (SCR0/SCR1)]

PEN:"0"

P,SBL,A/D:These bits make no sense.

CL:1 (8-bit data)

REC:0 (the error flag is cleared for initialization.)

RXE,TXE:At least one of the two bits is set to "1".

[Status register (SSR0/SSR1)]

RIE:1 when using interrupts; 0 when using no interrupts.

TIE:1 when using interrupts; 0 when using no interrupts.

## ● Starting communication

Write data to the output data register (SODR0/SODR1) to start communication. Temporary data must be written to SODR0/SODR1 to start communication for reception.

## ● Ending communication

The RDRF flag of the status register (SSR0/SSR1) is set to "1" when transmission or reception of a data frame is complete. During reception, check the overrun error flag bit (SSR0/SSR1) to see if communication is performing normally.

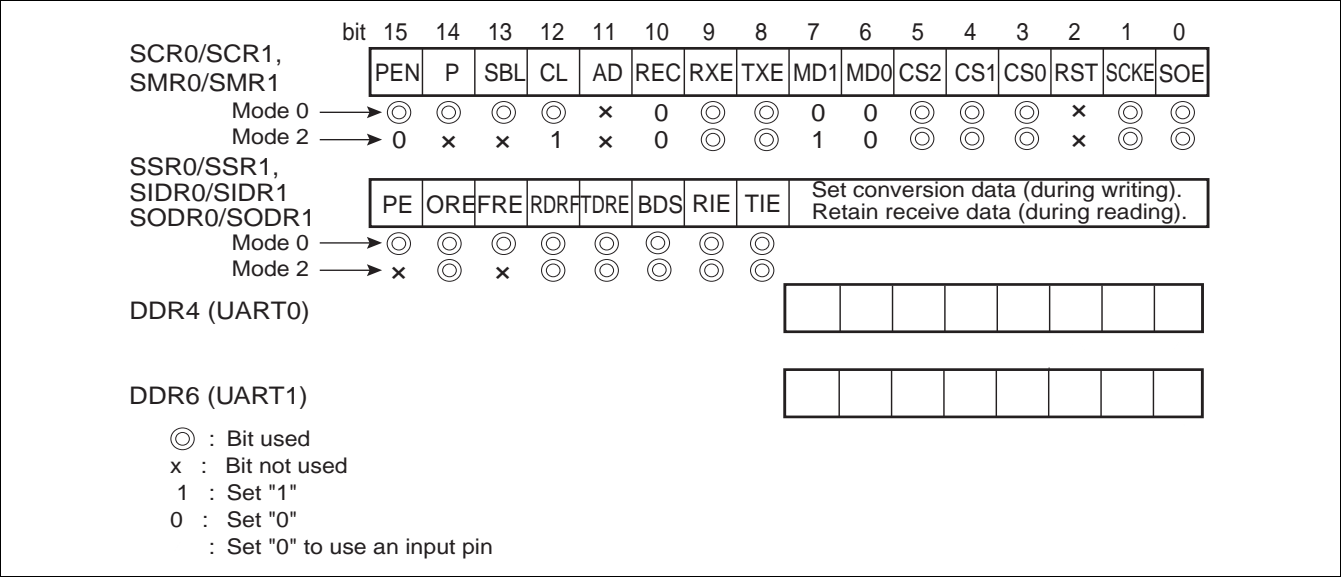
### 17.7.3 Bidirectional Communication Function (Normal Mode)

In operation mode 0 or 2, normal serial bidirectional communication (one-to-one connection) is available. Select operation mode 0 for asynchronous communication and operation mode 2 for synchronous communication.

#### ■ Bidirectional Communication Function

The settings shown in Figure 17.7-4 are required to operate UART in normal mode (operation mode 0 or 2).

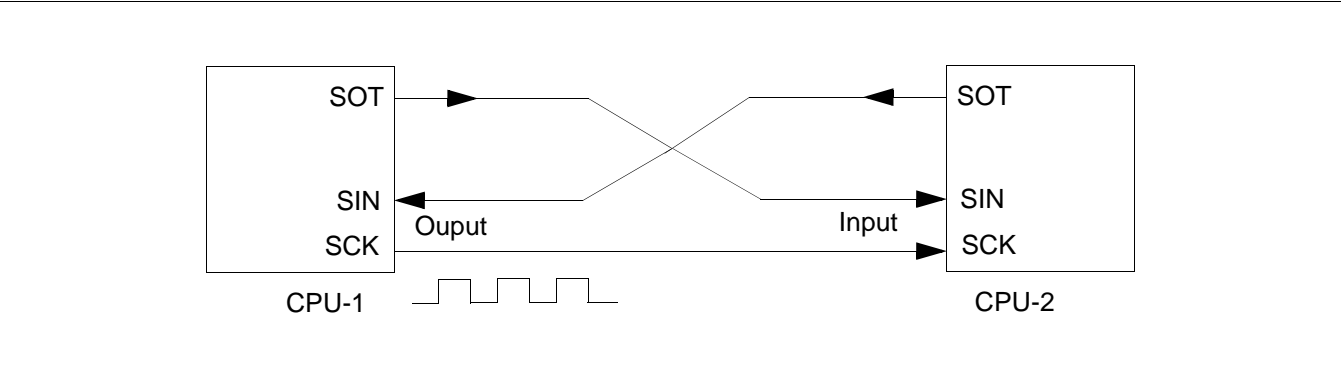
Figure 17.7-4 Settings for UART Operation Mode 0



#### ● Inter-CPU connection

As shown in Figure 17.7-5, interconnect two CPU's.

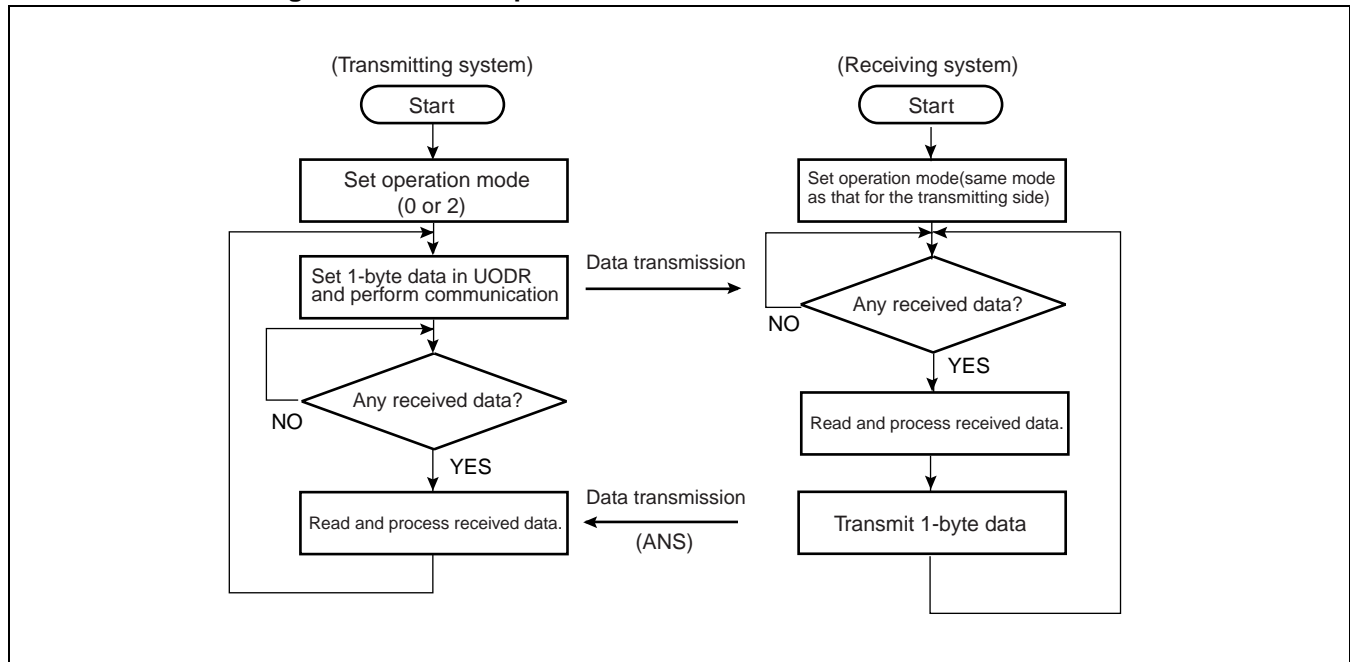
Figure 17.7-5 Connection Example of UART Bidirectional Communication



### ● Communication procedure

Communication starts from the transmitting system at an optional timing when transmission data has been prepared. An ANS is returned periodically (byte by byte in this example) when the receiving system receives transmission data. Figure 17.7-6 shows an example of a bidirectional communication flowchart.

**Figure 17.7-6 Example of Bidirectional Communication Flowchart**



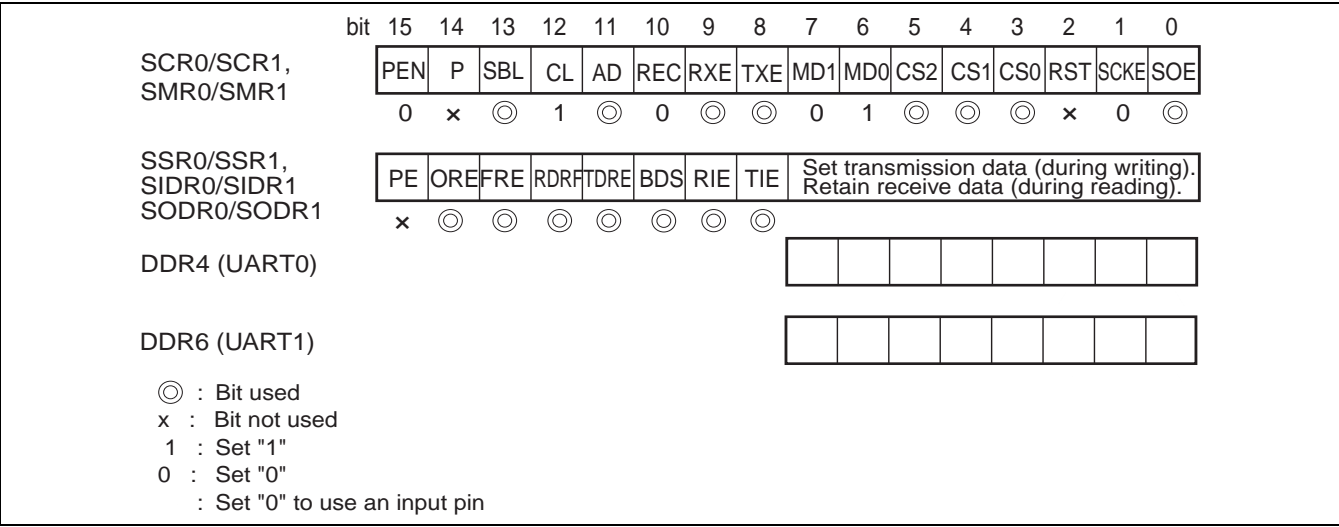
# 17.7.4 Master-slave Communication Function (Multiprocessor Mode)

With UART, communication with multiple CPUs connected in master-slave mode is available in operation mode 1. However, UART can be used only from the master system.

## ■ Master-slave Communication Function

The settings shown in Figure 17.7-7 are required to operate UART in multiprocessor mode (operation mode 1).

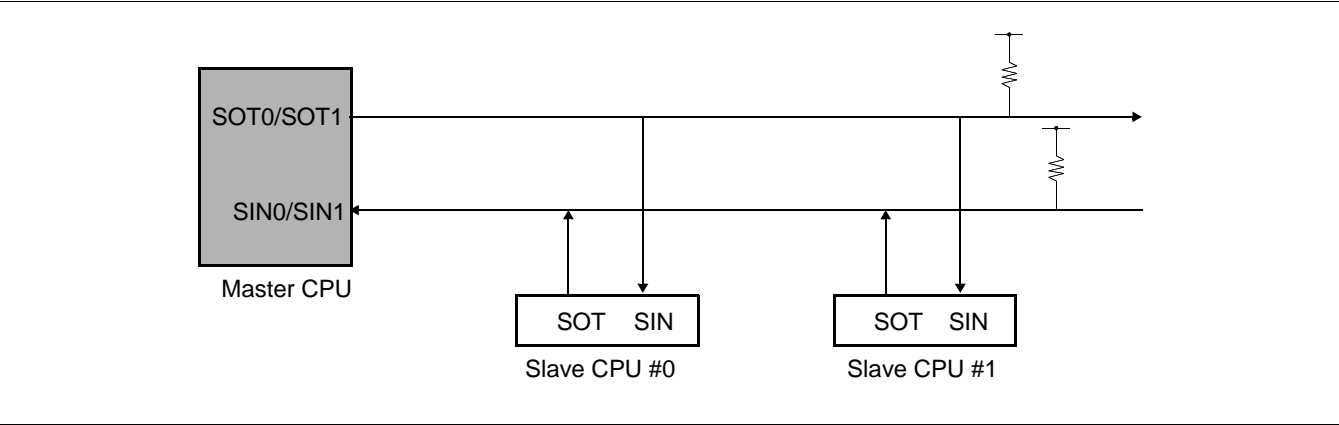
Figure 17.7-7 Settings for UART Operation Mode 1



### ● Inter-CPU connection

As shown in Figure 17.7-8, a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. UART can be used only from the master CPU.

Figure 17.7-8 Connection Example of UART Master-slave Communication



### ● Function selection

Select the operation mode and data transfer mode for master-slave communication as shown in Table

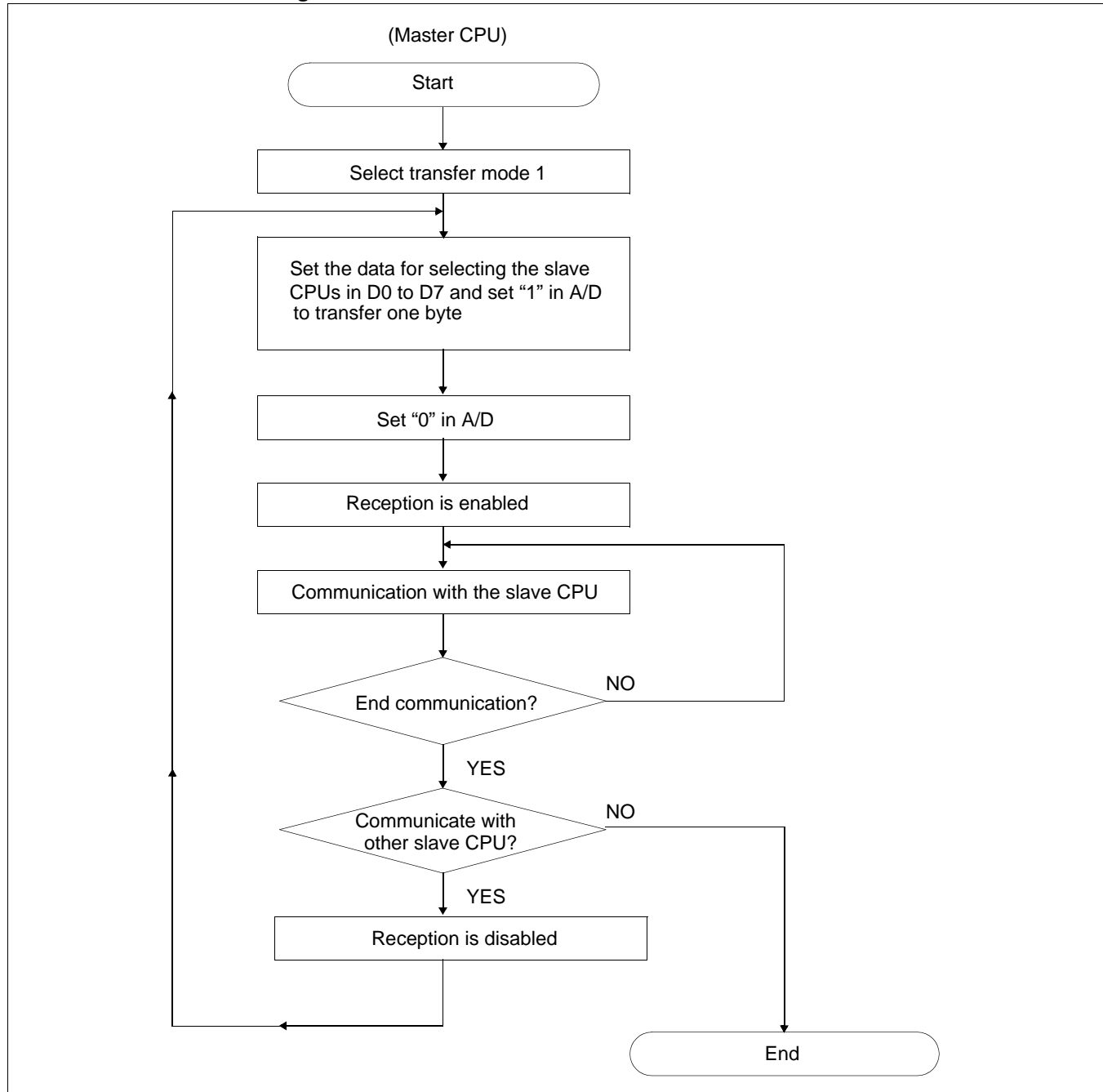
**Table 17.7-2 Selection of the Master-slave Communication Function**

	Operation mode		Data	Parity	Synchronizati on method	Stop bit
	Master CPU	Slave CPU				
Address transmission and reception	Mode 1	—	A/D = 1 + 8-bit address	None	Asynchronous	1 or 2 bits
Data transmission and reception			A/D = 0 + 8-bit data			

### ● Communication procedure

When the master CPU transmits address data, communication starts. The A/D bit in the address data is set to "1", and the communication destination slave CPU is selected. Each slave CPU checks the address data using a program. When the address data indicates the address assigned to a slave CPU, the slave CPU communicates with the master CPU (ordinary data).

Figure 17.7-9 shows a flowchart of master-slave communication (multiprocessor mode).

**Figure 17.7-9 Master-slave Communication Flowchart**

## 17.8 Usage Notes on UART

---

Notes on using UART are given below.

---

### ■ Notes on using UART

#### ● Enabling operations

In UART, the control register (SCR0/SCR1) has both TXE (transmission) and RXE (reception) operation enable bits. Both transmission and reception operations must be enabled before the transfer starts because they have been disabled as the default value (initial value). The transfer can also be canceled by disabling its operation as required.

#### ● Communication mode setting

Set the communication mode while the system is not operating. If the mode is set during transmission or reception, the transmission or reception data is not guaranteed.

#### ● Synchronous mode

UART clock synchronous mode (operation mode 2) uses clock control (I/O extended serial) mode, in which start and stop bits are not added to the data.

#### ● Transmission interrupt enabling timing

The default (initial value) of the transmission data empty flag bit (SSR0/SSR1: TRE) is "1" (no transmission data and transmission data write enable state). A transmission interrupt request is generated as soon as the transmission interrupt requests are enabled (SSR0/SSR1: TIE = 1). Be sure to set the TIE flag to "1" after setting the transmission data.



## 17.9 Sample Program for UART

---

**This section contains a sample program for UART.**

---

### ■ Sample Program for UART

#### ● Processing specifications

The UART1 bidirectional communication function (normal mode) is used to perform serial transmission and reception.

- Operation mode 0, asynchronous mode, eight data bits, two stop bits and no parity are set.
- The P60/SIN1 and P61/SOT1 pins are used for communication.
- The dedicated baud rate generator is used and the baud rate is set to about 9600 bps.
- Character 13<sub>H</sub> is transmitted from the SOT1 pin and is received using an interrupt.
- The machine clock ( $\phi$ ) is assumed to be 16 MHz.

#### ● Coding example

```

ICR13      EQU      0000BDH      ;UART1 transmission and reception interrupt control register
DDR6       EQU      000016H      ;Port-6 data direction register
CDCR1      EQU      00001BH      ;Communication prescaler register 1
SMR1       EQU      000024H      ;Mode control register 1
SCR1       EQU      000025H      ;Control register 1
SIDR1      EQU      000026H      ;Input data register 1
SODR1      EQU      000026H      ;Output data register 1
SSR1       EQU      000027H      ;Status register 1
REC        EQU      SCR1:2       ;Reception error flag clear bit

;-----Main program-----
CODE       CSEG      ABS = 0FFH
START:
;          :          ;Assumes that stack pointer (SP) has already been
;          :          ; initialized
          AND      CCR,#0BFH      ;Disables interrupts
          MOV      I:ICR13,#00H    ;Interrupt level 0 (highest)
          MOV      I:DDR6,#00000000B ;Sets SIN1 pin as input pin
          MOV      I:CDCR1,#080H    ;Enables communication prescaler
          MOV      I:SMR1,#00010001B ;Operation mode 0 (asynchronous)
          :          ;Uses dedicated baud rate generator (9615 bps)
          :          ;Disables clock pulse output and enables data output
          MOV      I:SCR1,#00010011B ;No parity and two stop bits
          :          ;Clears eight data bits and reception error flag

```

```

                                ;Enables transmission and reception operations
                                ;Disables transmission interrupts and enables reception
                                ; interrupts
MOV    I:SSR1,#00000010B
MOV    I:SODR1,#13H           ;Writes transmission data
MOV    ILM,#07H               ;Sets ILM in PS to level 7
OR     CCR,#40H               ;Enables interrupts
LOOP:  MOV    A,#00H           ;Endless loop
      MOV    A,#01H
      BRA    LOOP

;-----Interrupt program-----
WARI:
      MOV    A,SIDR1           ;Reads receive data
      CLRB   I:REC            ;Clears reception interrupt request flag
;      :
;      User processing
;      :
      RETI                     ;Returns from the interrupt
CODE   ENDS

;-----Vector setting-----
VECT   CSEG    ABS=0FFH
      ORG     0FF68H           ;Sets vector for interrupt #37 (25H)
      DSL     WARI
      ORG     0FFDCH           ;Sets reset vector
      DSL     START
      DB      00H              ;Sets single-chip mode
VECT   ENDS

```



# **CHAPTER 18**

---

# ***DTP/EXTERNAL INTERRUPT CIRCUIT***

**This chapter describes the functions and operation of the DTP/external interrupt circuit.**

- 18.1 Overview of the DTP/External Interrupt Circuit
- 18.2 Block Diagram of the DTP/External Interrupt Circuit
- 18.3 DTP/External Interrupt Circuit Pins
- 18.4 DTP/External Interrupt Circuit Registers
- 18.5 Operation of the DTP/External Interrupt Circuit
- 18.6 Usage Notes on the DTP/External Interrupt Circuit
- 18.7 Sample Programs for the DTP/External Interrupt Circuit

## 18.1 Overview of the DTP/External Interrupt Circuit

The data transfer peripheral (DTP)/external interrupt circuit is located between external peripherals and the F<sup>2</sup>MC-16LX CPU. It receives interrupt requests and data transfer requests from peripherals and passes them to the CPU to generate external interrupt requests or activate the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ DTP/external Interrupt Functions

The DTP/external interrupt circuit is activated by the signal supplied to a DTP/external interrupt pin. The CPU accepts the signal using the same procedure it uses for normal hardware interrupts and generates external interrupts or activates the extended intelligent I/O service (EI<sup>2</sup>OS).

If the extended intelligent I/O service (EI<sup>2</sup>OS) is disabled when an interrupt request is accepted by the CPU, the circuit executes its external interrupt function and branches to an interrupt routine. If EI<sup>2</sup>OS is enabled, the circuit executes its DTP function, which performs automatic data transfer using EI<sup>2</sup>OS and branches to an interrupt processing routine after the data transfer has been performed a specified number of times.

Table 18.1-1 provides an overview of the DTP/external interrupt circuit.

**Table 18.1-1 Overview of the DTP/external Interrupt Circuit**

	External interrupt function	DTP function
Input pins	Eight (P10/INT0/DTTI0 to P16/INT6, P63/INT7)	
Interrupt cause	By using the request level setting register (ELVR), the level or edge to be detected can be selected for each pin	
	Input of H level or L level or rising edge or falling edge	Input of H level or L level
Interrupt number	#20 (14 <sub>H</sub> ), #22 (16 <sub>H</sub> ), #25 (19 <sub>H</sub> ), #27 (1B <sub>H</sub> )	
Interrupt control	The output of interrupt requests is enabled and disabled using the DTP/interrupt enable register (ENIR)	
Interrupt flag	Interrupt causes are stored in the DTP/interrupt cause register (EIRR)	
Processing selection	EI <sup>2</sup> OS is disabled (ICR: ISE = 0)	EI <sup>2</sup> OS is enabled (ICR: ISE = 1)
Processing	The circuit branches to an external interrupt processing routine	The circuit performs automatic data transfer using EI <sup>2</sup> OS for a specified number of times and then branches to an interrupt routine

ICR: Interrupt control register

## ■ Interrupt of the DTP/external Interrupt Circuit and EI<sup>2</sup>OS

**Table 18.1-2 Interrupt of the DTP/external Interrupt Circuit and EI<sup>2</sup>OS**

Channel	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS
		Register name	Address	Lower	Middle	Upper	
INT0/INT1	#20 (14 <sub>H</sub> )	ICR04	0000B4 <sub>H</sub>	FFFFAC <sub>H</sub>	FFFFAD <sub>H</sub>	FFFFAE <sub>H</sub>	O
INT2/INT3	#22 (16 <sub>H</sub> )	ICR05	0000B5 <sub>H</sub>	FFFA4 <sub>H</sub>	FFFA5 <sub>H</sub>	FFFA6 <sub>H</sub>	
INT4/INT5	#25 (19 <sub>H</sub> )	ICR07	0000B7 <sub>H</sub>	FFF98 <sub>H</sub>	FFF99 <sub>H</sub>	FFF9A <sub>H</sub>	
INT6/INT7	#27 (1B <sub>H</sub> )	ICR08	0000B8 <sub>H</sub>	FFF90 <sub>H</sub>	FFF91 <sub>H</sub>	FFF92 <sub>H</sub>	

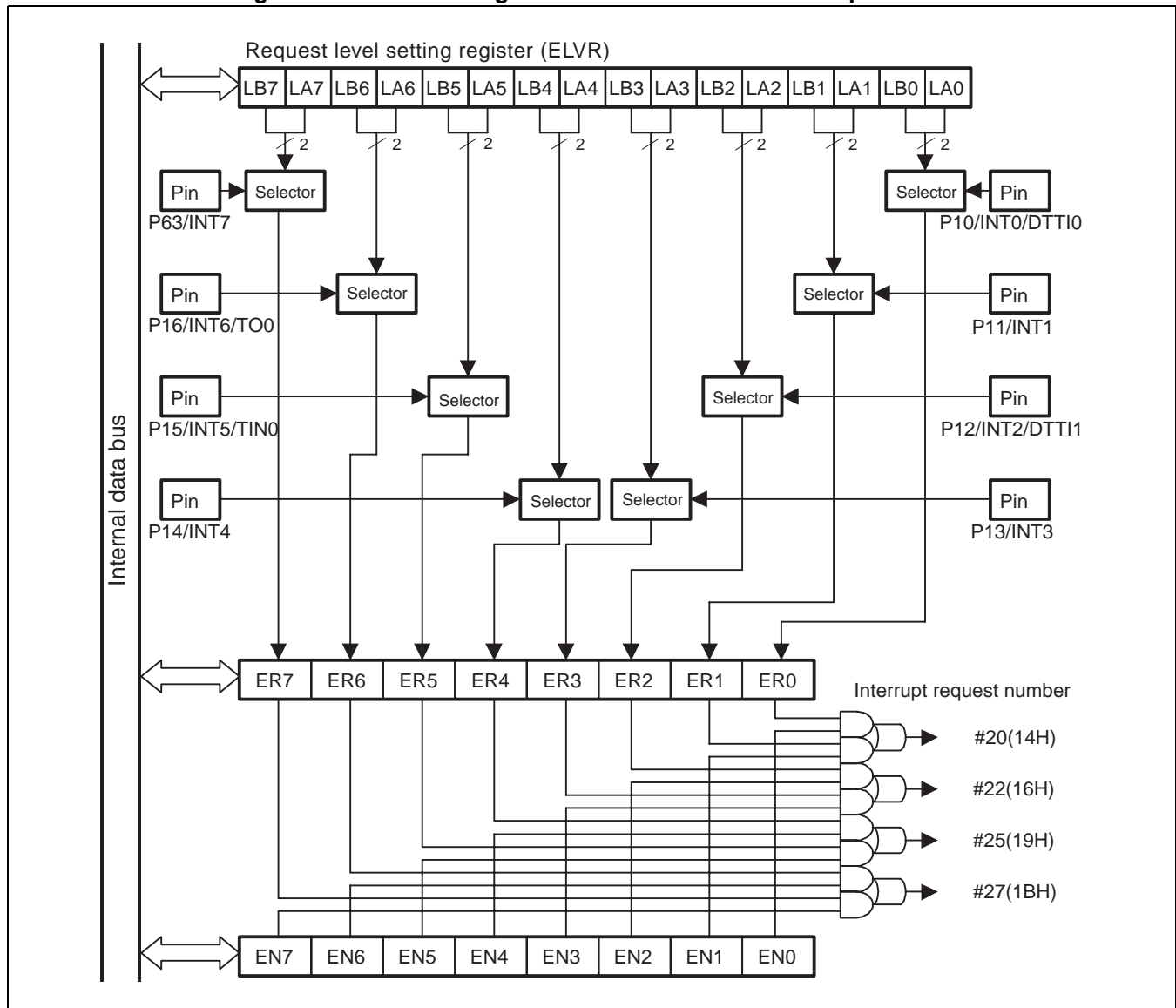
O: Can be used and interrupt request flag is cleared by EI<sup>2</sup>OS interrupt clear signal.

## 18.2 Block Diagram of the DTP/External Interrupt Circuit

The DTP/external interrupt circuit consists of four blocks, the block diagram is shown in Figure 18.2-1.

### ■ Block Diagram of the DTP/external Interrupt Circuit

Figure 18.2-1 Block Diagram of the DTP/external Interrupt Circuit



- DTP/external interrupt input detection circuit

Upon detecting the level or edge selected for each pin by the interrupt request level setting register (ELVR), this circuit sets to "1" the IR bit of the DTP/external interrupt cause register (EIRR) that corresponds to the pin.

- Request level setting register (ELVR)

This register selects the effective level or edge for each pin.

- DTP/interrupt cause register (EIRR)

This register stores DTP/external interrupt causes. It contains an external interrupt request flag bit for each pin. The bit is set to "1" if a valid signal is input to the corresponding pin.

- DTP/interrupt enable register (ENIR)

This register enables and disables external interrupts for each pin.



## 18.3 DTP/External Interrupt Circuit Pins

This section describes the DTP/external interrupt circuit pins and provides a pin block diagram.

### ■ DTP/external Interrupt Circuit Pins

The DTP/external interrupt circuit pins are also used as general ports. Table 18.3-1 lists the pin functions, I/O formats, and settings required to use the DTP/external interrupt circuit.

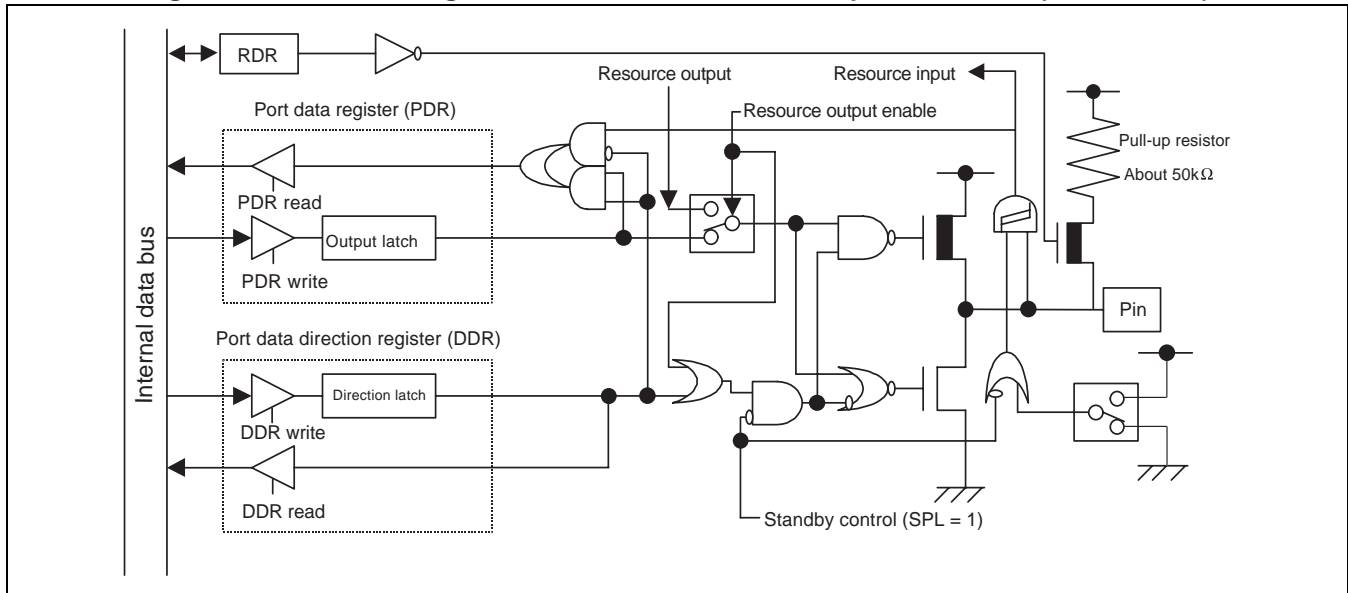
**Table 18.3-1 DTP/external Interrupt Circuit Pins**

Pin name	Function	I/O format	Pull-up resistor	Standby control	Setting required to use pins
P10/INT0/ DTTI0	Port 1 input-output/ external interrupt input/resource input- output	CMOS output /CMOS hysteresis input	Selectable	Provided	Set the pin as an input port (DDR1: bit8 = 0)
P11/INT1					Set the pin as an input port (DDR1: bit9 = 0)
P12/INT2/ DTTI1*					Set the pin as an input port (DDR1: bit10 = 0)
P13/INT3					Set the pin as an input port (DDR1: bit11 = 0)
P14/INT4					Set the pin as an input port (DDR1: bit12 = 0)
P15/INT5/ TIN0					Set the pin as an input port (DDR1: bit13 = 0)
P16/INT6/TO0	Port 6 input-output/ external interrupt input		Not provided		Set the pin as an input port (DDR1: bit14 = 0)
P63/INT7					Set the pin as an input port (DDR6: bit3 = 0)

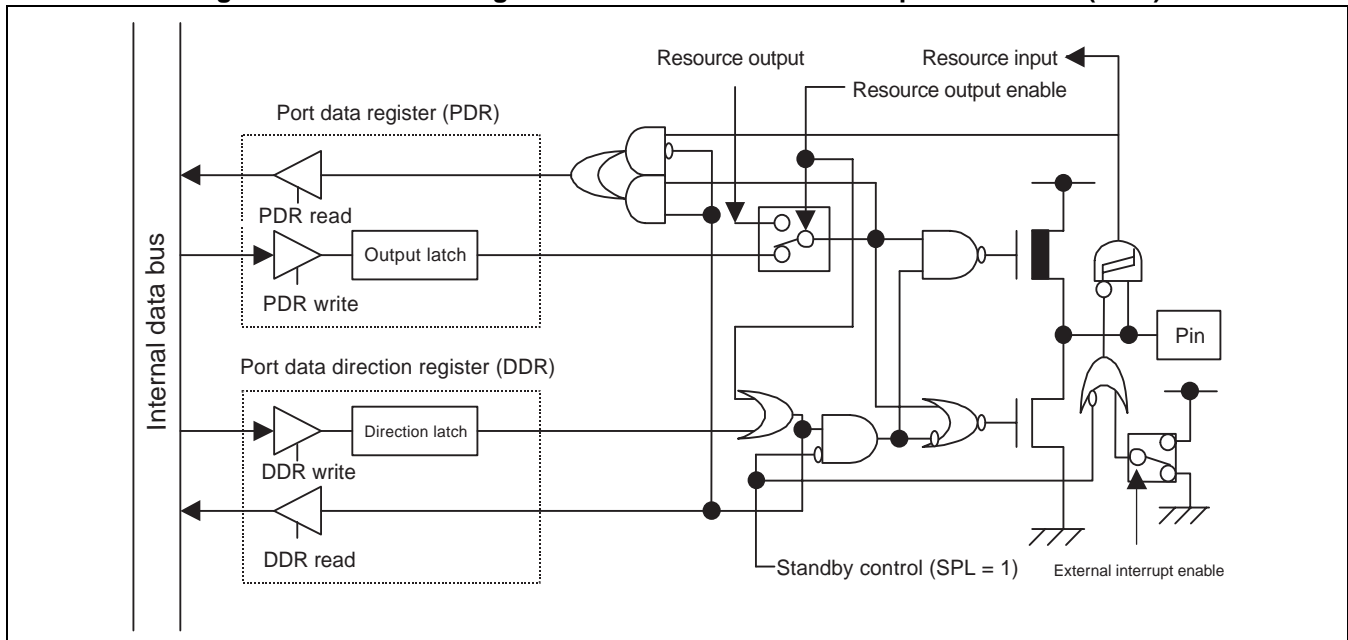
\*: Pin name not applicable to MB90465 series

## ■ Block Diagram of the DTP/external Interrupt Circuit Pins

**Figure 18.3-1 Block Diagram of the DTP/external Interrupt Circuit Pins (INT0 to INT6)**



**Figure 18.3-2 Block Diagram of the DTP/external Interrupt Circuit Pins (INT7)**



## 18.4 DTP/External Interrupt Circuit Registers

This section describes DTP/external interrupt circuit registers.

### ■ DTP/External Interrupt Circuit Registers

Figure 18.4-1 DTP/external Interrupt Circuit Registers

DTP / Interrupt Cause Register									
	bit	15	14	13	12	11	10	9	8
Address: 000031 <sub>H</sub>		ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0
		EIRR							
Read/write ⇨		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value ⇨		X	X	X	X	X	X	X	X
DTP / Interrupt Enable Register									
	bit	7	6	5	4	3	2	1	0
Address: 000030 <sub>H</sub>		EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
		ENIR							
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
Request Level Setting Register (Upper)									
	bit	15	14	13	12	11	10	9	8
Address: 000033 <sub>H</sub>		LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4
		ELVRH							
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
Request Level Setting Register (Lower)									
	bit	7	6	5	4	3	2	1	0
Address: 000032 <sub>H</sub>		LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0
		ELVRL							
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

## 18.4.1 DTP/interrupt Cause Register (EIRR)

The DTP/interrupt cause register (EIRR) stores and clears interrupt causes.

### ■ DTP/interrupt Cause Register (EIRR)

Figure 18.4-2 DTP/interrupt Cause Register (EIRR)

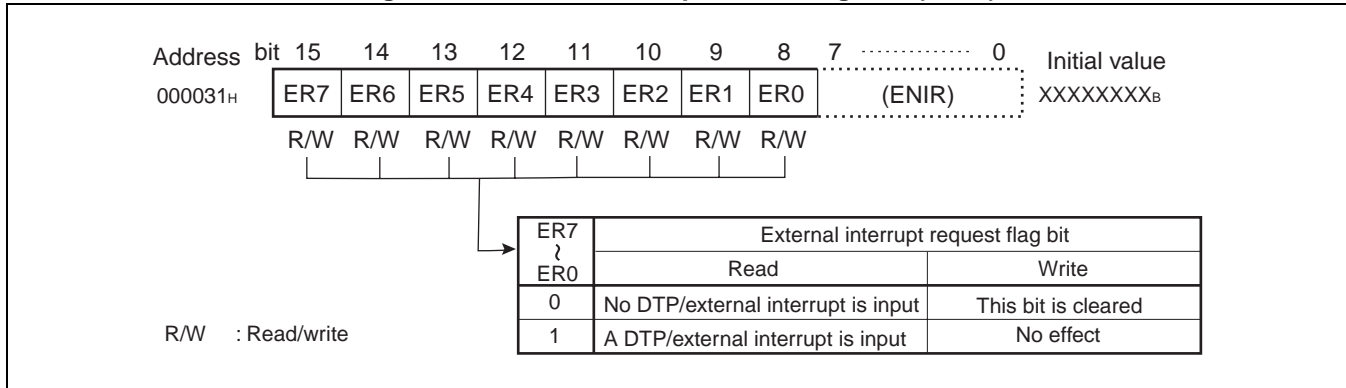


Table 18.4-1 Function Description of Each Bit of the DTP/interrupt Cause Register (EIRR)

Bit name		Function
bit15 to bit8	ER7 to ER0: External interrupt request flag bits	<ul style="list-style-type: none"> <li>Each of these bits is set to "1" if a signal with the edge or level selected by bits LB7, LA7 to LB0, LA0 of the request level setting register (ELVR) is input to the DTP/external interrupt pin (stores an interrupt cause).</li> <li>If these bits and corresponding bits EN7 to EN0 of the DTP/interrupt enable register (ENIR) are "1", an interrupt request is output to the CPU.</li> <li>Writing "0" to this bit clears the bit. Writing "1" to this bit does not change the bit value and has no effect on other bits.</li> </ul> <p>(Note) If more than one external interrupt request output is enabled (ENIR: EN7 to EN0 = 1), clear only the bit that caused the CPU to accept an interrupt (bits ER7 to ER0 set to "1"). Do not clear the other bits without a reason.</p> <p>(Reference) When the extended intelligent I/O service (EI<sup>2</sup>OS) is activated, the corresponding external interrupt request flag bit is automatically cleared when the transfer of one data ends.</p>

#### Notes:

- The value of the DTP/external interrupt request flag bit (EIRR:ER) is valid only when the corresponding DTP/external interrupt request enable bit (ENIR:EN) is set to "1". When the DTP/external interrupt is not enabled (ENIR:EN=0), the DTP/external interrupt cause bit may be set regardless of the presence or absence of a DTP/external interrupt cause.
- Immediately before enabling the DTP/external interrupt (ENIR:EN=1), clear the corresponding DTP/external interrupt request flag bit (EIRR:ER).

### 18.4.2 DTP/interrupt Enable Register (ENIR)

The DTP/interrupt enable register (ENIR) enables and disables the output of interrupt requests to the CPU.

■ DTP/interrupt Enable Register (ENIR)

Figure 18.4-3 DTP/interrupt Enable Register (ENIR)

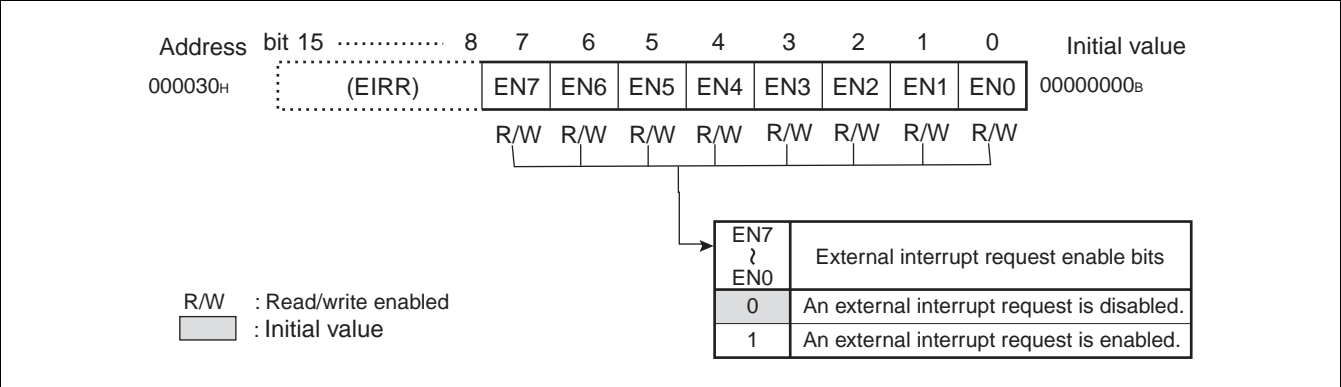


Table 18.4-2 Function Description of Each Bit of the DTP/interrupt Enable Register (ENIR)

Bit name		Function
bit7 to bit0	EN7 to EN0: External interrupt request enable bits	<p>Each of these bits enables and disables the output of interrupt requests to the CPU. If these bits and corresponding bits ER7 to ER0 of the DTP/interrupt cause register (EIRR) are "1", an interrupt request is output to the CPU.</p> <p>(References)</p> <ul style="list-style-type: none"><li>To use a DTP/external interrupt pin, write "0" to the corresponding bit of the port direction register to set the pin as an input port.</li><li>The states of the DTP/external interrupt pins can be read directly using the port data register regardless of the states of external interrupt request enable bits.</li><li>Bits ER7 to ER0 of the DTP/interrupt cause register (EIRR) are set to "1" if an interrupt cause is detected regardless of the values of external interrupt request enable bits.</li></ul>

Note: Immediately before enabling the DTP/external interrupt (ENIR:EN=1), clear the corresponding DTP/external interrupt request flag bit (EIRR:ER).

**Table 18.4-3 Correspondence between the DTP/interrupt Control Registers (EIRR, ENIR) and Each Channel**

DTP/external interrupt pin	Interrupt number	External interrupt request flag bit	External interrupt request enable bit
P63/INT7	#27 (1B <sub>H</sub> )	ER7	EN7
P16/INT6		ER6	EN6
P15/INT5	#25 (19 <sub>H</sub> )	ER5	EN5
P14/INT4		ER4	EN4
P13/INT3	#22 (16 <sub>H</sub> )	ER3	EN3
P12/INT2/DTTI1*		ER2	EN2
P11/INT1	#20 (14 <sub>H</sub> )	ER1	EN1
P10/INT0/DTTI0		ER0	EN0

\*: Pin name not applicable to MB90465 series

### 18.4.3 Request Level Setting Register (ELVR)

The request level setting register (ELVR) selects the level or edge of the signal input to each DTP/external interrupt pin that is to be detected as a DTP/external interrupt cause.

#### ■ Request Level Setting Register (ELVR)

Figure 18.4-4 Request Level Setting Register (ELVR)

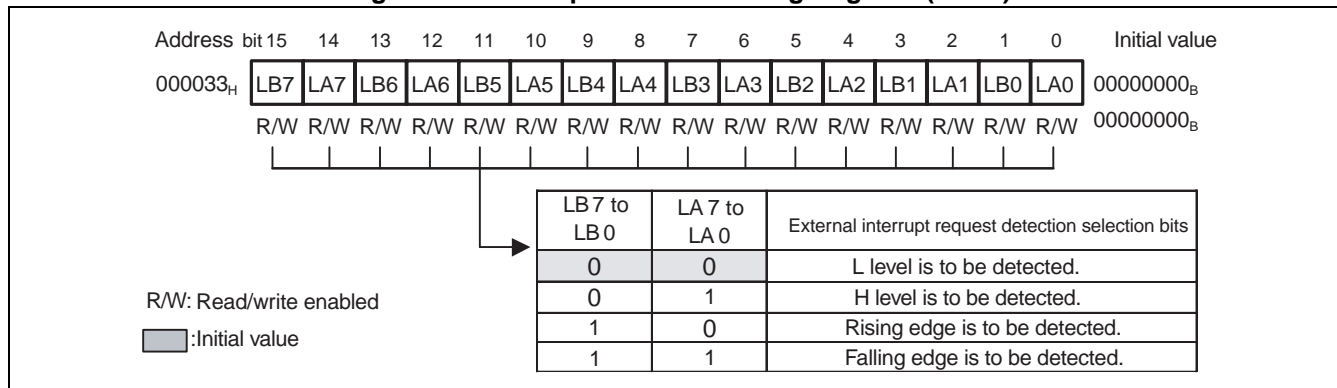


Table 18.4-4 Function Description of Each bit of the Request Level Setting Register (ELVR)

Bit name		Function
bit15 to bit0	LB7, LB0 to LA7, LA0: Request detection selection bits	<ul style="list-style-type: none"> <li>Each of these bits selects the level or edge of the signal input to the DTP/external interrupt pin to be detected as a DTP/external interrupt cause.</li> <li>Two bits are assigned to each pin.</li> </ul> (Reference) If the selected detection signal is input to a DTP/external interrupt pin, the external interrupt request flag bit is set to "1" regardless of the settings of the DTP/interrupt enable register (ENIR).

Table 18.4-5 Correspondence between Request Level Setting Register (ELVR) and Each Channel

DTP/external interrupt pin	Interrupt number	Bit name
P63/INT7	#27 (1B <sub>H</sub> )	LB7, LA7
P16/INT6		LB6, LA6
P15/INT5	#25 (19 <sub>H</sub> )	LB5, LA5
P14/INT4		LB4, LA4
P13/INT3	#22 (16 <sub>H</sub> )	LB3, LA3
P12/INT2/DTTI1*		LB2, LA2
P11/INT1	#20 (14 <sub>H</sub> )	LB1, LA1
P10/INT0/DTTI0		LB0, LA0

\*: Pin name not applicable to MB90465 series

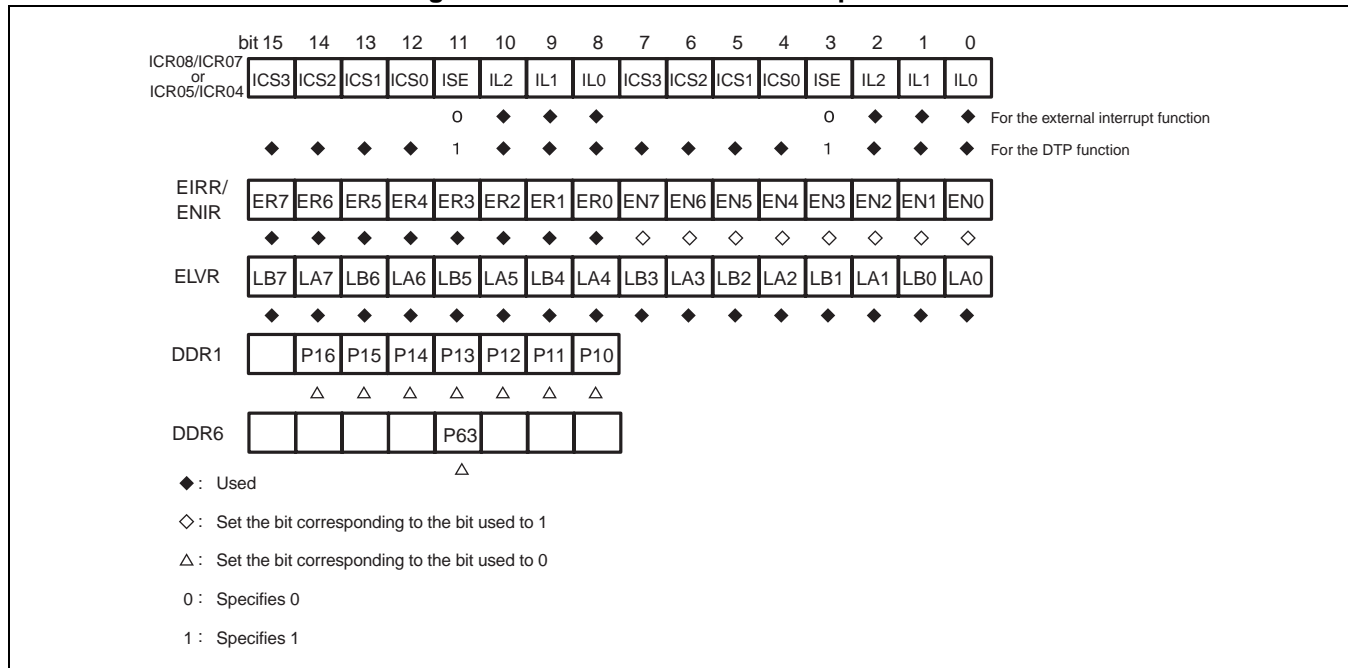
## 18.5 Operation of the DTP/External Interrupt Circuit

The DTP/external interrupt circuit provides the external interrupt function and the DTP function. This section describes the settings required for each function and the operation of the circuit.

### ■ Setting the DTP/external Interrupt Circuit

Figure 18.5-1 shows the settings required to operate the DTP/external interrupt circuit.

**Figure 18.5-1 DTP/external Interrupt Circuit**



Set the DTP/external interrupt circuit registers with the following procedure:

1. Set as an input port the general I/O port to be used also as a pin to input external interrupts.
2. Set the target bit of the DTP/interrupt enable register (ENIR) to disable interrupts.
3. Set the target bit of the request level setting register (ELVR).
4. Clear the target bit of the DTP/interrupt cause register (EIRR).
5. Set the target bit of the DTP/interrupt enable register (ENIR) to enable interrupts.

The procedure for setting the DTP/external interrupt circuit registers must start with disabling the output of external interrupt requests (ENIR:EN7 to EN0 = 0). Before the output of external interrupt requests can be enabled (ENIR:EN7 to EN0 = 1), the corresponding interrupt request flag bits must be cleared (ENIR:EN7 to EN0 = 0).

This is in order to avoid interrupt requests from being generated accidentally while the registers are being set.



● Switching between the external interrupt function and the DTP function

Switching between the external interrupt function and the DTP function is accomplished by the ISE bit of the corresponding interrupt control register (ICR). If the ISE bit is "1", the extended intelligent I/O service (EI<sup>2</sup>OS) is enabled and the circuit executes its DTP function. If it is "0", EI<sup>2</sup>OS is disabled and the circuit executes the its external interrupt function.

Note:

If multiple interrupt requests are assigned to a single ICR register, the interrupt level (IL2 to IL0) is common to all of the interrupt requests. As a rule, when one interrupt request uses EI<sup>2</sup>OS, the other interrupt requests cannot use it.

■ Operation of the DTP/external Interrupt Circuit

Table 18.5-1 shows the control bits and interrupt causes of the DTP/external interrupt circuit.

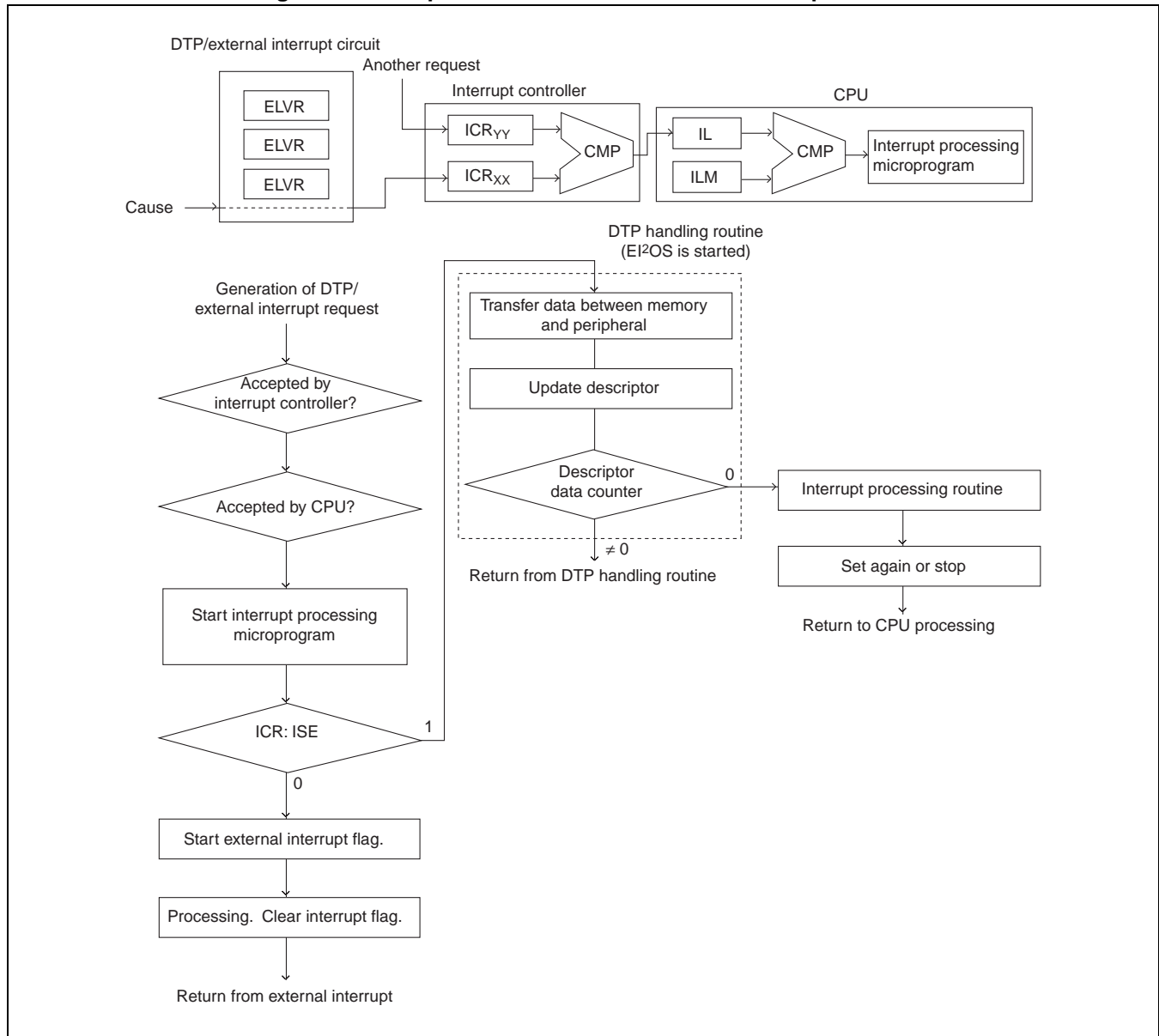
**Table 18.5-1 Control Bit and Interrupt Cause of the DTP/external Interrupt Circuit**

	DTP/external interrupt circuit
Interrupt request flag bit	EIRR: ER7 to ER0
Interrupt request enable bit	ENIR: EN7 to EN0
Interrupt cause	Input of an effective edge or level to pin INT7 to INT0

When DTP/external input requests are set, the resource will generate an interrupt request signal to the interrupt controller whenever an interrupt cause indicated in the request level setting register (ELVR) is received at the corresponding pin. If the ISE bit is "0", the interrupt processing microprogram is executed. If it is "1", the extended intelligent I/O service handling (DTP handling) microprogram is executed.

Figure 18.5-2 shows the operation of the DTP/external interrupt circuit.

**Figure 18.5-2 Operation of the DTP/external Interrupt Circuit**



## 18.5.1 External Interrupt Function

---

**The DTP/external interrupt circuit has an external interrupt function that generates an interrupt request when a selected signal level is input to a DTP/external interrupt pin.**

---

### ■ External Interrupt Function

If the edge or level selected for a DTP/external interrupt pin by the request level setting register (ELVR) is detected at that pin, the corresponding ER7 to ER0 bit of the DTP/interrupt cause register (EIRR) is set to "1". If, in this state, the corresponding interrupt request enable bit of the DTP/interrupt enable register is set to "1" to enable interrupts (ENIR:EN7 to EN0 = 1), the interrupt cause is reported to the interrupt controller. The interrupt controller checks the magnitude of the interrupt level (ICR:IL2 to IL0) in relation to those of the interrupt requests from other peripheral functions, the interrupt priority, etc. The CPU checks the magnitudes of the interrupt level mask register (PS:ILM2 to ILM0) and the interrupt level, the interrupt enable bit (PS:CCR: 1), etc. When the interrupt request is accepted by the CPU, the CPU executes an internal interrupt processing routine (microprogram) and branches to the interrupt processing routine. In the interrupt processing routine, 0 must be written to the corresponding interrupt request flag bit to clear the interrupt request.

---

Notes:

- An ER bit is set to "1" if a DTP/external interrupt cause is generated, regardless of the state of the corresponding EN bit.
  - When the interrupt routine is activated, the ER bit that caused the routine to be activated must be cleared. If the ER bit is kept at 1, control cannot return from the interrupt. Only clear the flag bit that caused the interrupt; do not clear the other bits without reason.
-

## 18.5.2 DTP Function

The DTP/external interrupt circuit has a DTP function that detects a signal supplied to a DTP/external interrupt pin from an external peripheral and activates the extended intelligent I/O service.

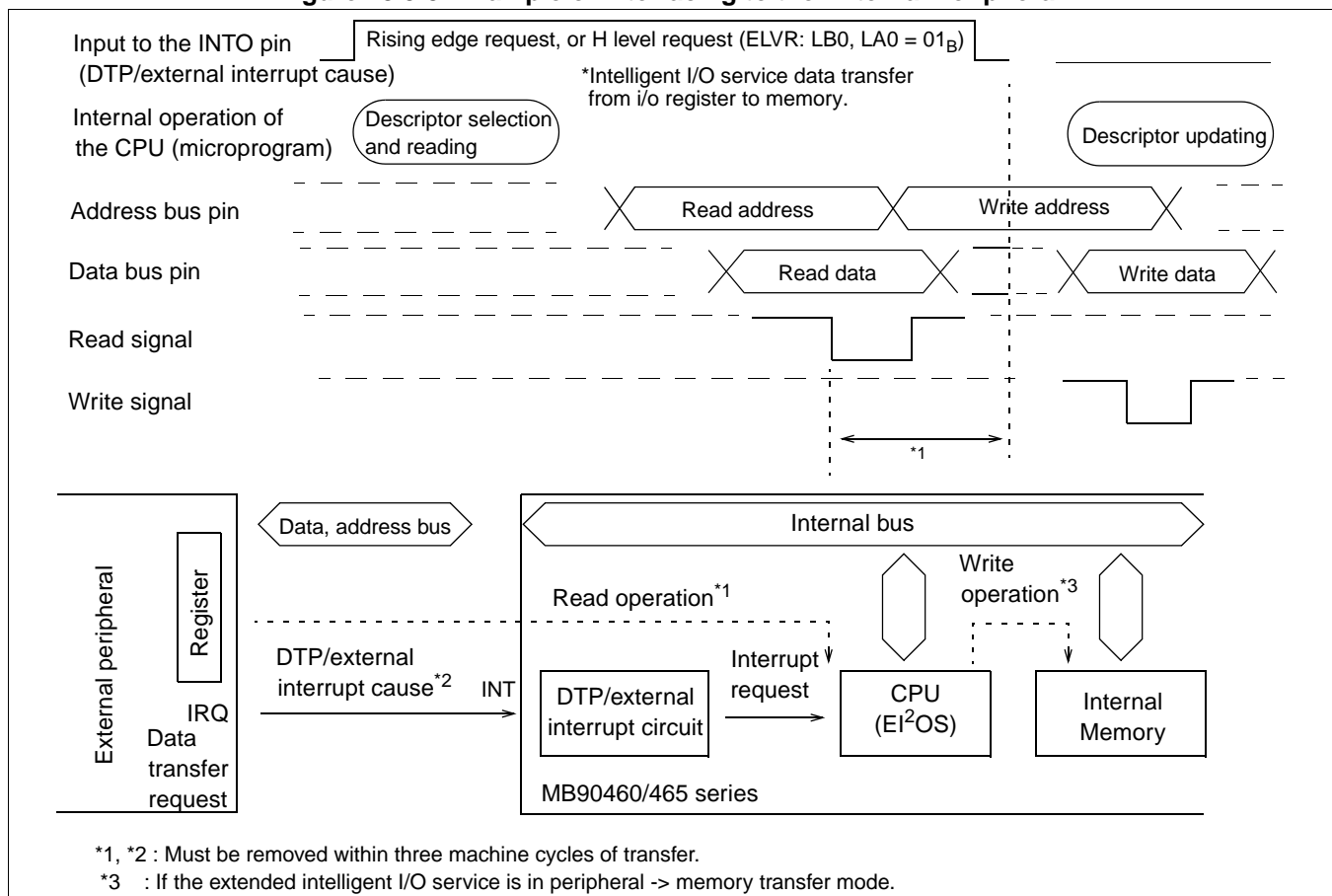
### ■ Operation of the DTP Function

The DTP function detects a data transfer request signal from an external peripheral to automatically transfer data between memory and the peripheral.

The extended intelligent I/O service (EI<sup>2</sup>OS) is activated by the external interrupt function using level detection. The operation of the DTP function is the same as that of the external interrupt function up to the point that the CPU accepts an interrupt request. If the operation of EI<sup>2</sup>OS is enabled (ICR:ISE = 1), EI<sup>2</sup>OS is activated to start data transfer when an interrupt request is accepted. When the transfer of one data unit ends, the descriptor is updated and the interrupt request flag bit is cleared to wait for the next request from the pin. When the entire transfer using EI<sup>2</sup>OS is completed, control is transferred to the interrupt processing routine.

The external peripheral must remove only the level of the data transfer request signal (DTP external interrupt cause) within three cycles of the first transfer.

**Figure 18.5-3 Example of Interfacing to the External Peripheral**



## 18.6 Usage Notes on the DTP/External Interrupt Circuit

Notes on the signal to be input to the DTP/external interrupt circuit, release from standby mode, and interrupts are given below.

### ■ Usage Notes on the DTP/external Interrupt Circuit

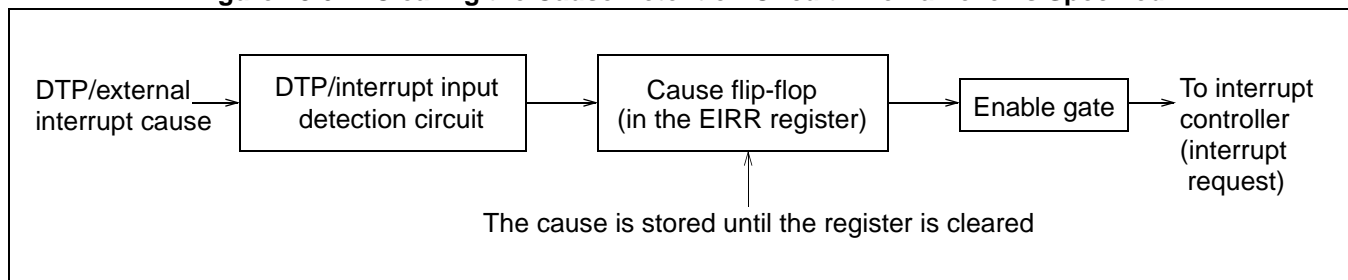
#### ● Conditions for external peripherals using the DTP function

To support the DTP function, external peripherals must be able to clear data transfer requests automatically in response to transfer operations. If a transfer request is not removed within three machine cycles of the start of transfer, the DTP/external interrupt circuit interprets the request as another transfer request.

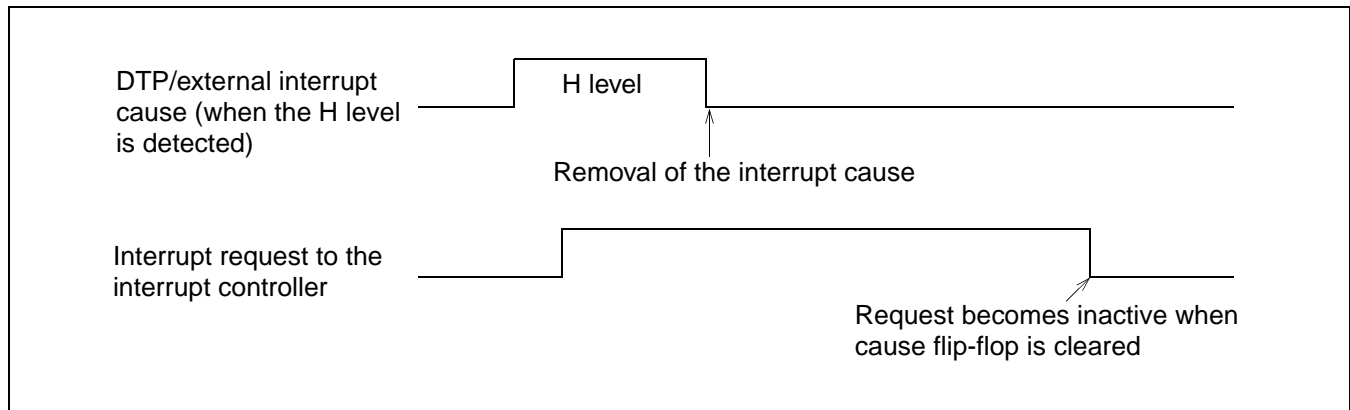
#### ● Input polarities of external interrupts

- If the request level setting register (ELVR) is set so that an edge is detected, the pulse width must be at least three machine cycles for the edge to be detected.
- If the request input level is level setting, the pulse width requires a longer period than the minimum pulse width stated on the data sheet. Also, as long as the interrupt input pin retains the active level, interrupt requests continue to be generated to the interrupt controller, even if the DTP/external interrupt cause register is cleared.
- If the register is set for level detection, and the level to be detected as an interrupt cause is input, cause F/F in the DTP/interrupt cause register (EIRR) is set to "1" to store the cause, as shown in Figure 18.6-1. Even if the cause is removed, the request to the interrupt controller remains active provided the output of interrupt requests is enabled. Thus, to cancel the request to the interrupt controller, clear the external interrupt request flag bit and cause F/F, as shown in Figure 18.6-2.

**Figure 18.6-1 Clearing the Cause Retention Circuit when a Level is Specified**



**Figure 18.6-2 DTP/external Interrupt Cause and Interrupt Request when the Output of Interrupt Requests is enabled**



● Notes about interrupts

When the external interrupt function is used, control cannot return from the interrupt processing routine if the external interrupt request flag bit is "1" and the output of interrupt requests is enabled. In the interrupt processing routine, the external interrupt request flag bit must be cleared. (When the DTP function is used, EI<sup>2</sup>OS automatically clears the bit.) For level detection, the external interrupt request flag bit is set again as soon as it is cleared if the level assumed as an interrupt cause continues to be input. Either disable the output of interrupt requests or remove the interrupt cause, if required.

## 18.7 Sample Programs for the DTP/External Interrupt Circuit

This section contains sample programs for the external interrupt function and the DTP function.

### ■ Sample Program for the External Interrupt Function

#### ● Processing

- The rising edge of the pulse input to the INT0 pin is detected, and an external interrupt is generated.

#### ● Coding example

```

ICR04      EQU      0000B4H      ;Interrupt control register for the DTP/external
                                   interrupt circuit
DDR6       EQU      000016H      ;
DDR1       EQU      000011H      ;Port 1 direction register
ENIR       EQU      000030H      ;DTP/interrupt enable register
EIRR       EQU      000031H      ;DTP/interrupt cause register
ELVRL      EQU      000032H      ;Request level setting register
ELVRH      EQU      000033H      ;Request level setting register
ER0        EQU      EIRR:0       ;INT0 interrupt flag bit
EN0        EQU      ENIR:0       ;INT0 interrupt enable bit
;-----Main program-----
CODE       CSEG
START:
;          :                      ;Assumes that stack pointer (SP) has already been
                                   initialized
          MOV      I:DDR1,#00000000B ;Sets DDR1 as an input port
          AND      CCR,#0BFH         ;Disables interrupts
          MOV      I:ICR04,#00H       ;Interrupt level: 0 (highest). Disables EI2OS
          CLRB     I:EN0              ;Disables INT0, using ENIR
          MOV      I:ELVRL,#00000010B ;Selects the rising edge for INT0
          CLRB     I:ER0              ;Clears the cause for INT0 using EIRR
          SETB     I:EN0              ;Enables INT0 using ENIR
          MOV      ILM,#07H          ;Sets ILM in PS to level 7
          OR       CCR,#40H          ;Enables interrupts
LOOP:      MOV     A,#00H             ;Endless loop
          MOV     A,#01H
          BRA     LOOP
;-----Interrupt program-----
WARI
          CLRB     I:ER0              ;Clears the interrupt request flag
;          :
;          :      User processing
;          :

```

```

                                RETI                                ;Returns from interrupt
CODE      ENDS
;-----Vector setting-----
VECT      CSEG      ABS=0FFH
                                ORG      0FFACH      ;Sets vector for interrupt #20 (14H)
                                DSL      WARI
                                ORG      0FFDCH      ;Sets reset vector
                                DSL      START
                                DB      00H      ;Sets single-chip mode
VECT      ENDS
                                END      START

```

## ■ Sample Program for the DTP Function

### ● Processing

- The H level of the signal input to the INT0 pin is detected, and channel 0 of the extended intelligent I/O service (EI<sup>2</sup>OS) is activated.
- Data is output from RAM to port 0 by DTP processing (EI<sup>2</sup>OS).

### ● Coding example

```

ICR04      EQU      0000B4H      ;Interrupt control register for the DTP/external
                                ;interrupt circuit
DDR0       EQU      000010H      ;Port 0 direction register
DDR1       EQU      000011H      ;Port 1 direction register
ENIR       EQU      000030H      ;DTP/interrupt enable register
EIRR       EQU      000031H      ;DTP/interrupt cause register
ELVRL      EQU      000032H      ;Request level setting register
ELVRH      EQU      000033H      ;Request level setting register
ER0        EQU      EIRR:0      ;INT0 interrupt flag bit
EN0        EQU      ENIR:0      ;INT0 interrupt enable bit
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
;-----Main program-----
CODE      CSEG
START:
;      :      ;Assumes that stack pointer (SP) has already been
                                ;initialized
                                MOV      I:DDR0,#11111111B ;Sets DDR0 as an output port
                                MOV      I:DDR1,#00000000B ;Sets DDR1 as an input port
                                AND      CCR,#0BFH      ;Disables interrupts
                                MOV      I:ICR04,#08H      ;Interrupt level: 0 (highest)

```



```

                                ;Enables EI2OS. Channel 0
MOV      BAPL,#00H            ;Sets the address of the output data
MOV      BAPM,#06H            ;
MOV      BAPH,#00H            ;
MOV      ISCS,#12H            ;Byte transfer. I/O address fixed. Buffer address
                                + 1. Transfer from memory to I/O
MOV      IOAL,#00H            ;Specifies port 0 (PDR0) as the transfer destination
MOV      IOAH,#00H            ;address pointer
MOV      DCTL,#0AH            ;Number of transfers: 10
MOV      DCTH,#00H            ;
CLRB     I:EN0                 ;Disables INT0 using ENIR
MOV      I:ELVRL,#00000001B;Selects H level for INT0
CLRB     I:ER0                 ;Clears the cause of INT0 using EIRR
SETB     I:EN0                 ;Enables INT0 using ENIR
MOV      ILM,#07H             ;Sets ILM in PS to level 7
OR        CCR,#40H             ;Enables interrupts
LOOP:    MOV      A,#00H        ;Endless loop
MOV      A,#01H                ;
BRA      LOOP                  ;
;-----Interrupt program-----
WARI:
        CLRB     I:ER0         ;Clears the interrupt request flag
;        :           ;          ;Switches the channel and changes the transfer
;                                address, if required
;        User processing        ;Specifies processing again, such as the termination
;                                of EI2OS. To terminate the processing, interrupts
;                                must be disabled
;        :
        RETI                 ;Returns from the interrupt
CODE    ENDS
;-----Vector setting-----
VECT    CSEG      ABS=0FFH
        ORG      0FFACH        ;Sets vector for interrupt #20 (14H)
        DSL      WARI
        ORG      0FFDCH        ;Sets reset vector
        DSL      START
        DB        00H          ;Sets single-chip mode
VECT    ENDS
END      STAR

```

# **CHAPTER 19**

---

# ***DELAYED INTERRUPT GENERATOR MODULE***

**This chapter describes the functions and operation of the delayed interrupt generator module.**

- 19.1 Overview of the Delayed Interrupt Generator Module
- 19.2 Delayed Interrupt Generator Module Register
- 19.3 Operation of the Delayed Interrupt Generator Module
- 19.4 Usage Notes on the Delayed Interrupt Generator Module

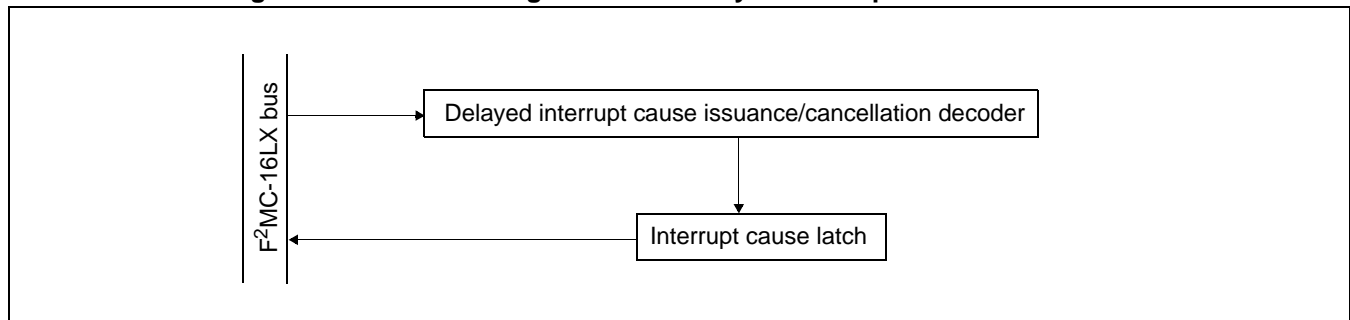
## 19.1 Overview of the Delayed Interrupt Generator Module

The delayed interrupt generator module generates interrupts for task switching. By using this module, software can issue and cancel interrupt requests for the F<sup>2</sup>MC-16LX CPU.

### ■ Block Diagram of the Delayed Interrupt Generator Module

Figure 19.1-1 shows the block diagram of the delayed interrupt generator module.

**Figure 19.1-1 Block Diagram of the Delayed Interrupt Generator Module**



## 19.2 Delayed Interrupt Generator Module Register

This section lists the delayed interrupt generator module register.

### ■ Delayed Interrupt Generator Module Register (DIRR)

Figure 19.2-1 Delayed Interrupt Generator Module Register (DIRR)

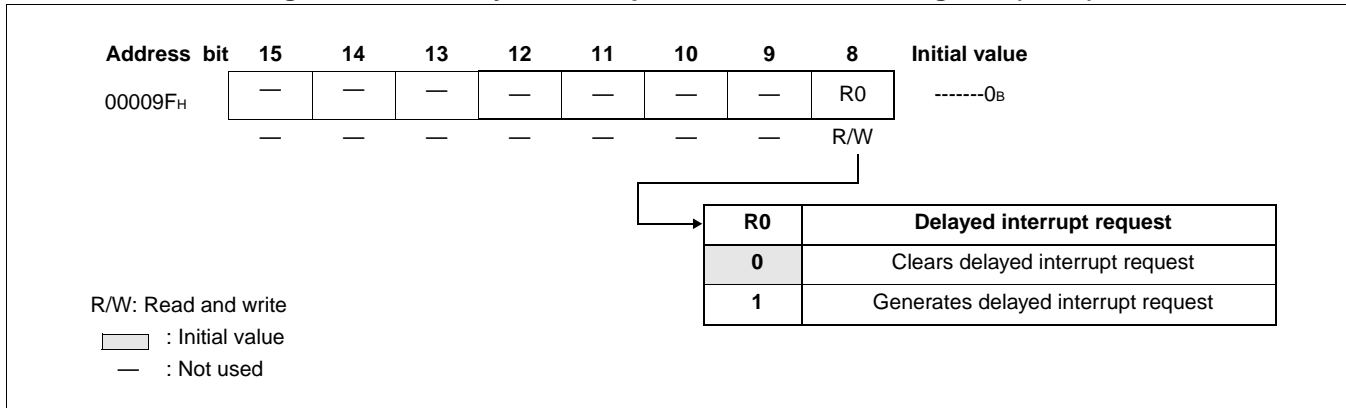


Table 19.2-1 Delayed Interrupt Generator Module Register (DIRR)

Bit name		Function
bit15 to bit9	Reserved bits	<ul style="list-style-type: none"> <li>Both "0" and "1" may be written to the reserved bit area, however, the set bit and clear bit instructions should be used to access this register to prepare for future expansion.</li> </ul>
bit8	R0: Delayed interrupt request bit	<ul style="list-style-type: none"> <li>This bit is used to controls the generation or clearing of a delayed interrupt request.</li> <li>Writing "1" to this register generates a delayed interrupt request.</li> <li>Writing "0" to this register clears the delayed interrupt request.</li> <li>The register is cleared at reset.</li> <li>Both "0" and "1" may be written to the reserved bit area. However, the set bit and clear bit instructions should be used to access this register to prepare for future expansion.</li> </ul>

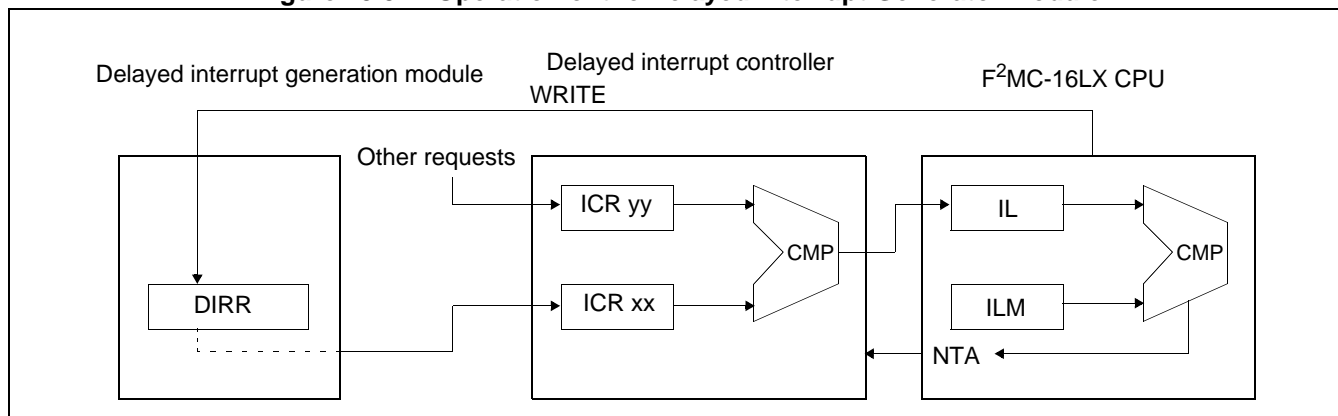
## 19.3 Operation of the Delayed Interrupt Generator Module

When software causes the CPU to write "1" to the relevant bit of DIRR, the request latch in the delayed interrupt generator module is set and an interrupt request is generated to the interrupt controller.

### ■ Operation of the Delayed Interrupt Generator Module

When software causes the CPU to write "1" to the relevant bit of DIRR, the request latch in the delayed interrupt generator module is set and an interrupt request is generated to the interrupt controller. If the priority of other interrupt requests is lower than that of this interrupt or no other interrupt request is generated, the interrupt controller generates an interrupt request to the F<sup>2</sup>MC-16LX CPU. The F<sup>2</sup>MC-16LX CPU compares the ILM bit of the internal CCR register and the interrupt request. When the request level is higher than that of the ILM bit, the CPU starts the hardware interrupt processing microprogram immediately after execution of the current instruction ends. As a result, the interrupt processing routine for this interrupt is executed. This interrupt cause is cleared and task switching is done by writing "0" to the relevant bit of DIRR in the interrupt processing routine. Figure 19.3-1 Operation of the delayed interrupt generator module shows the operation of the delayed interrupt generator module.

**Figure 19.3-1 Operation of the Delayed Interrupt Generator Module**



## 19.4 Usage Notes on the Delayed Interrupt Generator Module

---

Notes on using the delayed interrupt generator module are given below.

---

### ■ Usage Notes on the Delayed Interrupt Request Latch

- This latch is set by writing "1" to the relevant bit of DIRR and cleared by writing "0" to the same bit. Note that interrupt processing is restarted at the moment control returns from interrupt processing unless software is created to clear the cause in the interrupt processing routine.



# **CHAPTER 20**

---

## **8/10-BIT A/D CONVERTER**

**This chapter describes the functions and operation of the 8/10-bit A/D converter.**

- 20.1 Overview of the 8/10-bit A/D Converter
- 20.2 Block Diagram of the 8/10-bit A/D Converter
- 20.3 8/10-bit A/D Converter Pins
- 20.4 8/10-bit A/D Converter Registers
- 20.5 8/10-bit A/D Converter Interrupts
- 20.6 Operation of the 8/10-bit A/D Converter
- 20.7 Usage Notes on the 8/10-bit A/D Converter
- 20.8 Sample Program 1 for the 8/10-bit A/D Converter (Single Conversion Mode Using EI<sup>2</sup>OS)
- 20.9 Sample Program 2 for the 8/10-bit A/D Converter (Continuous Conversion Mode Using EI<sup>2</sup>OS)
- 20.10 Sample Program 3 for the 8/10-bit A/D Converter (Stop Conversion Mode Using EI<sup>2</sup>OS)



## 20.1 Overview of the 8/10-bit A/D Converter

Using the RC-type successive approximation conversion method, the 8/10-bit A/D converter converts an analog input voltage into a 10-bit or 8-bit digital value.

An input signal is selected from eight channels for analog input pins. The conversion can be activated by software, an internal clock, and 16-bit free-run timer zero detection.

### ■ Functions of the 8/10-bit A/D Converter

The converter converts the analog voltage input to an analog input pin (input voltage) to a digital value. The converter has the following features:

- The minimum conversion time is 6.13  $\mu\text{s}$  (for a machine clock of 16 MHz; includes the sampling time).
- The minimum sampling time is 3.75  $\mu\text{s}$  (for a machine clock of 16 MHz).
- The converter uses the RC-type successive approximation conversion method with a sample and hold circuit.
- A resolution of 10 bits or 8 bits can be selected.
- Up to eight channels for analog input pins can be selected by a program.
- At the end of A/D conversion, an interrupt request can be generated and EI<sup>2</sup>OS can be activated.
- In the interrupt-enabled state, the conversion data protection function prevents any part of the data from being lost through continuous conversion.
- The conversion can be activated by software, 16-bit reload timer 1 (rising edge) and 16-bit free-run timer zero detection edge.

Table 20.1-1 lists three types of conversion modes.

**Table 20.1-1 8/10-bit A/D Converter Conversion Modes**

	Single conversion	Scan conversion
Single conversion mode	Converts the input of a specified channel (single channel) just once.	Converts the inputs of two or more consecutive channels (up to eight channels) just once.
Continuous conversion mode	Converts the input of a specified channel (single channel) repeatedly.	Converts the inputs of two or more consecutive channels (up to eight channels) repeatedly.
Stop conversion mode	Converts the input of a specified channel (single channel), after which it is on standby for the next activation.	Converts the inputs of two or more consecutive channels (up to eight channels). When a channel has been converted, the converter is put on standby for the next activation.

## ■ 8/10-bit A/D Converter Interrupts and EI<sup>2</sup>OS

**Table 20.1-2 8/10-bit A/D Converter Interrupts and EI<sup>2</sup>OS**

Interrupt no.	Interrupt control register		Vector table address			EI <sup>2</sup> OS
	Register name	Address	Lower	Upper	Bank	
#11 (0B <sub>H</sub> )	ICR00	0000B0 <sub>H</sub>	FFFFD0 <sub>H</sub>	FFFFD1 <sub>H</sub>	FFFFD2 <sub>H</sub>	O

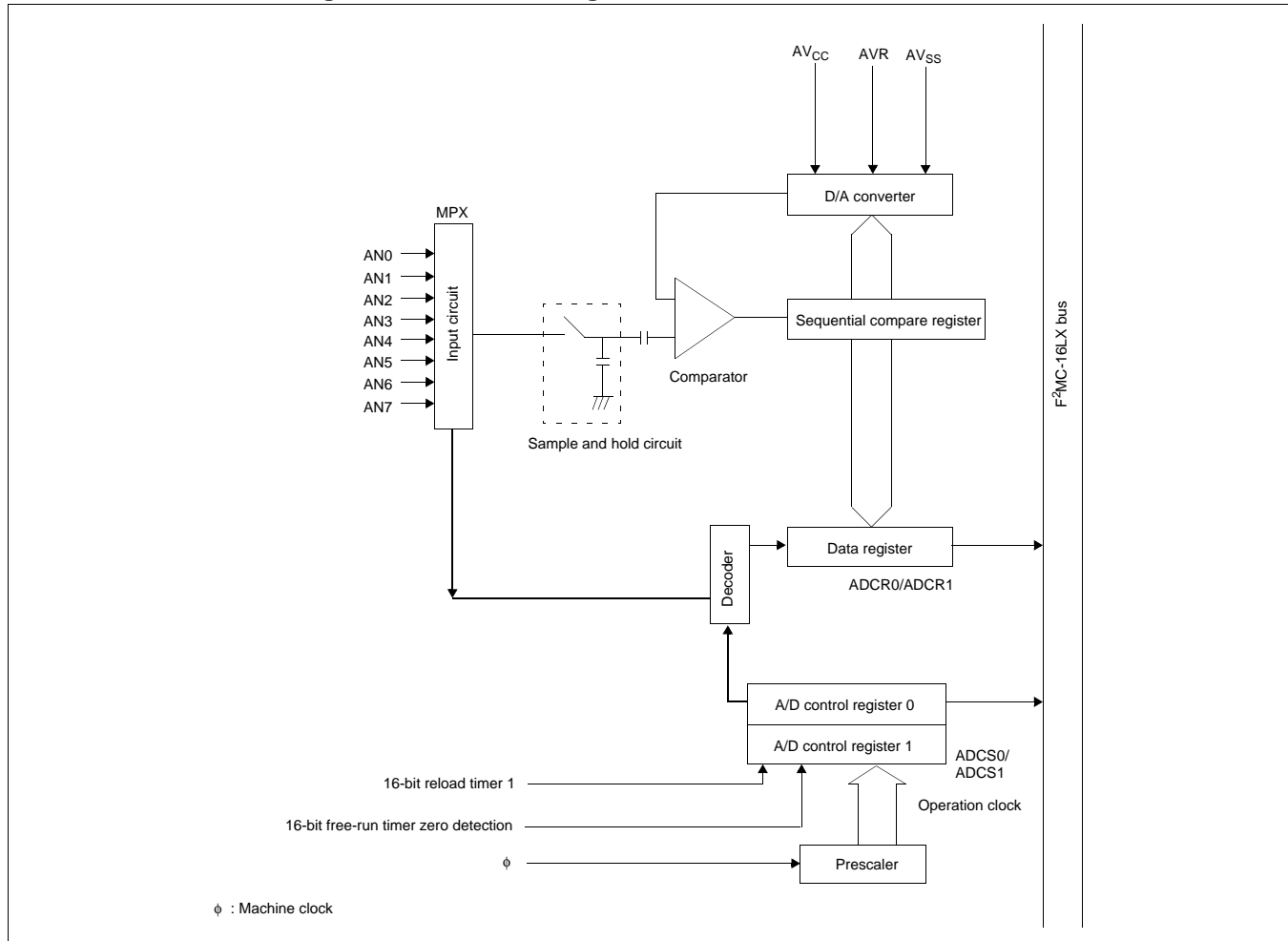
O: Can be used and interrupt request flag is cleared by EI<sup>2</sup>OS interrupt clear signal.

## 20.2 Block Diagram of the 8/10-bit A/D Converter

The 8/10-bit A/D converter has nine blocks, the block diagram is shown in Figure 20.2-1.

### ■ Block Diagram of the 8/10-bit A/D Converter

Figure 20.2-1 Block Diagram of the 8/10-bit A/D Converter



#### ● A/D control status register (ADCS0/ADCS1)

This register selects activation by software or another activation trigger, the conversion mode, and the A/D conversion channel. It also enables or disables interrupt requests, checks the interrupt request status, and indicates whether the conversion has halted or is in progress.

#### ● A/D data register (ADCR0/ADCR1)

This register holds the result of A/D conversion and selects the resolution for A/D conversion.

- Clock selector

The clock selector selects the clock for activating A/D conversion. Either 16-bit reload timer channel 1 output or 16-bit free-run timer zero detection can be used as the activation clock.

- Decoder

This circuit selects the analog input pin to be used based on the settings of the ANE0 to ANE2 bits and ANS0 to ANS2 bits of the A/D control status register (ADCS0).

- Analog channel selector

This circuit selects the pin to be used from eight analog input pins.

- Sample and hold circuit

This circuit maintains the input voltage of the channel selected by the analog channel selector. It samples and maintains the input voltage obtained immediately after the activation of A/D conversion. This circuit protects the A/D conversion from any variations in the input voltage during approximation.

- D/A converter

This circuit generates a reference voltage for comparison with the input voltage maintained by the sample and hold circuit.

- Comparator

This circuit compares the input voltage maintained by the sample and hold circuit with the output voltage of the D/A converter to determine which is greater.

- Control circuit

This circuit determines the A/D conversion value based on the decision signal generated by the comparator. When the A/D conversion has been completed, the circuit sets the conversion result in the A/D data register (ADCR0/ADCR1) and generates an interrupt request.

## 20.3 8/10-bit A/D Converter Pins

This section describes the 8/10-bit A/D converter pins and provides pin block diagrams.

### ■ 8/10-bit A/D Converter Pins

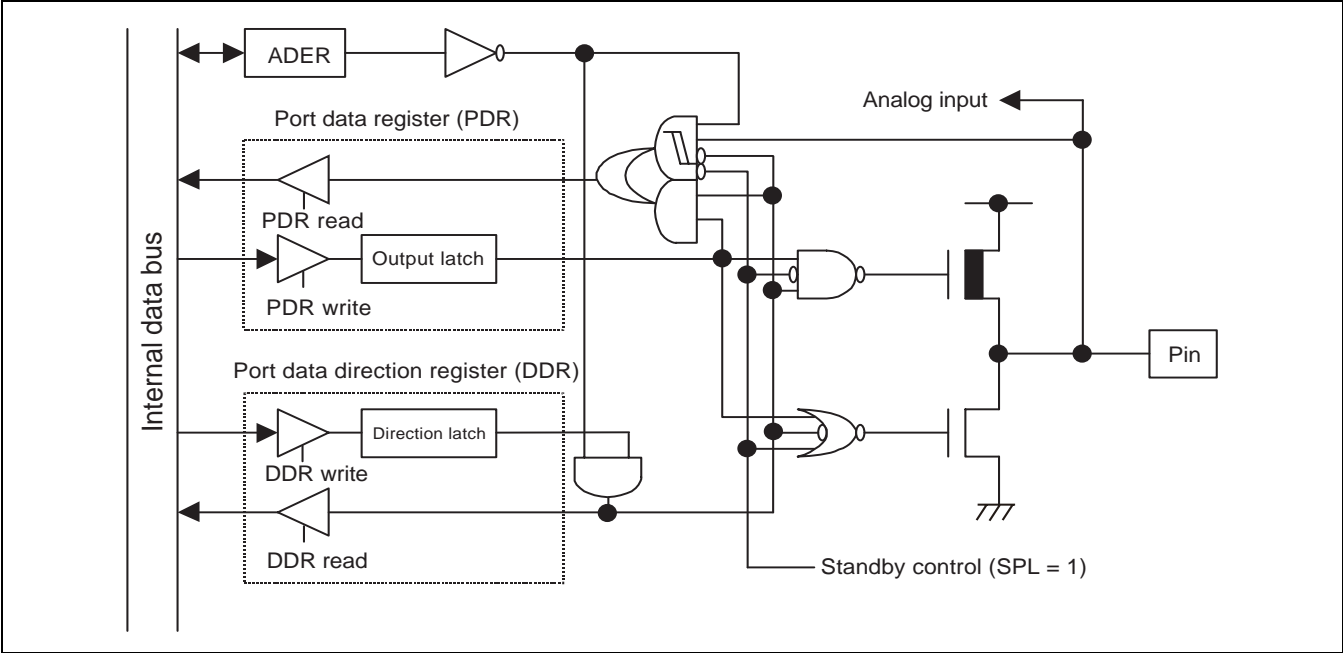
The A/D converter pins are also used as general ports. Table 20.3-1 lists the pin functions, I/O formats, and settings required to use the 8/10-bit A/D converter.

**Table 20.3-1 8/10-bit A/D Converter Pins**

Function	Pin name	Pin function	Input-output signal type	Pull-up option	Standby control	I/O port setting for using the pin
Channel 0	P50/AN0	Port 5 input/output or analog input	CMOS output/CMOS hysteresis input or analog input	Not selectable	Not selectable	Set port 5 as an input port (DDR5: bit8 to bit15 = 0). Set port 5 as an analog input port (ADER: bit8 to bit15 = 1)
Channel 1	P51/AN1					
Channel 2	P52/AN2					
Channel 3	P53/AN3					
Channel 4	P54/AN4					
Channel 5	P55/AN5					
Channel 6	P56/AN6					
Channel 7	P57/AN7					

### ■ Block Diagrams of the 8/10-bit A/D Converter Pins

**Figure 20.3-1 Block Diagram of the P50/AN0 to P57/AN7 Pins**



---

**Notes:**

- The MB90460/465 series runs only in single-chip mode so only internal ROM and RAM and internal peripheral address space can be accessed.
  - To use the pin as an analog input pin, set the corresponding bit of the ADER register to "1". The value read from the PDR5 register is "0".
-

## 20.4 8/10-bit A/D Converter Registers

This section lists the 8/10-bit A/D converter registers.

### ■ 8/10-bit A/D Converter Registers

Figure 20.4-1 8/10-bit A/D Converter Registers

Analog Input Enable Register

bit	15	14	13	12	11	10	9	8	
Address: 000017 <sub>H</sub>	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	ADER
Read/write ⇨	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨	1	1	1	1	1	1	1	1	

A/D Control Status Register 1

bit	15	14	13	12	11	10	9	8	
Address: 000035 <sub>H</sub>	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	RESV	ADCS1
Read/write ⇨	R/W	R/W	R/W	R/W	R/W	R/W	W	R/W	
Initial value ⇨	0	0	0	0	0	0	0	0	

A./D Control Status Register 2

bit	7	6	5	4	3	2	1	0	
Address: 000034 <sub>H</sub>	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS0
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	

A/D Data Register (Upper)

bit	15	14	13	12	11	10	9	8	
Address: 000037 <sub>H</sub>	S10	ST1	ST0	CT1	CT0	—	D9	D8	ADCR1
Read/write ⇨	R/W	W	W	W	W	—	R	R	
Initial value ⇨	0	0	0	0	0	—	X	X	

A/D Data Register (Lower)

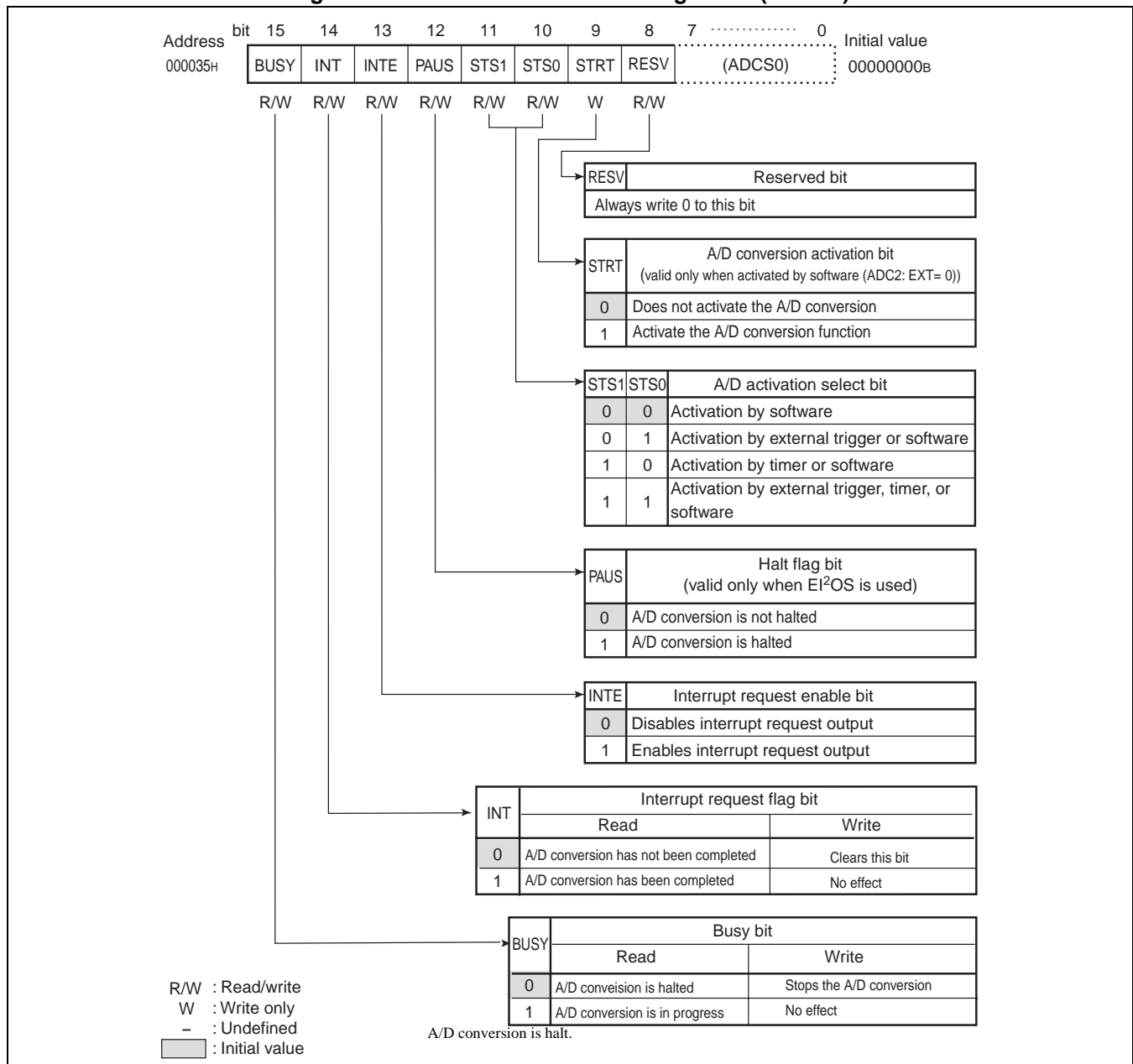
bit	7	6	5	4	3	2	1	0	
Address: 000036 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	ADCR0
Read/write ⇨	R	R	R	R	R	R	R	R	
Initial value ⇨	X	X	X	X	X	X	X	X	

## 20.4.1 A/D Control Status Register 1 (ADCS1)

A/D control status register 1 (ADCS1) selects activation by software or activation trigger, enables or disables interrupt requests, and indicates interrupt request status and whether conversion is halted or in progress.

### ■ A/D Control Status Register 1 (ADCS1)

Figure 20.4-2 A/D Control Status Register 1 (ADCS1)





**Table 20.4-1 A/D Dontrol Status Register 1 (ADCS1)**

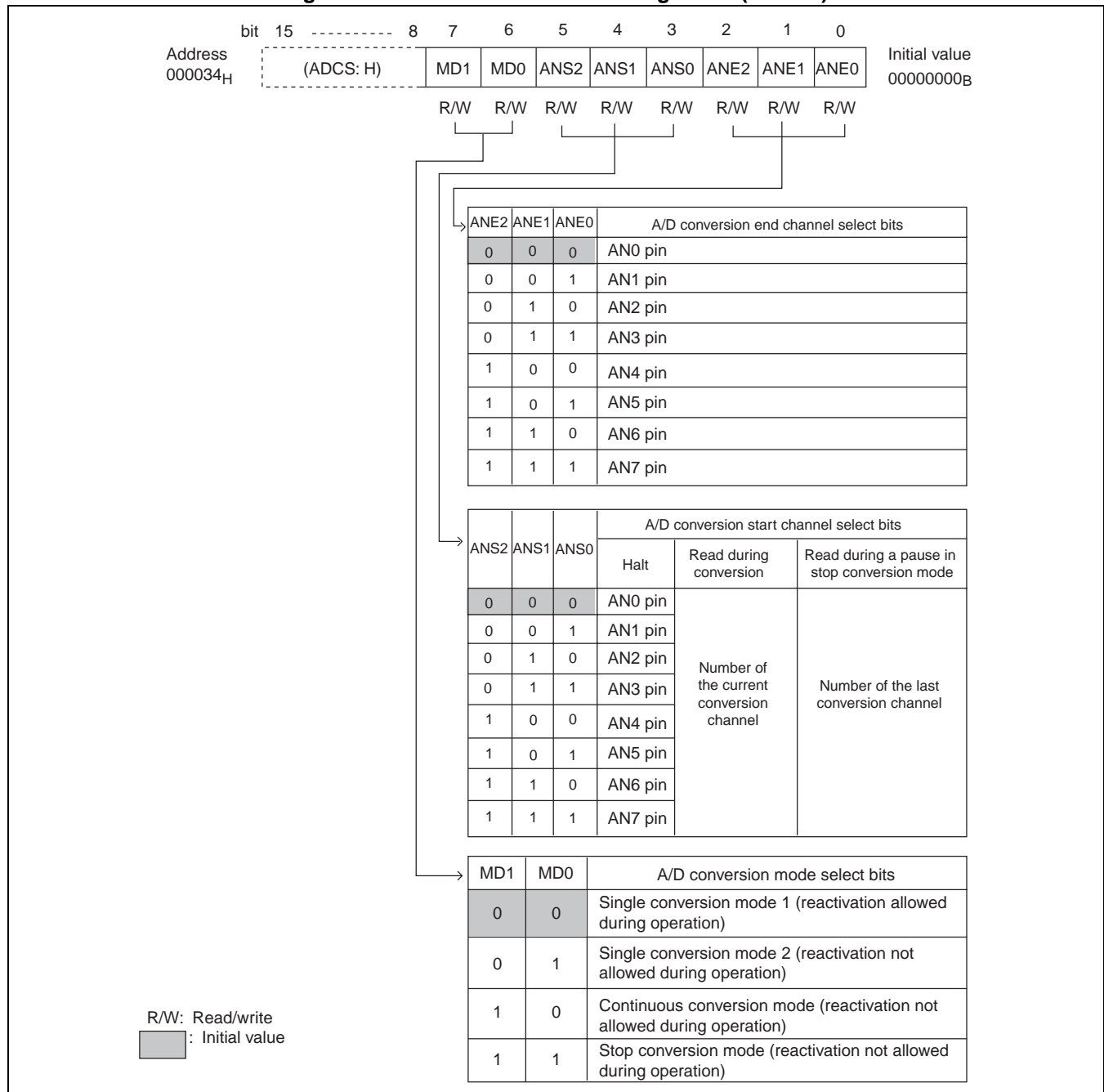
Bit name		Function
bit15	BUSY: Busy bit	<ul style="list-style-type: none"> <li>This bit indicates the operating status of the A/D converter.</li> <li>If the value read from this bit is "0", A/D conversion has halted. If the read value is "1", A/D conversion is in progress.</li> <li>Writing "0" to this bit forces the A/D conversion to stop. Writing "1" to this bit does not change the bit value and has no effect on other bits.</li> </ul> <p>(Note) Never select forced stop (BUSY = 0) and software activation (STRT = 1) simultaneously.</p>
bit14	INT: Interrupt request flag bit	<ul style="list-style-type: none"> <li>When A/D conversion data is set in the A/D data register, this bit is set to "1".</li> <li>When both this bit and the interrupt request enable bit (ADCS: INTE) are "1", an interrupt request is generated. If EI<sup>2</sup>OS has been enabled, it is activated.</li> <li>Writing "0" to this bit clears the bit. Writing "1" to this bit does not change the bit value and has no effect on other bits.</li> <li>When EI<sup>2</sup>OS is activated, this bit is cleared.</li> </ul> <p>(Note) When clearing this bit by writing "0" it, do so only while the A/D converter is not operating.</p>
bit13	INTE: Interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit enables or disables interrupt output to the CPU.</li> <li>When both this bit and the interrupt request flag bit (ADCS: INT) are set to "1", an interrupt request is generated.</li> <li>When EI<sup>2</sup>OS is used, set this bit to "1".</li> </ul>
bit12	PAUS: Halt flag bit	<ul style="list-style-type: none"> <li>When A/D conversion stops temporarily, this bit is set to "1".</li> <li>This A/D converter has just one A/D data register. In continuous conversion mode, if a conversion result were written before the previous conversion result was read by the CPU, the previous result would be lost. When continuous conversion mode is selected, the program must be written so that the conversion result is automatically transferred to memory by EI<sup>2</sup>OS each time a conversion is completed. This bit also protects against multiple interrupts preventing the completion of conversion data transfer before the next conversion. When a conversion is completed, this bit is set to "1". This status is maintained until EI<sup>2</sup>OS finishes transferring the contents of the data register. Meanwhile, the A/D conversion is halted so that no conversion data can be stored. When EI<sup>2</sup>OS completes the transfer, the A/D converter automatically resumes the conversion.</li> </ul> <p>(Note) This bit is valid only when EI<sup>2</sup>OS is used.</p>
bit11, bit10	STS1, STS0: A/D activation select bit	<ul style="list-style-type: none"> <li>These bits select how A/D conversion is to be activated.</li> <li>When two or more activation causes are shared, activation is the result of the cause that occurs first.</li> </ul> <p>(Note) Change the setting during A/D conversion only while there is no corresponding activation cause, since the change becomes effective immediately.</p>
bit9	STRT: A/D conversion activation bit	<ul style="list-style-type: none"> <li>This bit allows software to start A/D conversion.</li> <li>Writing "1" to this bit activates A/D conversion.</li> <li>In stop conversion mode, conversion cannot be reactivated with this bit.</li> <li>In byte/word instruction, "1" is read.</li> <li>In read-modify-write instruction, "0" is read.</li> </ul> <p>(Note) Never select forced stop (BUSY = 0) and software activation (STRT = 1) simultaneously.</p>
bit8	RESV: Reserved bit	<p>(Note) Always write "0" to this bit.</p>

## 20.4.2 A/D Control Status Register 0 (ADCS0)

A/D control status register 0 (ADCS0) selects the conversion mode and A/D conversion channel.

### ■ A/D Control Status Register 0 (ADCS0)

Figure 20.4-3 A/D Control Status Register 0 (ADCS0)



**Table 20.4-2 A/D Control Status Register 0 (ADCS0)**

Bit name		Function
bit7, bit6	MD1, MD0: A/D conversion mode select bits	<ul style="list-style-type: none"> <li>These bits select the conversion mode of the A/D conversion function.</li> <li>The two-bit value of the MD1 and MD0 bits determines the mode that is selected from among four modes: single conversion mode 1, single conversion mode 2, continuous conversion mode, and stop conversion mode.</li> <li>The operation in each mode is described below:            Single conversion mode 1:            Just a single A/D conversion from the channel set by ANS2 to ANS0 to the channel set by ANE2 to ANE0 is performed.            Reactivation during operation is allowed.            Single conversion mode 2:            Just a single A/D conversion from the channel set by ANS2 to ANS0 to the channel set by ANE2 to ANE0 is performed.            Reactivation during operation is not allowed.            Continuous conversion mode:            A/D conversion from the channel set by ANS2 to ANS0 to the channel set by ANE2 to ANE0 is performed repeatedly. The repeated conversion continues until it is stopped by the BUSY bit.            Reactivation during operation is not allowed.            Stop conversion mode:            A/D conversion from the channel set by ANS2 to ANS0 to the channel set by ANE2 to ANE0 is performed repeatedly with a pause after the conversion of each channel. The repeated conversion continues until it is stopped by the BUSY bit.            Reactivation during operation is not allowed. In the pause state, the conversion is reactivated when an activation cause selected by the STS1 and STS0 bits is generated.</li> </ul> <p>(Note)</p> <p>In the single conversion modes, continuous conversion mode, and stop conversion mode, no reactivation by a timer, external trigger, or software is allowed.</p>
bit5 to bit3	ANS2 to ANS0: A/D conversion start channel select bits	<ul style="list-style-type: none"> <li>These bits set the A/D conversion start channel and indicate the number of the current conversion channel.</li> <li>When activated, A/D conversion starts from the channel specified by these bits.</li> <li>During A/D conversion, the bits indicate the number of the current conversion channel. During a pause in stop conversion mode, the bits indicate the number of the last conversion channel.</li> <li>As for the lead value of this bit, even when the value is set to this bit, not the set value but the channel number in which A/D was converted last time is read until the analog to digital conversion is begun.</li> <li>At the reset, the bit is initialized to "000<sub>B</sub>".</li> </ul> <p>(Notes)</p> <ul style="list-style-type: none"> <li>Do not use a read-modify-write instruction to set the sampling time setting bits (ST1, ST0), comparison time setting bits (CT1, CT0), or A/D conversion end channel select bits (ANE2, ANE1, ANE0) after setting the A/D conversion start channel select bits (ANS2, ANS1, ANS0).</li> <li>As for ANS2, ANS1 and ANS0, the last conversion channel is read until A/D conversion starts. Therefore, if a read-modify-write instruction is used to set the ST1, ST0 bits, the CT1, CT0 bits, and the ANE2, ANE1 and ANE0 bits after the ANS2, ANS1 and ANS0 are set, the value of ANS2, ANS1 and ANS0 may be rewritten.</li> </ul>

**Table 20.4-2 A/D Control Status Register 0 (ADCS0)**

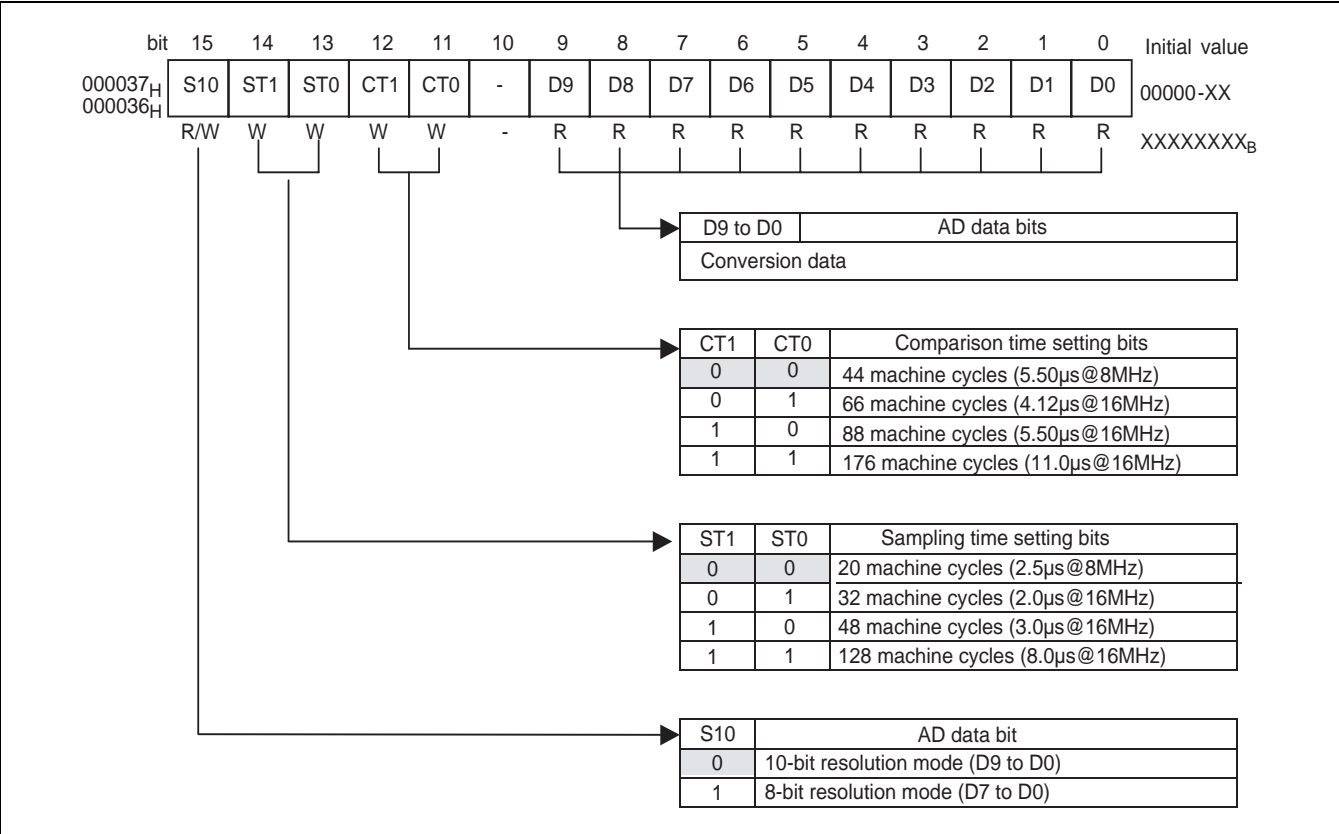
Bit name		Function
bit2 to bit0	ANE2 to ANE0: A/D conversion end channel select bits	<ul style="list-style-type: none"> <li>• These bits set the A/D conversion end channel.</li> <li>• When activated, A/D conversion is performed up to the channel specified by these bits.</li> <li>• When these bits specify the channel specified by ANS2 to ANS0, just that channel is converted. In continuous or stop conversion mode, the start channel specified by ANS2 to ANS0 is converted after the channel specified by these bits. If the start channel is greater than the end channel, the start channel to AN7 and AN0 to the end channel are converted in that order in a single series of conversions.</li> </ul>

### 20.4.3 A/D Data Register (ADCR0/ADCR1)

The A/D data register (ADCR0/ADCR1) holds the result of A/D conversion and selects the resolution of A/D conversion.

■ A/D Data Register (ADCR0/ADCR1)

Figure 20.4-4 A/D Data Register (ADCR0/ADCR1)



**Table 20.4-3 Function Description of Each bit of A/D Control Status Register 0 (ADCS0)**

Bit name		Function
bit15	S10: A/D conversion resolution selection bit	<ul style="list-style-type: none"> <li>This bit selects an A/D conversion resolution.</li> <li>Writing "0" to this bit selects a resolution of 10 bits, and writing "1" to this bit selects a resolution of 8 bits.</li> </ul> <p>(Note) The data bit to be used depends on the resolution.</p>
bit14, bit13	ST1, ST0: Sampling time setting bits	<ul style="list-style-type: none"> <li>These bits select the sampling time for A/D conversion.</li> <li>When A/D conversion is activated, analog input is fetched during the time set in this bit.</li> </ul> <p>(Note) Setting these bits to "00<sub>B</sub>" (for 8 MHz) during 16 MHz operation may disable normal fetching of the analog voltage.</p>
bit12, bit11	CT1, CT0: Comparison time setting bits	<ul style="list-style-type: none"> <li>These bits select the comparison time for A/D conversion.</li> <li>After analog input is fetched (i.e., sampling time elapses), conversion result data is defined and stored in bit9 to bit0 of this register after the time set in these bits.</li> </ul> <p>(Note) Setting these bits to "00<sub>B</sub>" (for 8 MHz) during 16 MHz operation may disable normal acquisition of the analog conversion value.</p>
bit10	Unused bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to this bit has no effect on the operation.</li> </ul>
bit9 to bit0	D9 to D0: A/D data bits	<ul style="list-style-type: none"> <li>The A/D conversion results are stored and the register is rewritten each time conversion ends.</li> <li>Usually, the last conversion value is stored.</li> <li>The initial value of this register is undefined.</li> </ul> <p>(Note) The conversion data protection function is provided. (See "20.6 Operation of the 8/10-bit A/D Converter".) Do not write data to these bits during A/D conversion.</p>

**Notes:**

- To rewrite the S10 bit, do so while the A/D is in a pause before conversion. If the bit is rewritten after the conversion, the contents of ADCR become undefined.
- To read the contents of the ADCR register in 10-bit mode, use a word transfer instruction (MOVW A, 0036H, etc.).

## 20.5 8/10-bit A/D Converter Interrupts

The 8/10-bit A/D converter can generate an interrupt request when the data for the A/D conversion is set in the A/D data register. This function supports the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ 8/10-bit A/D Converter Interrupts

Table 20.5-1 indicates the interrupt control bits of the 8/10-bit A/D converter and the interrupt cause.

**Table 20.5-1 Interrupt Control Bits of the 8/10-bit A/D Converter and the Interrupt Cause**

	8/10-bit A/D converter
Interrupt request flag bit	ADCS1: INT
Interrupt request enable bit	ADCS1: INTE
Interrupt cause	Writing the A/D conversion result to the A/D data register

When A/D conversion is performed and its result is set in the A/D data register (ADCR), the INT bit of the A/D control status register (ADCS1) is set to "1". If the interrupt request is enabled (ADCS1: INTE = 1), an interrupt request is output to the interrupt controller.

### ■ 8/10-bit A/D Converter Interrupts and EI<sup>2</sup>OS

**Table 20.5-2 8/10-bit A/D Converter Interrupts and EI<sup>2</sup>OS**

Interrupt no.	Interrupt control register		Vector table address			EI <sup>2</sup> OS
	Register name	Address	Lower	Upper	Bank	
#11 (0B <sub>H</sub> )	ICR00	0000B0 <sub>H</sub>	FFFFD0 <sub>H</sub>	FFFFD1 <sub>H</sub>	FFFFD2 <sub>H</sub>	O

O: Available

### ■ EI<sup>2</sup>OS Function of the 8/10-bit A/D Converter

Using the EI<sup>2</sup>OS function, the 8/10-bit A/D converter can transfer the A/D conversion result to memory. When the transfer is performed, a conversion data protection function halts the A/D conversion until the A/D conversion data is transferred to memory, and clears the INT bit. The function prevents any part of the data from being lost.

## 20.6 Operation of the 8/10-bit A/D Converter

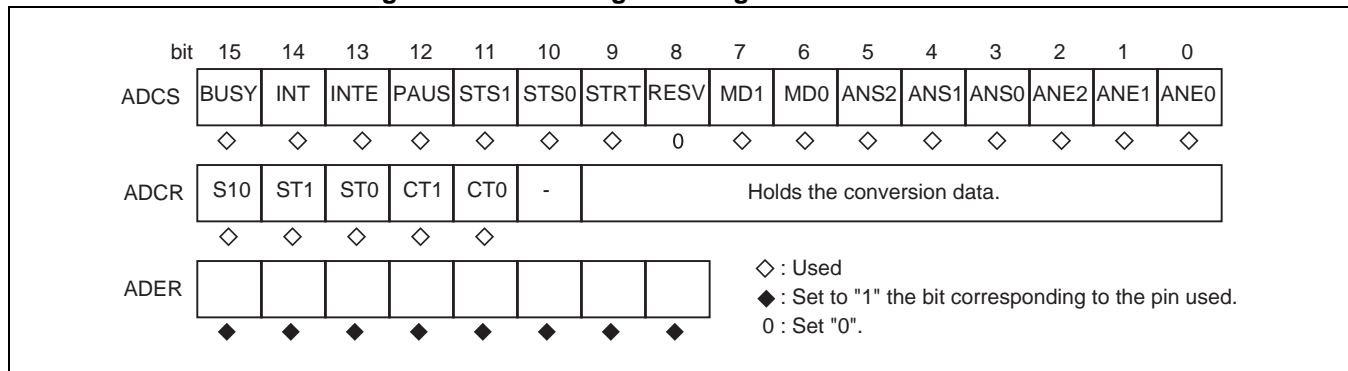
The 8/10-bit A/D converter has three conversion modes: single conversion mode, continuous conversion mode, and stop conversion mode. This section describes operation in each mode.

### ■ Operation in Single Conversion Mode

In single conversion mode, the analog inputs from the channel specified by the ANS bits to the channel specified by the ANE bits are sequentially converted. When the channels up to the end channel specified by the ANE bits have been converted, A/D conversion stops. If the start and end channels are the same (ANS = ANE), just the channel specified by the ANS bits is converted.

Figure 20.6-1 shows the settings required for operation in single conversion mode.

**Figure 20.6-1 Settings for Single Conversion Mode**



Reference:

The following are sample conversion sequences in single conversion mode:

ANS = 000<sub>B</sub>, ANE = 011<sub>B</sub> : AN0 → AN1 → AN2 → AN3 → End

ANS = 110<sub>B</sub>, ANE = 010<sub>B</sub> : AN6 → AN7 → AN0 → AN1 → AN2 End

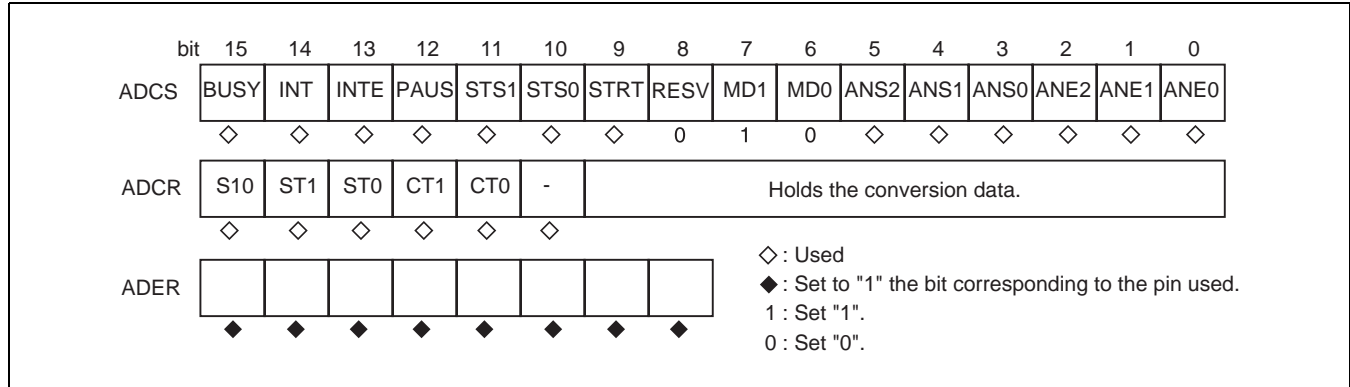
ANS = 011<sub>B</sub>, ANE = 011<sub>B</sub> : AN3 → End

### ■ Operation in Continuous Conversion Mode

In continuous conversion mode, the analog inputs from the channel specified by the ANS bits to the channel specified by the ANE bits are sequentially converted. When the end channel specified by the ANE bits has been processed, A/D conversion starts again from the channel specified by the ANS bits. If the start and end channels are the same (ANS = ANE), the conversion of the channel specified by the ANS bits is repeated.

Figure 20.6-2 shows the settings required for operation in continuous conversion mode.



**Figure 20.6-2 Settings for Continuous Conversion Mode****Reference:**

The following are sample conversion sequences in continuous conversion mode:

ANS = 000<sub>B</sub>, ANE = 011<sub>B</sub> : AN0 → AN1 → AN2 → AN0 → Repeat

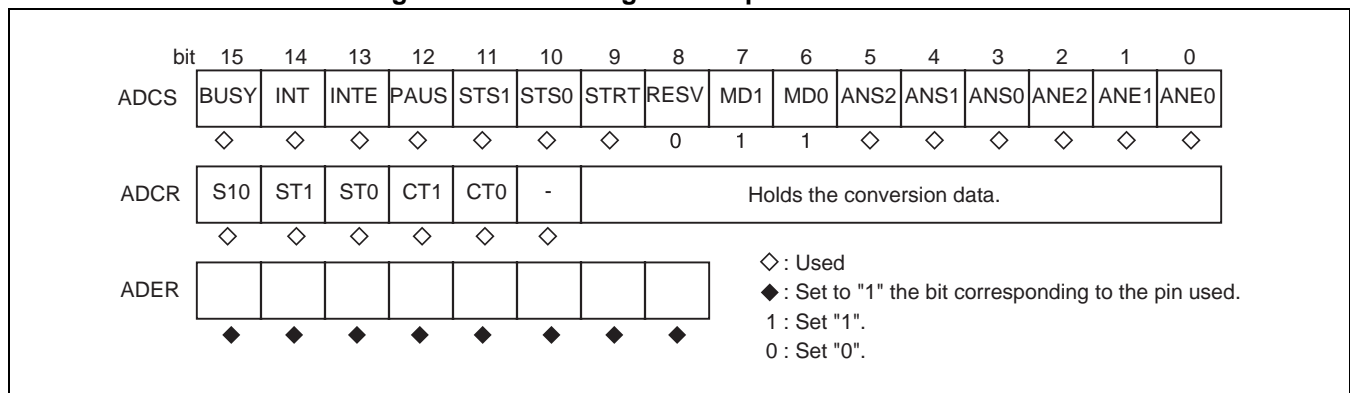
ANS = 110<sub>B</sub>, ANE = 010<sub>B</sub> : AN6 → AN7 → AN0 → AN1 → AN2 → AN6 → Repeat

ANS = 011<sub>B</sub>, ANE = 011<sub>B</sub> : AN3 → AN3 → Repeat

**■ Operation in Stop Conversion Mode**

In stop conversion mode, the analog inputs from the channel specified by the ANS bits to the channel specified by the ANE bits are sequentially converted with a pause after the conversion of each channel. When the end channel specified by the ANE bits has been processed, A/D conversion, with pauses, starts again with the channel specified by the ANS bits. If the start and end channels are the same (ANS = ANE), the conversion of the channel specified by the ANS bits is repeated. To reactivate conversion during a pause, generate the activation cause specified by the STS1 and STS0 bits.

Figure 20.6-3 shows the settings required for operation in stop conversion mode.

**Figure 20.6-3 Settings for Stop Conversion Mode**

---

**Reference:**

The following are sample conversion sequences in stop conversion mode:

ANS = 000<sub>B</sub>, ANE = 011<sub>B</sub> :

AN0 → Pause → AN1 → Pause → AN2 → Pause → AN0 → Repeat

ANS = 110<sub>B</sub>, ANE = 001<sub>B</sub> :

AN6 → Pause → AN7 → Pause → AN0 → Pause → AN1 → AN6 → Repeat

ANS = 011<sub>B</sub>, ANE = 011<sub>B</sub> :

AN3 → Pause → AN3 → Pause → Repeat

---

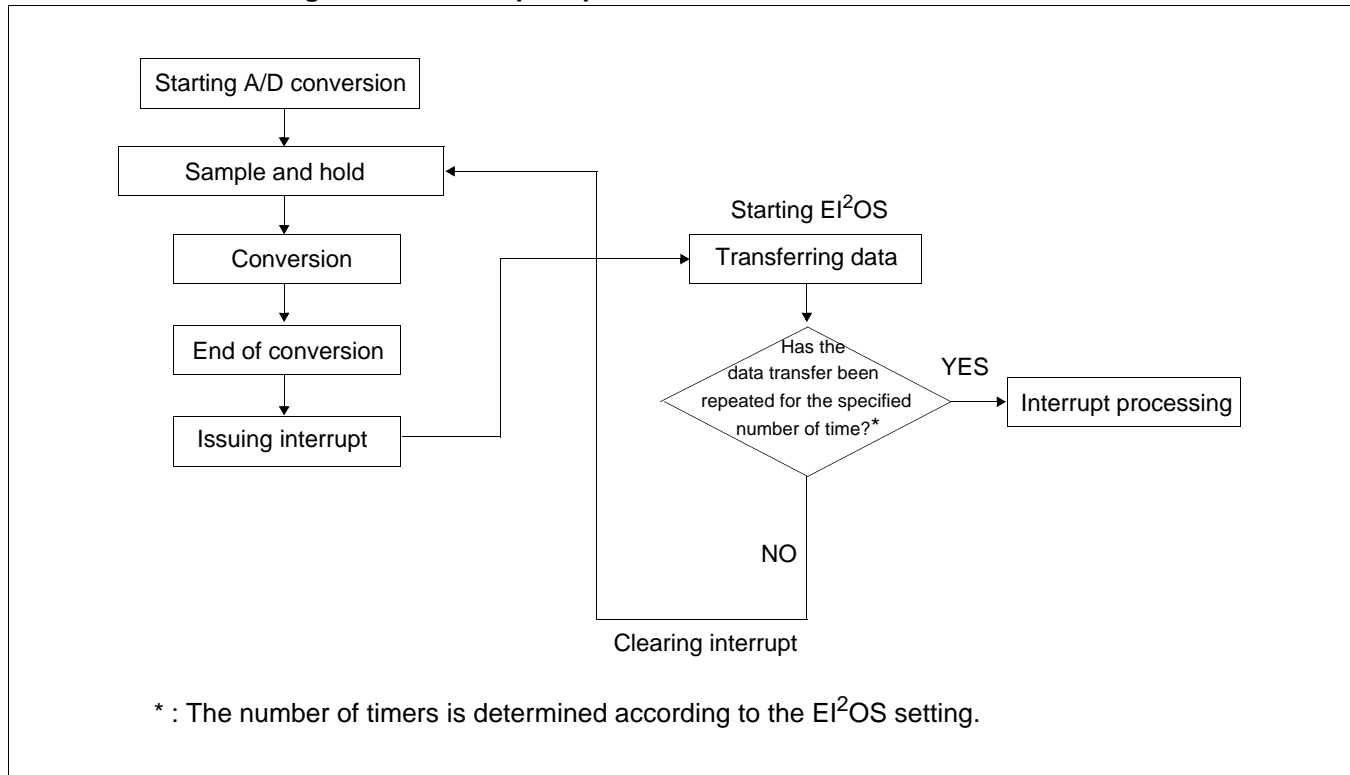
## 20.6.1 Conversion using EI<sup>2</sup>OS

The 8/10-bit A/D converter can use EI<sup>2</sup>OS transfer the A/D conversion result to memory.

### ■ Conversion using EI<sup>2</sup>OS

Figure 20.6-4 shows the operation flow when EI<sup>2</sup>OS is used.

**Figure 20.6-4 Sample Operation Flowchart when EI<sup>2</sup>OS is used**



When EI<sup>2</sup>OS is used, the conversion data protection function prevents any part of the data from being lost even in continuous conversion. Multiple data items can be safely transferred to memory.

## 20.6.2 A/D Conversion Data Protection Function

---

**When A/D conversion is performed in the interrupt enabled state, the conversion data protection function operates.**

---

### ■ A/D Conversion Data Protection Function

The A/D converter has just one data register that holds conversion data. When a single A/D conversion is completed, the data in the data register is rewritten.

If the conversion data were not transferred to memory before the next conversion data was stored, part of the conversion data would be lost. The data protection function operates in the interrupt enabled state (INTE = 1), as described below, to prevent loss of data.

#### ● Data protection function when EI<sup>2</sup>OS is not used

When conversion data is stored in the A/D data register (ADCR0/ADCR1), the INT bit of the A/D control status register1 (ADCS1) is set to "1".

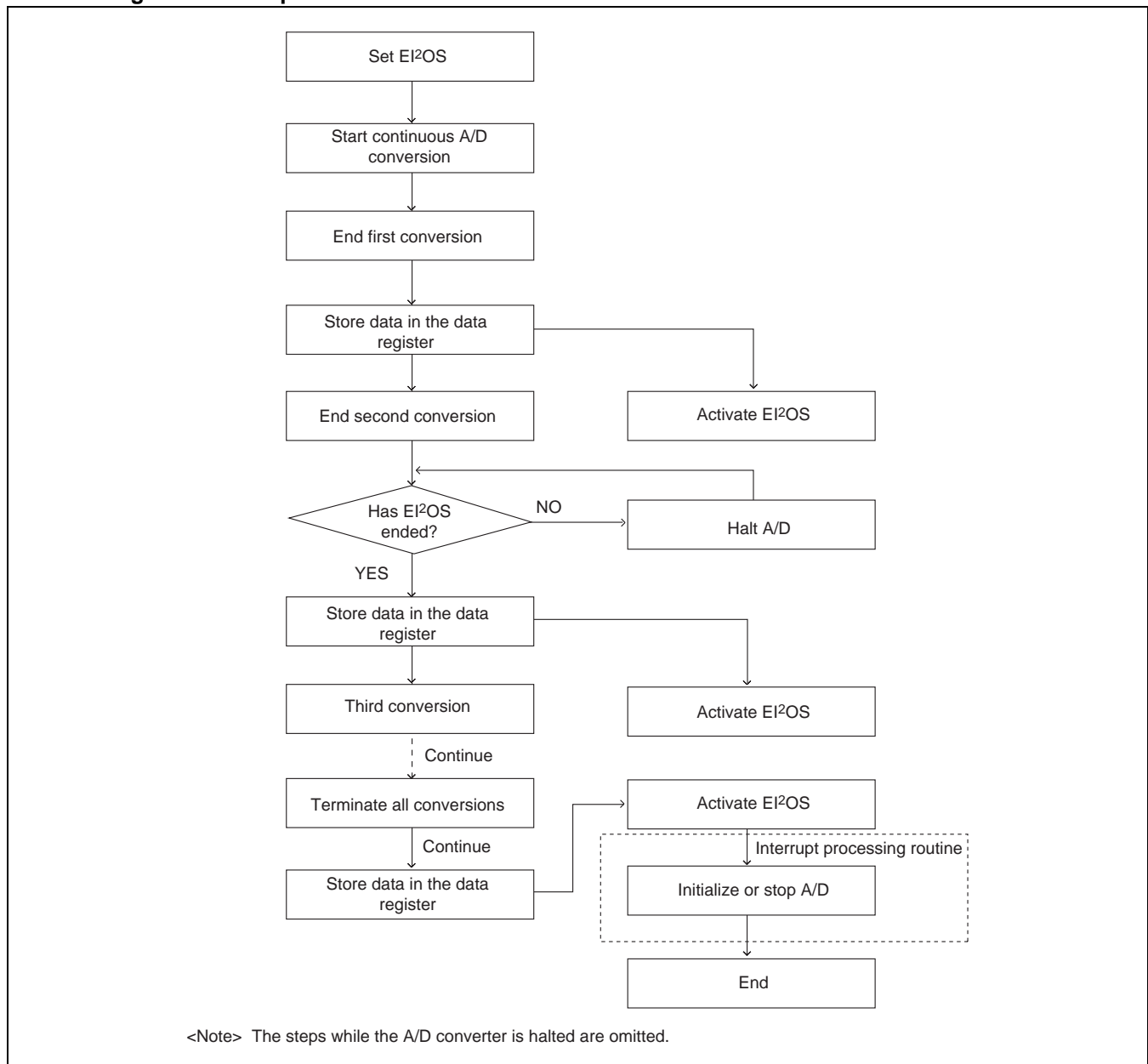
While the INT bit is "1", A/D conversion is halted.

Halt status is released when the INT bit is cleared after data in the A/D data register (ADCR0/ADCR1) has been transferred to memory by the interrupt routine.

#### ● Data protection function when EI<sup>2</sup>OS is used

In continuous conversion using EI<sup>2</sup>OS, the PAUS bit of the A/D control status register1 (ADCS1) is kept at "1" when a conversion ends. This status continues until EI<sup>2</sup>OS finishes transferring the conversion data from the data register to memory. In the meantime, the A/D conversion is halted, and the next conversion data is not stored. When the data transfer to memory is completed, the PAUS bit is cleared to "0" and conversion resumes.

Figure 20.6-5 shows the operation flow of the data protection function when EI<sup>2</sup>OS is used.

**Figure 20.6-5 Operation Flowchart of the Data Protection Function when EI<sup>2</sup>OS is used****Notes:**

- The conversion data protection function operates only in the interrupt enabled state (ADCS1: INTE = 1).
- If interrupts are disabled during a pause in A/D conversion while EI<sup>2</sup>OS is operating, A/D conversion may start again. This will cause new data to be written before the old data is transferred. Reactivation attempted during a pause will cause the old data to be destroyed.
- Reactivation attempted during a pause will destroy the standby data.

## 20.7 Usage Notes on the 8/10-bit A/D Converter

---

### Notes on using the 8/10-bit A/D converter.

---

#### ■ Usage Notes on the 8/10-bit A/D Converter

##### ● Analog input pin

The A/D input pins are also used as the I/O pins of port 5. The port 5 data register (DDR5) and analog input enable register (ADER) determine which pin is used for which purpose.

To use a pin as analog input, write "0" to the corresponding bit of DDR5 and change the port setting to input. Then, set the analog input mode (ADEx = 1) in the ADER register and determine the input gate of the port.

If an intermediate-level signal is input in the port input mode (ADEx = 0), a leakage current flows through the gate.

##### ● Note on using an internal timer

To start the A/D converter with an internal timer, set the STS1 and STS0 bits of A/D control status register 1 (ADCS1) accordingly. Set the input value of the internal timer at the inactive level (L for the internal timer). Otherwise, operation may start concurrently with writing to the ADCS register.

##### ● Sequence of turning on the A/D converter and analog input

Do not turn on power to the A/D converter (AVcc, AVR) and to the analog inputs (AN0 to AN7) before the digital power supply (Vcc) has been turned on.

Do not turn off the digital power supply (Vcc) before power to the A/D converter and the analog inputs has been turned off.

##### ● Supply voltage to the A/D converter

The supply voltage to the A/D converter (AVcc) must not exceed the digital power supply (Vcc); otherwise, latch-up may occur.

## 20.8 Sample Program 1 for the 8/10-bit A/D Converter (Single Conversion Mode Using EI<sup>2</sup>OS)

This section contains a sample program for A/D conversion in single conversion mode using EI<sup>2</sup>OS.

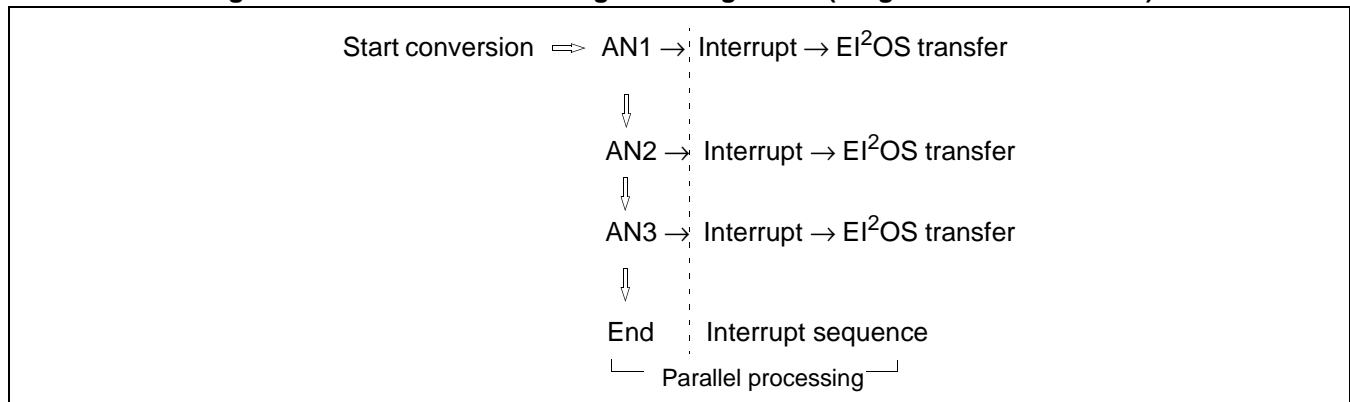
### ■ Sample Program for Single Conversion Mode using EI<sup>2</sup>OS

#### ● Processing

- Analog inputs AN1 to AN3 are converted once.
- The conversion data is sequentially transferred to addresses 200<sub>H</sub> to 205<sub>H</sub>.
- A resolution of 10 bits is selected.
- The conversion is activated by software.

Figure 20.8-1 shows a flowchart of the program using EI<sup>2</sup>OS (single conversion mode).

**Figure 20.8-1 Flowchart of Program using EI<sup>2</sup>OS (Single Conversion Mode)**



#### ● Coding example

BAPL	EQU	000100H	;Lower buffer address pointer
BAPM	EQU	000101H	;Intermediate buffer address pointer
BAPH	EQU	000102H	;Upper buffer address pointer
ISCS	EQU	000103H	;EI <sup>2</sup> OS status register
IOAL	EQU	000104H	;Lower I/O address register
IOAH	EQU	000105H	;Upper I/O address register
DCTL	EQU	000106H	;Lower data counter
DCTH	EQU	000107H	;Upper data counter
DDR5	EQU	000015H	;Port 5 direction register
ADER	EQU	000017H	;Analog input enable register
ICR00	EQU	0000B0H	;Interrupt control register for A/DC
ADCS0	EQU	000034H	;A/D control status register
ADCS1	EQU	000035H	;

```

ADCR0    EQU        000036H        ;A/D data register
ADCR1    EQU        000037H        ;
;-----Main program-----
CODE     CSEG
START:                                       ;Assumes that the stack pointer (SP) has already
                                           ;been initialized
        AND         CCR,#0BFH        ;Disables interrupts
        MOV         ICR00,#00H        ;Interrupt level: 0 (highest priority)
        MOV         BAPL,#00H        ;Sets the address to which the conversion data is
                                           ;transferred and stored
        MOV         BAPM,#02H        ;(Uses 200H to 205H.)
        MOV         BAPH,#00H        ;
        MOV         ISCS,#18H        ;Transfers word data, adds 1 to the address, then
                                           ;transfers the data from I/O to memory
        MOV         IOAL,#36H        ;Sets the address of the analog data register as the
        MOV         IOAH,#00H        ;transfer source address pointer
        MOV         DCTL,#03H        ;Sets the EI2OS transfer count to three, which is the
                                           ;same value as the conversion count
        MOV         DDR5,#11110001B  ;Sets P51 to P53 as input
        MOV         ADER,#00001110B  ;Sets P51/AN1 to P53/AN3 as analog inputs
        MOV         DCTH,#00H        ;
        MOV         ADCS0,#0BH        ;Single activation. Converts AN1 to AN3
        MOV         ADCS1,#0A2H        ;Software activation. Begins A/D conversion.
                                           ;Enables interrupts
        MOV         ILM,#07H        ;Sets ILM in PS to level 7
        OR          CCR,#40H        ;Enables interrupts
LOOP:    MOV         A,#00H        ;Endless loop
        MOV         A,#01H
        BRA         LOOP
;-----Interrupt program-----
ED_INT1:
        MOV         I:ADCS1,#00H        ;Stops A/D conversion. Clears and disables the
                                           ;interrupt flag
        RETI                        ;Returns from interrupt
CODE     ENDS
;-----Vector setting-----
VECT     CSEG        ABS=0FFH
        ORG         0FFD0H        ;Sets vector for interrupt #11 (0BH)
        DSL         ED_INT1
        ORG         0FFDCH        ;Sets reset vector
        DSL         START
        DB          00H        ;Sets single-chip mode
VECT     ENDS
        END          START

```



## 20.9 Sample Program 2 for the 8/10-bit A/D Converter (Continuous Conversion Mode Using EI<sup>2</sup>OS)

This section contains a sample program for A/D conversion in continuous conversion mode using EI<sup>2</sup>OS.

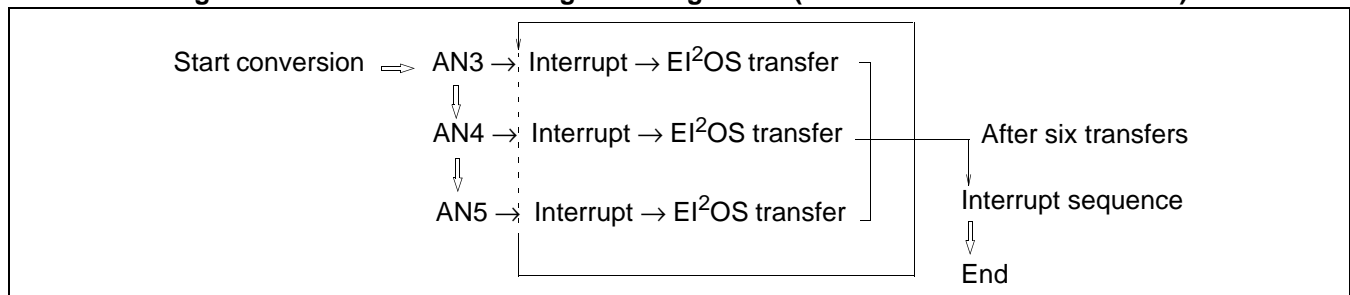
### ■ Sample Program for Continuous Conversion Mode using EI<sup>2</sup>OS

#### ● Processing

- Analog inputs AN3 to AN5 are converted twice. Two conversion data items are obtained for each channel.
- The conversion data is sequentially transferred to addresses 600<sub>H</sub> to 60B<sub>H</sub>.
- A resolution of 10 bits is selected.
- The conversion is activated by 16-bit reload timer 1.

Figure 20.9-1 shows a flowchart of the program using EI<sup>2</sup>OS (continuous conversion mode).

**Figure 20.9-1 Flowchart of Program using EI<sup>2</sup>OS (Continuous Conversion Mode)**



#### ● Coding example

BAPL	EQU	000100H	;Lower buffer address pointer
BAPM	EQU	000101H	;Middle buffer address pointer
BAPH	EQU	000102H	;Upper buffer address pointer
ISCS	EQU	000103H	;EI <sup>2</sup> OS status register
IOAL	EQU	000104H	;Lower I/O address register
IOAH	EQU	000105H	;Upper I/O address register
DCTL	EQU	000106H	;Lower data counter
DCTH	EQU	000107H	;Upper data counter
DDR5	EQU	000015H	;Port 5 direction register
ADER	EQU	000017H	;Analog input enable register
ICR00	EQU	0000B0H	;Interrupt control register for A/DC
ADCS0	EQU	000034H	;A/D control status register
ADCS1	EQU	000035H	;
ADCR0	EQU	000036H	;A/D data register
ADCR1	EQU	000037H	;
TMCSRL1	EQU	000086H	;Lower control status register 1
TMC SRH1	EQU	000087H	;
TMRD1	EQU	000088H	;16-bit reload register 1

;-----Main program-----

CODE            CSEG

```

START:                                     ;Assumes that the stack pointer (SP) has already
                                           ;been initialized
      AND      CCR,#0BFH                   ;Disables interrupts
      MOV      ICR10,#08H                 ;Interrupt level; 0 (highest priority). Enables
                                           ;interrupts
      MOV      BAPL,#00H                   ;Sets the address to which conversion data is stored
      MOV      BAPM,#06H                   ;(Uses 600H to 60BH.)
      MOV      BAPH,#00H                   ;
      MOV      ISCS,#18H                   ;Transfers word data, adds 1 to the address, then
                                           ;transfers from I/O to memory
      MOV      IOAL,#36H                   ;Sets the address of the analog data register as the
      MOV      IOAH,#00H                   ;transfer source address pointer
      MOV      DCTL,#06H                   ;Six transfers by EI2OS (two transfers each for three
                                           ;channels)
      MOV      DDR5,#00000000B             ;Sets P50 to P57 as input
      MOV      ADER,#00111000B             ;Sets P53/AN3 to P55/AN5 as analog input
      MOV      DCTH,#00H                   ;
      MOV      ADCS0,#9DH                   ;Continuous conversion mode. Converts
                                           ;AN3 to AN5 CH
      MOV      ADCS1,#0A8H                 ;Activates the 16-bit timer, starts A/D conversion,
                                           ;and enables interrupts
      MOV      WTMRD1,#0320H               ;Sets the timer value to 800 (320H), 100μs
      MOV      TMCSRH1,#00H               ;Sets the clock source to 125 ns and disables
                                           ;external trigger
      MOV      TMCSRL1,#12H               ;Disables timer output, disables interrupts, and
                                           ;enables reload
      MOV      TMCSRL1,#13H               ;Activates the 16-bit reload timer 1
      MOV      ILM,#07H                   ;Sets ILM in PS to level 7
      OR       CCR,#40H                   ;Enables interrupts
LOOP:  MOV      A,#00H                     ;Endless loop
      MOV      A,#01H
      BRA      LOOP

;-----Interrupt program-----
ED_INT1:
      MOV      I:ADCS1,#80H               ;Does not stop A/D conversion. Clears and disables
                                           ;the interrupt flag
      RETI                                 ;Returns from interrupt
CODE   ENDS

;-----Vector setting-----
VECT   CSEG      ABS=0FFH
      ORG      0FFD0H                     ;Sets vector for interrupt #11 (0BH)
      DSLED_INT1
      ORG      0FFDCH                     ;Sets reset vector
      DSL      START
      DB       00H                         ;Sets single-chip mode
VECT   ENDS
      ENDSTART

```

## 20.10 Sample Program 3 for the 8/10-bit A/D Converter (Stop Conversion Mode Using EI<sup>2</sup>OS)

This section contains a sample program for A/D conversion in stop conversion mode using EI<sup>2</sup>OS.

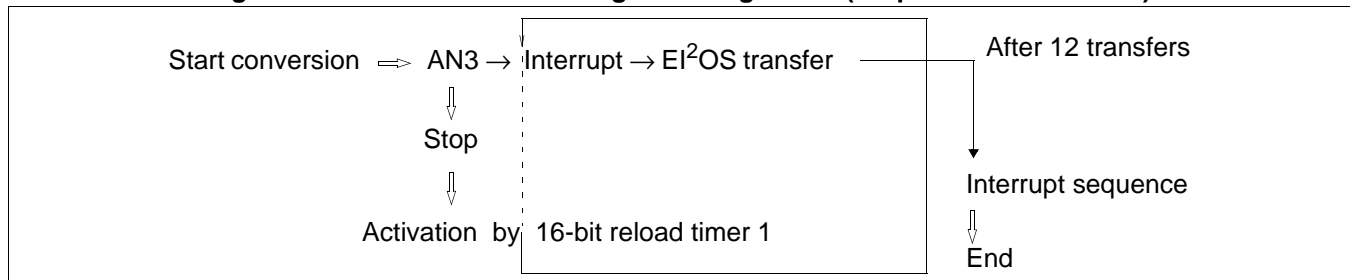
### ■ Sample Program for Stop Conversion Mode using EI<sup>2</sup>OS

#### ● Processing

- Analog input AN3 is converted 12 times at regular intervals.
- The conversion data is sequentially transferred to addresses 600<sub>H</sub> to 617<sub>H</sub>.
- A resolution of 10 bits is selected.
- The conversion is activated by 16-bit reload timer 1.

Figure 20.10-1 shows a flowchart of the program using EI<sup>2</sup>OS (stop conversion mode).

**Figure 20.10-1 Flowchart of Program using EI<sup>2</sup>OS (Stop Conversion Mode)**



#### ● Coding example

BAPL	EQU	000100H	;Lower buffer address pointer
BAPM	EQU	000101H	;Middle buffer address pointer
BAPH	EQU	000102H	;Upper buffer address pointer
ISCS	EQU	000103H	;EI <sup>2</sup> OS status register
IOAL	EQU	000104H	;Lower I/O address register
IOAH	EQU	000105H	;Upper I/O address register
DCTL	EQU	000106H	;Lower data counter
DCTH	EQU	000107H	;Upper data counter
DDR5	EQU	000015H	;Port 5 direction register
ADER	EQU	000017H	;Analog input enable register
ICR00	EQU	0000B0H	;Interrupt control register for A/DC
ADCS0	EQU	000034H	;A/D control status register
ADCS1	EQU	000035H	;
ADCR0	EQU	000036H	;A/D data register
ADCR1	EQU	000037H	;
TMCSRL1	EQU	000086H	;Lower control status register 1
TMCSRH1	EQU	000087H	;

```

TMRD1    EQU        000088H                ;16-bit reload register 1
;-----Main program-----
CODE      CSEG
START:                                         ;Assumes that the stack pointer (SP) has already
                                         ;been initialized
        AND         CCR,#0BFH              ;Disables interrupts
        MOV         ICR00,#08H              ;Interrupt level: TMCSR1:L0 (highest priority)
        MOV         BAPL,#00H              ;Sets the address to which conversion data is stored
        MOV         BAPM,#06H              ;(Uses 600H to 617H.)
        MOV         BAPH,#00H              ;
        MOV         ISCS,#19H              ;Transfers word data, adds 1 to the address,
                                         ;transfers from I/O to memory, then ends by a
                                         ;resource request
        MOV         IOAL,#36H              ;Sets the address of the analog data register as the
        MOV         IOAH,#00H              ;transfer source address pointer
        MOV         DCTL,#0CH              ;Transfers only channel 3 twelve times by EI2OS
        MOV         DDR5,#00000000B        ;Sets P50 to P57 as input
        MOV         ADER,#00001000B        ;Sets P53/AN3 as analog input
        MOV         ADCS0,#0DBH            ;Stop conversion mode. Converts AN3 CH
        MOV         ADCS1,#0A8H            ;Activates the 16-bit timer, starts A/D conversion,
                                         ;and enables interrupts
        MOV         WTMRD1,#0320H          ;Sets the timer value to 800 (320H), 100 µs
        MOV         TMCSRH1,#00H          ;Sets the clock source to 125 ns and disables
                                         ;external trigger
        MOV         TMCSRL1,#12H          ;Disables timer output, disables interrupts, and
                                         ;enables reload
        MOV         TMCSRL1,#13H          ;Activates the 16-bit reload timer 1
        MOV         ILM,#07H              ;Sets ILM in PS to level 7
        OR          CCR,#40H              ;Enables interrupts
LOOP:     MOVA,#00H                        ;Endless loop
        MOVA,#01H
        BRA         LOOP
;-----Interrupt program-----
ED_INT1:
        MOV         I:ADCS1,#80H          ;Does not stop A/D conversion. Clears and disables
                                         ;the interrupt flag
        RETI                               ;Returns from interrupt
CODE      ENDS
;-----Vector setting-----
VECT      CSEG        ABS=0FFH
        ORG0         FFD0H                ;Sets vector for interrupt #11 (0BH)
        DSL          ED_INT1
        ORG          0FFDCH              ;Sets reset vector
        DSL          START
        DB           00H                ;Sets single-chip mode
VECT      ENDS
        END          START

```



# ***CHAPTER 21***

---

# ***ROM CORRECTION FUNCTION***

**This chapter describes the functions and operation of the ROM correction function.**

- 21.1 Overview of the ROM Correction Function
- 21.2 Block Diagram of ROM Correction Function
- 21.3 ROM Correction Function Registers
- 21.4 Operation of the ROM Correction Function
- 21.5 Example of Using ROM Correction Function

## 21.1 Overview of the ROM Correction Function

---

An instruction code to be read by the CPU is replaced forcibly with an INT9 instruction code (01<sub>H</sub>) when the corresponding address is equal to the value set in a program address detect register. A program patch application function can be implemented by processing with the INT #9 interrupt routine.

---

### ■ Program Address Detection Registers (× 2)

There are two program address detection registers (PADR0/PADR1), each is provided with an interrupt enable bit and interrupt flag.

### ■ ROM Correction Interrupts

When the interrupt enable bit is "1", the value set in the program address detection register is compared with the address. If the value matches the address, "1" is set in the interrupt flag bit and the instruction code to be read to the CPU, is forcibly replaced with an INT9 instruction code.

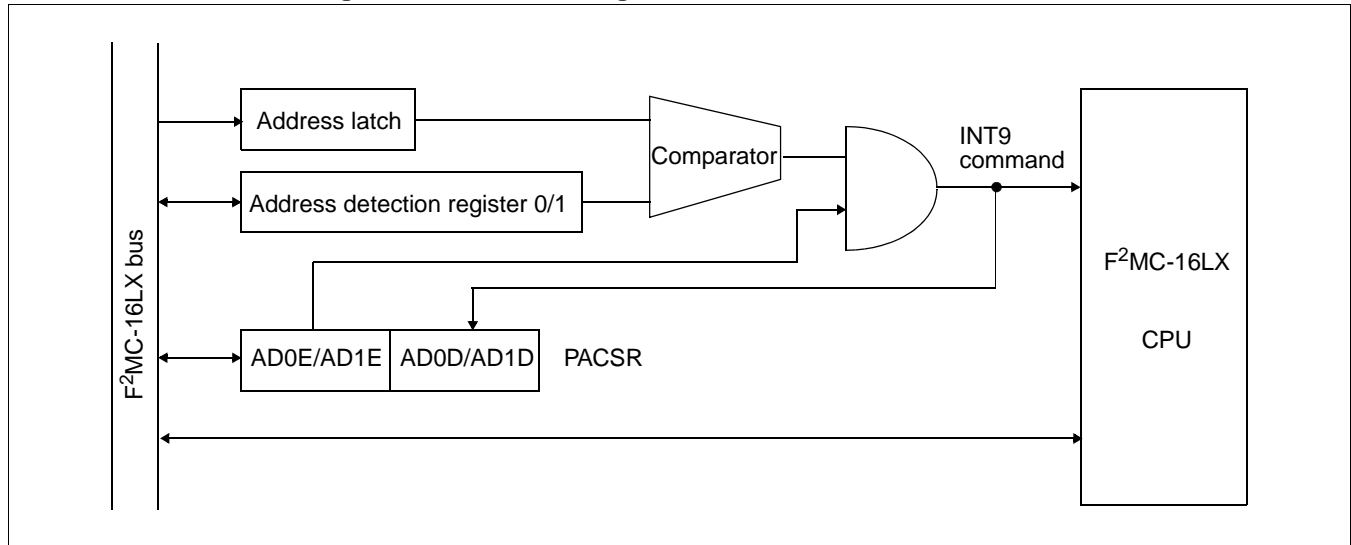
The interrupt flag bit is cleared to "0" by writing "0" to it using an instruction.

## 21.2 Block Diagram of ROM Correction Function

The block diagram of ROM correction function is shown as below.

### ■ Block Diagram of ROM Correction Function

Figure 21.2-1 Block Diagram of ROM Correction Function







## 21.3.1 Program Address Detection Register (PADR0/PADR1)

The program address detection register (PADR0/PADR1) is a 24-bit register and used to store the address to be compared with internal address bus.

### ■ Program Address Detection Register 0/1 (PADR0/PADR1)

Figure 21.3-2 Program Address Detection Register 0/1

Program Address Detection Register 0/1				
	Upper byte	Middle byte	Lower byte	
Address : 1FF2 <sub>H</sub> /1FF1 <sub>H</sub> /1FF0 <sub>H</sub>	PADRH0	PADRM0	PADRL0	PADR0
Address : 1FF5 <sub>H</sub> /1FF4 <sub>H</sub> /1FF3 <sub>H</sub>	PADRH1	PADRM1	PADRL1	PADR1
Read/write ⇨	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(XXXXXXXX <sub>B</sub> )	(XXXXXXXX <sub>B</sub> )	(XXXXXXXX <sub>B</sub> )	

The value written to this register is compared with a target address. If the value matches the address, and the corresponding interrupt enable bit of the PACSR register is "1", the corresponding interrupt bit is set to "1" to request the CPU to generate an INT9 instruction. If the corresponding interrupt enable bit is "0", no operation is performed.

Table 21.3-1 lists the correspondence between the program address detection register and PACSR.

Table 21.3-1 Correspondence between Program Address Detection Register and PACSR

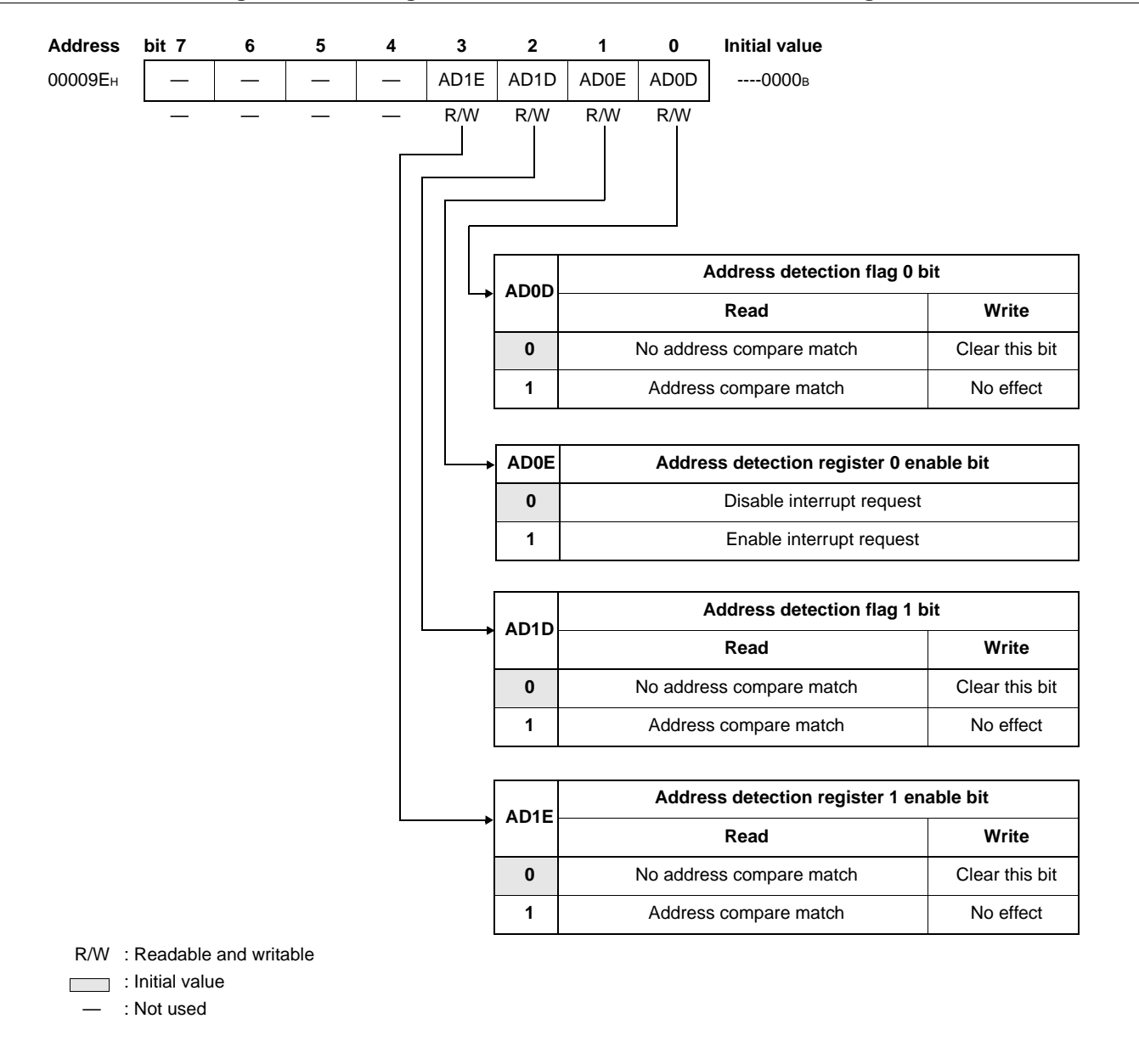
Program address detection register	Interrupt enable bit	Interrupt bit
PADR0	AD0E	AD0D
PADR1	AD1E	AD1D

### 21.3.2 Program Address Detection Control Status Register (PACSR)

The program address detection control status register (PACSR) is an 8-bit register and used to control the operation of ROM correction function.

#### ■ Program Address Detection Control Status Register (PACSR)

Figure 21.3-3 Program Address Detection Control Status Register



**Table 21.3-2 Program Address Detection Control Status Register**

Bit name		Function
bit7 to bit4	Reserved bits	<ul style="list-style-type: none"> <li>Always write “0” to these bits.</li> </ul>
bit3	AD1E: Address detection register 1 enable bit	<ul style="list-style-type: none"> <li>ADR1 operation enable bit.</li> <li>When this bit is “1”, the value set in the PADR1 register is compared with the address. If the two values are equal, an INT9 instruction is generated and the AD1D bit is set to “1”.</li> </ul>
bit2	AD1D: Address detection flag 1 bit	<ul style="list-style-type: none"> <li>ADR1 address match detection bit.</li> <li>This bit is set to “1” to indicate that the value set in the PADR1 register matches the address. It is cleared to “0” by writing “0” to it. It is left unchanged by writing “1” to it.</li> </ul>
bit1	AD0E: Address detection register 0 enable bit	<ul style="list-style-type: none"> <li>ADR0 operation enable bit.</li> <li>When this bit is “1”, the value set in the PADR0 register is compared to the address. If the two values are equal, an INT9 instruction is generated and the AD0D bit is set to “1”.</li> </ul>
bit0	AD0D: Address detection flag 0 bit	<ul style="list-style-type: none"> <li>ADR0 address match detection bit.</li> <li>This bit is set to “1” to indicate that the value set in the PADR0 register is equal to the address. It is cleared to “0” by writing “0” to it. It is left unchanged by writing “1” to it.</li> </ul>

## 21.4 Operation of the ROM Correction Function

---

If the program counter specifies the same address as that in program address detection register (PADR), the INT9 instruction is executed. The ROM correction function can be done by processing the INT9 instruction routine.

---

### ■ Operation of the ROM Correction Function

An instruction code to be read by the CPU is replaced forcibly with an INT9 instruction code (01<sub>H</sub>) when the corresponding address is equal to the value set in an address detection register. Therefore, the CPU executes the INT9 instruction when executing the set instruction.

A program patch application function can be implemented by processing with the INT #9 interrupt routine.

There are two address detection registers, of which each is provided with an interrupt enable bit and interrupt flag. When the address is equal to the value set in the address detection register, and the interrupt enable bit is "1", assume the following: the interrupt flag is set to "1", and the instruction code to be read by the CPU is replaced forcibly with the INT9 instruction code. The interrupt flag is cleared to "0" by writing "0" to it using an instruction.

---

#### Note:

The address match detection function fails if an address later than the first byte of the instruction is set in the address detection register. The value in the set address is replaced with "01H" so a wrong instruction is executed or an invalid address is accessed. Before changing the value set in the address detection register, set the interrupt enable bit to "0". If data is written while the interrupt enable bit is "1", the address may be wrongly detected during writing, causing a malfunction.

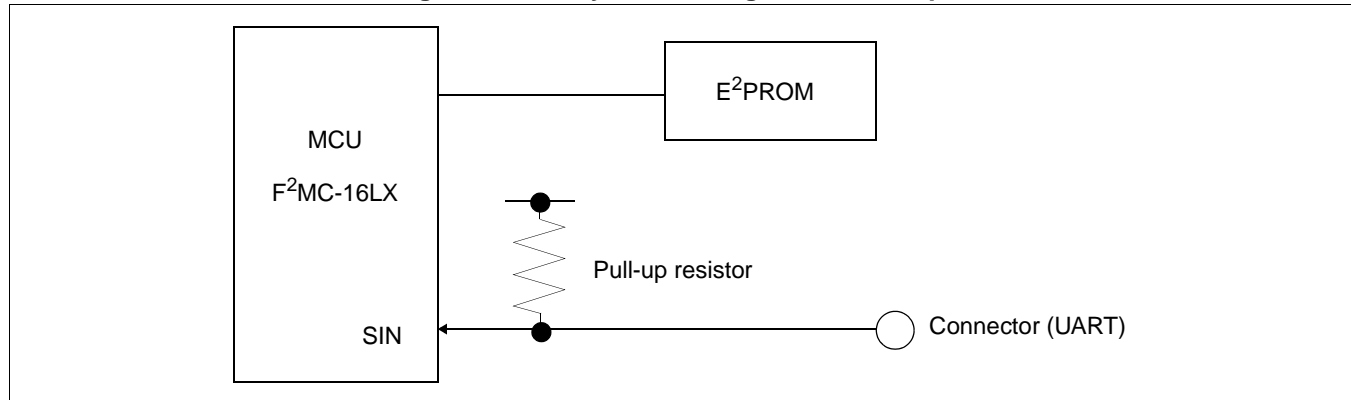
---

## 21.5 Example of Using ROM Correction Function

This section contains example of Using the Address Match Detection Function.

### ■ System Configuration

Figure 21.5-1 System Configuration Example



### ■ E²PROM Memory Map

Table 21.5-1 E²PROM Memory Map

Address	Meaning
0000 <sub>H</sub>	Number of bytes of patch program No. 0 (0 for no program error)
0001 <sub>H</sub>	Bit7 to bit0 of program address No. 0
0002 <sub>H</sub>	Bit15 to bit8 of program address No. 0
0003 <sub>H</sub>	Bit24 to bit26 of program address No. 0
0004 <sub>H</sub>	Number of bytes of patch program No. 1 (0 for no program error)
0005 <sub>H</sub>	Bit7 to bit0 of program address No. 1
0006 <sub>H</sub>	Bit15 to bit8 of program address No. 1
0007 <sub>H</sub>	Bit24 to bit16 of program address No. 1
to 0010 <sub>H</sub> <sup>+</sup> Number of bytes of patch program No. 0	Original of patch program No. 0

### ■ Initial State

The contents of E²PROM are all 0's.

### ■ If a Program Error Occurs

The original of a patch program and its address is transferred to the MCU via the connector (UART). The MCU writes the information to E<sup>2</sup>PROM.

### ■ Reset Sequence

After the reset sequence is completed, the MCU reads the value of E<sup>2</sup>PROM. If the number of bytes of the patch program is not 0, the MCU reads the original patch program and writes it to RAM. Then, the MCU sets the program address to PADDR0 or PADDR1 and enables the program to run. The first address of the program written to RAM is saved in RAM as specified for each address detection register.

### ■ INT9 Interrupt

During execution of an interrupt routine, control checks the interrupt flag for an address in which an interrupt was enabled, and branches to the corresponding program. The information stacked by the interrupt is deleted. The interrupt flag is also cleared.

**Figure 21.5-2 System Configuration Example**

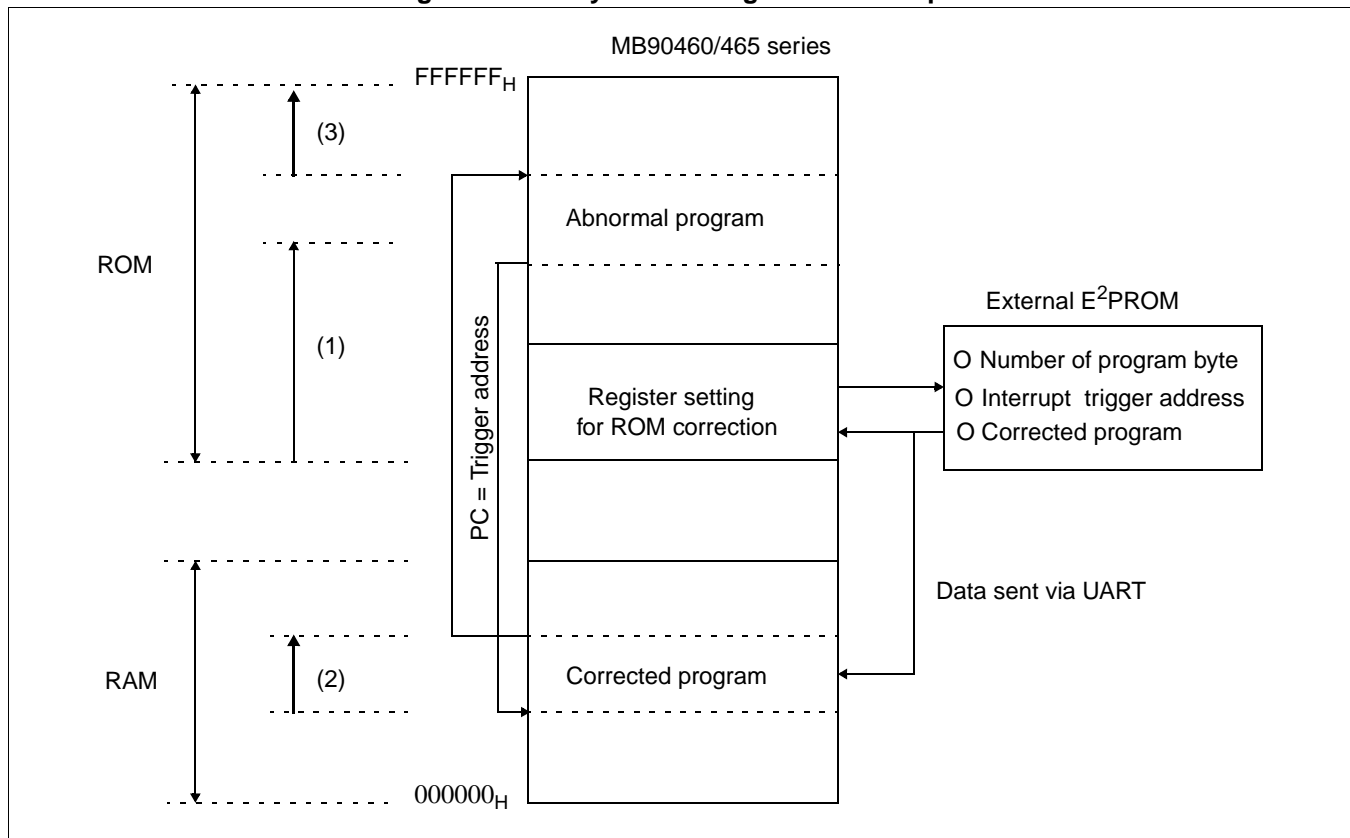
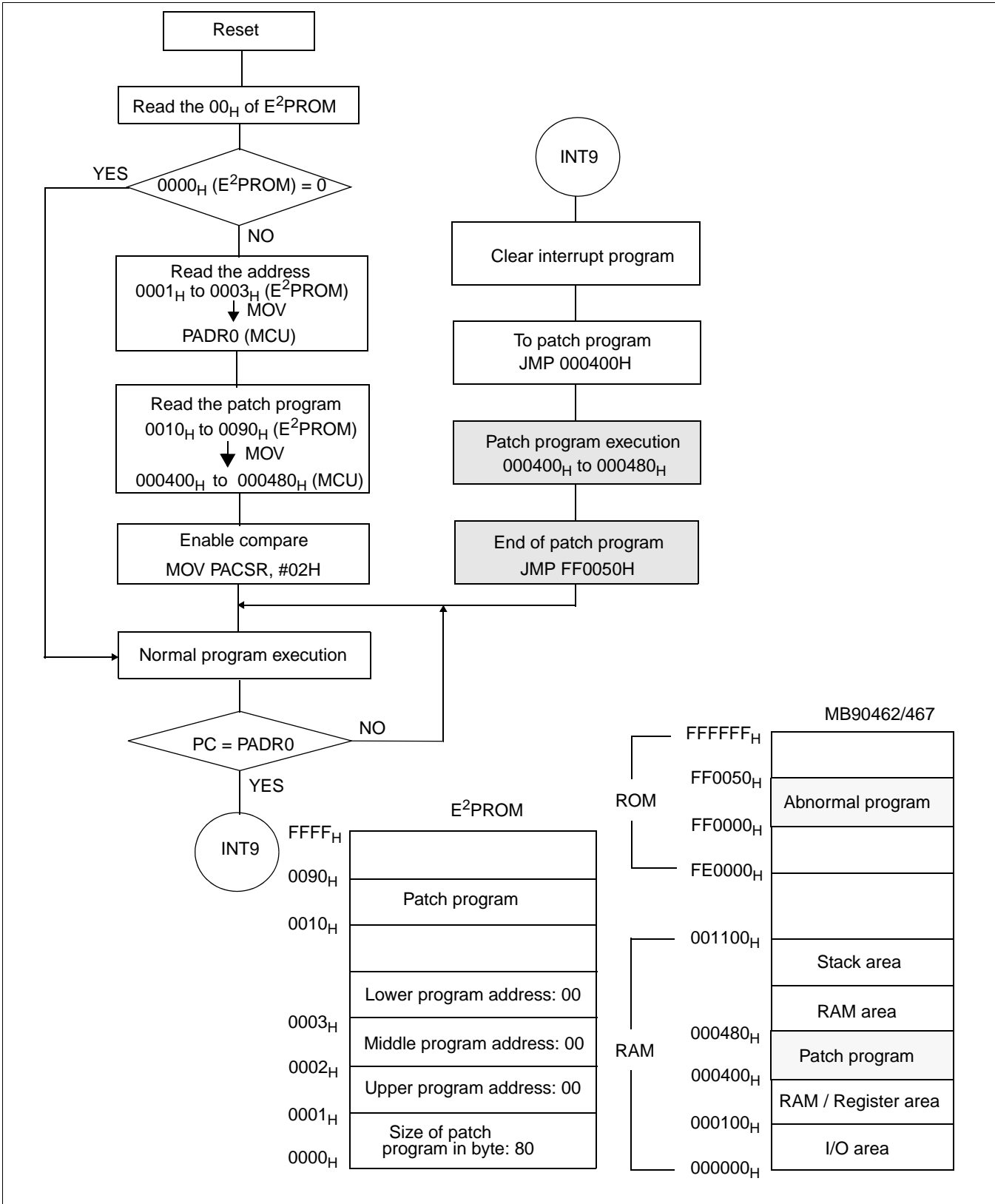


Figure 21.5-3 Flowchart of Program Patch Processing







# **CHAPTER 22**

---

## ***ROM MIRRORING FUNCTION SELECTION MODULE***

**This chapter explains the function and operation of the MB90460/465 series ROM mirroring function selection module.**

22.1 Overview of the ROM Mirroring Function Selection Module

22.2 ROM Mirroring Function Selection Register (ROMM)

## 22.1 Overview of the ROM Mirroring Function Selection Module

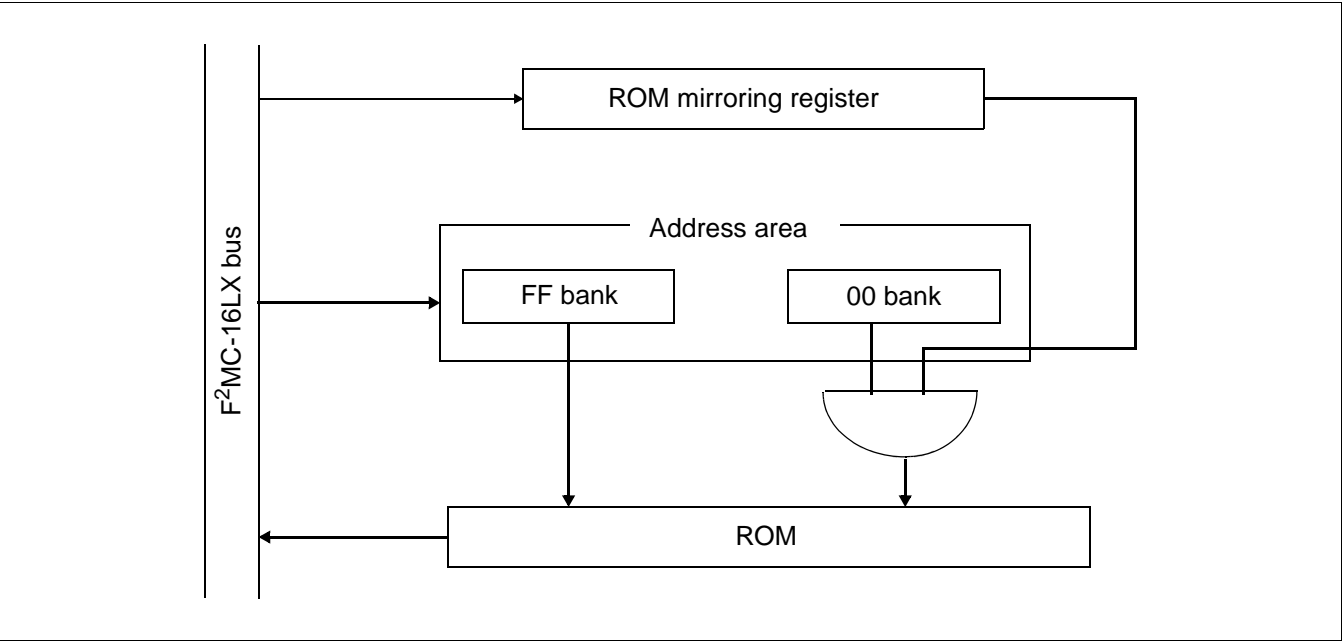
The ROM mirroring function selection module can access bank FF located in ROM from bank 00 by setting the register.

### ■ ROM Mirroring Function Selection Module Register

ROM Mirror Function Selection Register										
	bit	15	14	13	12	11	10	9	8	
Address : 00006F <sub>H</sub>		—	—	—	—	—	—	—	MI	ROMM
Read/write ➡		(-)	(-)	(-)	(-)	(-)	(-)	(-)	(W)	
Initial value ➡		(-)	(-)	(-)	(-)	(-)	(-)	(-)	(1)	

### ■ ROM Mirroring Function Selection Module Block Diagram

Figure 22.1-1 ROM Mirroring Function Selection Module Block Diagram



## 22.2 ROM Mirroring Function Selection Register (ROMM)

The ROM mirroring function selection register (ROMM) is used to enable mirroring function.

### ■ ROM Mirroring Function Selection Register (ROMM)

Figure 22.2-1 ROM Mirroring Function Selection Register

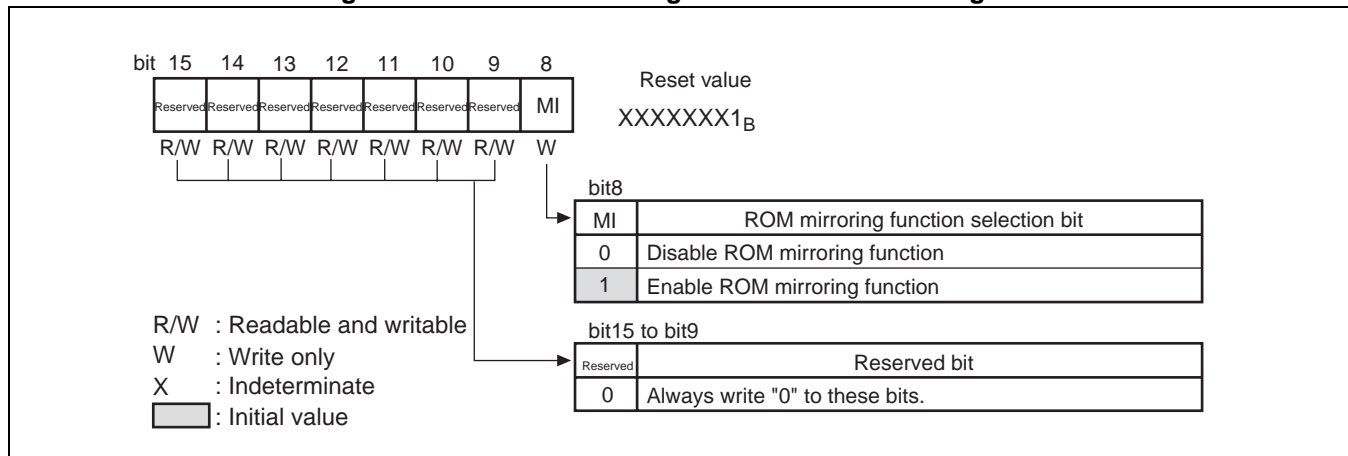


Table 22.2-1 ROM Mirroring Function Selection Register

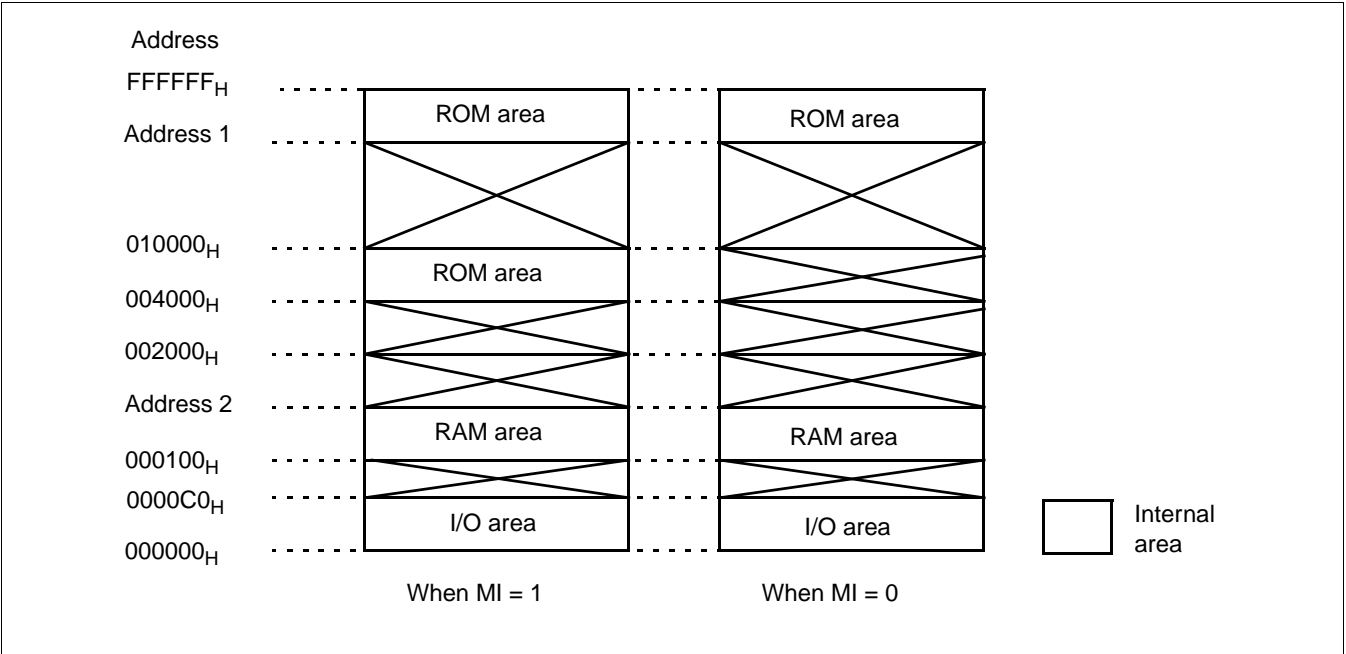
Bit name		Function
bit15 to bit9	Reserved bits	The read value is undefined. Always write "0" to these bits.
bit8	MI: ROM mirroring function select bit	This bit enables or disables the ROM mirroring function. When set to "0": Disables ROM mirroring function When set to "1": Enables ROM mirroring function When the ROM mirroring function is enabled (MI = 1), data at ROM addresses FF4000 <sub>H</sub> to FFFFFFF <sub>H</sub> can be read by accessing addresses 004000 <sub>H</sub> to 00FFFF <sub>H</sub> .

Note:

Bank 00 accesses FF4000<sub>H</sub> to FFFFFFF<sub>H</sub> from 004000<sub>H</sub> to 00FFFF<sub>H</sub>. Therefore, FFF000<sub>H</sub> to FF3FFF<sub>H</sub> cannot be accessed even by selecting the ROM mirroring function.

	MB90462	MB90467	MB90F462	MB90F462A	MB90F463A	MB90V460
Address 1	FF0000 <sub>H</sub>	FF0000 <sub>H</sub>	FF0000 <sub>H</sub>	FF0000 <sub>H</sub>	FE0000 <sub>H</sub>	FF0000 <sub>H</sub>
Address 2	000900 <sub>H</sub>	000900 <sub>H</sub>	000900 <sub>H</sub>	000900 <sub>H</sub>	000900 <sub>H</sub>	002100 <sub>H</sub>

Figure 22.2-2 Memory Space



# **CHAPTER 23**

---

## **512K / 1024K BIT FLASH MEMORY**

**The following explains the functions and operations of the 512K / 1024K bit flash memory.**

**Three methods of data writing/deleting to the flash memory are provided:**

- 23.1 Overview of the 512K / 1024K Bit Flash Memory
- 23.2 512K / 1024K Bit Flash Memory Sector Configuration
- 23.3 Flash Memory Control Status Register (FMCS)
- 23.4 Method of Starting the Automatic Algorithm in Flash Memory
- 23.5 Verifying Automatic Algorithm Execution Status
- 23.6 Detailed Explanation on the Flash Memory Write/Delete
- 23.7 Flash Security Feature
- 23.8 Programming Example of 512K Bit Flash Memory

## 23.1 Overview of the 512K / 1024K Bit Flash Memory

The 512K bit flash memory is allocated in the FF bank on the CPU memory map while 1024K bit flash memory is allocated in FE and FF bank. The function of the flash memory interface circuit enables the read/access or program access from the CPU to the flash memory, same as the mask ROM. The write/delete operation to the flash memory can be executed through the flash memory interface circuit by executing an instruction issued from the CPU. Therefore, the flash memory mounted can be rewritten under the control of the internal CPU, so that the program or data can be upgraded or updated more efficiently. However, no selector operation such as the enable sector protect can be used.

### ■ Characteristics of the 512K / 1024K Bit Flash Memory

- 512K Bit: 64K words x 8 bits/32K words x 16 bits  
(16K+8K+8K+32K) sector configuration
- 1024K Bit: 128K words x 8 bits/64K words x 16 bits  
(64K+16K+8K+8K+32K) sector configuration
- Automatic program algorithm (same as the Embedded Algorithm<sup>TM</sup> \*  
: MBM29F400TA)
- Installation of the deletion temporary stop/delete restart function
- Write/delete completion detected by the data polling or toggle bit
- Write/delete completion detected by the CPU interrupt
- Compatibility with the JEDEC standard-type command
- Each sector deletion can be executed (Sectors can be freely combined)
- Number of write/delete operations 10,000 times guaranteed

\*: Embedded Algorithm<sup>TM</sup> is the trademark of Advanced Micro Devices, Inc.

### ■ Procedure for Writing/Deleting the data to the Flash Memory

The write/delete operation of the flash memory cannot be executed simultaneously. In executing the data write/delete operation in the flash memory, only the write operation can be executed without a program access from the flash memory, by copying a program on the flash memory to RAM and executing the program.

### ■ Register on the Flash Memory

- Flash memory control status register (FMCS)

	bit	7	6	5	4	3	2	1	0
Address:0000AE <sub>H</sub>		INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0
Read/Write		(R/W)	(R/W)	(R/W)	(R)	(W)	(W)	(W)	(R/W)
Initial value		(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)

## 23.2 512K / 1024K Bit Flash Memory Sector Configuration

Figure 23.2-1 and Figure 23.2-2 and shows the sector configuration in the 512K bit flash memory. The address indicated in Figure 23.2-1 and Figure 23.2-2 is classified into the upper address and lower address of each sector.

### ■ Sector Configuration

When accessing the 512Kbit flash memory from the CPU, four sector addresses, SA0 to SA3, are allocated in the FF bank register.

**Figure 23.2-1 512K Bit Flash Memory Sector Configuration**

Flash memory	CPU address	*Writer address
SA3 (16 Kbytes)	FFFFFF <sub>H</sub>	7FFFF <sub>H</sub>
	FFC000 <sub>H</sub>	7C000 <sub>H</sub>
SA2 (8 Kbytes)	FFBFFF <sub>H</sub>	7BFFF <sub>H</sub>
	FFA000 <sub>H</sub>	7A000 <sub>H</sub>
SA1 (8 Kbytes)	FF9FFF <sub>H</sub>	79FFF <sub>H</sub>
	FF8000 <sub>H</sub>	78000 <sub>H</sub>
SA0 (32 Kbytes)	FF7FFF <sub>H</sub>	77FFF <sub>H</sub>
	FF0000 <sub>H</sub>	70000 <sub>H</sub>

When accessing the 1024Kbit flash memory from the CPU, five sector addresses, SA0 to SA4, are allocated in the FE and FF bank register.

**Figure 23.2-2 1024K Bit Flash Memory Sector Configuration**

Flash memory	CPU address	*Writer address
SA4 (16 Kbytes)	FFFFFF <sub>H</sub>	7FFFF <sub>H</sub>
	FFC000 <sub>H</sub>	7C000 <sub>H</sub>
SA3 (8 Kbytes)	FFBFFF <sub>H</sub>	7BFFF <sub>H</sub>
	FFA000 <sub>H</sub>	7A000 <sub>H</sub>
SA2 (8 Kbytes)	FF9FFF <sub>H</sub>	79FFF <sub>H</sub>
	FF8000 <sub>H</sub>	78000 <sub>H</sub>
SA1 (32 Kbytes)	FF7FFF <sub>H</sub>	77FFF <sub>H</sub>
	FF0000 <sub>H</sub>	70000 <sub>H</sub>
SA0 (64 Kbytes)	FEFFFF <sub>H</sub>	6FFFF <sub>H</sub>
	FE0000 <sub>H</sub>	60000 <sub>H</sub>

\*: Writer address

The writer address is equivalent to the CPU address when writing the data to the flash memory using the parallel writer. If the write/delete operation is executed using the general-purpose writer, the write/delete operation is executed using this address.



## 23.3 Flash Memory Control Status Register (FMCS)

The FMCS, which exists in the flash memory interface circuit, is used when data is written to or erased from flash memory.

### ■ Control Status Register (FMCS)

bit	7	6	5	4	3	2	1	0
Address:0000AE <sub>H</sub>	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0
Read/Write	(R/W)	(R/W)	(R/W)	(R)	(W)	(W)	(W)	(R/W)
Initial value	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)

#### ● Contents of the bits

##### [bit7] INTE (Interrupt Enable)

This bit generates an interrupt to the CPU when the write/delete operation to the flash memory is terminated.

When the INTE bit is "1" and the RDYINT bit is "1" an interrupt generated and sent to the CPU. If the INTE bit is "0", no interrupt is generated:

0: Interrupt disabled when the write/delete operation is terminated.

1: Interrupt enabled when the write/delete operation is terminated.

##### [bit6] RDYINT (Ready Interrupt)

This bit indicates the flash memory operating status.

After the write/delete to the flash memory is terminated, this bit is set to "1". While this bit is "0" after the end of write/delete operation to the flash memory, the flash memory cannot be written or deleted. After the write/delete operation is terminated and this bit is set to "1", the flash memory can be written or deleted. This bit is cleared to "0" by writing "0" and the writing of "1" is ignored. At the termination time of the automatic algorithm in the flash memory (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), this bit is set to "1". While using the read modify write (RMW) instruction, "1" can be read at any time.

0: During the write/delete operation

1: Write/delete operation terminated (An interrupt request is generated)

##### [bit5] WE (Write Enable)

This bit is the write enable bit for the flash memory area.

When this bit is "1", the write instruction after issuing command sequence to FF bank (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory") is equivalent to writing to the flash memory area. When this bit is "0", no write/delete signal is generated. This bit is used when the flash memory write/delete command is started.

0: Flash memory write/delete disabled

1: Flash memory write/delete enabled

**[bit4] RDY (Ready)**

This bit is the flash memory write/delete permission bit.

While this bit is "0", the write/delete cannot be executed to the flash memory. Even in this state, however, suspend commands such as the read/reset command and the sector deletion temporary stop can be accepted.

0: During the write/delete operation

1: Write/delete operation terminated (Next data write/delete operation permitted)

**[bit3, bit1] Reserved bits**

These bits are the reserved bits for testing. When these bits are usually used, they must be set to "0".

**[bit2, bit0] LPM1, LPM0 (Low-power Mode)**

These bits control the current consumption in the flash memory when accessing the flash memory. However, the access time from the CPU to the flash memory greatly dependent upon the setting. Therefore, the set values should be selected, depending on the CPU operating frequency.

01: Low-power consumption mode

(Internal operating frequency operating at 4 MHz or less)

10: Low-power consumption mode

(Internal operating frequency operating at 8 MHz or less)

11: Low-power consumption mode

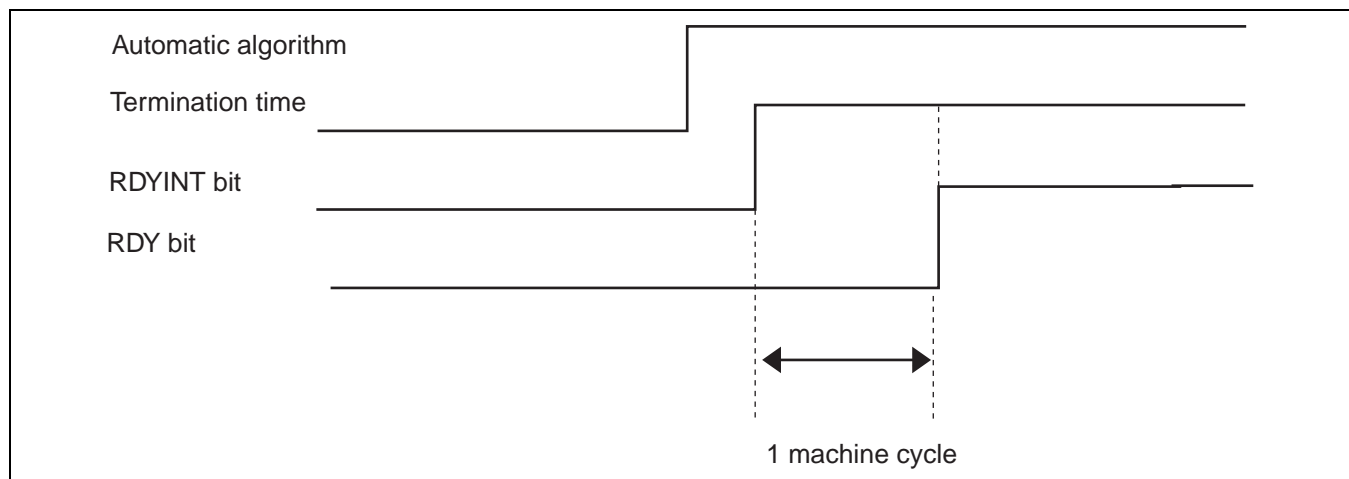
(Internal operating frequency operating at 10 MHz or less)

00: Normal power consumption mode

(Internal operating frequency operating at 12.58 MHz or less)

**Note:**

The RDYINT bit and RDY bit are not changed simultaneously. Create a program using one of the RDYINT bit or RDY bit for determination.



## 23.4 Method of Starting the Automatic Algorithm in Flash Memory

There are four types of commands for starting the automatic algorithm in the flash memory, i.e., the read/reset command, write command, and chip deletion command. In addition, the sector deletion command can be temporarily stopped and restarted.

### ■ Command Sequence Table

Table 23.4-1 lists the commands to be used for writing/deleting the data to the flash memory. All the data is written to the command register in units of bytes, though it should be accessed and written in units of words.

In this case, the data in the upper bytes is ignored.

**Table 23.4-1 Command Sequence Table**

Command sequence	Bus write cycle	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Ad-dress	Data	Address	Data	Address	Data
Read/reset	1	FFXXXX	XXF0	-	-	-	-	-	-	-	-	-	-
Read/reset	4	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XXF0	RA	RD	-	-	-	-
Write program	4	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip deletion	6	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XX80	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XX10
Sector deletion	6	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XX80	FFAAAA	XXAA	FF5554	XX55	SA (even)	XX30
Sector deletion temporary stop		Temporarily stops the sector deletion command when inputting Address "FFXXXX" Data(xxB0 <sub>H</sub> ).											
Sector deletion restart		Restarts the sector deletion command when inputting Address "FFXXXX" Data(xx30 <sub>H</sub> ).											

\* : Two types of read/reset commands can reset the flash memory to the read mode.

#### Note:

The addresses in the above table are the values on the CPU memory map. All the addresses and data are represented in hexadecimal. However, "X" is an optional value.

RA: Read address

PA: Write address. Only the even address can be specified.

SA: Sector address.

For details, see "23.2 512K / 1024K Bit Flash Memory Sector Configuration"

RD: Read data

PD: Write data. Only the word data can be specified

## 23.5 Verifying Automatic Algorithm Execution Status

The flash memory contains the hardware for posting the internal flash memory operating status or the flash memory operation completion, because the automatic algorithm executes the sequence of data writing/deleting procedures. This automatic algorithm can verify the internal flash memory operating status, depending on the following hardware sequence.

### ■ Hardware Sequence Flag

The hardware sequence flag consists of the four flag bits, DQ7, DQ6, DQ5, and DQ3. These flag bits have the data polling flag (DQ7) function, toggle bit flag (DQ6) function, time limit exceeded flag (DQ5) function, and sector deletion timer flag (DQ3) function, respectively. These functions can verify whether the write/chip sector deletion is terminated or whether the deletion code write is valid.

The hardware sequence flag can be referenced by accessing/reading the address of the target sector in the flash memory, after setting the command sequence (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"). Table 23.5-1 indicates the hardware sequence flag bit allocation.

**Table 23.5-1 Hardware Sequence Flag Bit Allocation**

Bit no.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

It can be determined whether automatic write/chip sector deletion is being performed, depending on the end of write processing, by checking the hardware sequence flag or the RDY bit in the flash memory control register (FMCS). After the write/delete operation is terminated, the flash memory is returned to the read/reset status. To actually create a program, it should be verified whether automatic write/delete operation is terminated, depending on any flag, and the next operation such as the data reading should be executed. Also, it can be verified whether the second sector deletion code write command or later commands are valid, depending on the hardware sequence flag. The following explains the hardware sequence flags. Table 23.5-2 lists the hardware sequence flag functions.

**Table 23.5-2 Hardware Sequence Flag Functions**

State		DQ7	DQ6	DQ5	DQ3
Status transition during normal operation	Write operation --> Write completion (when the write address is specified)	$\overline{DQ7}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3
	Chip/sector deletion operation --> Deletion completion	0 --> 1	Toggle --> Stop	0 --> 1	1
	Sector deletion wait --> Deletion start	0	Toggle	0	0 --> 1
	Deletion processing --> Sector deletion temporary stop (sector being deleted)	0 --> 1	Toggle --> 1	0	1 --> 0
	Sector deletion temporary stop --> Deletion restart (sector being deleted)	1 --> 0	1 --> Toggle	0	0 --> 1
	While the sector deletion is being temporarily stopped --> (sector not being deleted)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Write operation	$\overline{DQ7}$	Toggle	1	0
	Chip/sector deletion operation	0	Toggle	1	1

## 23.5.1 Data Polling Flag (DQ7)

---

The data polling flag (DQ7) indicates whether the automatic algorithm is being executed or has been terminated, using the data polling function. Table 23.5-3 shows the data polling flag status transition.

---

### ■ When the Write Operation is Executed.

When the read/access is executed during automatic write algorithm execution, the flash memory outputs the reverse data of bit7 in the last-written data, irrespective of the specified address. When the read/access is executed at the end of the automatic write algorithm, the flash memory outputs the data of bit7 in the specified read address. When the read/access is executed at the end of the automatic write algorithm, the flash memory outputs the data of bit7 in the specified read address.

### ■ When the Chip/Sector Deletion Operation is Executed.

When the read/access is executed during the chip/sector deletion algorithm execution, the flash memory outputs "0" from the sector being deleted by the sector deletion, or irrespective of the specified address during the chip deletion. Similarly, the flash memory outputs "1" at the end of chip/sector deletion algorithm.

### ■ When the Sector Deletion Temporary Stop is Executed.

When the access/read is executed while executing the sector deletion temporary stop, the flash memory outputs "1" if the specified address is the sector being deleted. However, the flash memory outputs the data of bit7 (DATA:7) of the specified read address, if the specified address is not the sector being deleted. By referencing this together with the toggle bit flag (DQ6), it can be determined whether the current sector is in the temporary stop state or which sector is being deleted.

---

#### Note:

When the automatic algorithm is started, the read/access to the specified address is ignored. As for the data reading, the end of data polling flag (DQ7) is posted, and then other data bit can be output. Therefore, the data read operation after the end of the automatic algorithm should be executed next to the read/access after verifying the end of data polling flag.

---

**Table 23.5-3 Data Polling Flag Station Transition**

- Status transition during normal operation

Operating status	Write operation --> Completion	Chip/sector deletion -->Completion	Sector deletion wait-->Start	Sector deletion -->Deletion temporary stop (Sector being deleted)	Sector deletion temporary stop -->Restart (Sector being deleted)	During the sector deletion temporary stop (Sector not being deleted)
DQ7	$\overline{\text{DQ7}}$ -->DATA:7	0-->1	0	0-->1	1-->0	DATA:7

- Status transition during abnormal operation

Operating status	Write operation	Chip/sector deletion operation
DQ7	$\overline{\text{DQ7}}$	0

## 23.5.2 Toggle Bit Flag (DQ6)

The toggle bit flag specifies whether the automatic algorithm is being executed or has been terminated, using the toggle bit function, the same as the data polling flag. Table 23.5-4 shows the toggle bit flag status transition.

### ■ When the Write Operation or Chip/Sector Deletion Operation is Executed.

When the continuous read/access is executed during the automatic write algorithm or chip/sector deletion algorithm execution, the flash memory outputs the toggle status, in which "1" and "0" are alternately output for each read operation, irrespective of the specified address. If the continuous read/access is executed at the end of the automatic write algorithm or chip/sector deletion algorithm, the flash memory stops the toggle operation in bit6 and outputs the data of bit6 (DATA:6) in the specified read address.

### ■ When the Sector Deletion Temporary Stop is Executed.

When the read/access is executed while executing the sector deletion temporary stop, the flash memory outputs "1" if the specified address belongs to the sector being deleted. The flash memory outputs the data of bit6 (DATA:6) in the specified read address unless the specified address belongs to the sector being deleted.

#### Reference:

When executing the write operation, the toggle bit executes the toggle operation for about 2 ms, then terminates it without rewriting the data, if the sector to be written is write-protected.

When executing the deletion operation, the toggle bit executes the toggle operation for about 100 ms, then returns to the read/reset status without rewriting the data, if all the selected sectors are protected from rewriting.

Toggle Bit Flag Status Transition

**Table 23.5-4 Toggle Bit Flag Status Transition**

- Status transition during normal operation

Operating status	Write operation --> Completion	Chip/sector deletion -->Completion	Sector deletion wait-->Start	Sector deletion -->Deletion temporary stop (Sector being deleted)	Sector deletion temporary stop -->Restart (Sector being deleted)	During the sector deletion temporary stop (Sector not being deleted)
DQ6	Toggle -->DATA:6	Toggle-->Stop	Toggle	Toggle-->1	1-->Toggle	DATA:6

- Status transition during abnormal operation

Operating status	Write operation	Chip/sector deletion operation
DQ6	Toggle	Toggle



### 23.5.3 Time limit Exceeded Flag (DQ5)

The time limit exceeded flag indicates that the automatic algorithm execution time has exceeded the time defined within the flash memory (i.e., internal pulse count). Table 23.5-5 shows the transition of the time limit exceeded flag status.

#### ■ When the Write Operation or Chip/Sector Deletion Operation is Executed.

When the read/access is executed after starting the write or chip/sector deletion automatic algorithm, this flag is set to "0" if the execution time is within the defined time (required for writing/deletion), or it outputs "1" if this execution time exceeds the defined time. This is not related to the state in which the automatic algorithm is being executed or has been terminated, so it can be determined whether the write/delete has succeeded or failed. Thus, when this flag is set to "1", it indicates that the write operation has failed if the automatic algorithm is being performed by the data polling function or toggle bit function.

For example, a fail occurs if an attempt is made to write "1" to the flash memory with "0" written. In this case, the flash memory is locked and the automatic algorithm is not terminated. Therefore, no valid data is set in data polling flag (DQ7). The toggle bit flag (DQ6) does not stop the toggle operation, so the execution time exceeds the time limit. Then, the time limit exceeded flag (DQ5) outputs "1". This event indicates that the flash memory has not been correctly used, but does not indicate that the flash memory is not good. If this event occurs, the reset command should be executed.

**Table 23.5-5 Transition of the Time Limit Exceeded Flag Status**

- Status transition during normal operation

Operating status	Write operation --> Completion	Chip/sector deletion -->Completion	Sector deletion wait-->Start	Sector deletion -->Deletion temporary stop (Sector being deleted)	Sector deletion temporary stop -->Restart (Sector being deleted)	During the sector deletion temporary stop (Sector not being deleted)
DQ5	0-->DATA:5	0-->1	0	0	0	DATA:5

- Status transition during abnormal operation

Operating status	Write operation	Chip/sector deletion operation
DQ5	1	1

## 23.5.4 Sector Deletion Timer Flag (DQ3)

After starting the sector deletion command, the sector deletion timer flag indicates whether it is "during the sector deletion waiting period". Table 23.5-6 shows the sector deletion timer flag status transition.

### ■ When the Sector Deletion Operation is Executed.

When the read/access is executed after starting the sector deletion command, the flash memory outputs "0" if this flag indicates "during the sector deletion waiting period" irrespective of the address specified by the address signal from the sector having issued the command. However, the flash memory outputs "1" if this flag exceeds the defined sector deletion waiting period.

When the deletion algorithm is being executed by the data polling function or toggle bit function, the internally-controlled deletion operation is started if this flag is "1". The succeeding sector deletion code to be written and other commands except the sector deletion temporary stop command are ignored until deletion is terminated.

If this flag is "0", the flash memory accepts the additional sector deletion code to be written. To verify this event, it is recommended that this flag status be checked before writing the succeeding sector deletion code. If this flag is "1" when the second time the status is checked, the additional sector deletion code may not be accepted.

### ■ When the Sector Deletion Operation is Executed.

When the read/access is executed while executing the sector deletion temporary stop, the flash memory outputs "1" if the specified address belongs to the sector being deleted. However, unless the specified address belongs to the sector being deleted, the flash memory outputs the data of bit3 (DATA:3) of the specified read address.

Sector Deletion Timer Flag Status Transition

**Table 23.5-6 Sector Deletion Timer Flag Status Transition**

- Status transition during normal operation

Operating status	Write operation --> Completion	Chip/sector deletion -->Completion	Sector deletion wait-->Start	Sector deletion -->Deletion temporary stop (Sector being deleted)	Sector deletion temporary stop -->Restart (Sector being deleted)	During the sector deletion temporary stop (Sector not being deleted)
DQ3	0-->DATA:3	1	0-->1	1-->0	0-->1	DATA:3

- Status transition during abnormal operation

Operating status	Write operation	Chip/sector deletion operation
DQ3	0	1

## 23.6 Detailed Explanation on the Flash Memory Write/Delete

---

**This section explains the procedures for issuing the command to start the automatic algorithm, reading/resetting the flash memory, writing the data to the flash memory, deleting the chip, deleting the sector, temporarily stopping the sector deletion, and restarting the sector deletion.**

---

### ■ Detailed Explanation on the Flash Memory Write/Delete

The read/reset, write, chip deletion, sector deletion, sector deletion temporary stop, or deletion start operation can be performed by the automatic algorithm which can be started by setting the command sequence (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory") to each bus write cycle. The write cycles to the respective buses must be executed continuously. The ending time of the automatic algorithm can be notified by the data polling function and so on. After the normal end of the automatic algorithm, the flash memory returns to the read/reset status.

23.6.1 Setting the Read/Reset Status

23.6.2 Writing the Data

23.6.3 Deleting the Data (Chip Deletion)

23.6.4 Deleting the Data (Sector Deletion)

23.6.5 Temporarily Stopping the Sector Deletion

23.6.6 Restarting the Sector Deletion

## 23.6.1 Setting the Read/Reset Status

---

**This section explains the procedure of issuing the read/reset command and setting the flash memory to the read/reset status.**

---

### ■ Setting the Read/Reset Status

When the flash memory is set to the read/reset status, the read/reset command can be executed by continuously sending the read/reset command, listed in the command sequence table (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), to the target sector in the flash memory.

There are two types of read/reset command sequences, one is that the bus operation is executed once and the other is that the bus operations are executed three times. However, these command sequences have no essential difference.

The read/reset status is the initial status of the flash memory. When the power supply is turned on, or when the command is normally terminated, the flash memory is always set to the read/reset status. The read/reset status means the status of the flash memory that is waiting for another command to be input.

In the read/reset status, data can be read from the flash memory by executing a usual read/access command. The data can be program-accessed from CPU same as the mask ROM. This command is not required for usual data reading. This command should be mainly used for initializing the automatic algorithm if the command has not been normally terminated for any reason.

## 23.6.2 Writing the Data

---

**This section explains the procedure of issuing the write command and writing the data to the flash memory. Figure 23.6-1 shows an example of procedure of writing data to the flash memory.**

---

### ■ Writing the Data

The automatic algorithm for writing the data to the flash memory can be performed by continuously sending the write command, listed in the command sequence table (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), to the target sector in the flash memory. When the data write operation to the target address in the 4th cycle has been terminated, the automatic algorithm is started for automatic writing.

### ■ How to Specify the Address

Only even addresses can be specified as the write address in the write data cycle. If an odd address is specified, data cannot be correctly written. That is, it is necessary to write the data in units of words to even addresses.

Data can be written to the flash memory, and any address sequence may be specified, even if data has been written across the sector boundary. However, only one-word data can be written by executing the write command once.

### ■ Notes on Writing the Data

By writing the data, data "0" cannot be returned to data "1". If data "1" is written to data "0", the data polling algorithm (DQ7) or toggle operation (DQ6) is not terminated and the flash memory element is determined to be bad. Then, the time limit exceeded flag (DQ5) error may be determined by the excess of the write defined time, or data "1" may be apparently written but is not actually done. However, if data is read from the flash memory in the read/reset status, data remains "0". Only the deletion operation enables data "0" to be changed to data "1".

All commands are ignored while automatic writing is being performed. Note that the data at the address for writing is not assured if the hardware is reset during automatic writing.

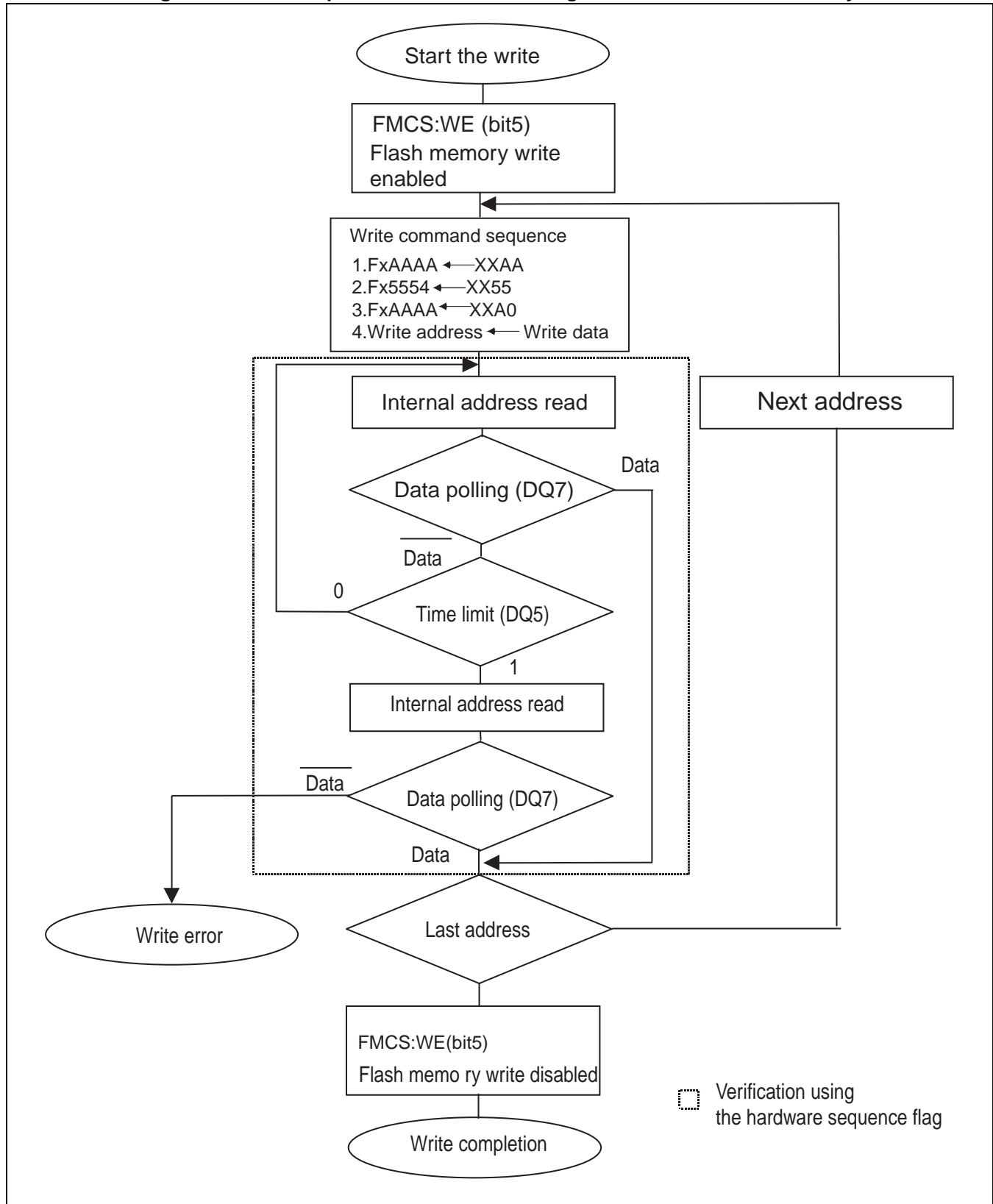
### ■ Procedure of Writing the Data to the Flash Memory

Figure 23.6-1 shows an example of procedure of writing the data to the flash memory. Using the hardware sequence flag (see "23.5 Verifying Automatic Algorithm Execution Status"), the status of the automatic algorithm within the flash memory can be determined. Here, the data polling flag (DQ7) is used for verifying the end of writing.

The data reading for checking the flag is started from the last-written address. The data polling flag (DQ7) is changed at the same time when the time limit exceeded flag (DQ5) is changed, so the data polling flag (DQ7) must be rechecked even if the time limit exceeded flag (DQ5) is "1".

Similarly, the toggle bit flag (DQ6) stops the toggle operation at the same time when the time limit exceeded flag (DQ5) is changed to "1", so the toggle bit flag (DQ6) must be rechecked.

Figure 23.6-1 Example of Procedure of Writing the Data to the Flash Memory



### 23.6.3 Deleting the Data (Chip Deletion)

---

**This section explains the procedure of issuing the chip deletion command and deleting all the data in the flash memory.**

---

#### ■ Deleting the data (Chip deletion)

All the data can be deleted from the flash memory by continuously sending the chip deletion command, listed in the command sequence table (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), to the target sector in the flash memory.

The chip deletion command is executed by executing the bus operation six times. When the 6th-cycle write operation has been completed, the chip deletion operation is started. The user need not write the data to the flash memory before chip deletion operation. During the automatic deletion algorithm execution, the flash memory writes data "0" to all the cells and verifies them before they are automatically deleted.

## 23.6.4 Deleting the Data (Sector Deletion)

---

**This section explains the procedure of issuing the sector deletion command and deleting any sector from the flash memory. This command enables each sector to be deleted, and two or more sectors to be specified at the same time.**

---

### ■ Deleting the data (Sector deletion)

Any sector can be deleted from the flash memory by continuously sending the sector deletion command, listed in the command sequence table (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), to the target sector in the flash memory.

#### Method of Specifying a Sector

The sector deletion command is executed by executing the bus operation six times. The sector deletion wait of 50  $\mu$ s is started by writing the sector deletion code (30<sub>H</sub>) to any accessible even address in the target sector in the 6th-cycle bus operation. To delete two or more sectors, the deletion code (30<sub>H</sub>) should be written to the address in the target sector just after the above operation.

### ■ Notes on Specifying Two or More Sectors

The deletion operation is started after the last sector deletion code is written and the sector deletion wait period of 50  $\mu$ s is terminated. Thus, the next deletion sector address and deletion code (i.e. in the 6th cycle of the command sequence) must be input each within 50  $\mu$ s to delete two or more sectors simultaneously, which may not be accepted later than 50  $\mu$ s. It can be checked whether the succeeding sector deletion code write operation is valid, using the sector deletion timer flag (DQ3). In this case, the address from which the sector deletion timer flag (DQ3) is read must indicate the sector to be deleted.

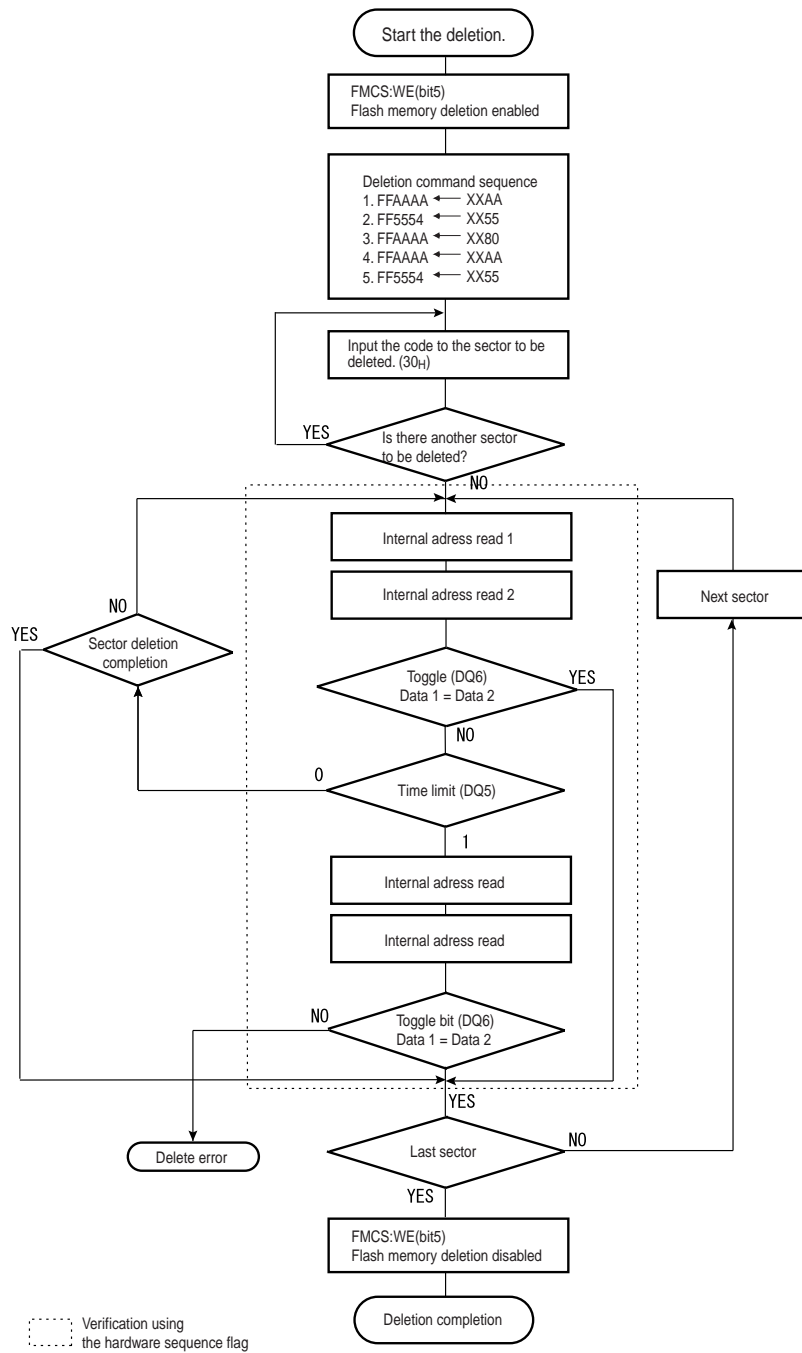
### ■ Procedure of Deleting a Sector

Using the hardware sequence flag (see "23.5 Verifying Automatic Algorithm Execution Status"), the status of the automatic algorithm within the flash memory can be determined. Figure 23.6-2 shows an example of procedure of deleting the sector from the flash memory. Here, the toggle bit flag (DQ6) is used for verifying the end of writing. Note that data to be used for checking the flag is read from the sector to be deleted.

The toggle bit flag (DQ6) stops the toggle operation at the same time when the time limit exceeded flag (DQ5) is changed to "1", so the toggle bit flag (DQ6) must be rechecked even if the time limit exceeded flag (DQ5) is "1".

Similarly, the data polling flag (DQ7) is changed at the same time when the time limit exceeded flag (DQ5) is changed, so the data polling flag (DQ7) must be rechecked.



**Figure 23.6-2 Example of Procedure of Deleting the Sector from the Flash Memory**

## 23.6.5 Temporarily Stopping the Sector Deletion

---

**This section explains the procedure of issuing the sector deletion temporary stop command and temporarily stopping the deletion of a sector from the flash memory. This command can read the data from the sector not being deleted.**

---

### ■ Temporarily Stopping the Sector Deletion

The sector deletion from the flash memory can be temporarily stopped by sending the sector deletion temporary stop command, listed in the command sequence table (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), to the target sector in the flash memory.

The sector deletion temporary stop command stops the sector deletion operation temporarily, and enables the data reading from the sector not being deleted. In this case, only the read operation can be executed, but the write operation cannot be done. This command is valid only during the sector deletion operation including the deletion waiting time, but it is ignored during the chip deletion operation or the write operation.

This command is executed by writing the deletion temporary stop code (B0<sub>H</sub>). In this case, the address must indicate an address within the flash memory. During the deletion temporary stop operation, the reissued deletion temporary stop command is ignored.

If the sector deletion temporary stop command is input during the sector deletion waiting period, the sector deletion wait is immediately terminated to stop the deletion operation, and the flash memory enters the deletion stop status. If the sector deletion temporary stop command is input during the sector deletion operation after the sector deletion waiting period, the flash memory enters the deletion temporary stop status after a lapse of up to 20  $\mu$ s. The sector deletion temporary stop command must be executed 20  $\mu$ s or more after the sector deletion command or sector deletion restart command is issued.

## 23.6.6 Restarting the Sector Deletion

---

**This section explains the procedure of issuing the sector deletion restart command and restarting the operation of deleting a sector from the flash memory, which has been temporarily stopped.**

---

### ■ Restarting the Sector Deletion

The sector deletion operation which has been temporarily stopped can be restarted by sending the sector deletion restart command, listed in the command sequence table (see "23.4 Method of Starting the Automatic Algorithm in Flash Memory"), to the target sector in the flash memory.

The sector deletion restart command is used to restart the sector deletion operation when the flash memory is in the sector deletion temporary stop status caused by the sector deletion temporary stop command. This command is executed by writing the deletion restart code (30<sub>H</sub>). In this case, the address must indicate an address within the flash memory area.

However, the sector deletion restart command issued during the sector deletion operation is ignored.

## 23.7 Flash Security Feature

---

**The Flash Security Controller provides possibilities to protect the content of the flash memory from being read from external pins.**

---

### ■ Flash Security Feature

One predefined address of the flash memory is assigned to the Flash Security Controller (MB90F462, MB90F462A: FF0001<sub>H</sub>; MB90F463A: FE0001<sub>H</sub>). If the protection code of "01<sub>H</sub>" is written in this address, access to the flash memory is restricted. Once the flash memory is protected, performing the chip erase operation only can unlock the function otherwise read/write access to the flash memory from any external pins is not generally possible.

This function is suitable for applications requiring security of self-containing program and data stored in the flash memory. If the target application requires any part of the program to locate outside the microcontroller, the Flash Security Controller can not offer the intended features. For this reason, the External Vector Fetch mode should not be used when the protection code is set.

Programming of the flash microcontroller by standard parallel programmer may require unique set-up. For example, with the programmer from Minato Electronics the device checking should be turned off. Writing the protection code is generally recommended to take place at the end of the flash programming. This is to avoid unnecessary protection during the programming.

In order to re-program the once protected flash memory, the chip erase operation should be preformed.

For further information, please contact Fujitsu.

## 23.8 Programming Example of 512K Bit Flash Memory

This section presents a programming example for the use of 512K bit flash memory.

### ■ Programming Example Using 512K Bit Flash Memory

```

NAME FLASHWE
TITLE FLASHWE
;-----
;512-KB FLASH  Sample program for 512-KB FLASH
;
;1: Transfer the program from the flash memory (address: FFBC00H sector: SA2) to RAM (address: 000700H).
;2: Run the transferred program in RAM.
;3: Write the value of PDR1 to the flash memory (address: FF0000H sector: SA0).
;4: Read the written value (address: FF0000H sector: SA0), and output the value to PDR2.
;5: Erase the sector (SA0) to which the value was written.
;6: Output a confirmation that the data was erased.
; Requirements:
;   - Number of bytes transferred to RAM: 100 H (256 B)
;   - Determination that writing or erasing has ended
;       Determined via DQ5 (timing limit exceeded flag)
;       Determined via DQ6 (toggle bit flag)
;       Determined via RDY (FMCS)
;   - Error handling
;       Output Hi to P00 to P07.
;       Issue a reset command.
;-----
;
RESOUS      IOSEG      ABS=00                      ;"RESOUS" I/O segment definition
            ORG        0000H
PDR0        RB         1
PDR1        RB         1
PDR2        RB         1
PDR3        RB         1
            ORG        0010H
DDR0        RB         1
DDR1        RB         1
DDR2        RB         1
DDR3        RB         1
            ORG        00A1H
CKSCR       RB         1
            ORG        00AEH
FMCS        RB         1
            ORG        006FH
ROMM        RB         1
RESOUS      ENDS
;
SSTA        SSEG
            RW         0127H
STA_T       RW         1
SSTA        ENDS
;
DATA        DSEG      ABS=0FFH                      ;Flash command address
            ORG        5554H
COMADR2     RW         1
            ORG        0AAAAH
COMADR1     RW         1

```

```

DATA ENDS
;////////////////////////////////////
;Main program (SA1)
;////////////////////////////////////
CODE CSEG
START:
;////////////////////////////////////
;Initialization
;////////////////////////////////////
MOV     CKSCR,#0BAH           ;Set a frequency multiplier of three.
MOV     RP,#0
MOV     A,#!STA_T
MOV     SSB,A
MOVBW   A,#STA_T
MOVBW   SP,A
MOV     ROMM,#00H           ;Mirror OFF
MOV     PDR0,#00H          ;For error check
MOV     DDR0,#0FFH
MOV     PDR1,#00H           ;Data input port
MOV     DDR1,#00H
MOV     PDR2,#00H           ;Data output port
MOV     DDR2,#0FFH
;////////////////////////////////////
;The flash memory write/erase program (FFBC00H) is transferred to RAM (1500H address).
;////////////////////////////////////
MOVBW   A,#0700H           ;Destination RAM area
MOVBW   A,#0BC00H          ;Source address (location of program)
MOVBW   RW0,#100H          ;Number of bytes of data transferred to RAM
MOVS    ADB,PCB            ;Transfer 100H from FFBC00H to 000700H.
CALLP   000700H            ;Jump to the address where the transferred program
                           ;exists.
;////////////////////////////////////
;Data output
;////////////////////////////////////
OUT      MOV     A,#0FEH
        MOV     ADB,A
        MOVBW   RW2,#0000H
        MOVBW   A,@RW2+00
        MOV     PDR2,A
END      JMP     *
CODE     ENDS
;////////////////////////////////////
; Flash program write/erase program (SA2)
;////////////////////////////////////
RAMPRG   CSEG      ABS=0FFH
        ORG      0BC00H
;
;      Initialization
;
;      MOVW     RW0,#0500H           ;RW0: Reserve RAM space for input area
;                                     00:0500 to
;
;      MOVW     RW2,#0000H          ;RW2: Flash memory write address
;                                     FF:0000 to
;
;      MOV     A,#00H              ;DTB modification
;      MOV     DTB,A              ;Bank specification for @RW0
;      MOV     A,#0FFH            ;ADB modification 1
;      MOV     ADB,A              ;Bank specification for write mode specification
;                                     ;address

```

```

        MOV     PDR3,#00H           ;Switch initialization
        MOV     DDR3,#00H
;
; WAIT1   BBC     PDR3:0,WAIT1      ;Writing starts if PDR3:0 Hi.
;
;//////////
; Write (SA0)
;//////////
        MOV     A,PDR1
        MOVW    @RW0+00,A           ;PDR1 data is stored in RAM.
        MOV     FMCS,#20H           ;Write mode setting
        MOVW    ADB:COMADR1,#00AAH  ;Flash write command 1
        MOVW    ADB:COMADR2,#0055H  ;Flash write command 2
        MOVW    ADB:COMADR1,#00A0H  ;Flash write command 3
;
        MOVW    A,@RW0+00           ; Input data (RW0) is written to flash memory (RW2).
        MOVW    @RW2+00,A
WRITE                                     ; Wait time check
;
;//////////
; If the "time-limit-exceeded" check flag is set while toggling is on, branches to ERROR.
;//////////
        MOVW    A,@RW2+00
        AND     A,#20H              ;DQ5 time limit check
        BZ      NTOW                ;Time limit exceeded
        MOVW    A,@RW2+00           ;AH
        MOVW    A,@RW2+00           ;AL
        XORW    A                   ;XOR of AH and AL (1 when the values
                                   ;are different.)
        AND     A,#40H              ;Checks whether the DQ6 toggle bit is
                                   ;different.
        BNZ     ERROR              ;If it is different, branches to ERROR
;
;//////////
; Write-end check (FMCS-RDY)
;//////////
NTOW   MOVW    A,FMCS
        AND     A,#10H              ;FMCS RDY bit (4 bit) is extracted.
        BZ      WRITE              ;Verifies that writing has ended.
        MOV     FMCS,#00H           ;Write mode is released.
;//////////
; Write data output
;//////////
        MOVW    RW2,#0000H          ;Write data output
        MOVW    A,@RW2+00
        MOV     PDR2,A
;
; WAIT2   BBC     PDR3:1,WAIT2      ;If PDR3:1 Hi, erasing of the sector starts.
;
;//////////
; Erasing sectors (SA0)
;//////////
        MOVW    @RW2+00,#0000H      ;Address initialization
        MOV     FMCS,#20H           ;Erase mode setting
        MOVW    ADB:COMADR1,#00AAH  ;Flash memory erase command 1
        MOVW    ADB:COMADR2,#0055H  ;Flash memory erase command 2
        MOVW    ADB:COMADR1,#0080H  ;Flash memory erase command 3
        MOVW    ADB:COMADR1,#00AAH  ;Flash memory erase command 4
        MOVW    ADB:COMADR2,#0055H  ;Flash memory erase command 5

```

```

MOVW    @RW2+00,#0030H      ;Issuing the erase command to the sector to be erased. 6
ELS     ; Wait-time check
;//////////////////////////////////////
; When the "time-limit-exceeded" check flag is set and toggling is on, branches to ERROR.
;//////////////////////////////////////
MOVW    A,@RW2+00
AND     A,#20H               ;DQ5 time limit check
BZ      NTOE                 ;Time limit exceeded
MOVW    A,@RW2+00           ;AH From DQ6 during writing
MOVW    A,@RW2+00           ;AL Hi and Low are output alternately by each read.
XORW    A                   ;XOR of AH and AL (If the DQ6 value is toggled,
                          XOR is 1, indicating that writing is in progress.)
AND     A,#40H              ;Checks whether the DQ6 toggle bit is Hi.
BNZ     ERROR               ;If the bit is Hi, branches to ERROR
;//////////////////////////////////////
; Erase-end check (FMCS-RDY)
;//////////////////////////////////////
NTOE    MOVW    A,FMCS       ;
AND     A,#10H              ;FMCS RDY bit (4 bit) is extracted.
BZ ELS   ;Verifies that erasing of the sector has ended.
MOV     FMCS,#00H          ;Flash memory erase mode is released.
RETP    ;Returns to the main program.
;//////////////////////////////////////
; Error
;//////////////////////////////////////
ERROR
        MOV     ADB:COMADR1,#0F0H ;Reset command (enabling data reading)
        MOV     FMCS,#00H         ;Flash command mode is released.
        MOV     PDR0,#0FFH       ;Confirmation of error processing.
        RETP    ;Returns to the main program.
RAMPRG ENDS
;//////////////////////////////////////
VECT    CSEG      ABS=0FFH
        ORG      0FFDCH
        DSL      START
        DB       00H
VECT    ENDS
;
END START
```





# **CHAPTER 24**

---

## ***EXAMPLE OF F<sup>2</sup>MC-16LX MB90F462/F462A/F463A CONNECTION FOR SERIAL WRITING***

**This chapter describes examples of F<sup>2</sup>MC-16LX MB90F462/F462A/F463A connections for serial writing.**

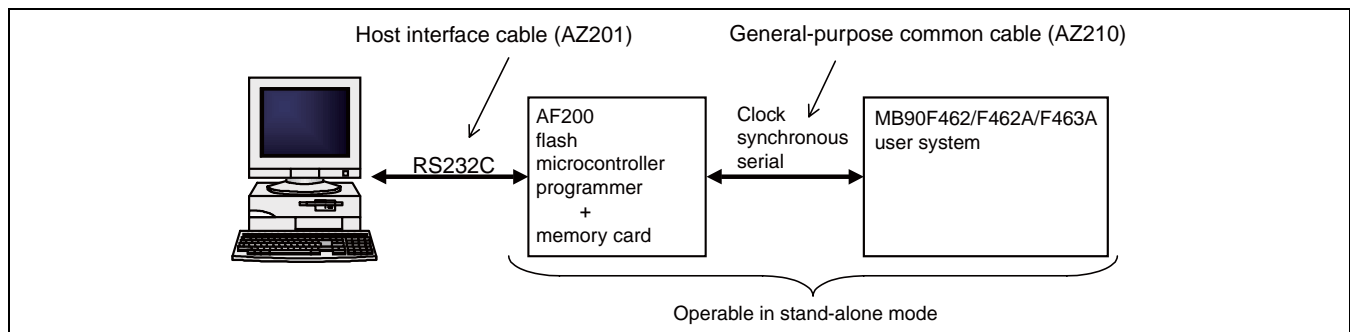
- 24.1 Standard Configuration for Serial On-board Writing (Fujitsu Standard)
- 24.2 Example of Connection for Serial Writing (When Power Supplied by User)
- 24.3 Example of Connection for Serial Writing (When Power Supplied from Writer)
- 24.4 Example of Minimum Connection with Flash Microcontroller Programmer (When Power Supplied by User)
- 24.5 Example of Minimum Connection with Flash Microcontroller Programmer (When Power Supplied from Writer)

## 24.1 Standard Configuration for Serial On-board Writing (Fujitsu Standard)

**MB90F462/F462A/F463A supports serial on-board writing (Fujitsu standard) to flash ROM. This describes the specifications for serial on-board writing.**

### ■ Standard Configuration for Fujitsu Standard Serial On-board Writing

The AF200 flash microcontroller programmer of Yokogawa Digital Computer Co., Ltd. is used for Fujitsu standard serial on-board writing.



Note:

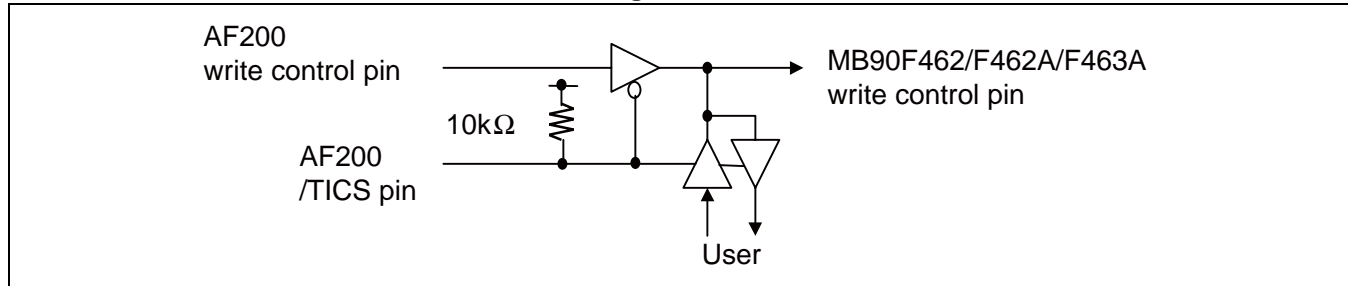
Contact Yokogawa Digital Computer Co., Ltd. for the functionality and operation of the AF200 flash microcontroller programmer and information on the general-purpose common cable (AZ210) and connectors.

**Table 24.1-1 Pins used for Fujitsu Standard Serial On-board Writing**

Pin	Function	Description
MD2, MD1, MD0	Mode pin	Used to enable write mode for the flash microcomputer programmer.
X0, X1	Oscillator pin	In write mode, since the operation clock is one times the CPU clock, the oscillation clock frequency is the internal operation clock. The resonator used for serial rewriting is therefore 1 MHz to 16 MHz.
P00, P01	Write program start pin	—
RSTX	Reset pin	—
SIN0	Serial data input pin	UART0 is used in CLK synchronous mode.
SOT0	Serial data output pin	
SCK0	Serial clock input pin	
C	C pin	Capacitance pin for power stabilization. Connect a ceramic capacitor of about 0.1 $\mu$ F to the outside.
VCC	Power supply pin	Write voltage (5 V $\pm$ 10%)
VSS	Ground pin	Used also as the ground pin for the flash microcontroller programmer.

**Note:**

When the P00, P01, SIN0, SOT0, SCK0 pins are also used by the user system, the control circuit shown below is required. (The /TICS signal of the flash microcontroller programmer can separate the user circuit during serial writing. See the connection example shown later.)

**Figure 24.1-1 Control Circuit**

Refer to the following four serial writing examples in Section 24.2 to 24.5.

- Example of serial write connection when power supplied by user
- Example of serial write connection when power supplied from writer
- Example of minimum connection to flash microcontroller programmer when power supplied by user
- Example of minimum connection to flash microcontroller programmer when power supplied from writer

**Table 24.1-2 System Configuration of AF200 Flash Microcontroller Programmer (Yokogawa Digital Computer Co., Ltd.)**

Type	Function
AF200 ACP	Flash microcontroller programmer/100 V power supply adapter
AF200 AC2P	Flash microcontroller programmer/power supply adapter with overseas specification
AZ201	RS232C cable for PC/AT
AZ210	Standard target probe (a) Length: 1 m
FF001	Control module for Fujitsu F <sup>2</sup> MC-16LX flash microcontroller
FF001 P2	2 MB PC Card (Option)
FF001 P4	4 MB PC Card (Option)

For more information, contact the Sales Department, Equipment Business Division, Yokogawa Digital Computer Co., Ltd. (Telephone: (81)-42-333-6224).

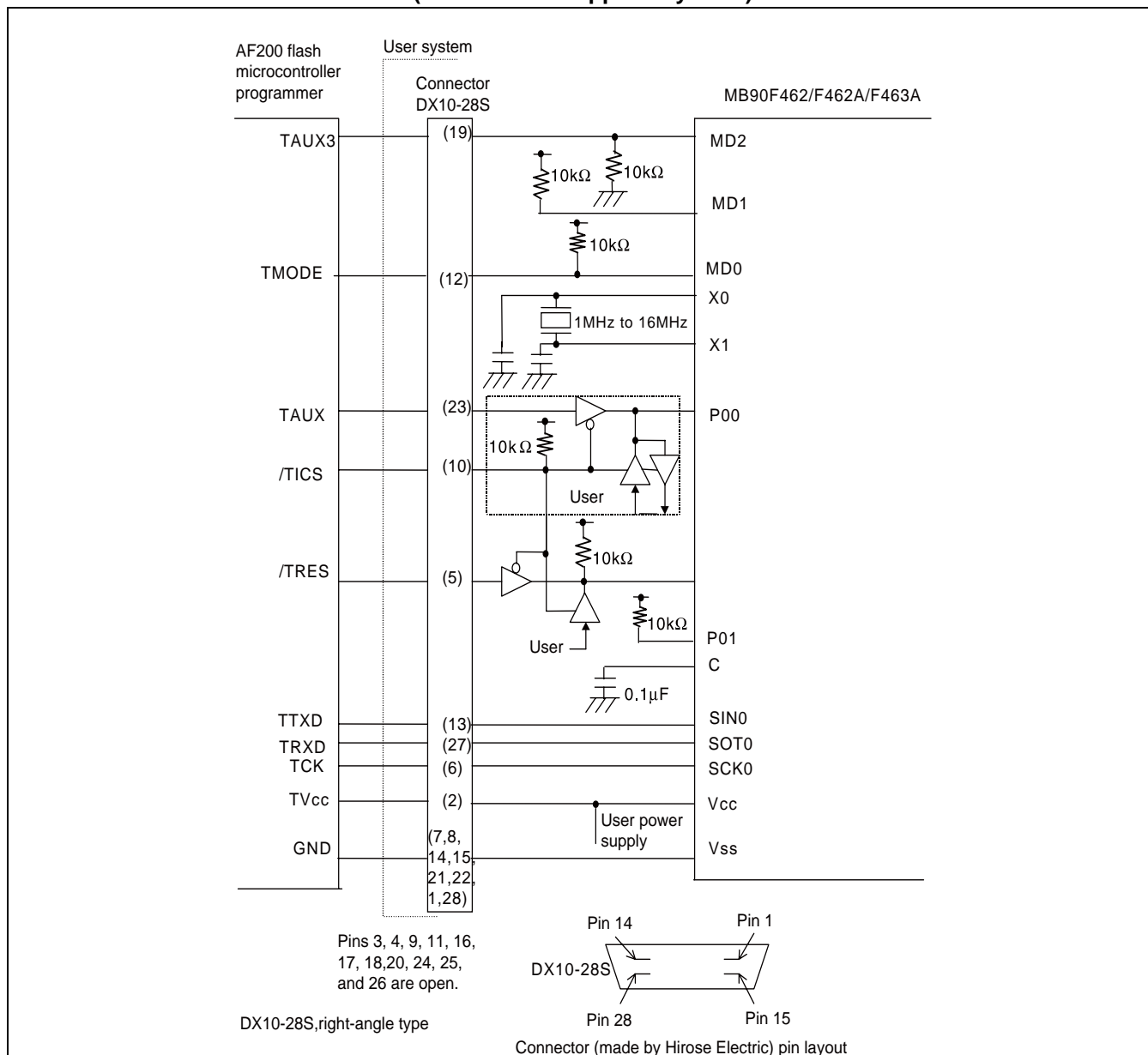
## 24.2 Example of Connection for Serial Writing (When Power Supplied by User)

Figure 24.2-1 is an example of serial write connection when power is supplied by the user.

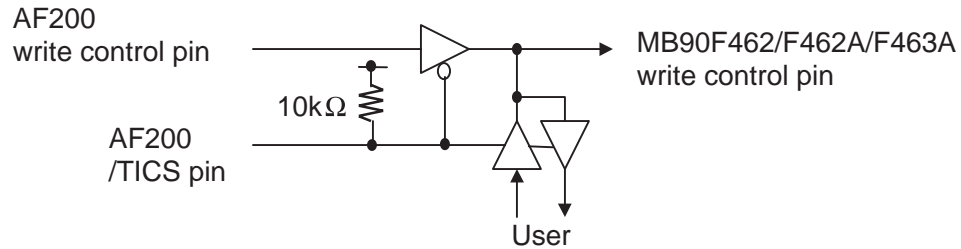
MD2=1 and MD0=0 are input from TAUX3 and TMODE respectively in AF200 flash microcontroller programmer. Serial write mode: MD2, MD1, MD0 = 110<sub>B</sub>

### ■ Example of Connection for Serial Writing (when Power Supplied by User)

Figure 24.2-1 Example of Connection for Serial Writing in MB90F462/F462A/F463A Internal Vector Mode (when Power Supplied by User)



- When the user system also uses pins SIN0, SOT0 and SCK0, the control circuit shown below is necessary, just as it is for P00. (During serial writing, the user circuit can be disconnected by the flash microcontroller programmer /TICS signal.)
- Before connecting the AF200, turn off the power supplied by the user.



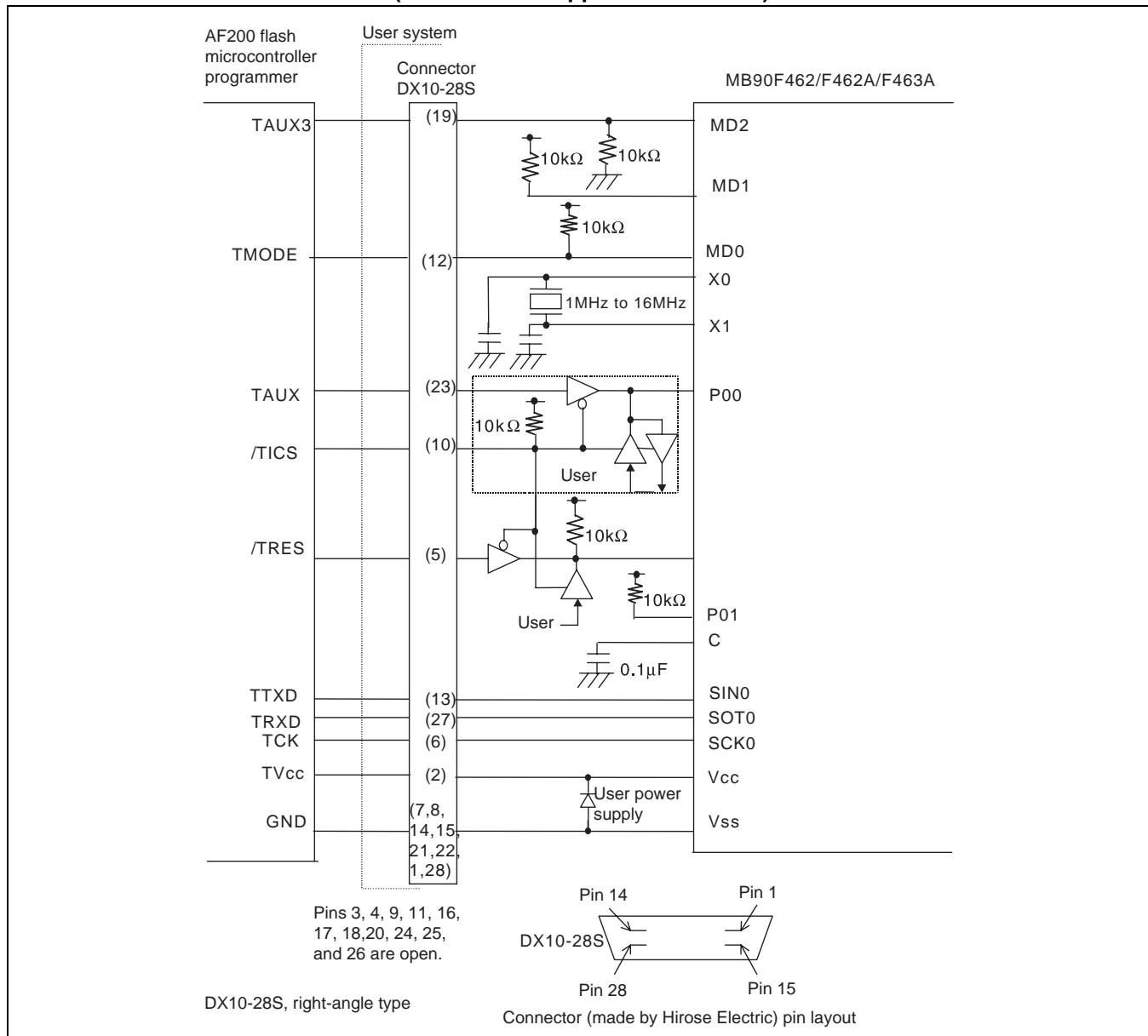
## 24.3 Example of Connection for Serial Writing (When Power Supplied from Writer)

Figure 24.3-1 is an example of serial write connection when power is supplied from the writer.

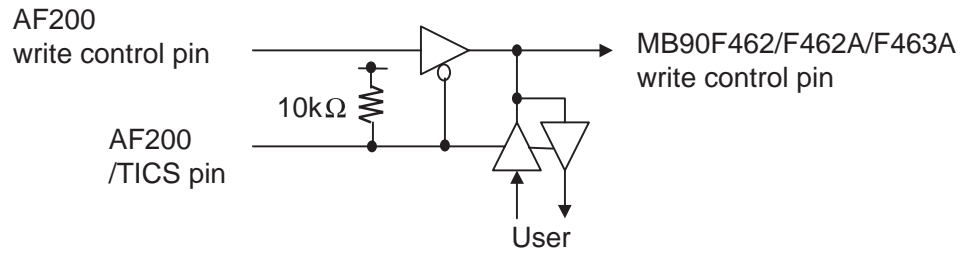
MD2=1 and MD0 are input from TAUX3 and TMODE respectively in AF200 flash microcontroller programmer. Serial write mode: MD2, MD1, MD0 = 110<sub>B</sub>

### ■ Example of Connection for Serial Writing (when Power Supplied from Writer)

Figure 24.3-1 Example of Connection for Serial Writing in MB90F462/F462A/F463A Internal Vector Mode (when Power Supplied from Writer)



- When the SIN0, SOT0 and SCK0 pins are also used by the user system, the control circuit shown below is necessary, just as it is for P00. (During serial writing, the user circuit can be disconnected by the flash microcontroller /TICS signal.)
- Before connecting the AF200, turn off the power supplied by the user.
- When supplying write power from the AF200, do not create a short with the power supplied by the user.





## 24.4 Example of Minimum Connection with Flash Microcontroller Programmer (When Power Supplied by User)

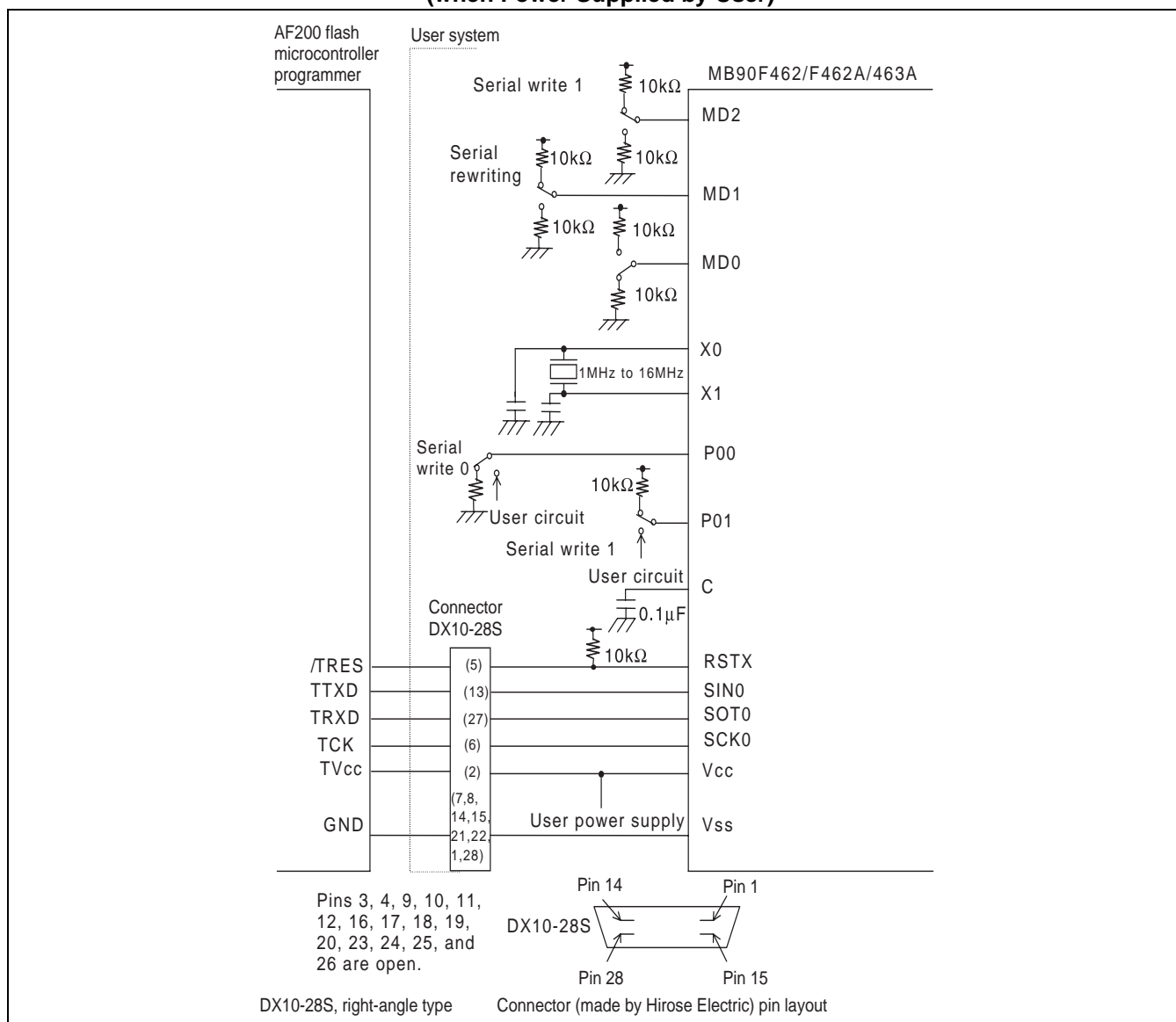
Figure 24.4-1 is an example of the minimum connection with the flash microcontroller programmer when power is supplied by the user.

Serial write mode: MD2, MD1, MD0 = 110<sub>B</sub>

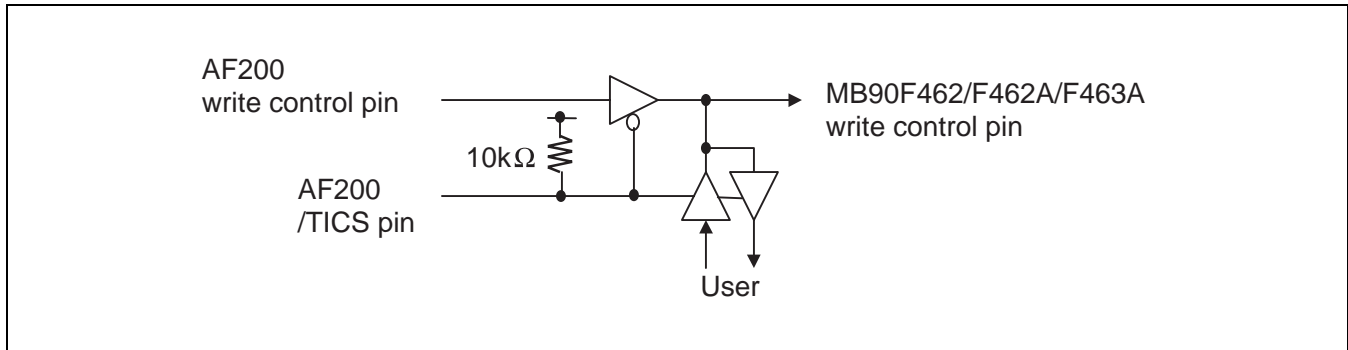
### ■ Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied by User)

If the pins are set as shown in Figure 24.4-1 during writing to flash memory, MD2, MD1, MD0, P00 and flash microcontroller programmer connection is unnecessary.

**Figure 24.4-1 Example of Minimum Connection with Flash Microcomputer Programmer (when Power Supplied by User)**



- When the user system also uses the SIN0, SOT0 and SCK0 pins the control circuit shown below is necessary. (During serial writing, the user circuit can be disconnected by the flash microcontroller programmer /TICS signal.)
- Before connecting the AF200, turn off the power supplied by the user.



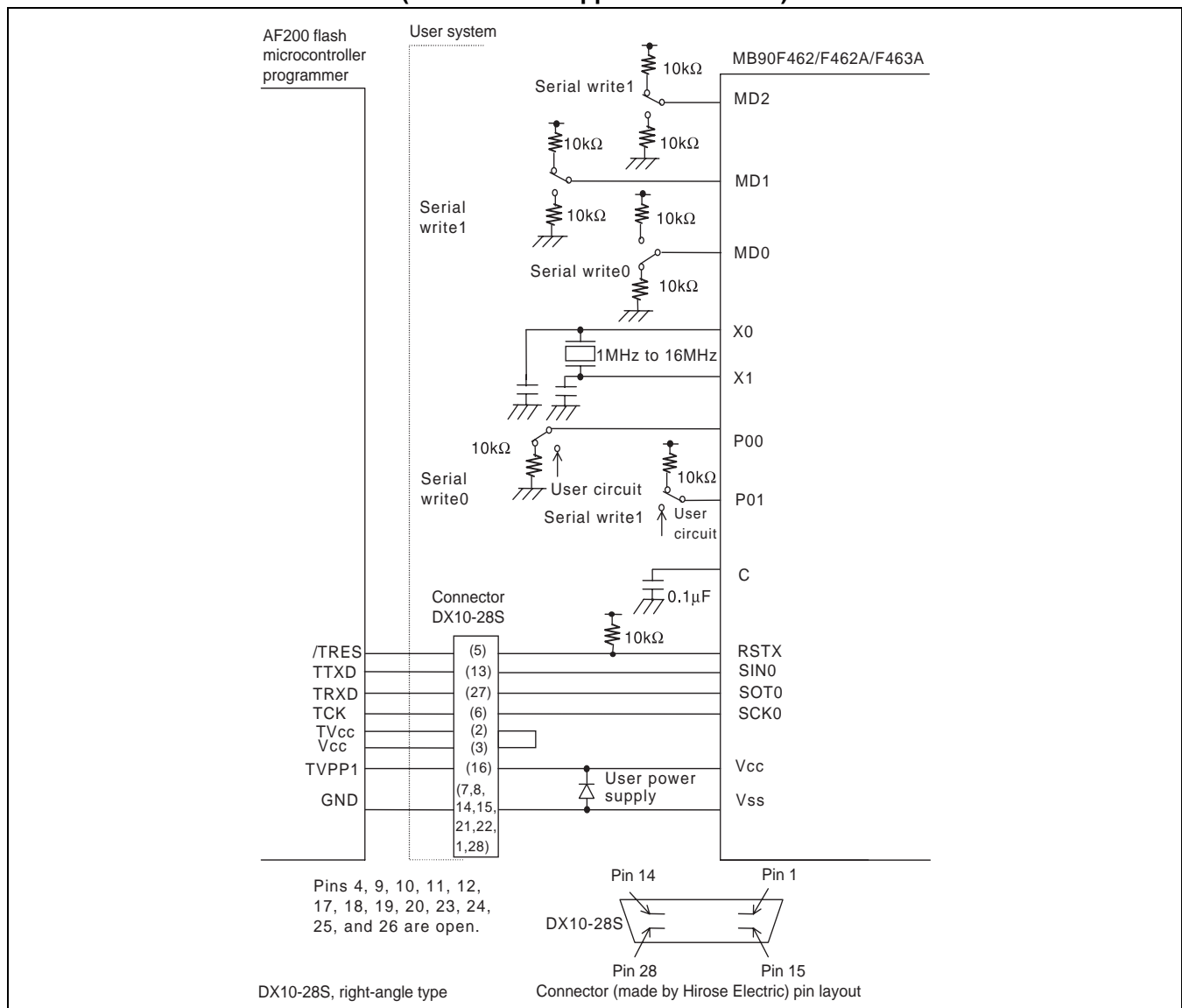
## 24.5 Example of Minimum Connection with Flash Microcontroller Programmer (When Power Supplied from Writer)

Figure 24.5-1 is an example of the minimum connection with the flash microcontroller programmer when power is supplied from the writer.

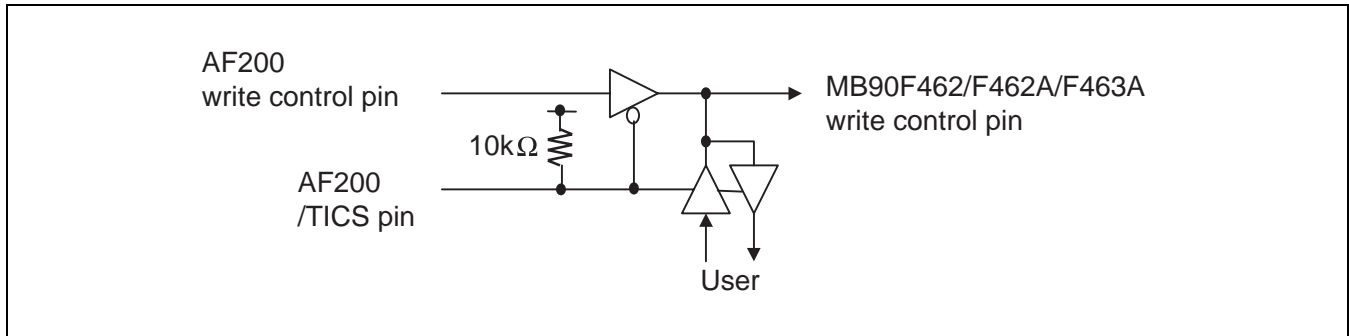
### ■ Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied from Writer)

If the pins are set as shown in Figure 24.5-1 during writing to flash memory, MD2, MD1, MD0, P00 and flash microcontroller programmer connection is unnecessary.

**Figure 24.5-1 Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied from Writer)**



- When the user system also uses the SIN0, SOT0, SCK0 pins, the control circuit shown below is necessary. (During serial writing, the user circuit can be disconnected by the flash microcontroller programmer /TICS signal.)
- Before connecting the AF200, turn off the power supplied by the user.
- When write power is supplied from the AF200, do not create a short with the power supplied by user.





# ***APPENDIX***

---

**The appendixes contain an I/O map and F<sup>2</sup>MC-16LX instructions description.**

APPENDIX A I/O MAP

APPENDIX B Instructions

## APPENDIX A I/O MAP

Table A-1 lists the addresses assigned to the registers for peripheral functions in the MB90460/465 series.

### ■ I/O Map

Table A-1 I/O Map (1/6)

Address	Abbreviation	Register	Byte access	Word access	Resource name	Initial value
000000 <sub>H</sub>	PDR0	Port 0 data register	R/W	R/W	Port 0	XXXXXXXX <sub>B</sub>
000001 <sub>H</sub>	PDR1	Port 1 data register	R/W	R/W	Port 1	XXXXXXXX <sub>B</sub>
000002 <sub>H</sub>	PDR2	Port 2 data register	R/W	R/W	Port 2	XXXXXXXX <sub>B</sub>
000003 <sub>H</sub>	PDR3	Port 3 data register	R/W	R/W	Port 3	XXXXXXXX <sub>B</sub>
000004 <sub>H</sub>	PDR4	Port 4 data register	R/W	R/W	Port 4	-XXXXXXXX <sub>B</sub>
000005 <sub>H</sub>	PDR5	Port 5 data register	R/W	R/W	Port 5	XXXXXXXX <sub>B</sub>
000006 <sub>H</sub>	PDR6	Port 6 data register	R/W	R/W	Port 6	----XXXX <sub>B</sub>
000007 <sub>H</sub>	Prohibited area					
000008 <sub>H</sub>	PWCSL0	PWC control status register CH.0 (lower byte)	R/W	R/W	PWC timer (CH.0)*	00000000 <sub>B</sub>
000009 <sub>H</sub>	PWCSH0	PWC control status register CH.0 (upper byte)	R/W	R/W		00000000 <sub>B</sub>
00000A <sub>H</sub>	PWC0	PWC data buffer register CH.0	-	R/W		XXXXXXXX <sub>B</sub>
00000B <sub>H</sub>						XXXXXXXX <sub>B</sub>
00000C <sub>H</sub>	DIV0	Divide ratio control register CH.0	R/W	R/W		-----00 <sub>B</sub>
00000D <sub>H</sub> to 0F <sub>H</sub>	Prohibited area					
000010 <sub>H</sub>	DDR0	Port 0 direction register	R/W	R/W	Port 0	00000000 <sub>B</sub>
000011 <sub>H</sub>	DDR1	Port 1 direction register	R/W	R/W	Port 1	00000000 <sub>B</sub>
000012 <sub>H</sub>	DDR2	Port 2 direction register	R/W	R/W	Port 2	00000000 <sub>B</sub>
000013 <sub>H</sub>	DDR3	Port 3 direction register	R/W	R/W	Port 3	00000000 <sub>B</sub>
000014 <sub>H</sub>	DDR4	Port 4 direction register	R/W	R/W	Port 4	-0000000 <sub>B</sub>
000015 <sub>H</sub>	DDR5	Port 5 direction register	R/W	R/W	Port 5	00000000 <sub>B</sub>
000016 <sub>H</sub>	DDR6	Port 6 direction register	R/W	R/W	Port 6	----0000 <sub>B</sub>
000017 <sub>H</sub>	ADER	Analog input enable register	R/W	R/W	Port 5, A/D	11111111 <sub>B</sub>
000018 <sub>H</sub>	Prohibited area					
000019 <sub>H</sub>	CDCR0	Clock division control register 0	R/W	R/W	Communication prescaler 0	0---0000 <sub>B</sub>
00001A <sub>H</sub>	Prohibited area					
00001B <sub>H</sub>	CDCR1	Clock division control register 1	R/W	R/W	Communication prescaler 1	0---0000 <sub>B</sub>
00001C <sub>H</sub>	RDR0	Port 0 pull-up resistor setting register	R/W	R/W	Port 0	00000000 <sub>B</sub>
00001D <sub>H</sub>	RDR1	Port 1 pull-up resistor setting register	R/W	R/W	Port 1	00000000 <sub>B</sub>
00001E <sub>H</sub> to 1F <sub>H</sub>	Prohibited area					
000020 <sub>H</sub>	SMR0	Serial mode register 0	R/W	R/W	UART0	00000000 <sub>B</sub>
000021 <sub>H</sub>	SCR0	Serial control register 0	R/W	R/W		00000100 <sub>B</sub>
000022 <sub>H</sub>	SIDR0 / SODR0	Input data register 0 / Output data register 0	R/W	R/W		XXXXXXXX <sub>B</sub>
000023 <sub>H</sub>	SSR0	Serial status register 0	R/W	R/W		00001000 <sub>B</sub>

Table A-1 I/O Map (2/6)

Address	Abbreviation	Register	Byte access	Word access	Resource name	Initial value
000024 <sub>H</sub>	SMR1	Serial mode register 1	R/W	R/W	UART1	00000000 <sub>B</sub>
000025 <sub>H</sub>	SCR1	Serial control register 1	R/W	R/W		00000100 <sub>B</sub>
000026 <sub>H</sub>	SIDR1 / SODR1	Input data register 1 / Output data register 1	R/W	R/W		XXXXXXXX <sub>B</sub>
000027 <sub>H</sub>	SSR1	Status register 1	R/W	R/W		00001000 <sub>B</sub>
000028 <sub>H</sub>	PWCSL1	PWC control status register CH.1 (lower byte)	R/W	R/W	PWC timer (CH.1)	00000000 <sub>B</sub>
000029 <sub>H</sub>	PWCSH1	PWC control status register CH.1 (upper byte)	R/W	R/W		00000000 <sub>BB</sub>
00002A <sub>H</sub>	PWC1	PWC data buffer register CH.1	-	R/W		XXXXXXXX <sub>B</sub>
00002B <sub>H</sub>						XXXXXXXX <sub>B</sub>
00002C <sub>H</sub>	DIV1	Divide ratio control register CH.1	R/W	R/W		-----00 <sub>B</sub>
00002D <sub>H</sub> to 2F <sub>H</sub>	Prohibited area					
000030 <sub>H</sub>	ENIR	Interrupt / DTP enable register	R/W	R/W	DTP/external interrupt	00000000 <sub>B</sub>
000031 <sub>H</sub>	EIRR	Interrupt / DTP cause register	R/W	R/W		XXXXXXXX <sub>B</sub>
000032 <sub>H</sub>	ELVRL	Request level setting register	R/W	R/W		00000000 <sub>B</sub>
000033 <sub>H</sub>	ELVRH		R/W	R/W		00000000 <sub>B</sub>
000034 <sub>H</sub>	ADCS0	A/D control status register (lower byte)	R/W	R/W	8/10-bit A/D converter	00000000 <sub>B</sub>
000035 <sub>H</sub>	ADCS1	A/D control status register (upper byte)	R/W	R/W		00000000 <sub>B</sub>
000036 <sub>H</sub>	ADCR0	A/D data register	R	R		XXXXXXXX <sub>B</sub>
000037 <sub>H</sub>	ADCR1		R/W	R/W		00000-XX <sub>B</sub>
000038 <sub>H</sub>	PDCR0	PPG0 down counter register	-	R	16-bit PPG timer (CH.0)	11111111 <sub>B</sub>
000039 <sub>H</sub>						11111111 <sub>B</sub>
00003A <sub>H</sub>	PCSR0	PPG0 period setting register	-	W		XXXXXXXX <sub>B</sub>
00003B <sub>H</sub>						XXXXXXXX <sub>B</sub>
00003C <sub>H</sub>	PDUT0	PPG0 duty setting register	-	W		XXXXXXXX <sub>B</sub>
00003D <sub>H</sub>						XXXXXXXX <sub>B</sub>
00003E <sub>H</sub>	PCNTL0	PPG0 control status register (lower byte)	R/W	R/W		--000000 <sub>B</sub>
00003F <sub>H</sub>	PCNTH0	PPG0 control status register (upper byte)	R/W	R/W		00000000 <sub>B</sub>
000040 <sub>H</sub>	PDCR1	PPG1 down counter register	-	R	16-bit PPG timer (CH.1)*	11111111 <sub>B</sub>
000041 <sub>H</sub>						11111111 <sub>B</sub>
000042 <sub>H</sub>	PCSR1	PPG1 period setting register	-	W		XXXXXXXX <sub>B</sub>
000043 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000044 <sub>H</sub>	PDUT1	PPG1 duty setting register	-	W		XXXXXXXX <sub>B</sub>
000045 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000046 <sub>H</sub>	PCNTL1	PPG1 control status register (lower byte)	R/W	R/W		--000000 <sub>B</sub>
000047 <sub>H</sub>	PCNTH1	PPG1 control status register (lower byte)	R/W	R/W		00000000 <sub>B</sub>
000048 <sub>H</sub>	PDCR2	PPG2 down counter register	-	R	16-bit PPG timer (CH.2)	11111111 <sub>B</sub>
000049 <sub>H</sub>						11111111 <sub>B</sub>
00004A <sub>H</sub>	PCSR2	PPG2 period setting register	-	W		XXXXXXXX <sub>B</sub>
00004B <sub>H</sub>						XXXXXXXX <sub>B</sub>
00004C <sub>H</sub>	PDUT2	PPG2 duty setting register	-	W		XXXXXXXX <sub>B</sub>
00004D <sub>H</sub>						XXXXXXXX <sub>B</sub>
00004E <sub>H</sub>	PCNTL2	PPG2 control status register (lower byte)	R/W	R/W		--000000 <sub>B</sub>
00004F <sub>H</sub>	PCNTH2	PPG2 control status register (upper byte)	R/W	R/W		00000000 <sub>B</sub>



# APPENDIX

**Table A-1 I/O Map (3/6)**

Address	Abbreviation	Register	Byte access	Word access	Resource name	Initial value
000050 <sub>H</sub>	TMRR0	16-bit timer register 0	-	R/W	Waveform generator	XXXXXXXX <sub>B</sub>
000051 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000052 <sub>H</sub>	TMRR1	16-bit timer register 1	-	R/W		XXXXXXXX <sub>B</sub>
000053 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000054 <sub>H</sub>	TMRR2	16-bit timer register 2	-	R/W		XXXXXXXX <sub>B</sub>
000055 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000056 <sub>H</sub>	DTCR0	16-bit timer control register 0	R/W	R/W		00000000 <sub>B</sub>
000057 <sub>H</sub>	DTCR1	16-bit timer control register 1	R/W	R/W		00000000 <sub>B</sub>
000058 <sub>H</sub>	DTCR2	16-bit timer control register 2	R/W	R/W		00000000 <sub>B</sub>
000059 <sub>H</sub>	SIGCR	Waveform control register	R/W	R/W		00000000 <sub>B</sub>
00005A <sub>H</sub>	CPCLRB / CPCLR	Compare clear buffer register / Compare clear register	-	R/W	16-bit free-run timer	11111111 <sub>B</sub>
00005B <sub>H</sub>						11111111 <sub>B</sub>
00005C <sub>H</sub>	TCDT	Timer data register	-	R/W		00000000 <sub>B</sub>
00005D <sub>H</sub>						00000000 <sub>B</sub>
00005E <sub>H</sub>	TCCSL	Timer control status register	R/W	R/W		00000000 <sub>B</sub>
00005F <sub>H</sub>	TCCSH		R/W	R/W		-0000000 <sub>B</sub>
000060 <sub>H</sub>	IPCP0	Input capture data register CH.0	-	R	16-bit input capture (CH.0 to CH.3)	XXXXXXXX <sub>B</sub>
000061 <sub>H</sub>		Input capture data register CH.0	-	R		XXXXXXXX <sub>B</sub>
000062 <sub>H</sub>	IPCP1	Input capture data register CH.1	-	R		XXXXXXXX <sub>B</sub>
000063 <sub>H</sub>		Input capture data register CH.1	-	R		XXXXXXXX <sub>B</sub>
000064 <sub>H</sub>	IPCP2	Input capture data register CH.2	-	R		XXXXXXXX <sub>B</sub>
000065 <sub>H</sub>		Input capture data register CH.2	-	R		XXXXXXXX <sub>B</sub>
000066 <sub>H</sub>	IPCP3	Input capture data register CH.3	-	R		XXXXXXXX <sub>B</sub>
000067 <sub>H</sub>		Input capture data register CH.3	-	R		XXXXXXXX <sub>B</sub>
000068 <sub>H</sub>	PICSL01	PPG output control / input capture control status register 01 (lower byte)	R/W	R/W		00000000 <sub>B</sub>
000069 <sub>H</sub>	PICSH01	PPG output control / input capture control status register 01 (upper byte)	R/W	R/W		00000000 <sub>B</sub>
00006A <sub>H</sub>	ICSL23	Input capture control status register 23 (lower byte)	R/W	R/W		00000000 <sub>B</sub>
00006B <sub>H</sub>	ICSH23	Input capture control status register 23 (upper byte)	R	R		-----00 <sub>B</sub>
00006C <sub>H</sub> to 6E <sub>H</sub>	Prohibited area					
00006F <sub>H</sub>	ROMM	ROM mirroring function selection register	W	W	ROM mirroring function	-----1 <sub>B</sub>

Table A-1 I/O Map (4/6)

Address	Abbreviation	Register	Byte access	Word access	Resource name	Initial value
000070 <sub>H</sub>	OCCPB0 / OCCP0	Output compare buffer register / Output compare register 0	-	R/W	Output compare (CH.0 to CH.5)	XXXXXXXX <sub>B</sub>
000071 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000072 <sub>H</sub>	OCCPB1 / OCCP1	Output compare buffer register / Output compare register 1	-	R/W		XXXXXXXX <sub>B</sub>
000073 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000074 <sub>H</sub>	OCCPB2 / OCCP2	Output compare buffer register / Output compare register 2	-	R/W		XXXXXXXX <sub>B</sub>
000075 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000076 <sub>H</sub>	OCCPB3 / OCCP3	Output compare buffer register / Output compare register 3	-	R/W		XXXXXXXX <sub>B</sub>
000077 <sub>H</sub>						XXXXXXXX <sub>B</sub>
000078 <sub>H</sub>	OCCPB4 / OCCP4	Output compare buffer register / Output compare register 4	-	R/W		XXXXXXXX <sub>B</sub>
000079 <sub>H</sub>						XXXXXXXX <sub>B</sub>
00007A <sub>H</sub>	OCCPB5 / OCCP5	Output compare buffer register / Output compare register 5	-	R/W		XXXXXXXX <sub>B</sub>
00007B <sub>H</sub>						XXXXXXXX <sub>B</sub>
00007C <sub>H</sub>	OCS0	Compare control register 0	R/W	R/W		00000000 <sub>B</sub>
00007D <sub>H</sub>	OCS1	Compare control register 1	R/W	R/W		-0000000 <sub>B</sub>
00007E <sub>H</sub>	OCS2	Compare control register 2	R/W	R/W		00000000 <sub>B</sub>
00007F <sub>H</sub>	OCS3	Compare control register 3	R/W	R/W		-0000000 <sub>B</sub>
000080 <sub>H</sub>	OCS4	Compare control register 4	R/W	R/W		00000000 <sub>B</sub>
000081 <sub>H</sub>	OCS5	Compare control register 5	R/W	R/W		-0000000 <sub>B</sub>
000082 <sub>H</sub>	TMCSRL0	Timer control status register CH.0 (lower byte)	R/W	R/W	16-bit reload timer (CH.0)	00000000 <sub>B</sub>
000083 <sub>H</sub>	TMCSRH0	Timer control status register CH.0 (upper byte)	R/W	R/W		----0000 <sub>B</sub>
000084 <sub>H</sub>	TMR0 / TMRD0	16-bit timer register / 16-bit reload register CH.0	-	R/W		XXXXXXXX <sub>B</sub>
000085 <sub>H</sub>					XXXXXXXX <sub>B</sub>	
000086 <sub>H</sub>	TMCSRL1	Timer control status register CH.1 (lower byte)	R/W	R/W	16-bit reload timer (CH.1)	00000000 <sub>B</sub>
000087 <sub>H</sub>	TMCSRH1	Timer control status register CH.1 (upper byte)	R/W	R/W		----0000 <sub>B</sub>
000088 <sub>H</sub>	TMR1 / TMRD1	16-bit timer register / 16-bit reload register CH.1	-	R/W		XXXXXXXX <sub>B</sub>
000089 <sub>H</sub>						XXXXXXXX <sub>B</sub>
00008A <sub>H</sub>	OPCLR	Output control lower register	R/W	R/W	Waveform sequencer*	00000000 <sub>B</sub>
00008B <sub>H</sub>	OPCUR	Output control upper register	R/W	R/W		00000000 <sub>B</sub>
00008C <sub>H</sub>	IPCLR	Input control lower register	R/W	R/W		00000000 <sub>B</sub>
00008D <sub>H</sub>	IPCUR	Input control upper register	R/W	R/W		00000000 <sub>B</sub>
00008E <sub>H</sub>	TCSR	Timer control status register	R/W	R/W		00000000 <sub>B</sub>
00008F <sub>H</sub>	NCCR	Noise cancellation control register	R/W	R/W		00000000 <sub>B</sub>
000090 <sub>H</sub> to 9D <sub>H</sub>	Prohibited area					
00009E <sub>H</sub>	PACSR	Program address detect control status register	R/W	R/W	Address match detection	----0000 <sub>B</sub>
00009F <sub>H</sub>	DIRR	Delayed interrupt cause / clear register	R/W	R/W	Delayed interrupt	-----0 <sub>B</sub>
0000A0 <sub>H</sub>	LPMCR	Low-power consumption mode register	R/W	R/W	Low-power consumption control register	00011000 <sub>B</sub>
0000A1 <sub>H</sub>	CKSCR	Clock selection register	R/W	R/W		11111100 <sub>B</sub>
0000A2 <sub>H</sub> to A7 <sub>H</sub>	Prohibited area					
0000A8 <sub>H</sub>	WDTC	Watchdog control register	R/W	R/W	Watchdog timer	X-XXX111 <sub>B</sub>
0000A9 <sub>H</sub>	TBTC	Time-base timer control register	R/W	R/W	Time-base timer	1--00100 <sub>B</sub>
0000AA <sub>H</sub> to AD <sub>H</sub>	Prohibited area					
0000AE <sub>H</sub>	FMCS	Flash memory control status register	R/W	R/W	Flash memory interface circuit	00010000 <sub>B</sub>
0000AF <sub>H</sub>	Prohibited area					

**Table A-1 I/O Map (5/6)**

Address	Abbreviation	Register	Byte access	Word access	Resource name	Initial value
0000B0 <sub>H</sub>	ICR00	Interrupt control register 00	R/W	R/W	Interrupt controller	00000111 <sub>B</sub>
0000B1 <sub>H</sub>	ICR01	Interrupt control register 01	R/W	R/W		00000111 <sub>B</sub>
0000B2 <sub>H</sub>	ICR02	Interrupt control register 02	R/W	R/W		00000111 <sub>B</sub>
0000B3 <sub>H</sub>	ICR03	Interrupt control register 03	R/W	R/W		00000111 <sub>B</sub>
0000B4 <sub>H</sub>	ICR04	Interrupt control register 04	R/W	R/W		00000111 <sub>B</sub>
0000B5 <sub>H</sub>	ICR05	Interrupt control register 05	R/W	R/W		00000111 <sub>B</sub>
0000B6 <sub>H</sub>	ICR06	Interrupt control register 06	R/W	R/W		00000111 <sub>B</sub>
0000B7 <sub>H</sub>	ICR07	Interrupt control register 07	R/W	R/W		00000111 <sub>B</sub>
0000B8 <sub>H</sub>	ICR08	Interrupt control register 08	R/W	R/W		00000111 <sub>B</sub>
0000B9 <sub>H</sub>	ICR09	Interrupt control register 09	R/W	R/W		00000111 <sub>B</sub>
0000BA <sub>H</sub>	ICR10	Interrupt control register 10	R/W	R/W		00000111 <sub>B</sub>
0000BB <sub>H</sub>	ICR11	Interrupt control register 11	R/W	R/W		00000111 <sub>B</sub>
0000BC <sub>H</sub>	ICR12	Interrupt control register 12	R/W	R/W		00000111 <sub>B</sub>
0000BD <sub>H</sub>	ICR13	Interrupt control register 13	R/W	R/W		00000111 <sub>B</sub>
0000BE <sub>H</sub>	ICR14	Interrupt control register 14	R/W	R/W		00000111 <sub>B</sub>
0000BF <sub>H</sub>	ICR15	Interrupt control register 15	R/W	R/W		00000111 <sub>B</sub>
0000C0 <sub>H</sub> to FF <sub>H</sub>	External area					
001FF0 <sub>H</sub>	PADRL0	Program address detection register 0 (lower byte)	R/W	R/W	Address match detection	XXXXXXXX <sub>B</sub>
001FF1 <sub>H</sub>	PADRM0	Program address detection register 1 (middle byte)	R/W	R/W		XXXXXXXX <sub>B</sub>
001FF2 <sub>H</sub>	PADRH0	Program address detection register 2 (upper byte)	R/W	R/W		XXXXXXXX <sub>B</sub>
001FF3 <sub>H</sub>	PADRL1	Program address detection register 3 (lower byte)	R/W	R/W		XXXXXXXX <sub>B</sub>
001FF4 <sub>H</sub>	PADRM1	Program address detection register 4 (middle byte)	R/W	R/W		XXXXXXXX <sub>B</sub>
001FF5 <sub>H</sub>	PADRH1	Program address detection register 5 (upper byte)	R/W	R/W		XXXXXXXX <sub>B</sub>

Table A-1 I/O Map (6/6)

Address	Abbreviation	Register	Byte access	Word access	Resource name	Initial value
003FE0 <sub>H</sub>	OPDBR0	Output data buffer register 0	-	R/W	Waveform sequencer*	00000000 <sub>B</sub>
003FE1 <sub>H</sub>		Output data buffer register 0	-	R/W		00000000 <sub>B</sub>
003FE2 <sub>H</sub>	OPDBR1	Output data buffer register 1	-	R/W		00000000 <sub>B</sub>
003FE3 <sub>H</sub>		Output data buffer register 1	-	R/W		00000000 <sub>B</sub>
003FE4 <sub>H</sub>	OPDBR2	Output data buffer register 2	-	R/W		00000000 <sub>B</sub>
003FE5 <sub>H</sub>		Output data buffer register 2	-	R/W		00000000 <sub>B</sub>
003FE6 <sub>H</sub>	OPDBR3	Output data buffer register 3	-	R/W		00000000 <sub>B</sub>
003FE7 <sub>H</sub>		Output data buffer register 3	-	R/W		00000000 <sub>B</sub>
003F78 <sub>H</sub>	OPDBR4	Output data buffer register 4	-	R/W		00000000 <sub>B</sub>
003FE9 <sub>H</sub>		Output data buffer register 4	-	R/W		00000000 <sub>B</sub>
003FEA <sub>H</sub>	OPDBR5	Output data buffer register 5	-	R/W		00000000 <sub>B</sub>
003FEB <sub>H</sub>		Output data buffer register 5	-	R/W		00000000 <sub>B</sub>
003FEC <sub>H</sub>	OPEBR6	Output data buffer register 6	-	R/W		00000000 <sub>B</sub>
003FED <sub>H</sub>		Output data buffer register 6	-	R/W		00000000 <sub>B</sub>
003FEE <sub>H</sub>	OPEBR7	Output data buffer register 7	-	R/W		00000000 <sub>B</sub>
003FEF <sub>H</sub>		Output data buffer register 7	-	R/W		00000000 <sub>B</sub>
003FF0 <sub>H</sub>	OPEBR8	Output data buffer register 8	-	R/W		00000000 <sub>B</sub>
003FF1 <sub>H</sub>		Output data buffer register 8	-	R/W		00000000 <sub>B</sub>
003FF2 <sub>H</sub>	OPEBR9	Output data buffer register 9	-	R/W		00000000 <sub>B</sub>
003FF3 <sub>H</sub>		Output data buffer register 9	-	R/W		00000000 <sub>B</sub>
003FF4 <sub>H</sub>	OPEBRA	Output data buffer register A	-	R/W		00000000 <sub>B</sub>
003FF5 <sub>H</sub>		Output data buffer register A	-	R/W		00000000 <sub>B</sub>
003FF6 <sub>H</sub>	OPEBRB	Output data buffer register B	-	R/W		00000000 <sub>B</sub>
003FF7 <sub>H</sub>		Output data buffer register B	-	R/W		00000000 <sub>B</sub>
003FF8 <sub>H</sub>	OPDR	Output data register	-	R		XXXXXXXX <sub>B</sub>
003FF9 <sub>H</sub>		Output data register	-	R		0000XXXX <sub>B</sub>
003FFA <sub>H</sub>	CPCR	Compare clear register	-	R/W		XXXXXXXX <sub>B</sub>
003FFB <sub>H</sub>		Compare clear register	-	R/W		XXXXXXXX <sub>B</sub>
003FFC <sub>H</sub>	TMBR	Timer buffer register	-	R		00000000 <sub>B</sub>
003FFD <sub>H</sub>		Timer buffer register	-	R		00000000 <sub>B</sub>
003FFE <sub>H</sub> to 003FFF <sub>H</sub> Prohibited area						

- Meaning of abbreviations used for reading and writing

R/W:Read and write enabled

R:Read-only

W:Write-only

- Explanation of initial values

0:The bit is initialized to 0.

1:The bit is initialized to 1.

X:The initial value of the bit is undefined.

-:The bit is not used. Its initial value is undefined.

- Instruction using IO addressing e.g. MOV A, io, is not supported for registers area 003FE0<sub>H</sub> to 003FFF<sub>H</sub>.

\*: These registers are not present in MB90465 series.

## 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	<table><tr><td>2</td><td>2</td><td>3</td><td>3</td><td>:</td><td>4</td><td>4</td><td>5</td><td>5</td></tr></table>	2	2	3	3	:	4	4	5	5
2	2	3	3	:	4	4	5	5			
After execution	A	<table><tr><td>4</td><td>4</td><td>5</td><td>5</td><td>:</td><td>1</td><td>2</td><td>1</td><td>2</td></tr></table> (Some instructions transfer AL to AH.)	4	4	5	5	:	1	2	1	2
4	4	5	5	:	1	2	1	2			

#### ● Register direct addressing

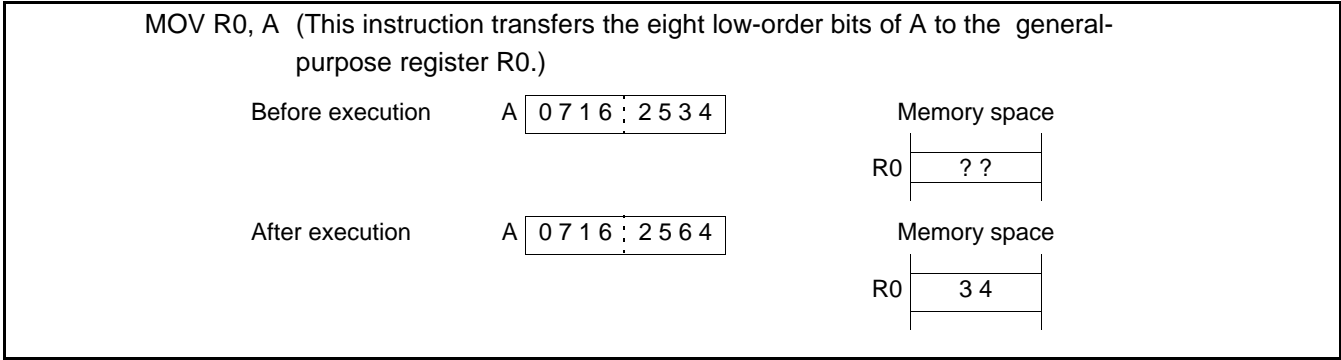
Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

**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.

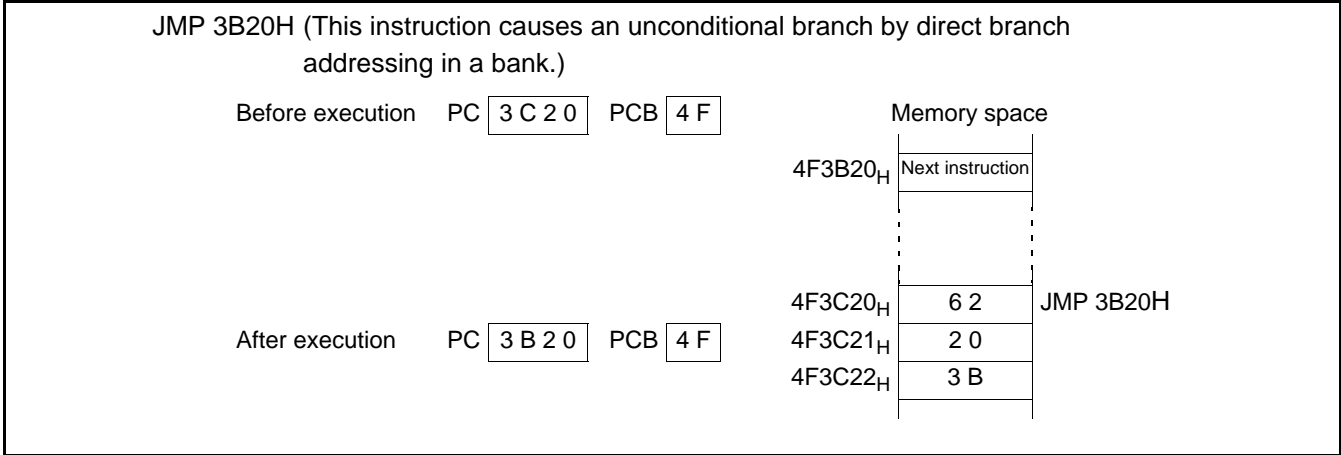
Figure B.3-2 Example of Register Direct Addressing



● 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).

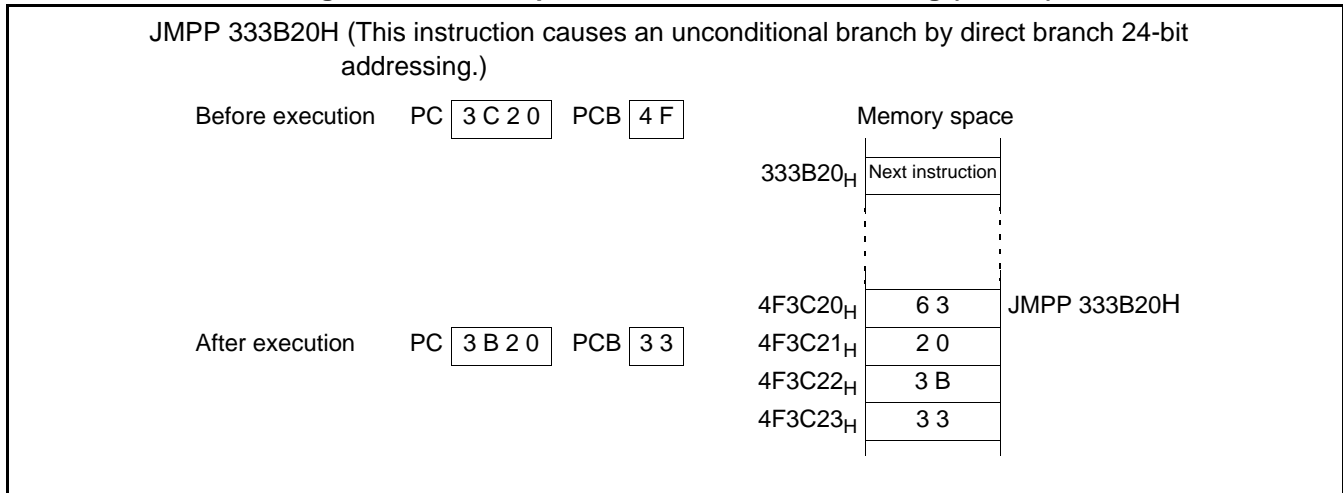
Figure B.3-3 Example of Direct Branch Addressing (addr16)



- 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.

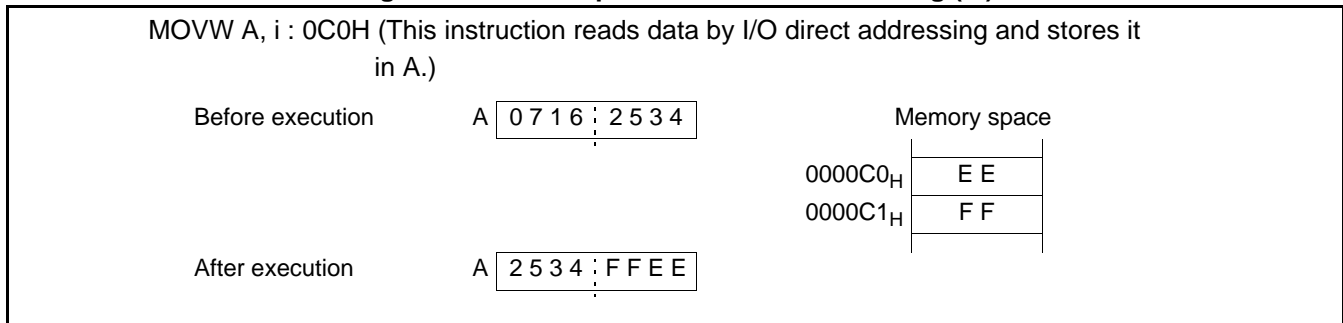
### Figure B.3-4 Example of Direct Branch Addressing (addr24)



- 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.

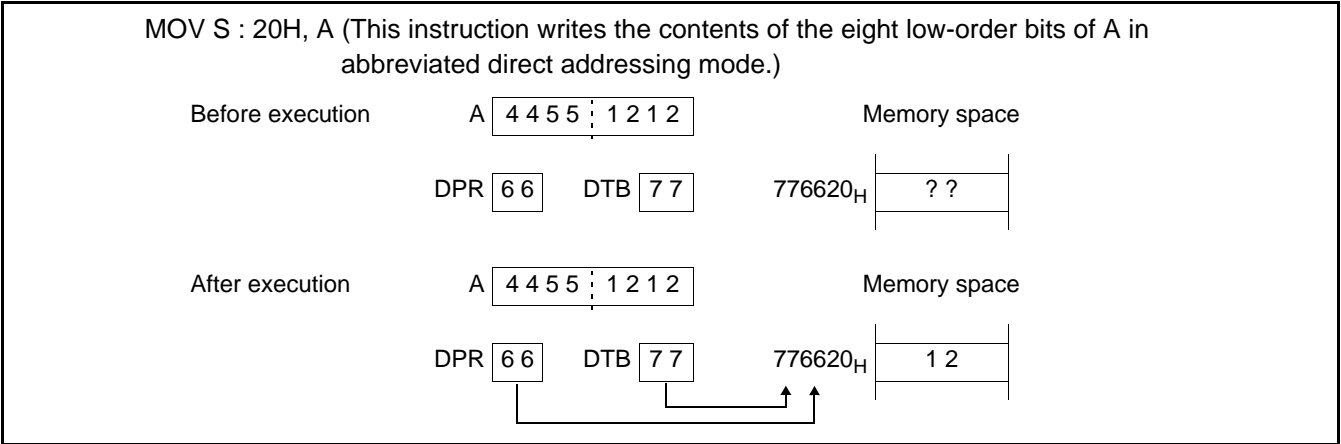
### Figure B.3-5 Example of I/O Direct Addressing (io)



● 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).

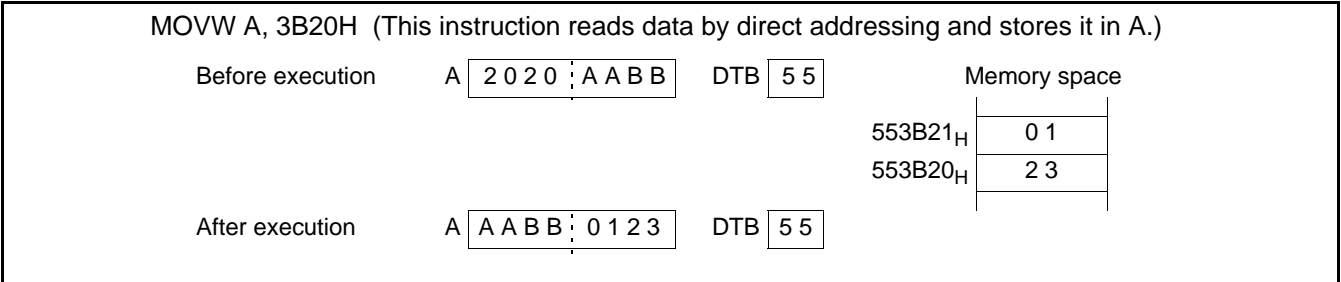
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)



● 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.

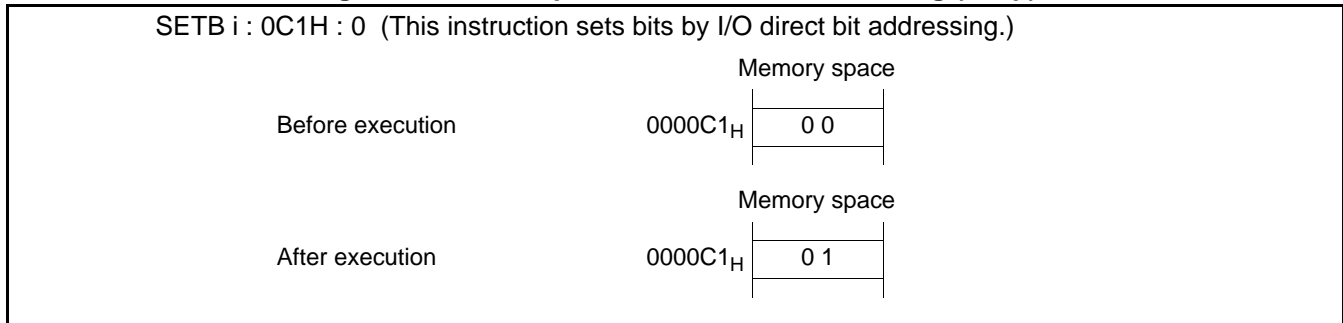
Figure B.3-7 Example of Direct Addressing (addr16)



● 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).

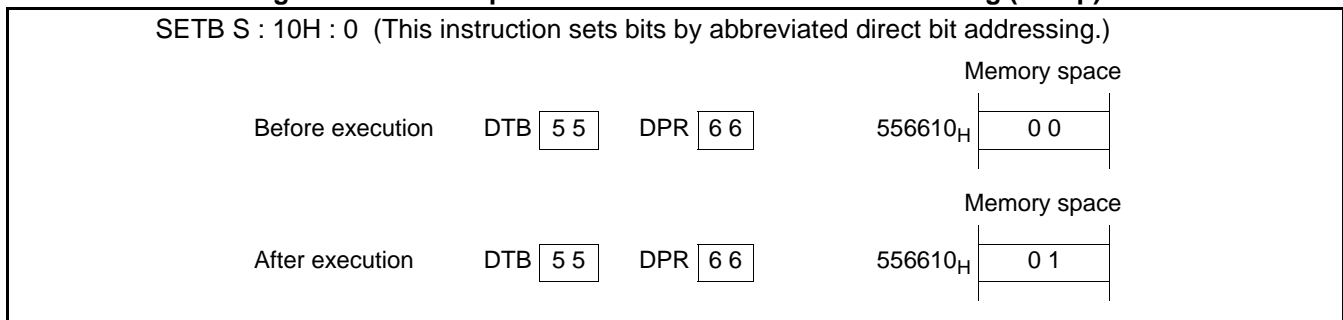
**Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)**



● 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).

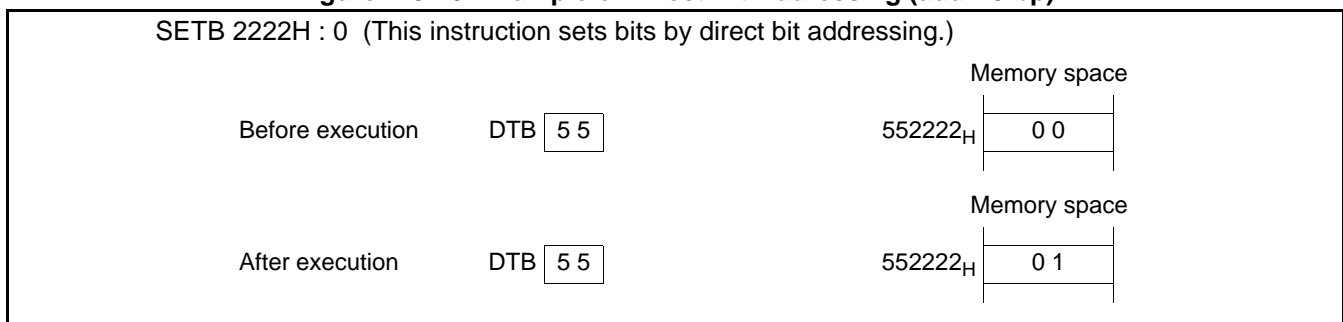
**Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)**



● 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)**



● 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)

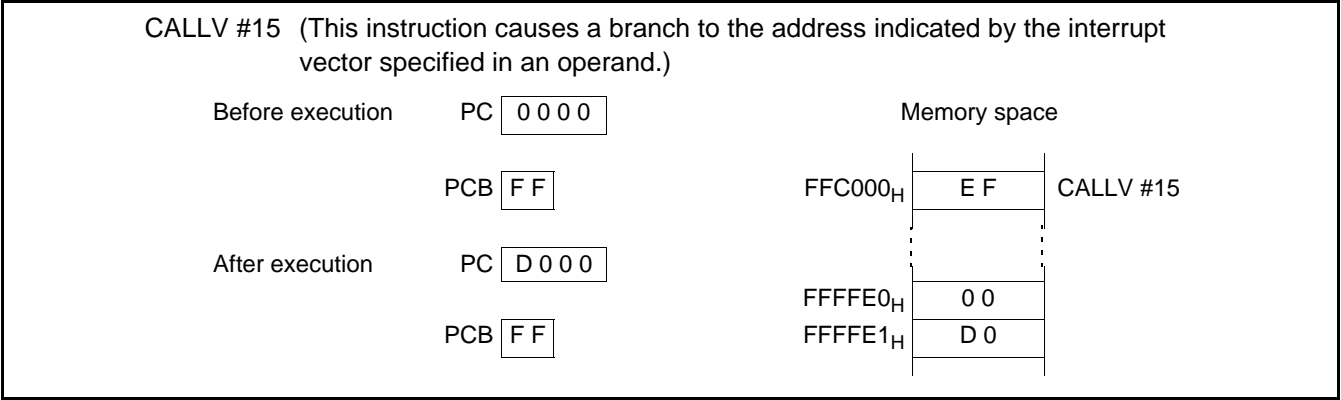


Table B.3-2 CALLV Vector List

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>	XXFFE <sub>F</sub> <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

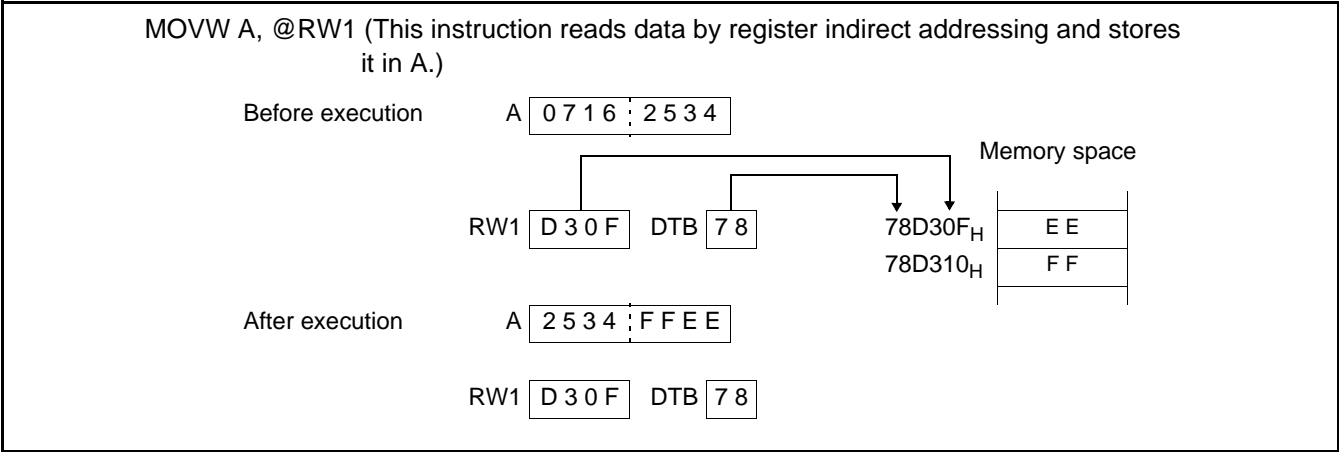
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

### ■ 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)

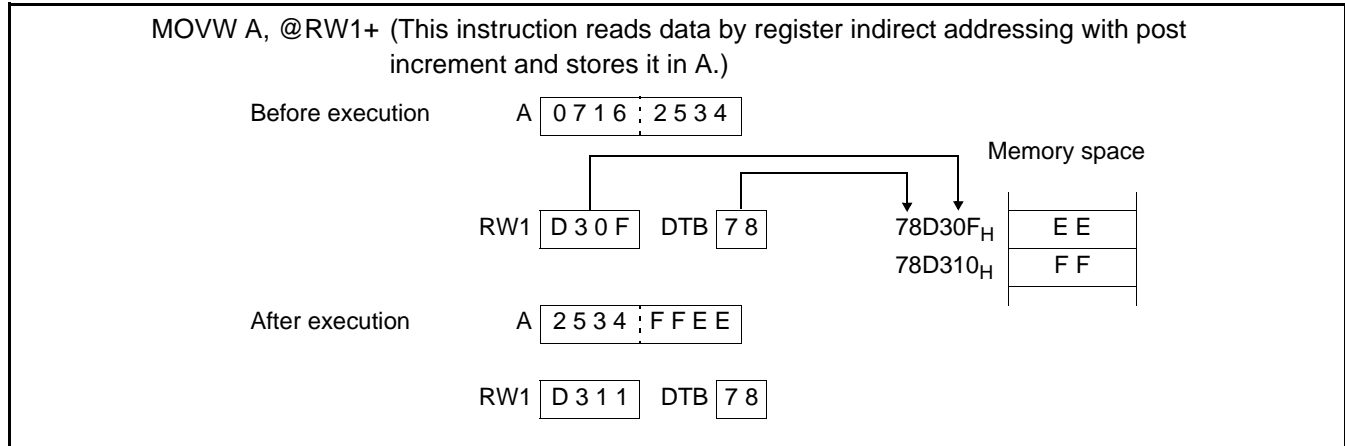


● Register indirect addressing with post increment (@RWj+ j = 0 to 3)

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.

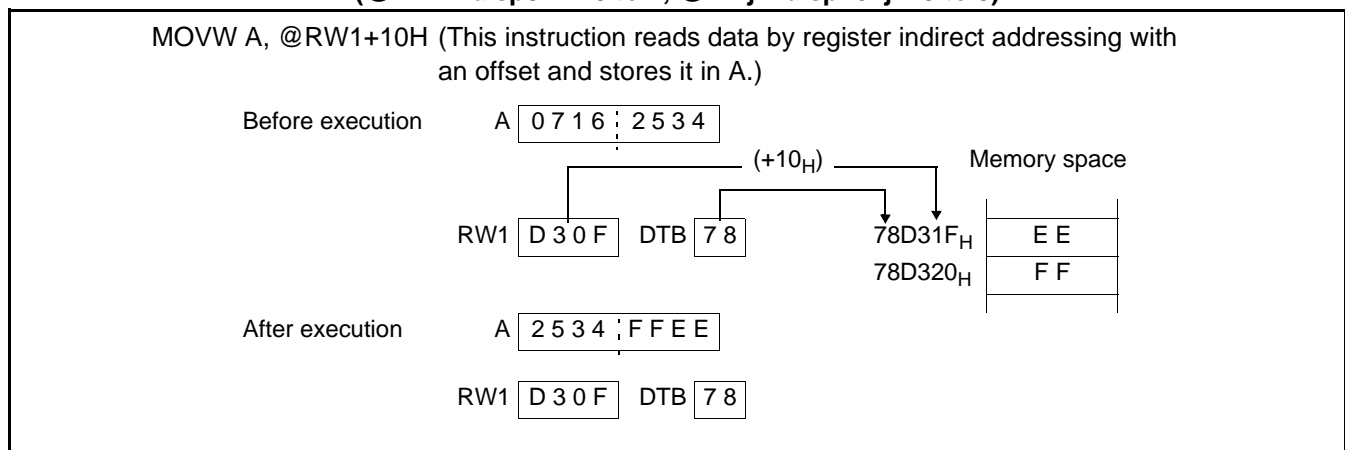
If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented 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)

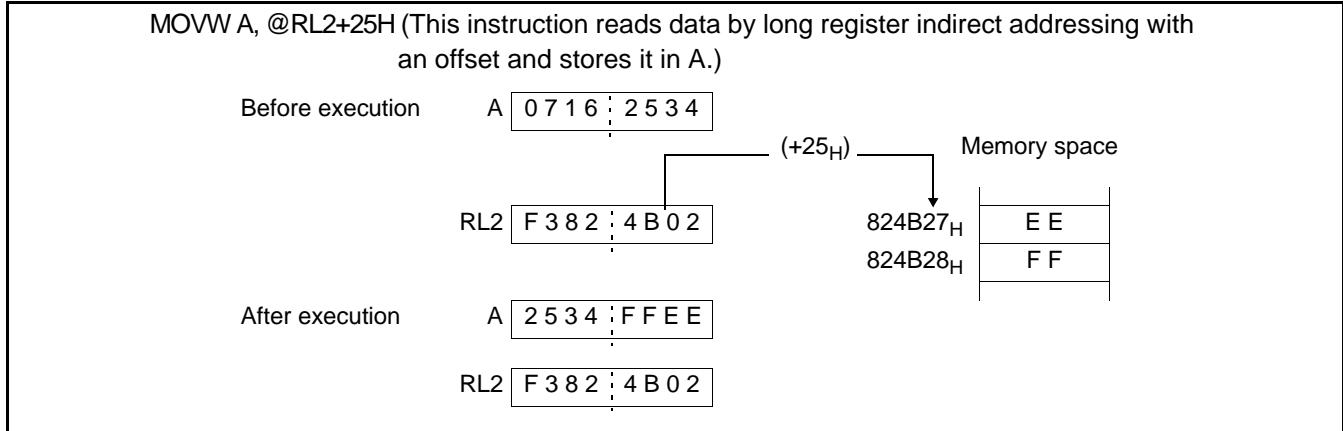
Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or 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)**

● Long register indirect addressing with offset ( $@RLi + \text{disp8}$   $i = 0$  to 3)

Memory is accessed using the address that is the 24 low-order bits obtained by adding an offset to the contents of general-purpose register  $RLi$ . The offset is 8-bits long and is added as a signed numeric value.

**Figure B.4-4 Example of Long Register Indirect Addressing with Offset ( $@RLi + \text{disp8}$   $i = 0$  to 3)**

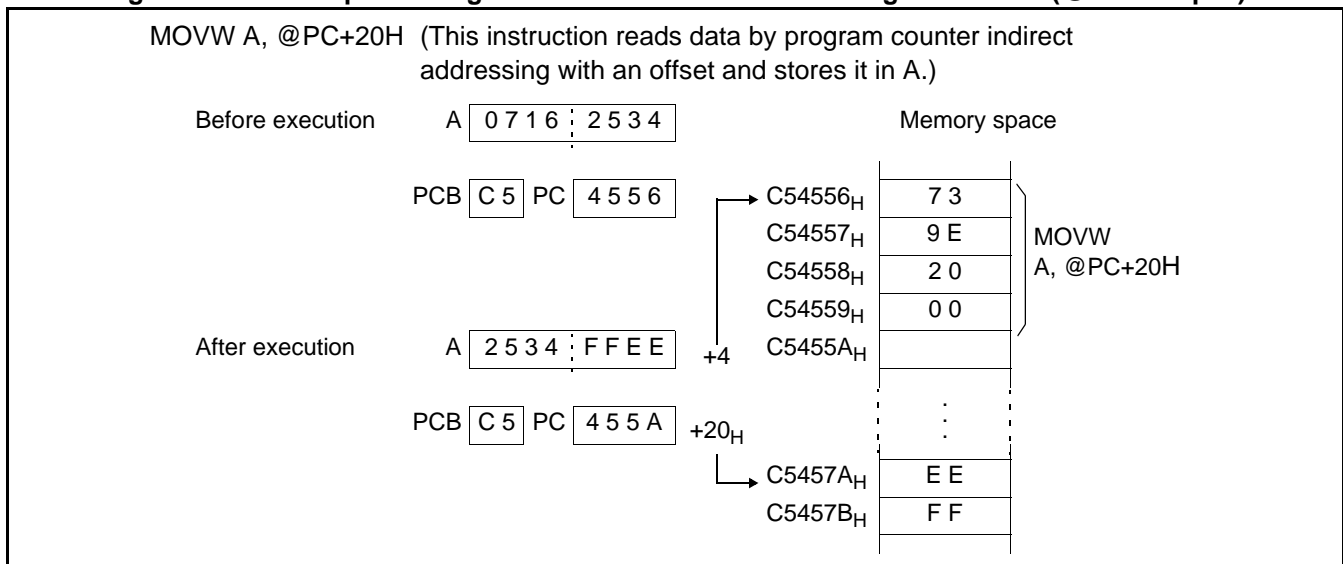


● Program counter indirect addressing with offset ( $@PC + \text{disp16}$ )

Memory is accessed using the address indicated by (instruction address + 4 +  $\text{disp16}$ ). The offset is one word long. Address bits 16 to 23 are specified by the program counter bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address +  $\text{disp16}$ ):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8
- MOVW eam, #imm16

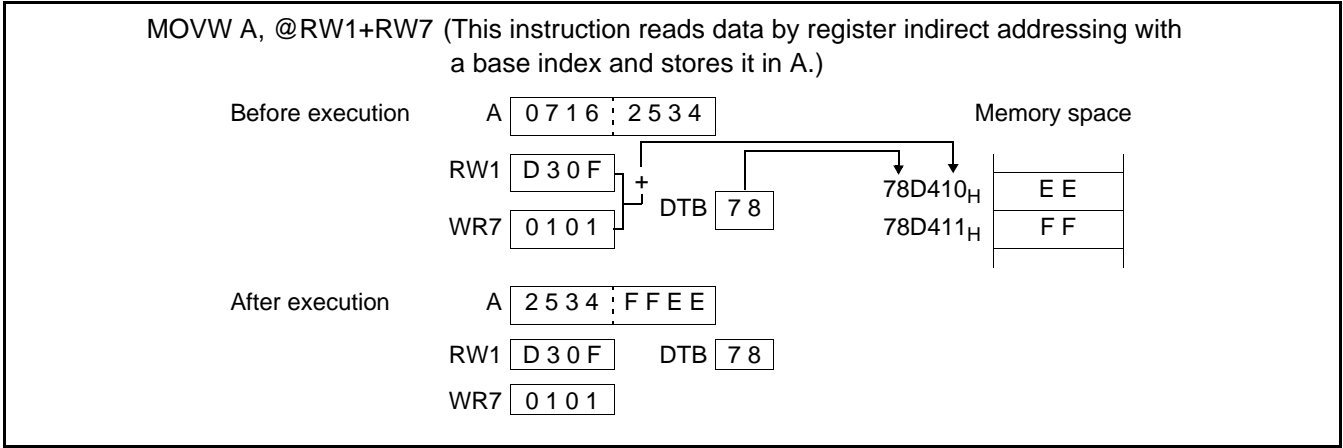
**Figure B.4-5 Example of Program Counter Indirect Addressing with Offset ( $@PC + \text{disp16}$ )**



● 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).

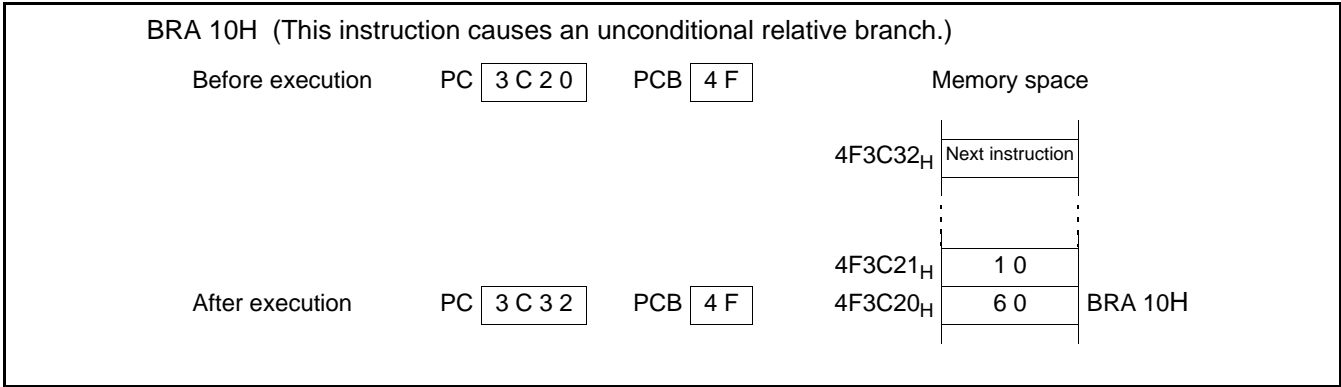
**Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)**



● 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**

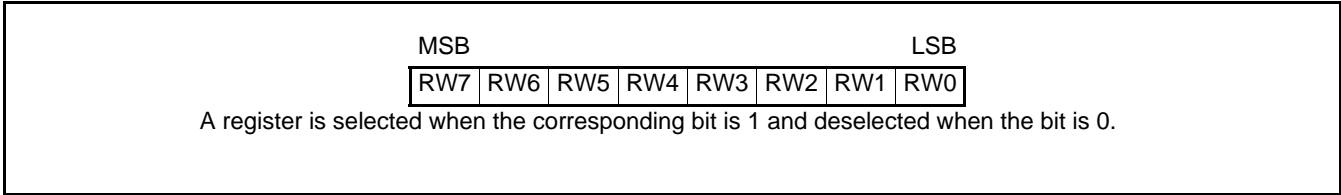
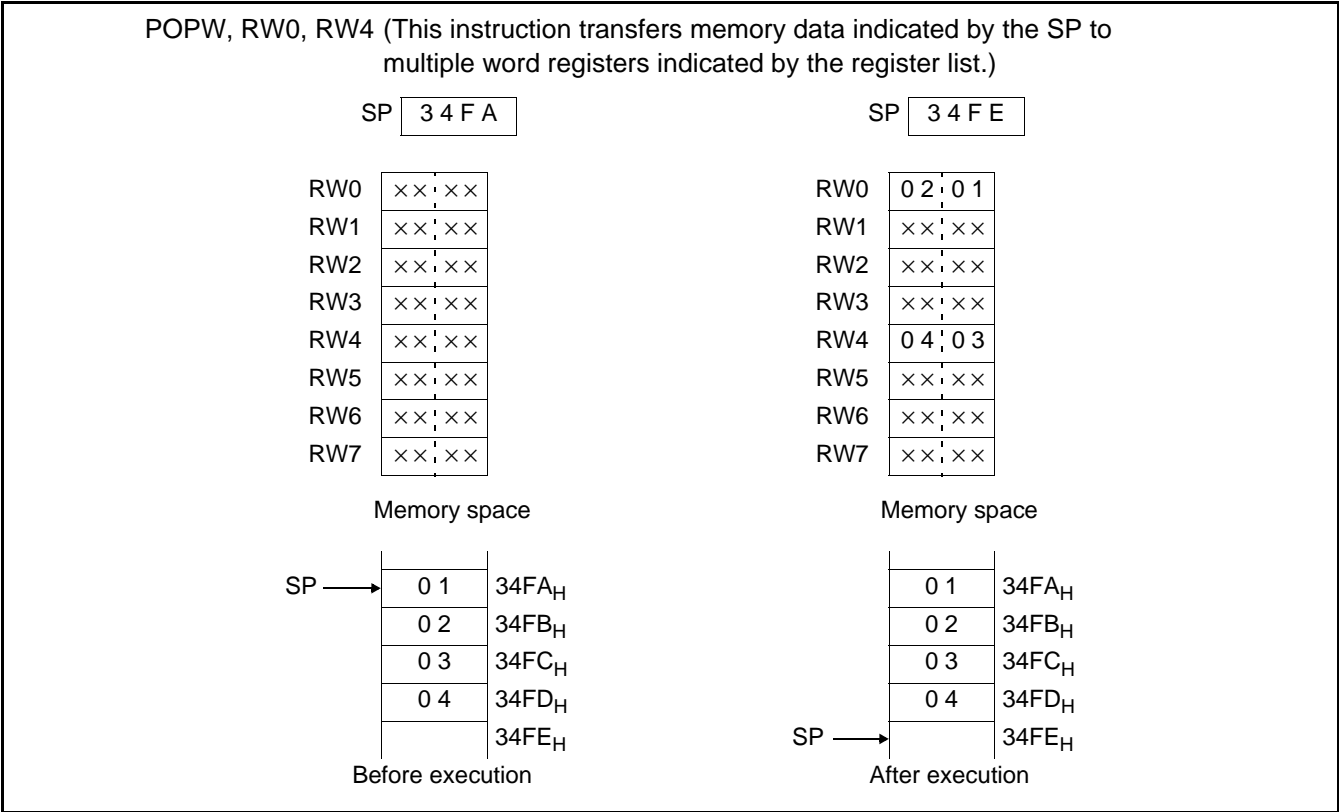


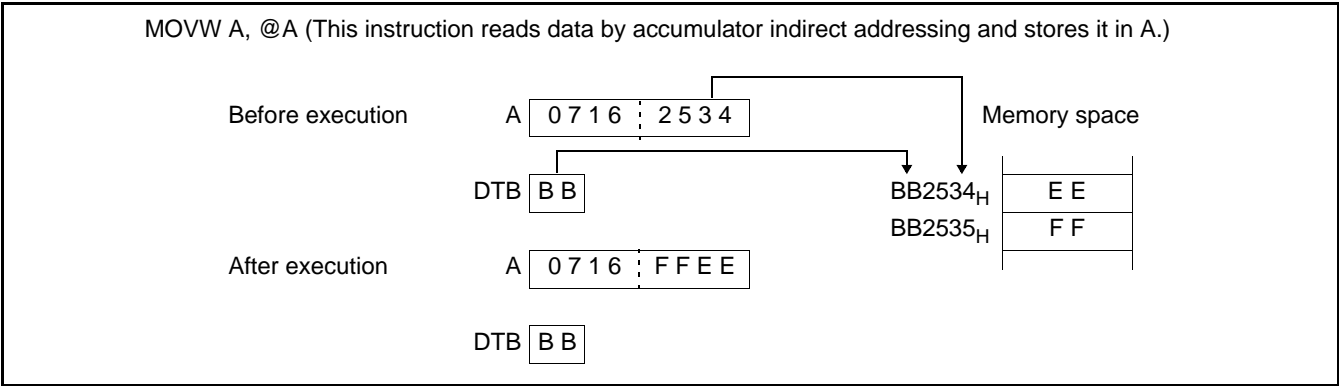
Figure B.4-9 Example of Register List (rlist)



● 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).

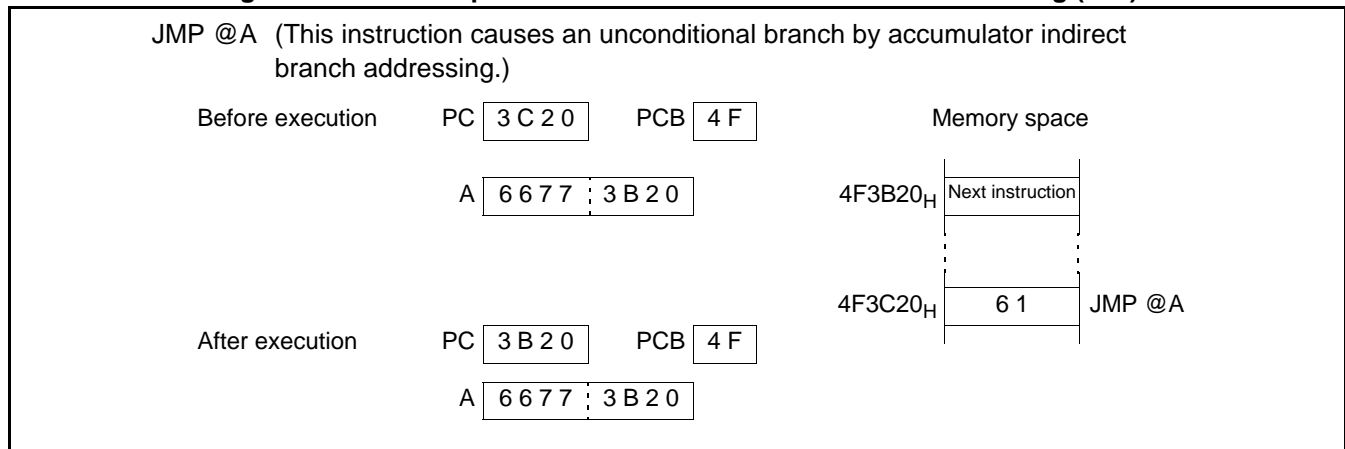
Figure B.4-10 Example of Accumulator Indirect Addressing (@A)



● 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.

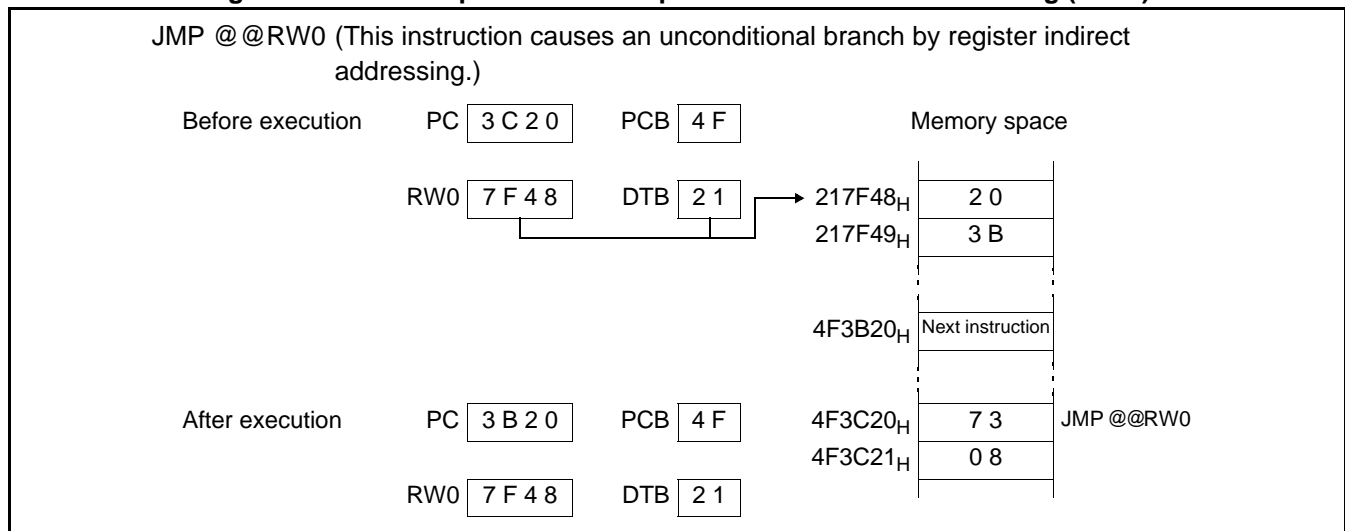
**Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)**



● Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

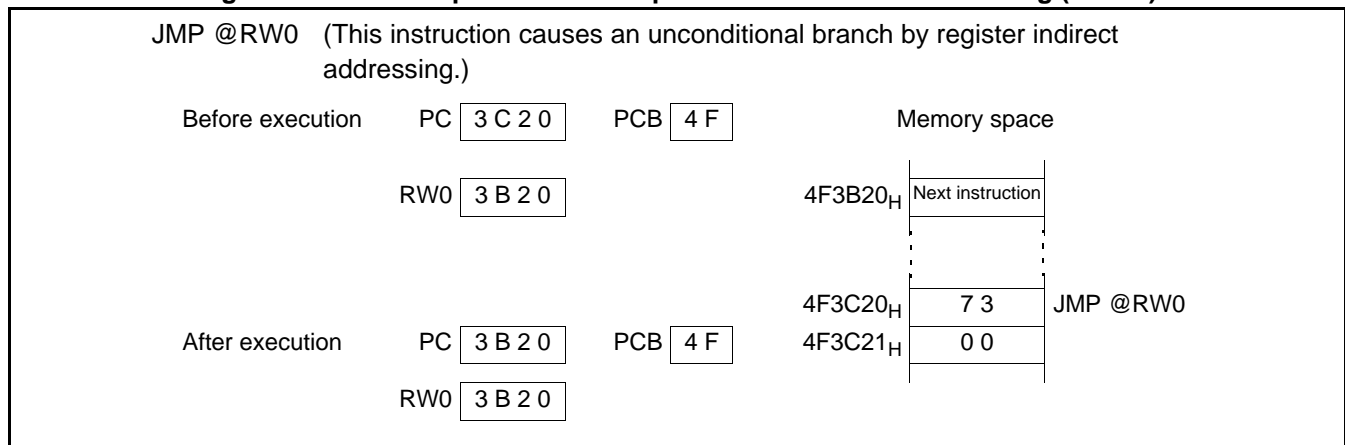
**Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)**



- Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by `eam`.

**Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)**



## 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	@RW0+RW7	4	2
1D	@RW1+RW7	4	2
1E	@PC+disp16	2	0
1F	addr16	1	0

\*: (a) is used for ~ (cycle count) and B (correction value) in "B.8 F<sup>2</sup>MC-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 F<sup>2</sup>MC-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 F<sup>2</sup>MC-16LX Instruction List".

## B.7 How to Read the Instruction List

Table B.7-1 describes the items used in "B.8 F<sup>2</sup>MC-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 (2/2)**

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 (2/2)**

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 to 07)
eam	Effective addressing (code 08 to 1F)
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) $\leftarrow$ (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	5	0	(b)	byte (A) $\leftarrow$ (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	4 + (a)	0	(b)	byte (A) $\leftarrow$ (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	3	2	0	byte (ear) $\leftarrow$ (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD eam,A	2+	5 + (a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	0	byte (A) $\leftarrow$ (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	4 + (a)	0	(b)	byte (A) $\leftarrow$ (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A	1	3	0	0	byte (A) $\leftarrow$ (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	0	byte (A) $\leftarrow$ (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	5	0	(b)	byte (A) $\leftarrow$ (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	4 + (a)	0	(b)	byte (A) $\leftarrow$ (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	3	2	0	byte (ear) $\leftarrow$ (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB eam,A	2+	5 + (a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	0	byte (A) $\leftarrow$ (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	4 + (a)	0	(b)	byte (A) $\leftarrow$ (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC A	1	3	0	0	byte (A) $\leftarrow$ (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	0	word (A) $\leftarrow$ (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	0	word (A) $\leftarrow$ (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	3	2	0	word (ear) $\leftarrow$ (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW eam,A	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	0	word (A) $\leftarrow$ (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	0	word (A) $\leftarrow$ (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	3	2	0	word (ear) $\leftarrow$ (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW eam,A	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL A,ear	2	6	2	0	long (A) $\leftarrow$ (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) $\leftarrow$ (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	0	long (A) $\leftarrow$ (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	6	2	0	long (A) $\leftarrow$ (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) $\leftarrow$ (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	0	long (A) $\leftarrow$ (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) $\leftarrow$ (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	4+(a)	0	(b)	byte (A) $\leftarrow$ (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	2	0	byte (ear) $\leftarrow$ (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND eam,A	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	0	byte (A) $\leftarrow$ (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	4+(a)	0	(b)	byte (A) $\leftarrow$ (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	2	0	byte (ear) $\leftarrow$ (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR eam,A	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	0	byte (A) $\leftarrow$ (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	3	1	0	byte (A) $\leftarrow$ (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	4+(a)	0	(b)	byte (A) $\leftarrow$ (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	2	0	byte (ear) $\leftarrow$ (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR eam,A	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	0	byte (A) $\leftarrow$ not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	3	2	0	byte (ear) $\leftarrow$ not (ear)	-	-	-	-	-	*	*	R	-	-
NOT eam	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	0	word (A) $\leftarrow$ (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	0	word (A) $\leftarrow$ (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	2	0	word (ear) $\leftarrow$ (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW eam,A	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	0	word (A) $\leftarrow$ (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	0	word (A) $\leftarrow$ (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	2	0	word (ear) $\leftarrow$ (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW eam,A	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	0	word (A) $\leftarrow$ (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	0	word (A) $\leftarrow$ (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	3	1	0	word (A) $\leftarrow$ (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	4+(a)	0	(c)	word (A) $\leftarrow$ (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	2	0	word (ear) $\leftarrow$ (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW eam,A	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	0	word (A) $\leftarrow$ not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	3	2	0	word (ear) $\leftarrow$ not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW eam	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ 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) $\leftarrow$ (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	long (A) $\leftarrow$ (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	long (A) $\leftarrow$ (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	long (A) $\leftarrow$ (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	long (A) $\leftarrow$ (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	long (A) $\leftarrow$ (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) $\leftarrow$ 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	byte (ear) $\leftarrow$ 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	word (A) $\leftarrow$ 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	word (ear) $\leftarrow$ 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ 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) $\leftarrow$ Shift left to the position where '1' is set for the first time. byte (R0) $\leftarrow$ 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) $\leftarrow$ (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ eam,rel	3+	*6	2	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*5	2	0	word (ear) $\leftarrow$ (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ eam,rel	3+	*6	2	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	20	0	8 $\times$ (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	16	0	6 $\times$ (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	17	0	6 $\times$ (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	20	0	8 $\times$ (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  $\times$  (b) + 2  $\times$  (c) when jumping to the next interruption request; 6  $\times$  (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:  $(\text{POP count}) \times (c)$  or  $(\text{PUSH count}) \times (c)$ \*5:  $(\text{POP count})$  or  $(\text{PUSH count})$ 

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

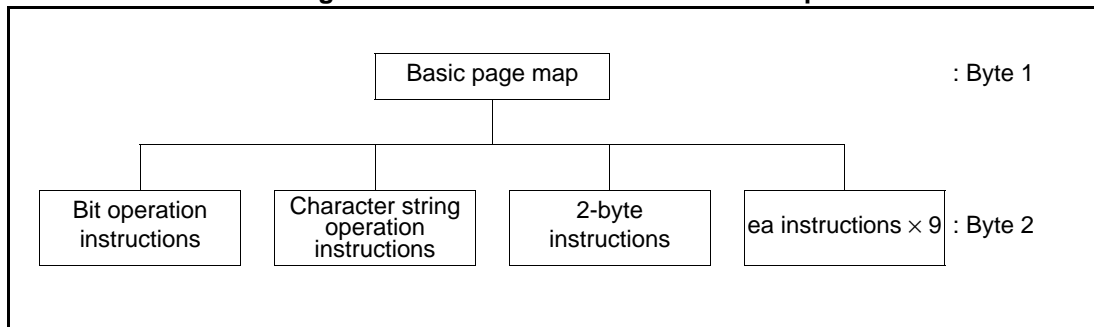
See Table B.5-1 and Table B.5-2 for information on (b) and (c) in the table.

## B.9 Instruction Map

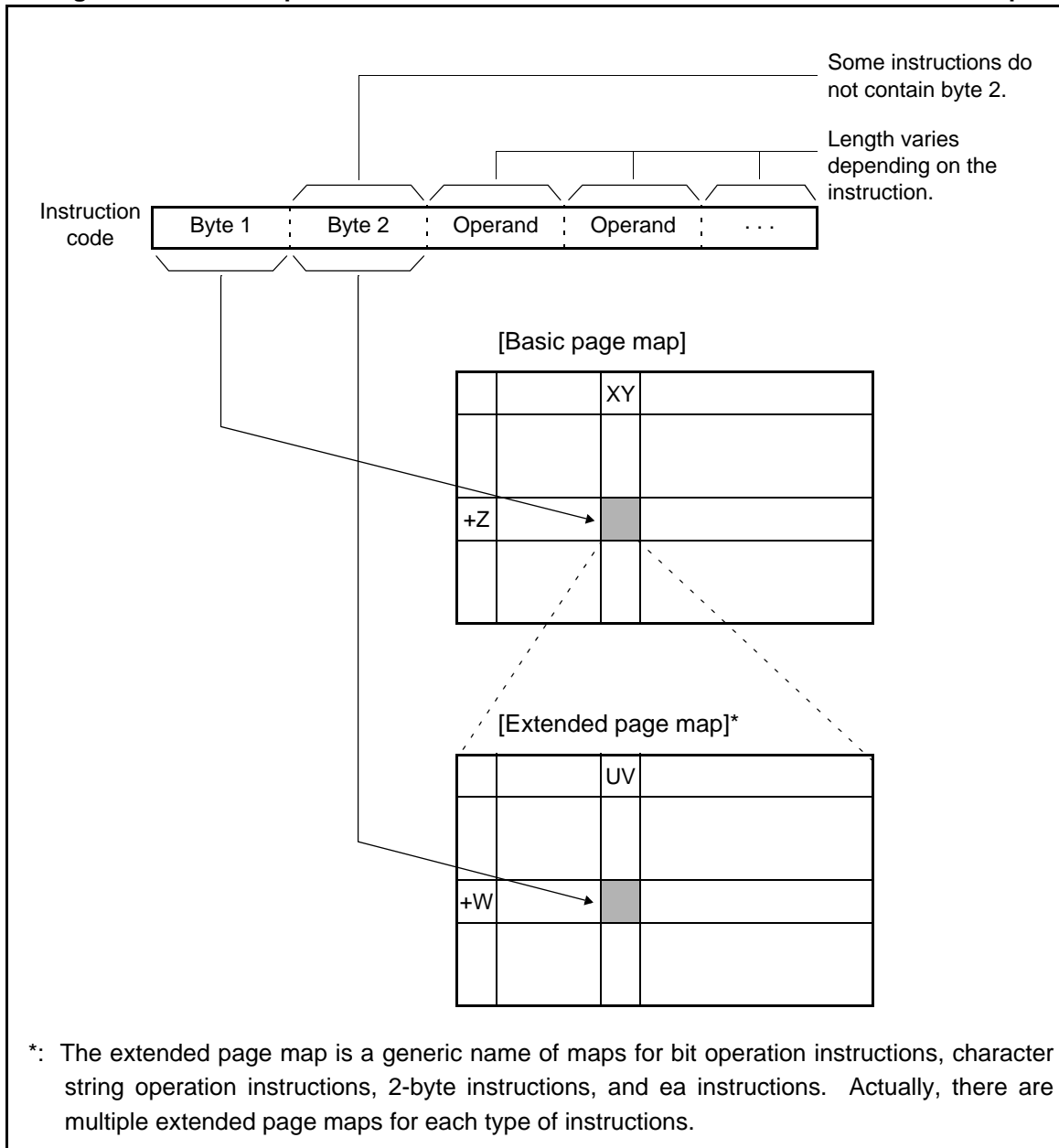
Each F<sup>2</sup>MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 to Table B.9-21 summarize the F<sup>2</sup>MC-16LX instruction map.

### ■ 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
@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	MOV A, Ri	MOVX A, @RWi+d8	MOV A, #4	CALL #4	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	MOVX A, @RWi+d8	MOVW @RWi+d8, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT	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	MOV Ri, ea								BLT rel
+B	NEGW A	CMPL A, #32	CMPL A	CMPL 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	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 rlist	POPW rlist	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 A, io:bp, A	MOVB io:bp, A	CLRB io:bp	CLRB io:bp	SETB io:bp	SETB io:bp	SETB addr16:bp	BBC io:bp, rel	BBC addr16:bp:rel	BBS io:bp, rel	BBS addr16:bp:rel	WBTS io:bp		WBTC io:bp	SBBS addr16:bp
+1																
+2																
+3																
+4																
+5																
+6																
+7																
+8	MOVB A, dir:bp	MOVB dir:bp, A	MOVB dir:bp, A	CLRB dir:bp	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				
+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	MOVSWI	MOVSWD					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB		FILSI 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 = 6F<sub>H</sub>)

	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		DIVU A									
+B																
+C	LSLW A, R0	LSL 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													

Table B.9-6 ea Instruction 1 (First Byte = 70<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CBNE ↓	CBNE ↓									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	ADDL A, A, @RW0	SUBL A, A, @RW0	SUBL A, A, @RW0	@RW0', @RW0+d16 #16, rel'	CMPL A, A, @RW0	CMPL A, A, @RW0	CMPL A, A, @RW0	ANDL A, A, @RW0	ANDL A, A, @RW0	ORL A, A, @RW0	ORL A, A, @RW0	XORL A, A, @RW0	XORL A, A, @RW0	@RW0', @RW0+d16 #8, rel'	@RW0', @RW0+d16 #8, rel'
+9	ADDL A, A, @RW1	ADDL A, A, @RW1	SUBL A, A, @RW1	SUBL A, A, @RW1	@RW1', @RW1+d16 #16, rel'	CMPL A, A, @RW1	CMPL A, A, @RW1	CMPL A, A, @RW1	ANDL A, A, @RW1	ANDL A, A, @RW1	ORL A, A, @RW1	ORL A, A, @RW1	XORL A, A, @RW1	XORL A, A, @RW1	@RW1', @RW1+d16 #8, rel'	@RW1', @RW1+d16 #8, rel'
+A	ADDL A, A, @RW2	ADDL A, A, @RW2	SUBL A, A, @RW2	SUBL A, A, @RW2	@RW2', @RW2+d16 #16, rel'	CMPL A, A, @RW2	CMPL A, A, @RW2	CMPL A, A, @RW2	ANDL A, A, @RW2	ANDL A, A, @RW2	ORL A, A, @RW2	ORL A, A, @RW2	XORL A, A, @RW2	XORL A, A, @RW2	@RW2', @RW2+d16 #8, rel'	@RW2', @RW2+d16 #8, rel'
+B	ADDL A, A, @RW3	ADDL A, A, @RW3	SUBL A, A, @RW3	SUBL A, A, @RW3	@RW3', @RW3+d16 #16, rel'	CMPL A, A, @RW3	CMPL A, A, @RW3	CMPL A, A, @RW3	ANDL A, A, @RW3	ANDL A, A, @RW3	ORL A, A, @RW3	ORL A, A, @RW3	XORL A, A, @RW3	XORL A, A, @RW3	@RW3', @RW3+d16 #8, rel'	@RW3', @RW3+d16 #8, rel'
+C	ADDL A, A, @RW0+	ADDL A, A, @RW0+	SUBL A, A, @RW0+	SUBL A, A, @RW0+	Use prohibited	CMPL A, A, @RW0+	CMPL A, A, @RW0+	CMPL A, A, @RW0+	ANDL A, A, @RW0+	ANDL A, A, @RW0+	ORL A, A, @RW0+	ORL A, A, @RW0+	XORL A, A, @RW0+	XORL A, A, @RW0+	Use prohibited	Use prohibited
+D	ADDL A, A, @RW1+	ADDL A, A, @RW1+	SUBL A, A, @RW1+	SUBL A, A, @RW1+	Use prohibited	CMPL A, A, @RW1+	CMPL A, A, @RW1+	CMPL A, A, @RW1+	ANDL A, A, @RW1+	ANDL A, A, @RW1+	ORL A, A, @RW1+	ORL A, A, @RW1+	XORL A, A, @RW1+	XORL A, A, @RW1+	Use prohibited	Use prohibited
+E	ADDL A, A, @RW2+	ADDL A, A, @RW2+	SUBL A, A, @RW2+	SUBL A, A, @RW2+	Use prohibited	CMPL A, A, @RW2+	CMPL A, A, @RW2+	CMPL A, A, @RW2+	ANDL A, A, @RW2+	ANDL A, A, @RW2+	ORL A, A, @RW2+	ORL A, A, @RW2+	XORL A, A, @RW2+	XORL A, A, @RW2+	Use prohibited	Use prohibited
+F	ADDL A, A, @RW3+	ADDL A, A, @RW3+	SUBL A, A, @RW3+	SUBL A, A, @RW3+	Use prohibited	CMPL A, A, @RW3+	CMPL A, A, @RW3+	CMPL A, A, @RW3+	ANDL A, A, @RW3+	ANDL A, A, @RW3+	ORL A, A, @RW3+	ORL A, A, @RW3+	XORL A, A, @RW3+	XORL A, A, @RW3+	Use prohibited	Use prohibited

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 @RLO', @RW0+d8	JMPP @RLO', @RW0+d8	CALLP @RLO', @RW0+d8	CALLP @RLO', @RW0+d8	INCL RLO', @RW0+d8	INCL RLO', @RW0+d8	DECL RLO', @RW0+d8	DECL RLO', @RW0+d8	MOVL A, RLO', @RW0+d8	MOVL A, RLO', @RW0+d8	MOVL RLO, A', @RW0+d8,A	MOVL RLO, A', @RW0+d8,A	MOV R0, #8', @RW0+d8,#8	MOV R0, #8', @RW0+d8,#8	MOVEA A, RWO', @RW0+d8	MOVEA A, RWO', @RW0+d8
+1	JMPP @RLO', @RW1+d8	JMPP @RLO', @RW1+d8	CALLP @RLO', @RW1+d8	CALLP @RLO', @RW1+d8	INCL RLO', @RW1+d8	INCL RLO', @RW1+d8	DECL RLO', @RW1+d8	DECL RLO', @RW1+d8	MOVL A, RLO', @RW1+d8	MOVL A, RLO', @RW1+d8	MOVL RLO, A', @RW1+d8,A	MOVL RLO, A', @RW1+d8,A	MOV R1, #8', @RW1+d8,#8	MOV R1, #8', @RW1+d8,#8	MOVEA A, RW1', @RW1+d8	MOVEA A, RW1', @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,A	MOVL RL1, A', @RW2+d8,A	MOV R2, #8', @RW2+d8,#8	MOV R2, #8', @RW2+d8,#8	MOVEA A, RW2', @RW2+d8	MOVEA A, RW2', @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,A	MOVL RL1, A', @RW3+d8,A	MOV R3, #8', @RW3+d8,#8	MOV R3, #8', @RW3+d8,#8	MOVEA A, RW3', @RW3+d8	MOVEA A, RW3', @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,A	MOVL RL2, A', @RW4+d8,A	MOV R4, #8', @RW4+d8,#8	MOV R4, #8', @RW4+d8,#8	MOVEA A, RW4', @RW4+d8	MOVEA A, RW4', @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,A	MOVL RL2, A', @RW5+d8,A	MOV R5, #8', @RW5+d8,#8	MOV R5, #8', @RW5+d8,#8	MOVEA A, RW5', @RW5+d8	MOVEA A, RW5', @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,A	MOVL RL3, A', @RW6+d8,A	MOV R6, #8', @RW6+d8,#8	MOV R6, #8', @RW6+d8,#8	MOVEA A, RW6', @RW6+d8	MOVEA A, RW6', @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,A	MOVL RL3, A', @RW7+d8,A	MOV R7, #8', @RW7+d8,#8	MOV R7, #8', @RW7+d8,#8	MOVEA A, RW7', @RW7+d8	MOVEA A, RW7', @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,A	MOVL @RW0,A', @RW0+d16,A	MOV @RW0, #8', @RW0+d16,#8	MOV @RW0, #8', @RW0+d16,#8	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,A	MOVL @RW1,A', @RW1+d16,A	MOV @RW1, #8', @RW1+d16,#8	MOV @RW1, #8', @RW1+d16,#8	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,A	MOVL @RW2,A', @RW2+d16,A	MOV @RW2, #8', @RW2+d16,#8	MOV @RW2, #8', @RW2+d16,#8	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,A	MOVL @RW3,A', @RW3+d16,A	MOV @RW3, #8', @RW3+d16,#8	MOV @RW3, #8', @RW3+d16,#8	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,A	MOVL @RW0+,A', @RW0+RW7,A	MOV @RW0+, #8', @RW0+RW7,#8	MOV @RW0+, #8', @RW0+RW7,#8	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,A	MOVL @RW1+,A', @RW1+RW7,A	MOV @RW1+, #8', @RW1+RW7,#8	MOV @RW1+, #8', @RW1+RW7,#8	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,A	MOVL @RW2+,A', @PC+d16,A	MOV @RW2+, #8', @PC+d16,#8	MOV @RW2+, #8', @PC+d16,#8	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,A	MOVL @RW3+,A', @addr16,A	MOV @RW3+, #8', @addr16,#8	MOV @RW3+, #8', @addr16,#8	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	ROLc R0, @RW0+d8	ROLc R0, @RW0+d8	RORc R0, @RW0+d8	RORc 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	ROLc R1, @RW1+d8	ROLc R1, @RW1+d8	RORc R1, @RW1+d8	RORc 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	ROLc R2, @RW2+d8	ROLc R2, @RW2+d8	RORc R2, @RW2+d8	RORc 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	ROLc R3, @RW3+d8	ROLc R3, @RW3+d8	RORc R3, @RW3+d8	RORc 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	ROLc R4, @RW4+d8	ROLc R4, @RW4+d8	RORc R4, @RW4+d8	RORc 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	ROLc R5, @RW5+d8	ROLc R5, @RW5+d8	RORc R5, @RW5+d8	RORc 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	ROLc R6, @RW6+d8	ROLc R6, @RW6+d8	RORc R6, @RW6+d8	RORc 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	ROLc R7, @RW7+d8	ROLc R7, @RW7+d8	RORc R7, @RW7+d8	RORc 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	ROLc @RW0, @RW0+d16	ROLc @RW0, @RW0+d16	RORc @RW0, @RW0+d16	RORc @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	ROLc @RW1, @RW1+d16	ROLc @RW1, @RW1+d16	RORc @RW1, @RW1+d16	RORc @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	ROLc @RW2, @RW2+d16	ROLc @RW2, @RW2+d16	RORc @RW2, @RW2+d16	RORc @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	ROLc @RW3, @RW3+d16	ROLc @RW3, @RW3+d16	RORc @RW3, @RW3+d16	RORc @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	ROLc @RW0+, @RW0+RW7	ROLc @RW0+, @RW0+RW7	RORc @RW0+, @RW0+RW7	RORc @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	ROLc @RW1+, @RW1+RW7	ROLc @RW1+, @RW1+RW7	RORc @RW1+, @RW1+RW7	RORc @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	ROLc @RW2+, @PC+d16	ROLc @RW2+, @PC+d16	RORc @RW2+, @PC+d16	RORc @RW2+, @PC+d16	INC @RW2+, @PC+d16	INC @RW2+, @PC+d16	DEC @RW2+, @PC+d16	DEC @RW2+, @PC+d16	MOV A, @RW2+, @PC+d16	MOV A, @RW2+, @PC+d16	MOV @RW2+, A, @PC+d16	MOV @RW2+, A, @PC+d16	MOVX A, @RW2+, @PC+d16	MOVX A, @RW2+, @PC+d16	XCH A, @RW2+, @PC+d16	XCH A, @RW2+, @PC+d16
+F	ROLc @RW3+, addr16	ROLc @RW3+, addr16	RORc @RW3+, addr16	RORc @RW3+, addr16	INC @RW3+, addr16	INC @RW3+, addr16	DEC @RW3+, addr16	DEC @RW3+, addr16	MOV A, @RW3+, addr16	MOV A, @RW3+, addr16	MOV @RW3+, A, addr16	MOV @RW3+, A, addr16	MOVX A, @RW3+, addr16	MOVX A, @RW3+, addr16	XCH A, @RW3+, addr16	XCH A, @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, #16	MOVW RW0, #16, @RW0+d8, #16	MOVW RW0, #16, @RW0+d8, #16	MOVW RW0, #16, @RW0+d8, #16	XCHW A, A, RW0, @RW0+d8	XCHW A, 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, #16	MOVW RW1, #16, @RW1+d8, #16	MOVW RW1, #16, @RW1+d8, #16	MOVW RW1, #16, @RW1+d8, #16	XCHW A, A, RW1, @RW1+d8	XCHW A, 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, #16	MOVW RW2, #16, @RW2+d8, #16	MOVW RW2, #16, @RW2+d8, #16	MOVW RW2, #16, @RW2+d8, #16	XCHW A, A, RW2, @RW2+d8	XCHW A, 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, #16	MOVW RW3, #16, @RW3+d8, #16	MOVW RW3, #16, @RW3+d8, #16	MOVW RW3, #16, @RW3+d8, #16	XCHW A, A, RW3, @RW3+d8	XCHW A, 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, #16	MOVW RW4, #16, @RW4+d8, #16	MOVW RW4, #16, @RW4+d8, #16	MOVW RW4, #16, @RW4+d8, #16	XCHW A, A, RW4, @RW4+d8	XCHW A, 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, #16	MOVW RW5, #16, @RW5+d8, #16	MOVW RW5, #16, @RW5+d8, #16	MOVW RW5, #16, @RW5+d8, #16	XCHW A, A, RW5, @RW5+d8	XCHW A, 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, #16	MOVW RW6, #16, @RW6+d8, #16	MOVW RW6, #16, @RW6+d8, #16	MOVW RW6, #16, @RW6+d8, #16	XCHW A, A, RW6, @RW6+d8	XCHW A, 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, #16	MOVW RW7, #16, @RW7+d8, #16	MOVW RW7, #16, @RW7+d8, #16	MOVW RW7, #16, @RW7+d8, #16	XCHW A, A, RW7, @RW7+d8	XCHW A, 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, #16	MOVW @RW0, #16, @RW0+d16, #16	MOVW @RW0, #16, @RW0+d16, #16	MOVW @RW0, #16, @RW0+d16, #16	XCHW A, A, @RW0, @RW0+d16	XCHW A, 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, #16	MOVW @RW1, #16, @RW1+d16, #16	MOVW @RW1, #16, @RW1+d16, #16	MOVW @RW1, #16, @RW1+d16, #16	XCHW A, A, @RW1, @RW1+d16	XCHW A, 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, #16	MOVW @RW2, #16, @RW2+d16, #16	MOVW @RW2, #16, @RW2+d16, #16	MOVW @RW2, #16, @RW2+d16, #16	XCHW A, A, @RW2, @RW2+d16	XCHW A, 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, #16	MOVW @RW3, #16, @RW3+d16, #16	MOVW @RW3, #16, @RW3+d16, #16	MOVW @RW3, #16, @RW3+d16, #16	XCHW A, A, @RW3, @RW3+d16	XCHW A, 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, #16	MOVW @RW0+, #16, @RW0+RW7, #16	MOVW @RW0+, #16, @RW0+RW7, #16	MOVW @RW0+, #16, @RW0+RW7, #16	XCHW A, A, @RW0+, @RW0+RW7	XCHW A, 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, #16	MOVW @RW1+, #16, @RW1+RW7, #16	MOVW @RW1+, #16, @RW1+RW7, #16	MOVW @RW1+, #16, @RW1+RW7, #16	XCHW A, A, @RW1+, @RW1+RW7	XCHW A, 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, #16	MOVW @RW2+, #16, @RW2+PC+d16, #16	MOVW @RW2+, #16, @RW2+PC+d16, #16	MOVW @RW2+, #16, @RW2+PC+d16, #16	XCHW A, A, @RW2+, @RW2+PC+d16	XCHW A, 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, #16	MOVW @RW3+, #16, @RW3+addr16, #16	MOVW @RW3+, #16, @RW3+addr16, #16	MOVW @RW3+, #16, @RW3+addr16, #16	XCHW A, A, @RW3+, @RW3+addr16	XCHW A, 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, r' RW0+d8, r	DBNZ @R0, r' RW0+d8, r
+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, r' RW1+d8, r	DBNZ @R1, r' RW1+d8, r
+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, r' RW2+d8, r	DBNZ @R2, r' RW2+d8, r
+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, r' RW3+d8, r	DBNZ @R3, r' RW3+d8, r
+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, r' RW4+d8, r	DBNZ @R4, r' RW4+d8, r
+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, r' RW5+d8, r	DBNZ @R5, r' RW5+d8, r
+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, r' RW6+d8, r	DBNZ @R6, r' RW6+d8, r
+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, r' RW7+d8, r	DBNZ @R7, r' RW7+d8, r
+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, r' W0+d16, r	DBNZ @RW0, r' W0+d16, r
+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, r' W1+d16, r	DBNZ @RW1, r' W1+d16, r
+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, r' W2+d16, r	DBNZ @RW2, r' W2+d16, r
+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, r' W3+d16, r	DBNZ @RW3, r' W3+d16, r
+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+, r' W0+RW7, r	DBNZ @RW0+, r' W0+RW7, r
+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+, r' W1+RW7, r	DBNZ @RW1+, r' W1+RW7, r
+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+, r' PC+d16, r	DBNZ @RW2+, r' PC+d16, r
+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+, r' addr16, r	DBNZ @RW3+, r' addr16, r

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 A, R0, @RW0+d8, A	SUBC A, R0, @RW0+d8, A	SUBC A, R0, @RW0+d8, A	NEG R0, @RW0+d8, A	NEG A, @RW0+d8, A	AND R0, A, @RW0+d8, A	AND R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	OR A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	NOT R0, @RW0+d8, A	NOT R0, @RW0+d8, A
+1	ADD R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB A, R1, @RW1+d8, A	SUBC A, R1, @RW1+d8, A	SUBC A, R1, @RW1+d8, A	NEG R1, @RW1+d8, A	NEG A, @RW1+d8, A	AND R1, A, @RW1+d8, A	AND R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	OR A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	NOT R1, @RW1+d8, A	NOT R1, @RW1+d8, A
+2	ADD R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB A, R2, @RW2+d8, A	SUBC A, R2, @RW2+d8, A	SUBC A, R2, @RW2+d8, A	NEG R2, @RW2+d8, A	NEG A, @RW2+d8, A	AND R2, A, @RW2+d8, A	AND R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	OR A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	NOT R2, @RW2+d8, A	NOT R2, @RW2+d8, A
+3	ADD R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB A, R3, @RW3+d8, A	SUBC A, R3, @RW3+d8, A	SUBC A, R3, @RW3+d8, A	NEG R3, @RW3+d8, A	NEG A, @RW3+d8, A	AND R3, A, @RW3+d8, A	AND R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	OR A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	NOT R3, @RW3+d8, A	NOT R3, @RW3+d8, A
+4	ADD R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB A, R4, @RW4+d8, A	SUBC A, R4, @RW4+d8, A	SUBC A, R4, @RW4+d8, A	NEG R4, @RW4+d8, A	NEG A, @RW4+d8, A	AND R4, A, @RW4+d8, A	AND R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	OR A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	NOT R4, @RW4+d8, A	NOT R4, @RW4+d8, A
+5	ADD R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB A, R5, @RW5+d8, A	SUBC A, R5, @RW5+d8, A	SUBC A, R5, @RW5+d8, A	NEG R5, @RW5+d8, A	NEG A, @RW5+d8, A	AND R5, A, @RW5+d8, A	AND R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	OR A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	NOT R5, @RW5+d8, A	NOT R5, @RW5+d8, A
+6	ADD R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB A, R6, @RW6+d8, A	SUBC A, R6, @RW6+d8, A	SUBC A, R6, @RW6+d8, A	NEG R6, @RW6+d8, A	NEG A, @RW6+d8, A	AND R6, A, @RW6+d8, A	AND R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	OR A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	NOT R6, @RW6+d8, A	NOT R6, @RW6+d8, A
+7	ADD R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB A, R7, @RW7+d8, A	SUBC A, R7, @RW7+d8, A	SUBC A, R7, @RW7+d8, A	NEG R7, @RW7+d8, A	NEG A, @RW7+d8, A	AND R7, A, @RW7+d8, A	AND R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	OR A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	NOT R7, @RW7+d8, A	NOT R7, @RW7+d8, A
+8	ADD @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB A, @RW0, @RW0+d16, A	SUBC A, @RW0, @RW0+d16, A	SUBC A, @RW0, @RW0+d16, A	NEG @RW0, @RW0+d16, A	NEG A, @RW0+d16, A	AND @RW0, A, @RW0+d16, A	AND @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	OR A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	NOT @RW0, @RW0+d16, A	NOT @RW0, @RW0+d16, A
+9	ADD @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB A, @RW1, @RW1+d16, A	SUBC A, @RW1, @RW1+d16, A	SUBC A, @RW1, @RW1+d16, A	NEG @RW1, @RW1+d16, A	NEG A, @RW1+d16, A	AND @RW1, A, @RW1+d16, A	AND @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	OR A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	NOT @RW1, @RW1+d16, A	NOT @RW1, @RW1+d16, A
+A	ADD @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB A, @RW2, @RW2+d16, A	SUBC A, @RW2, @RW2+d16, A	SUBC A, @RW2, @RW2+d16, A	NEG @RW2, @RW2+d16, A	NEG A, @RW2+d16, A	AND @RW2, A, @RW2+d16, A	AND @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	OR A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	NOT @RW2, @RW2+d16, A	NOT @RW2, @RW2+d16, A
+B	ADD @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB A, @RW3, @RW3+d16, A	SUBC A, @RW3, @RW3+d16, A	SUBC A, @RW3, @RW3+d16, A	NEG @RW3, @RW3+d16, A	NEG A, @RW3+d16, A	AND @RW3, A, @RW3+d16, A	AND @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	OR A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	NOT @RW3, @RW3+d16, A	NOT @RW3, @RW3+d16, A
+C	ADD @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB A, @RW0+, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7, A	NEG @RW0+, @RW0+RW7, A	NEG A, @RW0+RW7, A	AND @RW0+, A, @RW0+RW7, A	AND @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	OR A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	NOT @RW0+, @RW0+RW7, A	NOT @RW0+, @RW0+RW7, A
+D	ADD @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB A, @RW1+, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7, A	NEG @RW1+, @RW1+RW7, A	NEG A, @RW1+RW7, A	AND @RW1+, A, @RW1+RW7, A	AND @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	OR A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	NOT @RW1+, @RW1+RW7, A	NOT @RW1+, @RW1+RW7, A
+E	ADD @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB A, @RW2+, @PC+d16, A	SUBC A, @RW2+, @PC+d16, A	SUBC A, @RW2+, @PC+d16, A	NEG @RW2+, @PC+d16, A	NEG A, @PC+d16, A	AND @RW2+, A, @PC+d16, A	AND @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	OR A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	NOT @RW2+, @PC+d16, A	NOT @RW2+, @PC+d16, A
+F	ADD @RW3+, A, @addr16, A	SUB @RW3+, A, @addr16, A	SUB @RW3+, A, @addr16, A	SUB A, @RW3+, @addr16, A	SUBC A, @RW3+, @addr16, A	SUBC A, @RW3+, @addr16, A	NEG @RW3+, @addr16, A	NEG A, @addr16, A	AND @RW3+, A, @addr16, A	AND @RW3+, A, @addr16, A	OR @RW3+, A, @addr16, A	OR A, @addr16, A	XOR @RW3+, A, @addr16, A	XOR @RW3+, A, @addr16, A	NOT @RW3+, @addr16, A	NOT @RW3+, @addr16, A



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, A, RW0, @RW0+d8	ADDW A, A, RW0, @RW0+d8	SUBW A, A, RW0, @RW0+d8	SUBW A, A, RW0, @RW0+d8	ADDCW A, A, RW0, @RW0+d8	ADDCW A, A, RW0, @RW0+d8	CMPW A, A, RW0, @RW0+d8	CMPW A, A, RW0, @RW0+d8	ANDW A, A, RW0, @RW0+d8	ANDW A, A, RW0, @RW0+d8	ORW A, A, RW0, @RW0+d8	ORW A, A, RW0, @RW0+d8	XORW A, A, RW0, @RW0+d8	XORW A, A, RW0, @RW0+d8	DWBNZ RW0, r, @RW0+d8, r	DWBNZ RW0, r, @RW0+d8, r
+1	ADDW A, RW1, @RW1+d8	ADDW A, RW1, @RW1+d8	SUBW A, A, RW1, @RW1+d8	SUBW A, A, RW1, @RW1+d8	ADDCW A, A, RW1, @RW1+d8	ADDCW A, 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, r, @RW1+d8, r	DWBNZ RW1, r, @RW1+d8, r
+2	ADDW A, RW2, @RW2+d8	ADDW A, RW2, @RW2+d8	SUBW A, A, RW2, @RW2+d8	SUBW A, A, RW2, @RW2+d8	ADDCW A, A, RW2, @RW2+d8	ADDCW A, 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, r, @RW2+d8, r	DWBNZ RW2, r, @RW2+d8, r
+3	ADDW A, RW3, @RW3+d8	ADDW A, RW3, @RW3+d8	SUBW A, A, RW3, @RW3+d8	SUBW A, A, RW3, @RW3+d8	ADDCW A, A, RW3, @RW3+d8	ADDCW A, 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, r, @RW3+d8, r	DWBNZ RW3, r, @RW3+d8, r
+4	ADDW A, RW4, @RW4+d8	ADDW A, RW4, @RW4+d8	SUBW A, A, RW4, @RW4+d8	SUBW A, A, RW4, @RW4+d8	ADDCW A, A, RW4, @RW4+d8	ADDCW A, 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, r, @RW4+d8, r	DWBNZ RW4, r, @RW4+d8, r
+5	ADDW A, RW5, @RW5+d8	ADDW A, RW5, @RW5+d8	SUBW A, A, RW5, @RW5+d8	SUBW A, A, RW5, @RW5+d8	ADDCW A, A, RW5, @RW5+d8	ADDCW A, 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, r, @RW5+d8, r	DWBNZ RW5, r, @RW5+d8, r
+6	ADDW A, RW6, @RW6+d8	ADDW A, RW6, @RW6+d8	SUBW A, A, RW6, @RW6+d8	SUBW A, A, RW6, @RW6+d8	ADDCW A, A, RW6, @RW6+d8	ADDCW A, 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, r, @RW6+d8, r	DWBNZ RW6, r, @RW6+d8, r
+7	ADDW A, RW7, @RW7+d8	ADDW A, RW7, @RW7+d8	SUBW A, A, RW7, @RW7+d8	SUBW A, A, RW7, @RW7+d8	ADDCW A, A, RW7, @RW7+d8	ADDCW A, 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, r, @RW7+d8, r	DWBNZ RW7, r, @RW7+d8, r
+8	ADDW A, @RW0, @RW0+d16	ADDW A, @RW0, @RW0+d16	SUBW A, A, @RW0, @RW0+d16	SUBW A, A, @RW0, @RW0+d16	ADDCW A, A, @RW0, @RW0+d16	ADDCW A, A, @RW0, @RW0+d16	CMPW A, A, @RW0, @RW0+d16	CMPW A, A, @RW0, @RW0+d16	ANDW A, A, @RW0, @RW0+d16	ANDW A, A, @RW0, @RW0+d16	ORW A, A, @RW0, @RW0+d16	ORW A, A, @RW0, @RW0+d16	XORW A, A, @RW0, @RW0+d16	XORW A, A, @RW0, @RW0+d16	DWBNZ @RW0, r, @RW0+d16, r	DWBNZ @RW0, r, @RW0+d16, r
+9	ADDW A, @RW1, @RW1+d16	ADDW A, @RW1, @RW1+d16	SUBW A, A, @RW1, @RW1+d16	SUBW A, A, @RW1, @RW1+d16	ADDCW A, A, @RW1, @RW1+d16	ADDCW A, A, @RW1, @RW1+d16	CMPW A, A, @RW1, @RW1+d16	CMPW A, A, @RW1, @RW1+d16	ANDW A, A, @RW1, @RW1+d16	ANDW A, A, @RW1, @RW1+d16	ORW A, A, @RW1, @RW1+d16	ORW A, A, @RW1, @RW1+d16	XORW A, A, @RW1, @RW1+d16	XORW A, A, @RW1, @RW1+d16	DWBNZ @RW1, r, @RW1+d16, r	DWBNZ @RW1, r, @RW1+d16, r
+A	ADDW A, @RW2, @RW2+d16	ADDW A, @RW2, @RW2+d16	SUBW A, A, @RW2, @RW2+d16	SUBW A, A, @RW2, @RW2+d16	ADDCW A, A, @RW2, @RW2+d16	ADDCW A, A, @RW2, @RW2+d16	CMPW A, A, @RW2, @RW2+d16	CMPW A, A, @RW2, @RW2+d16	ANDW A, A, @RW2, @RW2+d16	ANDW A, A, @RW2, @RW2+d16	ORW A, A, @RW2, @RW2+d16	ORW A, A, @RW2, @RW2+d16	XORW A, A, @RW2, @RW2+d16	XORW A, A, @RW2, @RW2+d16	DWBNZ @RW2, r, @RW2+d16, r	DWBNZ @RW2, r, @RW2+d16, r
+B	ADDW A, @RW3, @RW3+d16	ADDW A, @RW3, @RW3+d16	SUBW A, A, @RW3, @RW3+d16	SUBW A, A, @RW3, @RW3+d16	ADDCW A, A, @RW3, @RW3+d16	ADDCW A, A, @RW3, @RW3+d16	CMPW A, A, @RW3, @RW3+d16	CMPW A, A, @RW3, @RW3+d16	ANDW A, A, @RW3, @RW3+d16	ANDW A, A, @RW3, @RW3+d16	ORW A, A, @RW3, @RW3+d16	ORW A, A, @RW3, @RW3+d16	XORW A, A, @RW3, @RW3+d16	XORW A, A, @RW3, @RW3+d16	DWBNZ @RW3, r, @RW3+d16, r	DWBNZ @RW3, r, @RW3+d16, r
+C	ADDW A, @RW0+, @RW0+RW7	ADDW A, @RW0+, @RW0+RW7	SUBW A, A, @RW0+, @RW0+RW7	SUBW A, A, @RW0+, @RW0+RW7	ADDCW A, A, @RW0+, @RW0+RW7	ADDCW A, A, @RW0+, @RW0+RW7	CMPW A, A, @RW0+, @RW0+RW7	CMPW A, A, @RW0+, @RW0+RW7	ANDW A, A, @RW0+, @RW0+RW7	ANDW A, A, @RW0+, @RW0+RW7	ORW A, A, @RW0+, @RW0+RW7	ORW A, A, @RW0+, @RW0+RW7	XORW A, A, @RW0+, @RW0+RW7	XORW A, A, @RW0+, @RW0+RW7	DWBNZ @RW0+, r, @RW0+RW7, r	DWBNZ @RW0+, r, @RW0+RW7, r
+D	ADDW A, @RW1+, @RW1+RW7	ADDW A, @RW1+, @RW1+RW7	SUBW A, A, @RW1+, @RW1+RW7	SUBW A, A, @RW1+, @RW1+RW7	ADDCW A, A, @RW1+, @RW1+RW7	ADDCW A, A, @RW1+, @RW1+RW7	CMPW A, A, @RW1+, @RW1+RW7	CMPW A, A, @RW1+, @RW1+RW7	ANDW A, A, @RW1+, @RW1+RW7	ANDW A, A, @RW1+, @RW1+RW7	ORW A, A, @RW1+, @RW1+RW7	ORW A, A, @RW1+, @RW1+RW7	XORW A, A, @RW1+, @RW1+RW7	XORW A, A, @RW1+, @RW1+RW7	DWBNZ @RW1+, r, @RW1+RW7, r	DWBNZ @RW1+, r, @RW1+RW7, r
+E	ADDW A, @RW2+, @PC+d16	ADDW A, @RW2+, @PC+d16	SUBW A, A, @RW2+, @PC+d16	SUBW A, A, @RW2+, @PC+d16	ADDCW A, A, @RW2+, @PC+d16	ADDCW A, A, @RW2+, @PC+d16	CMPW A, A, @RW2+, @PC+d16	CMPW A, A, @RW2+, @PC+d16	ANDW A, A, @RW2+, @PC+d16	ANDW A, A, @RW2+, @PC+d16	ORW A, A, @RW2+, @PC+d16	ORW A, A, @RW2+, @PC+d16	XORW A, A, @RW2+, @PC+d16	XORW A, A, @RW2+, @PC+d16	DWBNZ @RW2+, r, @PC+d16, r	DWBNZ @RW2+, r, @PC+d16, r
+F	ADDW A, @RW3+, addr 16	ADDW A, @RW3+, addr 16	SUBW A, A, @RW3+, addr 16	SUBW A, A, @RW3+, addr 16	ADDCW A, A, @RW3+, addr 16	ADDCW A, A, @RW3+, addr 16	CMPW A, A, @RW3+, addr 16	CMPW A, A, @RW3+, addr 16	ANDW A, A, @RW3+, addr 16	ANDW A, A, @RW3+, addr 16	ORW A, A, @RW3+, addr 16	ORW A, A, @RW3+, addr 16	XORW A, A, @RW3+, addr 16	XORW A, A, @RW3+, addr 16	DWBNZ @RW3+, r, addr 16, r	DWBNZ @RW3+, r, addr 16, r

Table B.9-13 ea Instruction 8 (First Byte = 77<sub>H</sub>)

	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 A, RW0, @RW0+d8	NEGW RW0, @RW0+d8	NEGW A, SUBCW A, RW0, @RW0+d8	NEGW A, SUBCW A, RW0, @RW0+d8	NEGW A, SUBCW A, RW0, @RW0+d8	NEGW A, SUBCW A, RW0, @RW0+d8	ANDW RW0, A, @RW0+d8, A	ANDW RW0, A, @RW0+d8, A	ORW RW0, A, @RW0+d8, A	ORW A, @RW0+d8, A	XORW RW0, A, @RW0+d8, A	XORW 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 A, RW1, @RW1+d8	NEGW RW1, @RW1+d8	NEGW A, SUBCW A, RW1, @RW1+d8	NEGW A, SUBCW A, RW1, @RW1+d8	NEGW A, SUBCW A, RW1, @RW1+d8	NEGW A, SUBCW A, RW1, @RW1+d8	ANDW RW1, A, @RW1+d8, A	ANDW RW1, A, @RW1+d8, A	ORW RW1, A, @RW1+d8, A	ORW A, @RW1+d8, A	XORW RW1, A, @RW1+d8, A	XORW 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 A, RW2, @RW2+d8	NEGW RW2, @RW2+d8	NEGW A, SUBCW A, RW2, @RW2+d8	NEGW A, SUBCW A, RW2, @RW2+d8	NEGW A, SUBCW A, RW2, @RW2+d8	NEGW A, SUBCW A, RW2, @RW2+d8	ANDW RW2, A, @RW2+d8, A	ANDW RW2, A, @RW2+d8, A	ORW RW2, A, @RW2+d8, A	ORW A, @RW2+d8, A	XORW RW2, A, @RW2+d8, A	XORW 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 A, RW3, @RW3+d8	NEGW RW3, @RW3+d8	NEGW A, SUBCW A, RW3, @RW3+d8	NEGW A, SUBCW A, RW3, @RW3+d8	NEGW A, SUBCW A, RW3, @RW3+d8	NEGW A, SUBCW A, RW3, @RW3+d8	ANDW RW3, A, @RW3+d8, A	ANDW RW3, A, @RW3+d8, A	ORW RW3, A, @RW3+d8, A	ORW A, @RW3+d8, A	XORW RW3, A, @RW3+d8, A	XORW 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 A, RW4, @RW4+d8	NEGW RW4, @RW4+d8	NEGW A, SUBCW A, RW4, @RW4+d8	NEGW A, SUBCW A, RW4, @RW4+d8	NEGW A, SUBCW A, RW4, @RW4+d8	NEGW A, SUBCW A, RW4, @RW4+d8	ANDW RW4, A, @RW4+d8, A	ANDW RW4, A, @RW4+d8, A	ORW RW4, A, @RW4+d8, A	ORW A, @RW4+d8, A	XORW RW4, A, @RW4+d8, A	XORW 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 A, RW5, @RW5+d8	NEGW RW5, @RW5+d8	NEGW A, SUBCW A, RW5, @RW5+d8	NEGW A, SUBCW A, RW5, @RW5+d8	NEGW A, SUBCW A, RW5, @RW5+d8	NEGW A, SUBCW A, RW5, @RW5+d8	ANDW RW5, A, @RW5+d8, A	ANDW RW5, A, @RW5+d8, A	ORW RW5, A, @RW5+d8, A	ORW A, @RW5+d8, A	XORW RW5, A, @RW5+d8, A	XORW 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 A, RW6, @RW6+d8	NEGW RW6, @RW6+d8	NEGW A, SUBCW A, RW6, @RW6+d8	NEGW A, SUBCW A, RW6, @RW6+d8	NEGW A, SUBCW A, RW6, @RW6+d8	NEGW A, SUBCW A, RW6, @RW6+d8	ANDW RW6, A, @RW6+d8, A	ANDW RW6, A, @RW6+d8, A	ORW RW6, A, @RW6+d8, A	ORW A, @RW6+d8, A	XORW RW6, A, @RW6+d8, A	XORW 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 A, RW7, @RW7+d8	NEGW RW7, @RW7+d8	NEGW A, SUBCW A, RW7, @RW7+d8	NEGW A, SUBCW A, RW7, @RW7+d8	NEGW A, SUBCW A, RW7, @RW7+d8	NEGW A, SUBCW A, RW7, @RW7+d8	ANDW RW7, A, @RW7+d8, A	ANDW RW7, A, @RW7+d8, A	ORW RW7, A, @RW7+d8, A	ORW A, @RW7+d8, A	XORW RW7, A, @RW7+d8, A	XORW 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 A, @RW0, @RW0+d16	NEGW @RW0, @RW0+d16	NEGW A, SUBCW A, @RW0, @RW0+d16	NEGW A, SUBCW A, @RW0, @RW0+d16	NEGW A, SUBCW A, @RW0, @RW0+d16	NEGW A, SUBCW A, @RW0, @RW0+d16	ANDW @RW0, A, @RW0+d16, A	ANDW @RW0, A, @RW0+d16, A	ORW @RW0, A, @RW0+d16, A	ORW A, @RW0+d16, A	XORW @RW0, A, @RW0+d16, A	XORW 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 A, @RW1, @RW1+d16	NEGW @RW1, @RW1+d16	NEGW A, SUBCW A, @RW1, @RW1+d16	NEGW A, SUBCW A, @RW1, @RW1+d16	NEGW A, SUBCW A, @RW1, @RW1+d16	NEGW A, SUBCW A, @RW1, @RW1+d16	ANDW @RW1, A, @RW1+d16, A	ANDW @RW1, A, @RW1+d16, A	ORW @RW1, A, @RW1+d16, A	ORW A, @RW1+d16, A	XORW @RW1, A, @RW1+d16, A	XORW 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 A, @RW2, @RW2+d16	NEGW @RW2, @RW2+d16	NEGW A, SUBCW A, @RW2, @RW2+d16	NEGW A, SUBCW A, @RW2, @RW2+d16	NEGW A, SUBCW A, @RW2, @RW2+d16	NEGW A, SUBCW A, @RW2, @RW2+d16	ANDW @RW2, A, @RW2+d16, A	ANDW @RW2, A, @RW2+d16, A	ORW @RW2, A, @RW2+d16, A	ORW A, @RW2+d16, A	XORW @RW2, A, @RW2+d16, A	XORW 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 A, @RW3, @RW3+d16	NEGW @RW3, @RW3+d16	NEGW A, SUBCW A, @RW3, @RW3+d16	NEGW A, SUBCW A, @RW3, @RW3+d16	NEGW A, SUBCW A, @RW3, @RW3+d16	NEGW A, SUBCW A, @RW3, @RW3+d16	ANDW @RW3, A, @RW3+d16, A	ANDW @RW3, A, @RW3+d16, A	ORW @RW3, A, @RW3+d16, A	ORW A, @RW3+d16, A	XORW @RW3, A, @RW3+d16, A	XORW A, @RW3+d16, A	NOTW @RW3, @RW3+d16	NOTW @RW3, @RW3+d16
+C	ADDW @RW0+, A, @RW0+RW7, A	SUBW @RW0+, A, @RW0+RW7, A	SUBW A, @RW0+, @RW0+RW7	NEGW @RW0+, @RW0+RW7	NEGW A, SUBCW A, @RW0+, @RW0+RW7	NEGW A, SUBCW A, @RW0+, @RW0+RW7	NEGW A, SUBCW A, @RW0+, @RW0+RW7	NEGW A, SUBCW A, @RW0+, @RW0+RW7	ANDW @RW0+, A, @RW0+RW7, A	ANDW @RW0+, A, @RW0+RW7, A	ORW @RW0+, A, @RW0+RW7, A	ORW A, @RW0+RW7, A	XORW @RW0+, A, @RW0+RW7, A	XORW A, @RW0+RW7, A	NOTW @RW0+, @RW0+RW7	NOTW @RW0+, @RW0+RW7
+D	ADDW @RW1+, A, @RW1+RW7, A	SUBW @RW1+, A, @RW1+RW7, A	SUBW A, @RW1+, @RW1+RW7	NEGW @RW1+, @RW1+RW7	NEGW A, SUBCW A, @RW1+, @RW1+RW7	NEGW A, SUBCW A, @RW1+, @RW1+RW7	NEGW A, SUBCW A, @RW1+, @RW1+RW7	NEGW A, SUBCW A, @RW1+, @RW1+RW7	ANDW @RW1+, A, @RW1+RW7, A	ANDW @RW1+, A, @RW1+RW7, A	ORW @RW1+, A, @RW1+RW7, A	ORW A, @RW1+RW7, A	XORW @RW1+, A, @RW1+RW7, A	XORW A, @RW1+RW7, A	NOTW @RW1+, @RW1+RW7	NOTW @RW1+, @RW1+RW7
+E	ADDW @RW2+, A, @PC+d16, A	SUBW @RW2+, A, @PC+d16, A	SUBW A, @RW2+, @PC+d16	NEGW @RW2+, @PC+d16	NEGW A, SUBCW A, @RW2+, @PC+d16	NEGW A, SUBCW A, @RW2+, @PC+d16	NEGW A, SUBCW A, @RW2+, @PC+d16	NEGW A, SUBCW A, @RW2+, @PC+d16	ANDW @RW2+, A, @PC+d16, A	ANDW @RW2+, A, @PC+d16, A	ORW @RW2+, A, @PC+d16, A	ORW A, @PC+d16, A	XORW @RW2+, A, @PC+d16, A	XORW A, @PC+d16, A	NOTW @RW2+, @PC+d16	NOTW @RW2+, @PC+d16
+F	ADDW @RW3+, A, addr16, A	SUBW @RW3+, A, addr16, A	SUBW A, @RW3+, addr16, A	NEGW @RW3+, addr16, A	NEGW A, SUBCW A, @RW3+, addr16, A	NEGW A, SUBCW A, @RW3+, addr16, A	NEGW A, SUBCW A, @RW3+, addr16, A	NEGW A, SUBCW A, @RW3+, addr16, A	ANDW @RW3+, A, addr16, A	ANDW @RW3+, A, addr16, A	ORW @RW3+, A, addr16, A	ORW A, addr16, A	XORW @RW3+, A, addr16, A	XORW A, addr16, A	NOTW @RW3+, addr16, A	NOTW @RW3+, addr16, A

### Table B.9-14 ea Instruction 9 (First Byte = 78<sub>H</sub>)

	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	MUL A, R0' @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	MUL A, R1' @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	MUL A, R2' @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	MUL A, R3' @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	MUL A, R4' @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	MUL A, R5' @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	MUL A, R6' @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	MUL A, R7' @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, R0' @RW0+d16	MULU A, R0' @RW0+d16	MULUW A, RW0' @RW0+d16	MULUW A, RW0' @RW0+d16	MUL A, R0' @RW0+d16	MUL A, R0' @RW0+d16	MULW A, RW0' @RW0+d16	MULW A, RW0' @RW0+d16	DIVU A, R0' @RW0+d16	DIVU A, R0' @RW0+d16	DIVUW A, RW0' @RW0+d16	DIVUW A, RW0' @RW0+d16	DIV A, R0' @RW0+d16	DIV A, R0' @RW0+d16	DIVW A, RW0' @RW0+d16	DIVW A, RW0' @RW0+d16
+9	MULU A, R0' @RW1+d16	MULU A, R0' @RW1+d16	MULUW A, RW1' @RW1+d16	MULUW A, RW1' @RW1+d16	MUL A, R0' @RW1+d16	MUL A, R0' @RW1+d16	MULW A, RW1' @RW1+d16	MULW A, RW1' @RW1+d16	DIVU A, R0' @RW1+d16	DIVU A, R0' @RW1+d16	DIVUW A, RW1' @RW1+d16	DIVUW A, RW1' @RW1+d16	DIV A, R0' @RW1+d16	DIV A, R0' @RW1+d16	DIVW A, RW1' @RW1+d16	DIVW A, RW1' @RW1+d16
+A	MULU A, R0' @RW2+d16	MULU A, R0' @RW2+d16	MULUW A, RW2' @RW2+d16	MULUW A, RW2' @RW2+d16	MUL A, R0' @RW2+d16	MUL A, R0' @RW2+d16	MULW A, RW2' @RW2+d16	MULW A, RW2' @RW2+d16	DIVU A, R0' @RW2+d16	DIVU A, R0' @RW2+d16	DIVUW A, RW2' @RW2+d16	DIVUW A, RW2' @RW2+d16	DIV A, R0' @RW2+d16	DIV A, R0' @RW2+d16	DIVW A, RW2' @RW2+d16	DIVW A, RW2' @RW2+d16
+B	MULU A, R0' @RW3+d16	MULU A, R0' @RW3+d16	MULUW A, RW3' @RW3+d16	MULUW A, RW3' @RW3+d16	MUL A, R0' @RW3+d16	MUL A, R0' @RW3+d16	MULW A, RW3' @RW3+d16	MULW A, RW3' @RW3+d16	DIVU A, R0' @RW3+d16	DIVU A, R0' @RW3+d16	DIVUW A, RW3' @RW3+d16	DIVUW A, RW3' @RW3+d16	DIV A, R0' @RW3+d16	DIV A, R0' @RW3+d16	DIVW A, RW3' @RW3+d16	DIVW A, RW3' @RW3+d16
+C	MULU A, R0' @RW0+R7	MULU A, R0' @RW0+R7	MULUW A, RW0' @RW0+R7	MULUW A, RW0' @RW0+R7	MUL A, R0' @RW0+R7	MUL A, R0' @RW0+R7	MULW A, RW0' @RW0+R7	MULW A, RW0' @RW0+R7	DIVU A, R0' @RW0+R7	DIVU A, R0' @RW0+R7	DIVUW A, RW0' @RW0+R7	DIVUW A, RW0' @RW0+R7	DIV A, R0' @RW0+R7	DIV A, R0' @RW0+R7	DIVW A, RW0' @RW0+R7	DIVW A, RW0' @RW0+R7
+D	MULU A, R0' @RW1+R7	MULU A, R0' @RW1+R7	MULUW A, RW1' @RW1+R7	MULUW A, RW1' @RW1+R7	MUL A, R0' @RW1+R7	MUL A, R0' @RW1+R7	MULW A, RW1' @RW1+R7	MULW A, RW1' @RW1+R7	DIVU A, R0' @RW1+R7	DIVU A, R0' @RW1+R7	DIVUW A, RW1' @RW1+R7	DIVUW A, RW1' @RW1+R7	DIV A, R0' @RW1+R7	DIV A, R0' @RW1+R7	DIVW A, RW1' @RW1+R7	DIVW A, RW1' @RW1+R7
+E	MULU A, R0' @PC+d16	MULU A, R0' @PC+d16	MULUW A, RW2' @PC+d16	MULUW A, RW2' @PC+d16	MUL A, R0' @RW2' @PC+d16	MUL A, R0' @RW2' @PC+d16	MULW A, RW2' @PC+d16	MULW A, RW2' @PC+d16	DIVU A, R0' @RW2' @PC+d16	DIVU A, R0' @RW2' @PC+d16	DIVUW A, RW2' @PC+d16	DIVUW A, RW2' @PC+d16	DIV A, R0' @RW2' @PC+d16	DIV A, R0' @RW2' @PC+d16	DIVW A, RW2' @PC+d16	DIVW A, RW2' @PC+d16
+F	MULU A, R0' @RW3+addr16	MULU A, R0' @RW3+addr16	MULUW A, RW3' @RW3+addr16	MULUW A, RW3' @RW3+addr16	MUL A, R0' @RW3' @RW3+addr16	MUL A, R0' @RW3' @RW3+addr16	MULW A, RW3' @RW3+addr16	MULW A, RW3' @RW3+addr16	DIVU A, R0' @RW3' @RW3+addr16	DIVU A, R0' @RW3' @RW3+addr16	DIVUW A, RW3' @RW3+addr16	DIVUW A, RW3' @RW3+addr16	DIV A, R0' @RW3' @RW3+addr16	DIV A, R0' @RW3' @RW3+addr16	DIVW A, RW3' @RW3+addr16	DIVW A, RW3' @RW3+addr16

### Table B.9-15 MOVEA RWi, ea Instruction (First Byte = 79<sub>H</sub>)

[illegible]

**Table B.9-16 MOV Ri, ea Instruction (First Byte = 7A<sub>H</sub>)**

[illegible]

**Table B.9-17 MOVW RWi, ea Instruction (First Byte = 7B<sub>H</sub>)**

[illegible]

**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	MOV	R0, R0	MOV	R0, R1	MOV	R0, R1	MOV	R0, R2	MOV	R0, R3	MOV	R0, R3	MOV	R0, R4	MOV	R0, R4	MOV	R0, R5	MOV	R0, R5	MOV	R0, R6	MOV	R0, R6	MOV	R0, R7	MOV	R0, R7	MOV	R0, R7
+1	MOV	R1, R0	MOV	R0, R1	MOV	R1, R1	MOV	R1, R1	MOV	R1, R2	MOV	R1, R3	MOV	R1, R3	MOV	R1, R4	MOV	R1, R4	MOV	R1, R5	MOV	R1, R5	MOV	R1, R6	MOV	R1, R6	MOV	R1, R7	MOV	R1, R7	MOV	R1, R7
+2	MOV	R2, R0	MOV	R0, R1	MOV	R2, R1	MOV	R2, R1	MOV	R2, R2	MOV	R2, R3	MOV	R2, R3	MOV	R2, R4	MOV	R2, R4	MOV	R2, R5	MOV	R2, R5	MOV	R2, R6	MOV	R2, R6	MOV	R2, R7	MOV	R2, R7	MOV	R2, R7
+3	MOV	R3, R0	MOV	R0, R1	MOV	R3, R1	MOV	R3, R1	MOV	R3, R2	MOV	R3, R3	MOV	R3, R3	MOV	R3, R4	MOV	R3, R4	MOV	R3, R5	MOV	R3, R5	MOV	R3, R6	MOV	R3, R6	MOV	R3, R7	MOV	R3, R7	MOV	R3, R7
+4	MOV	R4, R0	MOV	R0, R1	MOV	R4, R1	MOV	R4, R1	MOV	R4, R2	MOV	R4, R3	MOV	R4, R3	MOV	R4, R4	MOV	R4, R4	MOV	R4, R5	MOV	R4, R5	MOV	R4, R6	MOV	R4, R6	MOV	R4, R7	MOV	R4, R7	MOV	R4, R7
+5	MOV	R5, R0	MOV	R0, R1	MOV	R5, R1	MOV	R5, R1	MOV	R5, R2	MOV	R5, R3	MOV	R5, R3	MOV	R5, R4	MOV	R5, R4	MOV	R5, R5	MOV	R5, R5	MOV	R5, R6	MOV	R5, R6	MOV	R5, R7	MOV	R5, R7	MOV	R5, R7
+6	MOV	R6, R0	MOV	R0, R1	MOV	R6, R1	MOV	R6, R1	MOV	R6, R2	MOV	R6, R3	MOV	R6, R3	MOV	R6, R4	MOV	R6, R4	MOV	R6, R5	MOV	R6, R5	MOV	R6, R6	MOV	R6, R6	MOV	R6, R7	MOV	R6, R7	MOV	R6, R7
+7	MOV	R7, R0	MOV	R0, R1	MOV	R7, R1	MOV	R7, R1	MOV	R7, R2	MOV	R7, R3	MOV	R7, R3	MOV	R7, R4	MOV	R7, R4	MOV	R7, R5	MOV	R7, R5	MOV	R7, R6	MOV	R7, R6	MOV	R7, R7	MOV	R7, R7	MOV	R7, R7
+8	MOV	RW0, R0	MOV	R0, R1	MOV	RW0, R1	MOV	RW0, R1	MOV	RW0, R2	MOV	RW0, R3	MOV	RW0, R3	MOV	RW0, R4	MOV	RW0, R4	MOV	RW0, R5	MOV	RW0, R5	MOV	RW0, R6	MOV	RW0, R6	MOV	RW0, R7	MOV	RW0, R7	MOV	RW0, R7
+9	MOV	RW1, R0	MOV	R0, R1	MOV	RW1, R1	MOV	RW1, R1	MOV	RW1, R2	MOV	RW1, R3	MOV	RW1, R3	MOV	RW1, R4	MOV	RW1, R4	MOV	RW1, R5	MOV	RW1, R5	MOV	RW1, R6	MOV	RW1, R6	MOV	RW1, R7	MOV	RW1, R7	MOV	RW1, R7
+A	MOV	RW2, R0	MOV	R0, R1	MOV	RW2, R1	MOV	RW2, R1	MOV	RW2, R2	MOV	RW2, R3	MOV	RW2, R3	MOV	RW2, R4	MOV	RW2, R4	MOV	RW2, R5	MOV	RW2, R5	MOV	RW2, R6	MOV	RW2, R6	MOV	RW2, R7	MOV	RW2, R7	MOV	RW2, R7
+B	MOV	RW3, R0	MOV	R0, R1	MOV	RW3, R1	MOV	RW3, R1	MOV	RW3, R2	MOV	RW3, R3	MOV	RW3, R3	MOV	RW3, R4	MOV	RW3, R4	MOV	RW3, R5	MOV	RW3, R5	MOV	RW3, R6	MOV	RW3, R6	MOV	RW3, R7	MOV	RW3, R7	MOV	RW3, R7
+C	MOV	RW0+, R0	MOV	R0, R1	MOV	RW0+, R1	MOV	RW0+, R1	MOV	RW0+, R2	MOV	RW0+, R3	MOV	RW0+, R3	MOV	RW0+, R4	MOV	RW0+, R4	MOV	RW0+, R5	MOV	RW0+, R5	MOV	RW0+, R6	MOV	RW0+, R6	MOV	RW0+, R7	MOV	RW0+, R7	MOV	RW0+, R7
+D	MOV	RW1+, R0	MOV	R0, R1	MOV	RW1+, R1	MOV	RW1+, R1	MOV	RW1+, R2	MOV	RW1+, R3	MOV	RW1+, R3	MOV	RW1+, R4	MOV	RW1+, R4	MOV	RW1+, R5	MOV	RW1+, R5	MOV	RW1+, R6	MOV	RW1+, R6	MOV	RW1+, R7	MOV	RW1+, R7	MOV	RW1+, R7
+E	MOV	RW2+, R0	MOV	R0, R1	MOV	RW2+, R1	MOV	RW2+, R1	MOV	RW2+, R2	MOV	RW2+, R3	MOV	RW2+, R3	MOV	RW2+, R4	MOV	RW2+, R4	MOV	RW2+, R5	MOV	RW2+, R5	MOV	RW2+, R6	MOV	RW2+, R6	MOV	RW2+, R7	MOV	RW2+, R7	MOV	RW2+, R7
+F	MOV	RW3+, R0	MOV	R0, R1	MOV	RW3+, R1	MOV	RW3+, R1	MOV	RW3+, R2	MOV	RW3+, R3	MOV	RW3+, R3	MOV	RW3+, R4	MOV	RW3+, R4	MOV	RW3+, R5	MOV	RW3+, R5	MOV	RW3+, R6	MOV	RW3+, R6	MOV	RW3+, R7	MOV	RW3+, R7	MOV	RW3+, R7

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 R0, R0, @RW0+8, RW0	MOVW R0, RW1, @RW0+8, RW1	MOVW R0, RW2, @RW0+8, RW2	MOVW R0, RW3, @RW0+8, RW3	MOVW R0, RW4, @RW0+8, RW4	MOVW R0, RW5, @RW0+8, RW5	MOVW R0, RW6, @RW0+8, RW6	MOVW R0, RW7, @RW0+8, RW7	MOVW R0, RW8, @RW0+8, RW8	MOVW R0, RW9, @RW0+8, RW9	MOVW R0, RW10, @RW0+8, RW10	MOVW R0, RW11, @RW0+8, RW11	MOVW R0, RW12, @RW0+8, RW12	MOVW R0, RW13, @RW0+8, RW13	MOVW R0, RW14, @RW0+8, RW14	MOVW R0, RW15, @RW0+8, RW15
+1	MOVW RW1, RW0, @RW1+8, RW0	MOVW RW1, RW1, @RW1+8, RW1	MOVW RW1, RW2, @RW1+8, RW2	MOVW RW1, RW3, @RW1+8, RW3	MOVW RW1, RW4, @RW1+8, RW4	MOVW RW1, RW5, @RW1+8, RW5	MOVW RW1, RW6, @RW1+8, RW6	MOVW RW1, RW7, @RW1+8, RW7	MOVW RW1, RW8, @RW1+8, RW8	MOVW RW1, RW9, @RW1+8, RW9	MOVW RW1, RW10, @RW1+8, RW10	MOVW RW1, RW11, @RW1+8, RW11	MOVW RW1, RW12, @RW1+8, RW12	MOVW RW1, RW13, @RW1+8, RW13	MOVW RW1, RW14, @RW1+8, RW14	MOVW RW1, RW15, @RW1+8, RW15
+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, RW4, @RW2+8, RW4	MOVW RW2, RW5, @RW2+8, RW5	MOVW RW2, RW6, @RW2+8, RW6	MOVW RW2, RW7, @RW2+8, RW7	MOVW RW2, RW8, @RW2+8, RW8	MOVW RW2, RW9, @RW2+8, RW9	MOVW RW2, RW10, @RW2+8, RW10	MOVW RW2, RW11, @RW2+8, RW11	MOVW RW2, RW12, @RW2+8, RW12	MOVW RW2, RW13, @RW2+8, RW13	MOVW RW2, RW14, @RW2+8, RW14	MOVW RW2, RW15, @RW2+8, RW15
+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, RW5, @RW3+8, RW5	MOVW RW3, RW6, @RW3+8, RW6	MOVW RW3, RW7, @RW3+8, RW7	MOVW RW3, RW8, @RW3+8, RW8	MOVW RW3, RW9, @RW3+8, RW9	MOVW RW3, RW10, @RW3+8, RW10	MOVW RW3, RW11, @RW3+8, RW11	MOVW RW3, RW12, @RW3+8, RW12	MOVW RW3, RW13, @RW3+8, RW13	MOVW RW3, RW14, @RW3+8, RW14	MOVW RW3, RW15, @RW3+8, RW15
+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, RW6, @RW4+8, RW6	MOVW RW4, RW7, @RW4+8, RW7	MOVW RW4, RW8, @RW4+8, RW8	MOVW RW4, RW9, @RW4+8, RW9	MOVW RW4, RW10, @RW4+8, RW10	MOVW RW4, RW11, @RW4+8, RW11	MOVW RW4, RW12, @RW4+8, RW12	MOVW RW4, RW13, @RW4+8, RW13	MOVW RW4, RW14, @RW4+8, RW14	MOVW RW4, RW15, @RW4+8, RW15
+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, RW8, @RW5+8, RW8	MOVW RW5, RW9, @RW5+8, RW9	MOVW RW5, RW10, @RW5+8, RW10	MOVW RW5, RW11, @RW5+8, RW11	MOVW RW5, RW12, @RW5+8, RW12	MOVW RW5, RW13, @RW5+8, RW13	MOVW RW5, RW14, @RW5+8, RW14	MOVW RW5, RW15, @RW5+8, RW15
+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, RW8, @RW6+8, RW8	MOVW RW6, RW9, @RW6+8, RW9	MOVW RW6, RW10, @RW6+8, RW10	MOVW RW6, RW11, @RW6+8, RW11	MOVW RW6, RW12, @RW6+8, RW12	MOVW RW6, RW13, @RW6+8, RW13	MOVW RW6, RW14, @RW6+8, RW14	MOVW RW6, RW15, @RW6+8, RW15
+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, RW8, @RW7+8, RW8	MOVW RW7, RW9, @RW7+8, RW9	MOVW RW7, RW10, @RW7+8, RW10	MOVW RW7, RW11, @RW7+8, RW11	MOVW RW7, RW12, @RW7+8, RW12	MOVW RW7, RW13, @RW7+8, RW13	MOVW RW7, RW14, @RW7+8, RW14	MOVW RW7, RW15, @RW7+8, RW15
+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, RW8, +d16, RW8	MOVW @RW0, RW9, +d16, RW9	MOVW @RW0, RW10, +d16, RW10	MOVW @RW0, RW11, +d16, RW11	MOVW @RW0, RW12, +d16, RW12	MOVW @RW0, RW13, +d16, RW13	MOVW @RW0, RW14, +d16, RW14	MOVW @RW0, RW15, +d16, RW15
+9	MOVW @RW1, RW1, +d16, RW1	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, RW8, +d16, RW8	MOVW @RW1, RW9, +d16, RW9	MOVW @RW1, RW10, +d16, RW10	MOVW @RW1, RW11, +d16, RW11	MOVW @RW1, RW12, +d16, RW12	MOVW @RW1, RW13, +d16, RW13	MOVW @RW1, RW14, +d16, RW14	MOVW @RW1, RW15, +d16, RW15
+A	MOVW @RW2, RW2, +d16, RW2	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, RW8, +d16, RW8	MOVW @RW2, RW9, +d16, RW9	MOVW @RW2, RW10, +d16, RW10	MOVW @RW2, RW11, +d16, RW11	MOVW @RW2, RW12, +d16, RW12	MOVW @RW2, RW13, +d16, RW13	MOVW @RW2, RW14, +d16, RW14	MOVW @RW2, RW15, +d16, RW15
+B	MOVW @RW3, RW3, +d16, RW3	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, RW8, +d16, RW8	MOVW @RW3, RW9, +d16, RW9	MOVW @RW3, RW10, +d16, RW10	MOVW @RW3, RW11, +d16, RW11	MOVW @RW3, RW12, +d16, RW12	MOVW @RW3, RW13, +d16, RW13	MOVW @RW3, RW14, +d16, RW14	MOVW @RW3, RW15, +d16, RW15
+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+, RW8, +RW7, RW8	MOVW @RW0+, RW9, +RW7, RW9	MOVW @RW0+, RW10, +RW7, RW10	MOVW @RW0+, RW11, +RW7, RW11	MOVW @RW0+, RW12, +RW7, RW12	MOVW @RW0+, RW13, +RW7, RW13	MOVW @RW0+, RW14, +RW7, RW14	MOVW @RW0+, RW15, +RW7, RW15
+D	MOVW @RW1+, RW1, +RW7, RW1	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+, RW8, +RW7, RW8	MOVW @RW1+, RW9, +RW7, RW9	MOVW @RW1+, RW10, +RW7, RW10	MOVW @RW1+, RW11, +RW7, RW11	MOVW @RW1+, RW12, +RW7, RW12	MOVW @RW1+, RW13, +RW7, RW13	MOVW @RW1+, RW14, +RW7, RW14	MOVW @RW1+, RW15, +RW7, RW15
+E	MOVW @RW2+, RW2, +d16, RW2	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+, RW8, +d16, RW8	MOVW @RW2+, RW9, +d16, RW9	MOVW @RW2+, RW10, +d16, RW10	MOVW @RW2+, RW11, +d16, RW11	MOVW @RW2+, RW12, +d16, RW12	MOVW @RW2+, RW13, +d16, RW13	MOVW @RW2+, RW14, +d16, RW14	MOVW @RW2+, RW15, +d16, RW15
+F	MOVW @RW3+, RW3, +d16, RW3	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+, RW8, +d16, RW8	MOVW @RW3+, RW9, +d16, RW9	MOVW @RW3+, RW10, +d16, RW10	MOVW @RW3+, RW11, +d16, RW11	MOVW @RW3+, RW12, +d16, RW12	MOVW @RW3+, RW13, +d16, RW13	MOVW @RW3+, RW14, +d16, RW14	MOVW @RW3+, RW15, +d16, RW15



**Table B.9-20 XCH Ri, ea Instruction (First Byte = 7E<sub>H</sub>)**

[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

### 1024K Bit Flash Memory

Characteristics of the 512K/1024K Bit Flash Memory  
..... 588

### 16-bit Free-run Timer

Block Diagram of 16-bit Free-run Timer ..... 283

### 16-bit Free-run Timer

16-bit Free-run Timer ( $\times 1$ ) ..... 280  
16-bit Free-run Timer Interrupts ..... 320  
16-bit Free-run Timer Interrupts and EI<sup>2</sup>OS ..... 320  
16-bit Free-run Timer Registers ..... 289  
Sample Program for 16-bit Free-run Timer ..... 351  
Usage Notes on the 16-bit Free-run Timer ..... 349

### 16-bit Input Capture

16-bit Input Capture ( $\times 4$ ) ..... 281  
16-bit Input Capture Input Timing ..... 338  
16-bit Input Capture Interrupts ..... 322  
16-bit Input Capture Interrupts and EI<sup>2</sup>OS ..... 322  
16-bit Input Capture Operation ..... 337  
Block Diagram of 16-bit Input Capture ..... 284  
Usage Notes on the 16-bit Input Capture ..... 350

### 16-bit Output Compare

16-bit Output Compare ( $\times 6$ ) ..... 280  
16-bit Output Compare Interrupts ..... 321  
16-bit Output Compare Interrupts and EI<sup>2</sup>OS ..... 321  
16-bit Output Compare Operation ..... 332  
16-bit Output Compare Registers ..... 290  
16-bit Output Compare Timing ..... 336  
Block Diagram of 16-bit Output Compare ..... 284  
Sample Program for 16-bit Output Compare ..... 352  
Usage Notes on the 16-bit Output Compare ..... 349

### 16-bit PPG Timer

16-bit PPG Timer ( $\times 1$ ) ..... 281  
16-bit PPG Timer ( $\times 3$ , PPG1 is not present in  
MB90465 Series) ..... 258  
16-bit PPG Timer Interrupts ..... 271  
16-bit PPG Timer Interrupts and EI<sup>2</sup>OS ..... 272  
16-bit PPG Timer Pins ..... 260  
16-bit PPG Timer Registers ..... 262  
Block Diagram of 16-bit PPG Timer ..... 259  
Block Diagram of the 16-bit PPG Timer Pins ..... 260  
EI<sup>2</sup>OS Function of the 16-bit PPG Timer ..... 272  
Sample Program for the 16-bit PPG Timer ..... 277  
Usage Notes on the 16-bit PPG Timer ..... 276

### 16-bit Reload Register

16-bit Reload Register (TMRD0/TMRD1) ..... 242

### 16-bit Reload Timer

16-bit Reload Timer Interrupts ..... 243  
16-bit Reload Timer Interrupts and EI<sup>2</sup>OS ..... 243  
16-bit Reload Timer Interrupts and EI<sup>2</sup>OS ..... 232

16-bit Reload Timer Pins ..... 235

16-bit Reload Timer Registers ..... 236

16-bit Reload Timer Settings ..... 244

Baud Rates determined using the Internal Timer

(16-bit Reload Timer 0) ..... 495

Block Diagram of the 16-bit Reload Timer ..... 233

Block Diagram of the 16-bit Reload Timer Pins

..... 235

EI<sup>2</sup>OS Function of the 16-bit Reload Timer ..... 243

Overview of the 16-bit Reload Timer ..... 230

Usage Notes on the 16-bit Reload Timer ..... 252

### 16-bit Timer

16-bit Timer Buffer Operation Timing Diagram

..... 427

16-bit Timer in Multi-pulse Generator Operation

Diagram ..... 428

16-bit Timer Operation ..... 425

16-bit Timer Timing ..... 426

Block Diagram of 16-bit Timer ..... 363

PPG0 Output Pulse from Rising Edge of RT to 16-bit  
Timer Underflow (DTCR0/DTCR1/

DTCR2:TMD2 to TMD0=010<sub>B</sub>) ..... 342

The Use of the 16-bit Timer in Multi-pulse Generator

..... 428

Usage Notes on the 16-bit Timer ..... 430

### 16-bit Timer Buffer Operation Timing Diagram

16-bit Timer Buffer Operation Timing Diagram

..... 427

### 16-bit Timer Control Register

16-bit Timer Control Register (DTCR0/DTCR2)

..... 314

16-bit Timer Control Register (DTCR1) ..... 316

### 16-bit Timer Register

16-bit Timer Register (TMR0/TMR1) ..... 241

### 16-bit Timer Registers

16-bit Timer Registers (TMRR0/TMRR1/TMRR2)

..... 313

### 24-bit Operand

Linear Addressing by 24-bit Operand Specification

..... 34

### 512K

Characteristics of the 512K/1024K Bit Flash Memory  
..... 588

### 512K Bit Flash Memory

Programming Example Using 512K Bit Flash

Memory ..... 610

### 8/10-bit A/D Converter

8/10-bit A/D Converter Interrupts ..... 556

8/10-bit A/D Converter Interrupts and EI<sup>2</sup>OS

..... 543, 556

8/10-bit A/D Converter Pins.....	546
8/10-bit A/D Converter Registers .....	548
Block Diagram of the 8/10-bit A/D Converter.....	544
Block Diagrams of the 8/10-bit A/D Converter Pins .....	546
EI <sup>2</sup> OS Function of the 8/10-bit A/D Converter .....	556
Functions of the 8/10-bit A/D Converter .....	542
Usage Notes on the 8/10-bit A/D Converter.....	563
<b>A</b>	
A	
Accumulator (A) .....	42
A/D Control Status Register	
A/D Control Status Register 0 (ADCS0) .....	551
A/D Control Status Register 1 (ADCS1) .....	549
A/D Conversion	
A/D Conversion Data Protection Function .....	561
A/D Converter	
8/10-bit A/D Converter Interrupts .....	556
8/10-bit A/D Converter Interrupts and EI <sup>2</sup> OS .....	543, 556
8/10-bit A/D Converter Pins .....	546
8/10-bit A/D Converter Registers .....	548
Block Diagram of the 8/10-bit A/D Converter .....	544
Block Diagrams of the 8/10-bit A/D Converter Pins .....	546
EI <sup>2</sup> OS Function of the 8/10-bit A/D Converter .....	556
Functions of the 8/10-bit A/D Converter .....	542
Usage Notes on the 8/10-bit A/D Converter .....	563
A/D Data Register	
A/D Data Register (ADCR0/ADCR1) .....	554
Access Space	
Bank Registers and Access Space.....	35
Accumulator	
Accumulator (A) .....	42
ADB	
Bank Registers (PCB,DTB,USB,SSB,ADB) .....	54
Bank Select Prefixes (PCB,DTB,ADB,SPB) .....	58
ADCR	
A/D Data Register (ADCR0/ADCR1) .....	554
ADCS	
A/D Control Status Register 0 (ADCS0) .....	551
A/D Control Status Register 1 (ADCS1) .....	549
Addressing	
Addressing.....	637
Addressing by Indirect Specification with a 32-bit .....	34
Bank Addressing and Default Space .....	36
Direct Addressing .....	639
Indirect Addressing.....	645
Linear Addressing and Bank Addressing.....	33
Linear Addressing by 24-bit Operand Specification .....	34
Assignment	
DIP-64P-M01 Pin Assignment.....	10
FPT-64P-M06 Pin Assignment .....	8
FPT-64P-M09 Pin Assignment .....	9
Asynchronous Mode	
Operation in Asynchronous Mode .....	500

## INDEX

<b>B</b>	
Bank Addressing	
Bank Addressing and Default Space.....	36
Linear Addressing and Bank Addressing .....	33
Bank Registers	
Bank Registers (PCB,DTB,USB,SSB,ADB) .....	54
Bank Registers and Access Space .....	35
Bank Select Prefixes	
Bank Select Prefixes (PCB,DTB,ADB,SPB) .....	58
BAP	
Buffer Address Pointer (BAP) .....	143
Baud Rate	
UART Baud Rate Selection.....	490
Baud Rates	
Baud Rates determined using the Dedicated Baud Rate Generator .....	492
Baud Rates determined using the External Clock .....	497
Baud Rates determined using the Internal Timer (16-bit Reload Timer 0) .....	495
Bidirectional Communication	
Bidirectional Communication Function .....	504
Bit Flash Memory	
Characteristics of the 512K/1024K Bit Flash Memory .....	588
Block Diagram	
Block Diagram of 16-bit Free-run Timer .....	283
Block Diagram of 16-bit Input Capture.....	284
Block Diagram of 16-bit Output Compare .....	284
Block Diagram of 16-bit PPG Timer .....	259
Block Diagram of 16-bit Timer.....	363
Block Diagram of Data Write Control Unit.....	364
Block Diagram of Multi-functional Timer .....	282
Block Diagram of Multi-functional Timer Pins .....	287
Block Diagram of Multi-pulse Generator .....	359
Block Diagram of Multi-pulse Generator Pins .....	368
Block Diagram of Port 0 Pins .....	168
Block Diagram of Port 1 Pins .....	174
Block Diagram of Port 2 Pins .....	179
Block Diagram of Port 3 Pins .....	184
Block Diagram of Port 4 Pins .....	189
Block Diagram of Port 5 Pins .....	194
Block Diagram of Port 6 Pins .....	199
Block Diagram of Position Detection Circuit .....	366
Block Diagram of ROM Correction Function .....	573
Block Diagram of the 16-bit PPG Timer Pins .....	260
Block Diagram of the 16-bit Reload Timer .....	233
Block Diagram of the 16-bit Reload Timer Pins .....	235
Block Diagram of the 8/10-bit A/D Converter .....	544
Block Diagram of the Clock Generation Block .....	80
Block Diagram of the Delayed Interrupt Generator Module .....	536
Block Diagram of the DTP/external Interrupt Circuit .....	516
Block Diagram of the DTP/external Interrupt Circuit Pins .....	519
Block Diagram of the Low Power Consumption Control Circuit.....	92
Block Diagram of the PWC Timer Pins.....	436
Block Diagram of the Time-base Timer .....	208
Block Diagram of the Watchdog Timer.....	221
Block Diagram of UART .....	470
Block Diagram of UART Pins.....	474
Block Diagram of Waveform Generator .....	285
Block Diagram of Waveform Sequencer .....	360
DTT11 Circuit Block Diagram.....	421
MB90460/465 Series Block Diagram.....	7
Output Data Register Block Diagram.....	397
PWC Timer Block Diagram .....	435
ROM Mirroring Function Selection Module Block Diagram .....	584
Block Diagrams	
Block Diagrams of the 8/10-bit A/D Converter Pins .....	546
Block Diagrams of the External Reset Pin .....	69
Both Edges Detection	
Both Edges Detection and SN1x/RDAx Comparison Timing Diagram (CMPE=1) .....	400
Buffer Address Pointer	
Buffer Address Pointer (BAP).....	143
Bus Mode	
Bus Mode.....	158
Bus Mode Setting Bits .....	160
<b>C</b>	
Calculating	
Calculating the Execution Cycle Count .....	654
CCR	
Condition Code Register (CCR) Configuration .....	48
CDCR	
Communication Prescaler Control Register (CDCR) .....	484
Characteristics	
Characteristics of the 512K/1024K Bit Flash Memory .....	588
Chip	
When the Chip/Sector Deletion Operation is Executed. .....	595
When the Write Operation or Chip/Sector Deletion Operation is Executed.....	597, 598
Chip deletion	
Deleting the data (Chip deletion) .....	604

Circuit	
DTTI1 Circuit Block Diagram.....	421
Circuit Timing Diagram	
DTTI1 Circuit Timing Diagram (D1,D0=00 <sub>B</sub> )	
.....	422
CKSCR	
Configuration of the Clock Selection Register	
(CKSCR) .....	82
Clock	
Block Diagram of the Clock Generation Block	
.....	80
Clock .....	78
Clock Mode.....	91
Clock Mode Transition .....	84
Clock Supply Function .....	207, 213
Count Clock and Maximum Period .....	454
Event Count Mode (External Clock Mode).....	231
External Count Clock Selected .....	331
Internal Clock Mode.....	230
Internal Clock Mode (Single-shot Mode).....	248
Machine Clock .....	84
Operation in Internal Clock Mode (reload mode)	
.....	246
Selection of a PLL Clock Multiplier .....	84
Switching the Clock Mode .....	111
Clock Generation Block	
Block Diagram of the Clock Generation Block	
.....	80
Clock Mode	
Clock Mode.....	91
Clock Mode Transition .....	84
Switching the Clock Mode .....	111
Clock Selection Register	
Configuration of the Clock Selection Register	
(CKSCR) .....	82
Clock Supply	
Clock Supply Function .....	207, 213
Clock Supply Map	
Clock Supply Map .....	79
CMR	
Common register bank prefix (CMR).....	60
Command Sequence Table	
Command Sequence Table.....	592
Common register bank prefix	
Common register bank prefix (CMR).....	60
Communication	
Bidirectional Communication Function .....	504
Master-slave Communication Function .....	506
Communication Prescaler Control Register	
Communication Prescaler Control Register (CDCR)	
.....	484
Compare Clear Buffer	
Compare Clear Buffer .....	327
Compare Clear Buffer Register	
Compare Clear Buffer Register (CPCLRB) .....	293
Compare Clear Register	
Compare Clear Register (CPCLR).....	293
Compare Clear Register (CPCR).....	388
Compare Control Register	
Compare Control Register,Upper Byte	
(OCS1/OCS3/OCS5) .....	301
Compare Control Register,Lower Byte	
(OCS0/OCS2/OCS4) .....	303
Condition Code Register	
Condition Code Register (CCR) Configuration	
.....	48
Connection	
Example of Connection for Serial Writing (when	
Power Supplied by User) .....	618
Example of Connection for Serial Writing (when	
Power Supplied from Writer) .....	620
Example of Minimum Connection with Flash	
Microcontroller Programmer (when Power	
Supplied by User) .....	622
Example of Minimum Connection with Flash	
Microcontroller Programmer (when Power	
Supplied from Writer) .....	624
Continuous Conversion Mode	
Operation in Continuous Conversion Mode.....	557
Sample Program for Continuous Conversion Mode	
using EI <sup>2</sup> OS .....	566
Continuous Measurement Mode	
Single Measurement Mode and Continuous	
Measurement Mode .....	456
Control Circuit	
Block Diagram of the Low Power Consumption	
Control Circuit .....	92
Control Status Register	
Control Status Register (FMCS).....	590
Conversion	
Conversion using EI <sup>2</sup> OS.....	560
Conversion Mode	
Operation in Continuous Conversion Mode.....	557
Operation in Single Conversion Mode .....	557
Operation in Stop Conversion Mode.....	558
Sample Program for Continuous Conversion Mode	
using EI <sup>2</sup> OS .....	566
Sample Program for Single Conversion Mode using	
EI <sup>2</sup> OS .....	564
Sample Program for Stop Conversion Mode using	
EI <sup>2</sup> OS .....	568
Count Clock	
Count Clock and Maximum Period.....	454
Counter Operating State	
Counter Operating State .....	245
CPCLR	
Compare Clear Register (CPCLR).....	293



## INDEX

CPCLRB	
Compare Clear Buffer Register (CPCLRB).....	293
CPCR	
Compare Clear Register (CPCR).....	388
CPU	
CPU .....	28
CPU Intermittent Operation Mode .....	91
CPU Intermittent Operation Mode	
CPU Intermittent Operation Mode .....	91, 97
CPU Operating Modes	
CPU Operating Modes and Current Consumption .....	90
Current Consumption	
CPU Operating Modes and Current Consumption .....	90
<b>D</b>	
Data Counter	
Data Counter (DCT) .....	142
Data Protection	
A/D Conversion Data Protection Function .....	561
Data Write Control Unit	
Block Diagram of Data Write Control Unit .....	364
Operation of Data Write Control Unit.....	401
DCT	
Data Counter (DCT) .....	142
Dedicated Baud Rate	
Baud Rates determined using the Dedicated Baud Rate Generator .....	492
Dedicated Registers	
Configuration of Dedicated Registers .....	40
Dedicated Registers and General-purpose Registers .....	39
Default Space	
Bank Addressing and Default Space.....	36
Delayed Interrupt Generator Module	
Block Diagram of the Delayed Interrupt Generator Module.....	536
Operation of the Delayed Interrupt Generator Module .....	538
Delayed Interrupt Generator Module Register	
Delayed Interrupt Generator Module Register (DIRR) .....	537
Delayed Interrupt Request Latch	
Usage Notes on the Delayed Interrupt Request Latch .....	539
Delete	
Detailed Explanation on the Flash Memory Write/Delete .....	600
Deleting	
Deleting the data (Chip deletion) .....	604
Deleting the data (Sector deletion) .....	605
Procedure for Writing/Deleting the data to the Flash Memory .....	588
Procedure of Deleting a Sector .....	605
Deletion Operation	
When the Chip/Sector Deletion Operation is Executed. ....	595
When the Write Operation or Chip/Sector Deletion Operation is Executed.....	597, 598
Description	
Description of Instruction Presentation Items and Symbols .....	657
Descriptor	
Configuration of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	141
Devices	
Notes on Handling Devices.....	24
DIP-64P-M01	
DIP-64P-M01 Package Dimensions .....	11
DIP-64P-M01 Pin Assignment .....	10
Direct Addressing	
Direct Addressing .....	639
Direct Page Register	
Direct Page Register (DPR) .....	53
DIRR	
Delayed Interrupt Generator Module Register (DIRR) .....	537
DIV	
Division Rate Control Register (DIV0/DIV1) .....	444
Division Rate Control Register	
Division Rate Control Register (DIV0/DIV1) .....	444
DPR	
Direct Page Register (DPR) .....	53
DTB	
Bank Registers (PCB,DTB,USB,SSB,ADB).....	54
Bank Select Prefixes (PCB,DTB,ADB,SPB) .....	58
DTCR	
16-bit Timer Control Register (DTCR0/DTCR2) .....	314
16-bit Timer Control Register (DTCR1).....	316
DTP	
Operation of the DTP Function .....	529
Sample Program for the DTP Function.....	533
DTP/external Interrupt	
DTP/external Interrupt Functions .....	514
DTP/External Interrupt Circuit	
DTP/External Interrupt Circuit Registers .....	520
DTP/external Interrupt Circuit	
Block Diagram of the DTP/external Interrupt Circuit .....	516
Block Diagram of the DTP/external Interrupt Circuit Pins .....	519
DTP/external Interrupt Circuit Pins.....	518

Interrupt of the DTP/external Interrupt Circuit and EI <sup>2</sup> OS .....	515
Operation of the DTP/external Interrupt Circuit ..	526
Setting the DTP/external Interrupt Circuit .....	525
Usage Notes on the DTP/external Interrupt Circuit .....	530
DTP/interrupt Cause Register	
DTP/interrupt Cause Register (EIRR) .....	521
DTP/interrupt Enable Register	
DTP/interrupt Enable Register (ENIR) .....	522
DTTI	
DTTI0 Interrupt .....	348
DTTI0 Pin Input Operation .....	347
DTTI0 Pin Noise Cancellation Function .....	348
DTTI1 Circuit Block Diagram .....	421
DTTI1 Circuit Timing Diagram (D1,D0=00 <sub>B</sub> ) .....	422
Operation of DTTI1 Input Control .....	421
Relationship between DTTI1 and OPTx Output .....	423
<b>E</b>	
E <sup>2</sup> PROM	
E <sup>2</sup> PROM Memory Map .....	579
Edge Detection Timing Diagram	
Edge Detection Timing Diagram (CMPE=0) .....	399
Effective Address Field	
Effective Address Field .....	638, 656
EI <sup>2</sup> OS	
16-bit Reload Timer Interrupts and EI <sup>2</sup> OS .....	243
EI <sup>2</sup> OS Function of the 16-bit Reload Timer .....	243
EI <sup>2</sup> OS	
16-bit Free-run Timer Interrupts and EI <sup>2</sup> OS .....	320
16-bit Input Capture Interrupts and EI <sup>2</sup> OS .....	322
16-bit Output Compare Interrupts and EI <sup>2</sup> OS .....	321
16-bit PPG Timer Interrupts and EI <sup>2</sup> OS .....	272
16-bit Reload Timer Interrupts and EI <sup>2</sup> OS .....	232
8/10-bit A/D Converter Interrupts and EI <sup>2</sup> OS .....	543, 556
Configuration of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	141
Conversion using EI <sup>2</sup> OS .....	560
EI <sup>2</sup> OS Function of the 16-bit PPG Timer .....	272
EI <sup>2</sup> OS Function of the 8/10-bit A/D Converter .....	556
EI <sup>2</sup> OS Function of the Multi-functional Timer .....	323
EI <sup>2</sup> OS Function of the PWC Timer .....	446
Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	139
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	143
Interrupt of the DTP/external Interrupt Circuit and EI <sup>2</sup> OS .....	515
Multi-pulse Generator EI <sup>2</sup> OS Functions .....	396
Multi-pulse Generator Interrupts and EI <sup>2</sup> OS .....	396
Operation flow of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	145
Operation of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	140
Procedure for using the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	146
Processing Specifications of Sample Program for Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	153
Processing Time (one transfer time) of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	147
PWC Timer Interrupts and EI <sup>2</sup> OS .....	445
Sample Program for Continuous Conversion Mode using EI <sup>2</sup> OS .....	566
Sample Program for Single Conversion Mode using EI <sup>2</sup> OS .....	564
Sample Program for Stop Conversion Mode using EI <sup>2</sup> OS .....	568
Time-base Timer Interrupts and EI <sup>2</sup> OS .....	211
UART EI <sup>2</sup> OS Functions .....	487
UART Interrupt and EI <sup>2</sup> OS .....	469
UART Interrupts and EI <sup>2</sup> OS .....	487
Waveform Generator Interrupts and EI <sup>2</sup> OS .....	323
EIRR	
DTP/interrupt Cause Register (EIRR) .....	521
ELVR	
Request Level Setting Register (ELVR) .....	524
ENIR	
DTP/interrupt Enable Register (ENIR) .....	522
Event Count Mode	
Event Count Mode .....	250
Event Count Mode (External Clock Mode) .....	231
Sample Program in Event Count Mode .....	254
Exception Processing	
Exception Processing .....	149
Execution Cycle Count	
Calculating the Execution Cycle Count .....	654
Execution Cycle Count .....	653
Extended	
Processing Time (one transfer time) of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	147
Extended Intelligent I/O Service	
Configuration of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	141
Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	139
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	143
Operation flow of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	145
Operation of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	140
Procedure for using the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	146

## INDEX

Processing Specifications of Sample Program for Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	153
External Clock	
Baud Rates determined using the External Clock .....	497
Connection of an Oscillator or an External Clock to the Microcontroller .....	87
Event Count Mode (External Clock Mode) .....	231
External Count Clock	
External Count Clock Selected .....	331
External Interrupt	
External Interrupt Function .....	528
Sample Program for the External Interrupt Function .....	532
External Reset	
Block Diagrams of the External Reset Pin .....	69
<b>F</b>	
F <sup>2</sup> MC-16LX Instruction List	
F <sup>2</sup> MC-16LX Instruction List .....	660
Fetch	
Mode Fetch.....	72
Flag	
Hardware Sequence Flag.....	593
Flag Change Suppression Prefix	
Flag Change Suppression Prefix (NCC).....	61
Flash Memory	
Characteristics of the 512K/1024K Bit Flash Memory .....	588
Detailed Explanation on the Flash Memory Write/ Delete .....	600
Procedure for Writing/Deleting the data to the Flash Memory .....	588
Procedure of Writing the Data to the Flash Memory .....	602
Programming Example Using 512K Bit Flash Memory .....	610
Register on the Flash Memory .....	588
Flash Microcontroller Programmer	
Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied by User).....	622
Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied from Writer).....	624
Flash Security	
Flash Security Feature .....	609
Flowchart	
Flowchart of Pulse-width Measurement Operation .....	460
Flowchart of Timer Mode Operation .....	455

FMCS	
Control Status Register (FMCS) .....	590
FPT-64P-M06	
FPT-64P-M06 Package Dimensions .....	12
FPT-64P-M06 Pin Assignment.....	8
FPT-64P-M09	
FPT-64P-M09 Package Dimensions .....	13
FPT-64P-M09 Pin Assignment.....	9
Free-run Timer	
Block Diagram of 16-bit Free-run Timer.....	283
Free-run Timer	
16-bit Free-run Timer ( 1).....	280
16-bit Free-run Timer Interrupts .....	320
16-bit Free-run Timer Interrupts and EI <sup>2</sup> OS.....	320
16-bit Free-run Timer Registers.....	289
Sample Program for 16-bit Free-run Timer.....	351
Usage Notes on the 16-bit Free-run Timer .....	349
Fujitsu Standard	
Standard Configuration for Fujitsu Standard Serial On-board Writing.....	616
<b>G</b>	
GATE	
Output Condition of RTO0 to RTO5 and GATE .....	339
Gate	
Gate Triggered PPG0 Output .....	340
GATE Signal	
Generating GATE Signal during Each RTx is at "H" Level when GTENx is active (DTCR0/ DTCR1/DTCR2:TMD2 to TMD0=001 <sub>B</sub> or 111 <sub>B</sub> ) .....	340
Generating GATE Signal from Rising Edge of Each RTx until 16-bit Timer 0/1/2 Underflow when GTENx is active (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ) .....	341
Gate Trigger	
Gate Trigger (PPG channel 0 only) .....	275
General-purpose Register	
Configuration of a General-purpose Register .....	55
General-purpose Register Area and Register Bank Pointer .....	50
General-purpose Registers	
Dedicated Registers and General-purpose Registers .....	39
GTENx	
Generating GATE Signal during Each RTx is at "H" Level when GTENx is active (DTCR0/ DTCR1/DTCR2:TMD2 to TMD0=001 <sub>B</sub> or 111 <sub>B</sub> ) .....	340
Generating GATE Signal from Rising Edge of Each RTx until 16-bit Timer 0/1/2 Underflow when GTENx is active (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ) .....	341

**H****Handling Devices**

Notes on Handling Devices ..... 24

**Hardware Interrupt**

Hardware Interrupt ..... 126  
 Hardware Interrupt Activation ..... 129  
 Hardware Interrupt Operation ..... 130  
 Hardware Interrupt Processing Time ..... 135  
 Hardware Interrupt Structure ..... 127  
 Hardware Interrupt Suppression ..... 127  
 Procedure for using Hardware Interrupt ..... 132  
 Returning from a Hardware Interrupt ..... 129

**Hardware Sequence Flag**

Hardware Sequence Flag ..... 593

**hold**

Prefix Codes and Interrupt/hold Suppression  
 Instructions ..... 62

**I****I/O Area**

I/O Area ..... 30

**I/O Circuit**

I/O Circuit Types ..... 19

**I/O Map**

I/O Map ..... 628

**I/O Pins**

I/O Pins and Pin Functions ..... 14

**I/O Port**

I/O Port Functions ..... 164  
 Sample I/O Port Program ..... 203

**I/O Ports**

Registers for I/O Ports ..... 166

**I/O Register Address Pointer**

I/O Register Address Pointer (IOA) ..... 142

**I/O Service**

Processing Time (one transfer time) of the Extended  
 Intelligent I/O Service (EI<sup>2</sup>OS) ..... 147

**ICR**

Configuration of Interrupt Control Registers (ICR)  
 ..... 123  
 Interrupt Control Registers (ICR00 to ICR15)  
 ..... 121

**ICSH**

Input Capture Control Status Register, Upper Byte  
 (ICSH23) ..... 306

**ICSL**

Input Capture Control Status Register, Lower Byte  
 (ICSL23) ..... 307

**ILM**

Interrupt Level Mask Register (ILM) ..... 51

**Indirect Addressing**

Indirect Addressing ..... 645

**Indirect Specification**

Addressing by Indirect Specification with a 32-bit  
 ..... 34

**Initial State**

Initial State ..... 579

**Input Capture**

16-bit Input Capture ( 4) ..... 281  
 16-bit Input Capture Input Timing ..... 338  
 16-bit Input Capture Interrupts ..... 322  
 16-bit Input Capture Interrupts and EI<sup>2</sup>OS ..... 322  
 16-bit Input Capture Operation ..... 337  
 Block Diagram of 16-bit Input Capture ..... 284  
 Input Capture Registers ..... 291  
 Usage Notes on the 16-bit Input Capture ..... 350

**Input Capture Control Status Register**

Input Capture Control Status Register, Lower Byte  
 (ICSL23) ..... 307  
 Input Capture Control Status Register, Lower Byte  
 (PICSL01) ..... 311  
 Input Capture Control Status Register, Upper Byte  
 (ICSH23) ..... 306

**Input Capture Register**

Input Capture Register (IPCP0 to IPCP3) ..... 305

**Input Control**

Operation of DTTI1 Input Control ..... 421

**Input Control Lower Register**

Input Control Lower Register (IPCLR) ..... 386

**Input Control Upper Register**

Input Control Upper Register (IPCUR) ..... 384

**Input Data Register**

Input Data Register (SIDR0/SIDR1) ..... 482

**Instruction**

Description of Instruction Presentation Items and  
 Symbols ..... 657  
 F<sup>2</sup>MC-16LX Instruction List ..... 660  
 Instruction Types ..... 636  
 Structure of Instruction Map ..... 674

**Instruction Presentation Items and Symbols**

Description of Instruction Presentation Items and  
 Symbols ..... 657

**Instructions**

Prefix Codes and Interrupt/hold Suppression  
 Instructions ..... 62

**INT9**

INT9 Interrupt ..... 580

**Intelligent**

Processing Time (one transfer time) of the Extended  
 Intelligent I/O Service (EI<sup>2</sup>OS) ..... 147

**Internal Clock Mode**

Internal Clock Mode ..... 230  
 Internal Clock Mode (Single-shot Mode) ..... 248  
 Operation in Internal Clock Mode (reload mode)  
 ..... 246  
 Sample Program in Internal Clock Mode ..... 253

## INDEX

Internal Peripheral	
Internal Peripheral Features .....	3
Internal Timer	
Baud Rates determined using the Internal Timer (16-bit Reload Timer 0) .....	495
Interrupt	
DTTIO Interrupt .....	348
Hardware Interrupt .....	126
Hardware Interrupt Activation .....	129
Hardware Interrupt Operation .....	130
Hardware Interrupt Processing Time .....	135
Hardware Interrupt Structure .....	127
Hardware Interrupt Suppression .....	127
INT9 Interrupt .....	580
Interrupt Causes and Interrupt Vectors/interrupt Control Registers .....	117
Interrupt of the DTP/external Interrupt Circuit and EI <sup>2</sup> OS .....	515
Interrupt Operation .....	115
Interrupt Request Generation .....	453, 458
Interrupt Types and Functions .....	114
Multi-pulse Generator Interrupt Source .....	395
Prefix Codes and Interrupt/hold Suppression Instructions .....	62
Procedure for using Hardware Interrupt .....	132
Processing for Interrupt Operation .....	131
Reception Interrupt Generation and Flag Set Timing .....	488
Returning from a Hardware Interrupt .....	129
Returning from a Software Interrupt .....	137
Sample Programs for Interrupt Processing .....	152
Software Interrupt Activation .....	137
Software Interrupt Operation .....	138
Stack Operations at the Start of Interrupt Processing .....	150
Stack Operations on Return from Interrupt Processing .....	150
Transmission Interrupt Heneration and Flag Set Timing .....	489
UART Interrupt and EI <sup>2</sup> OS .....	469
Interrupt Causes	
Interrupt Causes and Interrupt Vectors/interrupt Control Registers .....	117
Interrupt Control Register	
Interrupt Control Register Functions .....	120, 123
Interrupt Control Registers	
Configuration of Interrupt Control Registers (ICR) .....	123
Interrupt Control Registers .....	119
Interrupt Control Registers (ICR00 to ICR15) .....	121
interrupt Control Registers	
Interrupt Causes and Interrupt Vectors/interrupt Control Registers .....	117
Interrupt Level Mask Register	
Interrupt Level Mask Register (ILM) .....	51
Interrupt Mask Function	
Interrupt Mask Function .....	330
Interrupt Request	
Interrupt Request Generation .....	453, 458
Interrupt Vectors	
Interrupt Causes and Interrupt Vectors/interrupt Control Registers .....	117
Interrupt Vectors .....	116
Interrupts	
16-bit Free-run Timer Interrupts .....	320
16-bit Free-run Timer Interrupts and EI <sup>2</sup> OS .....	320
16-bit Input Capture Interrupts .....	322
16-bit Input Capture Interrupts and EI <sup>2</sup> OS .....	322
16-bit Output Compare Interrupts .....	321
16-bit Output Compare Interrupts and EI <sup>2</sup> OS .....	321
16-bit PPG Timer Interrupts .....	271
16-bit PPG Timer Interrupts and EI <sup>2</sup> OS .....	272
16-bit Reload Timer Interrupts .....	243
16-bit Reload Timer Interrupts and EI <sup>2</sup> OS .....	243
16-bit Reload Timer Interrupts and EI <sup>2</sup> OS .....	232
8/10-bit A/D Converter Interrupts .....	556
8/10-bit A/D Converter Interrupts and EI <sup>2</sup> OS .....	543, 556
Multiple Interrupts .....	133
Multi-pulse Generator Interrupts and EI <sup>2</sup> OS .....	396
Multi-pulse Interrupts .....	394
PPG Interrupts .....	275
PWC Timer Interrupts .....	445
PWC Timer Interrupts and EI <sup>2</sup> OS .....	445
ROM Correction Interrupts .....	572
Time-base Timer Interrupts .....	211
Time-base Timer Interrupts and EI <sup>2</sup> OS .....	211
Timer Interrupts .....	329
UART Interrupts .....	486
UART Interrupts and EI <sup>2</sup> OS .....	487
Waveform Generator Interrupts .....	322
Waveform Generator Interrupts and EI <sup>2</sup> OS .....	323
Interval Timer	
Interval Timer Function .....	206
Operation of the Interval Timer Function (Time-base Timer) .....	212
IOA	
I/O Register Address Pointer (IOA) .....	142
IPCLR	
Input Control Lower Register (IPCLR) .....	386
IPCP	
Input Capture Register (IPCP0 to IPCP3) .....	305
IPCUR	
Input Control Upper Register (IPCUR) .....	384
ISCS	
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	143
ISD	
Configuration of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	141

## L

### Latch

Usage Notes on the Delayed Interrupt Request Latch .....	539
--	-----

### Linear Addressing

Linear Addressing and Bank Addressing .....	33
Linear Addressing by 24-bit Operand Specification .....	34

### Low Power Consumption

Block Diagram of the Low Power Consumption Control Circuit .....	92
--	----

### Low Power Consumption Mode

Low Power Consumption Mode Operating States .....	107
---	-----

### Low Power Consumption Mode Control Register

Access to the Low Power Consumption Mode Control Register .....	96
Low Power Consumption Mode Control Register (LPMCR) .....	94

### LPMCR

Low Power Consumption Mode Control Register (LPMCR) .....	94
---	----

## M

### Machine Clock

Machine Clock .....	84
---------------------	----

### Main Clock Mode

Main Clock Mode and PLL Clock Mode .....	84
--	----

### Mask

Interrupt Mask Function .....	330
-------------------------------	-----

### Master-slave Communication

Master-slave Communication Function .....	506
---	-----

### Maximum Period

Count Clock and Maximum Period .....	454
--------------------------------------	-----

### MB90460/465 Series

MB90460/465 Series Block Diagram .....	7
MB90460/465 Series Features .....	2
MB90460/465 Series Product Line-up .....	5

### MB90465 Series

16-bit PPG Timer (×3, PPG1 is not present in MB90465 Series) .....	258
PWC Timer (×2, PWC Timer 0 is not present in MB90465 Series) .....	434

### MD

Mode Pins (MD2 to MD0) .....	159
------------------------------	-----

### Measurement Mode

Measurement Mode and Measurement Operation .....	458
--	-----

### Measurement Operation

Measurement Mode and Measurement Operation .....	458
--	-----

### Measurement Range

Pulse Width/period Measurement Range .....	457
--	-----

### Measurement Result

Measurement Result Data .....	456
-------------------------------	-----

### Memory Map

E <sup>2</sup> PROM Memory Map .....	579
--------------------------------------	-----

### Memory Maps

Memory Maps .....	31
-------------------	----

### Memory Space

Memory Space .....	29
--------------------	----

### Microcontroller

Connection of an Oscillator or an External Clock to the Microcontroller .....	87
---	----

#### Example of Minimum Connection with Flash

Microcontroller Programmer (when Power Supplied by User) .....	622
--	-----

#### Example of Minimum Connection with Flash

Microcontroller Programmer (when Power Supplied from Writer) .....	624
--	-----

### Minimum Connection

#### Example of Minimum Connection with Flash

Microcontroller Programmer (when Power Supplied by User) .....	622
--	-----

#### Example of Minimum Connection with Flash

Microcontroller Programmer (when Power Supplied from Writer) .....	624
--	-----

### Minimum Input Pulse Width

Minimum Input Pulse Width .....	457
---------------------------------	-----

### Mode

Bus Mode .....	158
----------------	-----

Bus Mode Setting Bits .....	160
-----------------------------	-----

Clock Mode .....	91
------------------	----

Clock Mode Transition .....	84
-----------------------------	----

CPU Intermittent Operation Mode .....	91, 97
---------------------------------------	--------

Event Count Mode .....	250
------------------------	-----

Event Count Mode (External Clock Mode) .....	231
--	-----

Flowchart of Timer Mode Operation .....	455
---	-----

Internal Clock Mode .....	230
---------------------------	-----

Internal Clock Mode (Single-shot Mode) .....	248
--	-----

Main Clock Mode and PLL Clock Mode .....	84
--	----

Measurement Mode and Measurement Operation .....	458
--	-----

Mode Data .....	160
-----------------	-----

Mode Fetch .....	72
------------------	----

Mode Pins .....	71
-----------------	----

Mode Pins (MD2 to MD0) .....	159
------------------------------	-----

Mode Setting .....	158
--------------------	-----

Notes on Standby Mode .....	110
-----------------------------	-----

One-shot Operation Mode .....	453
-------------------------------	-----

Operating Status during Standby Mode .....	98
--	----

Operation in Asynchronous Mode .....	500
--------------------------------------	-----

Operation in Continuous Conversion Mode .....	557
---	-----

Operation in Internal Clock Mode (reload mode) .....	246
--	-----

Operation in Single Conversion Mode .....	557
---	-----

Operation in Stop Conversion Mode .....	558
---	-----

Operation in Synchronous Mode (Operation Mode 2) .....	502
--	-----

## INDEX

Operation Mode Selection .....	450
PWM Mode (PCNTL: MDSE=0).....	273
Relationship between Mode Pins and Mode Data .....	161
Release of Sleep Mode.....	99
Release of Stop Mode .....	104, 111
Release of Time-base Timer Mode.....	102, 111
Reload Operation Mode .....	453
RUN Mode .....	158
Sample Program for Continuous Conversion Mode using EI <sup>2</sup> OS .....	566
Sample Program for Single Conversion Mode using EI <sup>2</sup> OS .....	564
Sample Program for Stop Conversion Mode using EI <sup>2</sup> OS .....	568
Sample Program in Event Count Mode.....	254
Sample Program in Internal Clock Mode .....	253
Single Measurement Mode and Continuous Measurement Mode .....	456
Single-shot Mode (PCNTL: MDSE=1).....	274
Standby Mode.....	91
State of Pins in Single-chip Mode .....	109
Status of Pins after Mode Data is read .....	75
Switching the Clock Mode .....	111
Switching to Sleep Mode .....	99
Switching to Stop Mode .....	104
Switching to Time-base Timer Mode .....	102
Timer Mode .....	326
<b>Mode Data</b>	
Mode Data.....	160
Relationship between Mode Pins and Mode Data .....	161
Status of Pins after Mode Data is read .....	75
<b>Mode Fetch</b>	
Mode Fetch.....	72
<b>Mode Pins</b>	
Mode Pins .....	71
Mode Pins (MD2 to MD0) .....	159
Relationship between Mode Pins and Mode Data .....	161
<b>Modes</b>	
CPU Operating Modes and Current Consumption .....	90
Operating Modes.....	158
<b>Multi-byte Data</b>	
Multi-byte Data Access.....	38
Storage of Multi-byte Data in a Stack.....	38
Storage of Multi-byte Data in RAM .....	37
<b>Multi-byte Operand</b>	
Storage of Multi-byte Operand .....	37
<b>Multi-functional Timer</b>	
Block Diagram of Multi-functional Timer .....	282
Block Diagram of Multi-functional Timer Pins .....	287
EI <sup>2</sup> OS Function of the Multi-functional Timer .....	323
Multi-functional Timer Pins .....	286
Operation of Multi-functional Timer.....	324
<b>Multiple Interrupts</b>	
Multiple Interrupts .....	133
<b>Multiplier</b>	
Selection of a PLL Clock Multiplier .....	84
<b>Multi-pulse</b>	
Multi-pulse Interrupts .....	394
<b>Multi-pulse Generator</b>	
16-bit Timer in Multi-pulse Generator Operation Diagram .....	428
Block Diagram of Multi-pulse Generator .....	359
Block Diagram of Multi-pulse Generator Pins .....	368
Multi-pulse Generator EI <sup>2</sup> OS Functions .....	396
Multi-pulse Generator Interrupt Source .....	395
Multi-pulse Generator Interrupts and EI <sup>2</sup> OS .....	396
Pins of Multi-pulse Generator .....	367
Registers of Multi-pulse Generator .....	369
Sample Program for the Multi-pulse Generator .....	431
The Use of the 16-bit Timer in Multi-pulse Generator .....	428
<b>Multi-pulse Generator Operation Diagram</b>	
16-bit Timer in Multi-pulse Generator Operation Diagram .....	428
<b>N</b>	
<b>NCC</b>	
Flag Change Suppression Prefix (NCC) .....	61
<b>NCCR</b>	
Noise Cancellation Control Register (NCCR) .....	392
<b>Noise Cancellation</b>	
DTTIO Pin Noise Cancellation Function.....	348
Operation of Noise Cancellation Function .....	424
<b>Noise Cancellation Control Register</b>	
Noise Cancellation Control Register (NCCR) .....	392
<b>O</b>	
<b>OCCP</b>	
Output Compare Registers (OCCP0 to OCCP5) .....	300
<b>OCCPB</b>	
Output Compare Buffer Registers (OCCPB0 to OCCPB5) .....	299
<b>OCS</b>	
Compare Control Register,Upper Byte (OCS1/OCS3/OCS5).....	301
Compare Control Register,Lower Byte (OCS0/OCS2/OCS4).....	303
<b>One-shot Operation Mode</b>	
One-shot Operation Mode.....	453

One-shot Position Detection	
Timing Generated by One-shot Position Detection (OPS2 to OPS0=110 <sub>B</sub> ) .....	418
Timing Generated by One-shot Position Detection and Timer Underflow (OPS2 to OPS0 = 111 <sub>B</sub> ) .....	419
Timing Generated by One-shot Position Detection or Timer Underflow (OPS2 to OPS0 = 101 <sub>B</sub> ) .....	420
When One-shot Position Detection .....	418
When One-shot Position Detection and Timer Underflow .....	419
When One-shot Position Detection or Timer Underflow .....	420
OPCLR	
Output Control Lower Register (OPCLR).....	374
OPCUR	
Output Control Upper Register (OPCUR).....	372
OPDBR	
Output Data Buffer Lower Register (OPDBR) .....	382
Output Data Buffer Upper Register (OPDBR) .....	380
Signal Flow Diagram for OPDBR0 by Setting OPS2 to OPS0=000 <sub>B</sub> .....	401
Timing Generated by OPDBR0 Write (OPS2 to OPS0=000 <sub>B</sub> ) .....	408
OPDR	
OPDR Register Write Timing Diagram (OPS2 to OPS0=000 <sub>B</sub> ) .....	402
OPDR Register Write Timing Diagram (OPS2 to OPS0=001 <sub>B</sub> ,010 <sub>B</sub> ,011 <sub>B</sub> ,100 <sub>B</sub> ,101 <sub>B</sub> ,110 <sub>B</sub> , 111 <sub>B</sub> ) .....	404
Output Data Lower Register (OPDR).....	378
Output Data Register (OPDR) .....	398
Output Data Upper Register (OPDR) .....	376
OPDR Register Write Timing Diagram	
OPDR Register Write Timing Diagram (OPS2 to OPS0=000 <sub>B</sub> ) .....	402
OPDR Register Write Timing Diagram (OPS2 to OPS0=001 <sub>B</sub> ,010 <sub>B</sub> ,011 <sub>B</sub> ,100 <sub>B</sub> ,101 <sub>B</sub> ,110 <sub>B</sub> , 111 <sub>B</sub> ) .....	404
Operand	
Linear Addressing by 24-bit Operand Specification .....	34
Storage of Multi-byte Operand .....	37
Operating Modes	
Operating Modes .....	158
Operation Mode	
CPU Intermittent Operation Mode .....	91, 97
One-shot Operation Mode.....	453
Operation in Synchronous Mode (Operation Mode 2) .....	502
Operation Mode Selection.....	450
Reload Operation Mode.....	453
OPS	
OPDR Register Write Timing Diagram (OPS2 to OPS0=000 <sub>B</sub> ) .....	402
OPDR Register Write Timing Diagram (OPS2 to OPS0=001 <sub>B</sub> ,010 <sub>B</sub> ,011 <sub>B</sub> ,100 <sub>B</sub> ,101 <sub>B</sub> ,110 <sub>B</sub> , 111 <sub>B</sub> ).....	404
Signal Flow Diagram for Position Detection by Setting OPS2 to OPS0=010 <sub>B</sub> or 110 <sub>B</sub> .....	403
Signal Flow Diagram for Reload Timer 0 and Position Detection by Setting OPS2 to OPS0 = 011 <sub>B</sub> or 111 <sub>B</sub> .....	403
Signal Flow Diagram for Reload Timer 0 or Position Detection by Setting OPS2 to OPS0 = 100 <sub>B</sub> or 101 <sub>B</sub> .....	404
Signal Flow Diagram for Reload Timer 0 Underflow by Setting OPS2 to OPS0=001 <sub>B</sub> .....	402
Timing Generated by One-shot Position Detection (OPS2 to OPS0=110 <sub>B</sub> ) .....	418
Timing Generated by One-shot Position Detection and Timer Underflow (OPS2 to OPS0 = 111 <sub>B</sub> ) .....	419
Timing Generated by One-shot Position Detection or Timer Underflow (OPS2 to OPS0 = 101 <sub>B</sub> ) .....	420
Timing Generated by OPDBR0 Write (OPS2 to OPS0=000 <sub>B</sub> ) .....	408
Timing Generated by Position Detection (OPS2 to OPS0=010 <sub>B</sub> ) .....	412
Timing Generated by Position Detection and Timer Underflow (OPS2 to OPS0=011 <sub>B</sub> ).....	415
Timing Generated by Position Detection or Timer Underflow (OPS2 to OPS0=100 <sub>B</sub> ).....	417
Timing Generated by Reload Timer Underflow (OPS2 to OPS0=001 <sub>B</sub> ) .....	410
OPTx	
OPTx Output Waveform Timing Diagram (WTS1,WTS0=00 <sub>B</sub> ).....	398
Relationship between DTTI1 and OPTx Output .....	423
OPTx Output Waveform Timing Diagram	
OPTx Output Waveform Timing Diagram (WTS1,WTS0=00 <sub>B</sub> ).....	398
Oscillation Stabilization Time	
Oscillation Stabilization Time Timer Function .....	213
Oscillation Stabilization Time Timer	
Oscillation Stabilization Time Timer Function .....	213
Oscillation Stabilization Wait	
Oscillation Stabilization Wait and Reset State .....	68
Oscillation Stabilization Wait Interval .....	86, 111
Reset Causes and Oscillation Stabilization Wait Intervals.....	68
Oscillator	
Connection of an Oscillator or an External Clock to the Microcontroller .....	87



## INDEX

Output Compare	
16-bit Output Compare (×6) .....	280
16-bit Output Compare Interrupts .....	321
16-bit Output Compare Interrupts and EI <sup>2</sup> OS .....	321
16-bit Output Compare Operation .....	332
16-bit Output Compare Registers .....	290
16-bit Output Compare Timing.....	336
Block Diagram of 16-bit Output Compare .....	284
Sample Program for 16-bit Output Compare .....	352
Usage Notes on the 16-bit Output Compare .....	349
Output Compare Buffer Registers	
Output Compare Buffer Registers (OCCPB0 to OCCPB5).....	299
Output Compare Registers	
Output Compare Registers (OCCP0 to OCCP5) .....	300
Output Condition	
Output Condition of RTO0 to RTO5 and GATE .....	339
Output Control Lower Register	
Output Control Lower Register (OPCLR).....	374
Output Control Upper Register	
Output Control Upper Register (OPCUR).....	372
Output Data Buffer Lower Register	
Output Data Buffer Lower Register (OPDBR) .....	382
Output Data Buffer Register	
Operation of Output Data Buffer Register.....	405
Output Data Buffer Upper Register	
Output Data Buffer Upper Register (OPDBR) .....	380
Output Data Lower Register	
Output Data Lower Register (OPDR) .....	378
Output Data Register	
Operation of Data Transfer of Output Data Register .....	407
Output Data Register (OPDR) .....	398
Output Data Register (SODR0/SODR1) .....	482
Output Data Register Block Diagram .....	397
Output Data Upper Register	
Output Data Upper Register (OPDR) .....	376
Output Pulse	
PPG0 Output Pulse from Rising Edge of RT to 16-bit Timer Underflow (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ).....	342
Output Waveform	
OPTx Output Waveform Timing Diagram (WTS1,WTS0=00 <sub>B</sub> ) .....	398
Overlap	
Making Non-overlap Signals by using PPG in Inverted Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111 <sub>B</sub> ).....	346
Making Non-overlap Signals by using PPG in Normal Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111 <sub>B</sub> ) .....	345
Making Non-overlap Signals by using RT1/RT3/RT5 in Inverted Polarity (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=100 <sub>B</sub> ) .....	344
Making Non-overlap Signals by using RT1/RT3/RT5 in Normal Polarity (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=100 <sub>B</sub> ) .....	343
<b>P</b>	
Package Dimensions	
DIP-64P-M01 Package Dimensions .....	11
FPT-64P-M06 Package Dimensions .....	12
FPT-64P-M09 Package Dimensions .....	13
PACSR	
Program Address Detection Control Status Register (PACSR).....	576
PADR	
Program Address Detection Register 0/1 (PADR0/PADR1) .....	575
PC	
Program Counter (PC) .....	52
PCB	
Bank Registers (PCB,DTB,USB,SSB,ADB).....	54
Bank Select Prefixes (PCB,DTB,ADB,SPB) .....	58
PCNTH	
PPG Control Status Register,Upper Byte (PCNTH0 to PCNTH2) .....	267
PCNTL	
PPG Control Status Register,Lower Byte (PCNTL0 to PCNTL2) .....	269
PCSR	
PPG Period Setting Buffer Register (PCSR0 to PCSR2) .....	265
PDCR	
PPG Down Counter Register (PDCR0 to PDCR2) .....	264
PDUT	
PPG Duty Setting Buffer Register (PDUT0 to PDUT2) .....	266
period	
Calculating Pulse Width/period .....	457
Pulse Width/period Measurement Range .....	457
PICSH	
PPG Output Control/Input Capture Control Status Register,Upper Byte (PICSH01).....	309
PICSL	
Input Capture Control Status Register,Lower Byte (PICSL01).....	311
Pin Functions	
I/O Pins and Pin Functions.....	14
PLL Clock Mode	
Main Clock Mode and PLL Clock Mode.....	84

PLL Clock Multiplier		
Selection of a PLL Clock Multiplier .....	84	
Pointer		
Register Bank Pointer (RP) .....	50	
Polarity		
Making Non-overlap Signals by using PPG in Inverted Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111 <sub>B</sub> ) .....	346	
Making Non-overlap Signals by using PPG in Normal Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111 <sub>B</sub> ) .....	345	
Making Non-overlap Signals by using RT1/RT3/RT5 in Inverted Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=100 <sub>B</sub> ) .....	344	
Making Non-overlap Signals by using RT1/RT3/RT5 in Normal Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=100 <sub>B</sub> ) .....	343	
Port 0		
Block Diagram of Port 0 Pins .....	168	
Functions of Port 0 Registers.....	169	
Operation of Port 0.....	171	
Port 0 Configuration.....	167	
Port 0 Pins.....	167	
Port 0 Registers.....	168	
Port 1		
Block Diagram of Port 1 Pins .....	174	
Functions of Port 1 Registers.....	175	
Operation of Port 1.....	176	
Port 1 Configuration.....	173	
Port 1 Pins.....	173	
Port 1 Registers.....	174	
Port 2		
Block Diagram of Port 2 Pins .....	179	
Functions of Port 2 Registers.....	180	
Operation of Port 2.....	181	
Port 2 Configuration.....	178	
Port 2 Pins.....	178	
Port 2 Registers.....	179	
Port 3		
Block Diagram of Port 3 Pins .....	184	
Functions of Port 3 Registers.....	185	
Operation of Port 3.....	186	
Port 3 Configuration.....	183	
Port 3 Pins.....	183	
Port 3 Registers.....	184	
Port 4		
Block Diagram of Port 4 Pins .....	189	
Functions of Port 4 Registers.....	190	
Operation of Port 4.....	191	
Port 4 Configuration.....	188	
Port 4 Pins.....	188	
Port 4 Registers.....	189	
Port 5		
Block Diagram of Port 5 Pins .....	194	
Functions of Port 5 Registers.....	195	
Operation of Port 5 .....	196	
Port 5 Configuration .....	193	
Port 5 Pins .....	193	
Port 5 Registers .....	194	
Port 6		
Block Diagram of Port 6 Pins .....	199	
Functions of Port 6 Registers .....	200	
Operation of Port 6 .....	201	
Port 6 Configuration .....	198	
Port 6 Pins .....	198	
Port 6 Registers .....	199	
Position		
Signal Flow Diagram for Reload Timer 0 or Position Detection by Setting OPS2 to OPS0 = 100 <sub>B</sub> or 101 <sub>B</sub> .....	404	
Position Detection		
Operation of Position Detection .....	399	
Signal Flow Diagram for Position Detection by Setting OPS2 to OPS0=010 <sub>B</sub> or 110 <sub>B</sub> .....	403	
Signal Flow Diagram for Reload Timer 0 and Position Detection by Setting OPS2 to OPS0 = 011 <sub>B</sub> or 111 <sub>B</sub> .....	403	
Timing Generated by One-shot Position Detection (OPS2 to OPS0=110 <sub>B</sub> ) .....	418	
Timing Generated by One-shot Position Detection and Timer Underflow (OPS2 to OPS0 = 111 <sub>B</sub> ) .....	419	
Timing Generated by One-shot Position Detection or Timer Underflow (OPS2 to OPS0 = 101 <sub>B</sub> ) .....	420	
Timing Generated by Position Detection .....	411	
Timing Generated by Position Detection (OPS2 to OPS0=010 <sub>B</sub> ) .....	412	
Timing Generated by Position Detection and Timer Underflow .....	413	
Timing Generated by Position Detection and Timer Underflow (OPS2 to OPS0=011 <sub>B</sub> ).....	415	
Timing Generated by Position Detection or Timer Underflow .....	416	
Timing Generated by Position Detection or Timer Underflow (OPS2 to OPS0=100 <sub>B</sub> ).....	417	
When One-shot Position Detection .....	418	
When One-shot Position Detection and Timer Underflow .....	419	
When One-shot Position Detection or Timer Underflow .....	420	
Position Detection Circuit		
Block Diagram of Position Detection Circuit.....	366	
Power Supplied		
Example of Connection for Serial Writing (when Power Supplied by User) .....	618	
Example of Connection for Serial Writing (when Power Supplied from Writer) .....	620	
Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied by User) .....	622	

## INDEX

Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied from Writer).....	624
<b>PPG</b>	
16-bit PPG Timer (×3, PPG1 is not present in MB90465 Series) .....	258
Gate Trigger (PPG channel 0 only) .....	275
Gate Triggered PPG0 Output.....	340
Making Non-overlap Signals by using PPG in Inverted Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111 <sub>B</sub> ).....	346
Making Non-overlap Signals by using PPG in Normal Polarity (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=111 <sub>B</sub> ).....	345
PPG Interrupts .....	275
PPG0 Output Control.....	340
PPG0 Output Pulse from Rising Edge of RT to 16-bit Timer Underflow (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ).....	342
<b>PPG Control Status Register</b>	
PPG Control Status Register, Lower Byte (PCNTL0 to PCNTL2) .....	269
PPG Control Status Register, Upper Byte (PCNTH0 to PCNTH2).....	267
<b>PPG Down Counter Register</b>	
PPG Down Counter Register (PDCR0 to PDCR2) .....	264
<b>PPG Duty Setting Buffer Register</b>	
PPG Duty Setting Buffer Register (PDUT0 to PDUT2) .....	266
<b>PPG Output Control/Input Capture Control Status Register</b>	
PPG Output Control/Input Capture Control Status Register, Upper Byte (PICSH01) .....	309
<b>PPG Period Setting Buffer Register</b>	
PPG Period Setting Buffer Register (PCSR0 to PCSR2).....	265
<b>PPG Timer</b>	
16-bit PPG Timer (×1).....	281
16-bit PPG Timer (×3, PPG1 is not present in MB90465 Series) .....	258
16-bit PPG Timer Interrupts .....	271
16-bit PPG Timer Interrupts and EI <sup>2</sup> OS.....	272
16-bit PPG Timer Pins .....	260
16-bit PPG Timer Registers.....	262
Block Diagram of 16-bit PPG Timer .....	259
Block Diagram of the 16-bit PPG Timer Pins.....	260
EI <sup>2</sup> OS Function of the 16-bit PPG Timer.....	272
Sample Program for the 16-bit PPG Timer.....	277
Usage Notes on the 16-bit PPG Timer .....	276
<b>Prefix</b>	
Consecutive Prefix Codes .....	63
Prefix Codes .....	57
Prefix Codes and Interrupt/hold Suppression Instructions.....	62
<b>Processing Time</b>	
Hardware Interrupt Processing Time.....	135
Processing Time (one transfer time) of the Extended Intelligent I/O Service (EI <sup>2</sup> OS).....	147
<b>Processor Status</b>	
Processor Status (PS) Configuration .....	47
<b>Product Line-up</b>	
MB90460/465 Series Product Line-up .....	5
<b>Program</b>	
Sample I/O Port Program.....	203
<b>Program Address Detection Control Status Register</b>	
Program Address Detection Control Status Register (PACSR).....	576
<b>Program Address Detection Register</b>	
Program Address Detection Register 0/1 (PADR0/PADR1) .....	575
<b>Program Address Detection Registers</b>	
Program Address Detection Registers (×2) .....	572
<b>Program Counter</b>	
Program Counter (PC) .....	52
<b>Program Error</b>	
If a Program Error Occurs.....	580
<b>Programmer</b>	
Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied by User) .....	622
Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied from Writer) .....	624
<b>Programming Example</b>	
Programming Example Using 512K Bit Flash Memory .....	610
<b>Protection</b>	
A/D Conversion Data Protection Function.....	561
<b>PS</b>	
Processor Status (PS) Configuration .....	47
<b>Pull-up Resistor</b>	
Software Pull-up Resistor .....	109
<b>Pulse</b>	
Calculating Pulse Width/period .....	457
Pulse Width/period Measurement Range .....	457
<b>Pulse Width</b>	
Calculating Pulse Width/period .....	457
Minimum Input Pulse Width .....	457
Pulse Width/period Measurement Range .....	457
<b>Pulse-width Measurement</b>	
Flowchart of Pulse-width Measurement Operation .....	460
Pulse-width Measurement Function .....	448
Starting and Stopping Timer and Pulse-width Measurement .....	451
<b>PWC</b>	
PWC Data Buffer Register (PWC0/PWC1) .....	443

PWC Control Status Register		
PWC Control Status Register,Upper Byte		
(PWCSH0/PWCSH1).....	439	
PWC control Status Register		
PWC control Status Register,Lower Byte		
(PWCSL0/PWCSL1).....	441	
PWC Data Buffer Register		
PWC Data Buffer Register (PWC0/PWC1) .....	443	
PWC Timer		
Block Diagram of the PWC Timer Pins .....	436	
EI <sup>2</sup> OS Function of the PWC Timer .....	446	
PWC Timer (×2,PWC Timer 0 is not present in		
MB90465 Series) .....	434	
PWC Timer Block Diagram .....	435	
PWC Timer Interrupts .....	445	
PWC Timer Interrupts and EI <sup>2</sup> OS.....	445	
PWC Timer Pins .....	436	
PWC Timer Registers.....	438	
Sample Program for the PWC Timer.....	464	
Usage Notes on the PWC Timer .....	461	
PWCSH		
PWC Control Status Register,Upper Byte		
(PWCSH0/PWCSH1).....	439	
PWCSL		
PWC control Status Register,Lower Byte		
(PWCSL0/PWCSL1).....	441	
PWM		
PWM Mode (PCNTL: MDSE=0) .....	273	
PWM Mode		
PWM Mode (PCNTL: MDSE=0) .....	273	
<b>R</b>		
RAM		
RAM Area .....	30	
Storage of Multi-byte Data in RAM .....	37	
Read		
Setting the Read/Reset Status .....	601	
Reception Interrupt		
Reception Interrupt Generation and Flag Set Timing		
.....	488	
Register		
16-bit Reload Register (TMRD0/TMRD1) .....	242	
16-bit Timer Control Register (DTCR0/DTCR2)		
.....	314	
16-bit Timer Control Register (DTCR1).....	316	
16-bit Timer Register (TMR0/TMR1) .....	241	
A/D Control Status Register 0 (ADCS0) .....	551	
A/D Control Status Register 1 (ADCS1) .....	549	
A/D Data Register (ADCR0/ADCR1) .....	554	
Communication Prescaler Control Register (CDCR)		
.....	484	
Compare Clear Buffer Register (CPCLRB) .....	293	
Compare Clear Register (CPCLR) .....	293	
Compare Clear Register (CPCR) .....	388	
Compare Control Register,Upper Byte		
(OCS1/OCS3/OCS5) .....	301	
Condition Code Register (CCR) Configuration .....	48	
Control Status Register (FMCS).....	590	
Delayed Interrupt Generator Module Register (DIRR)		
.....	537	
Direct Page Register (DPR) .....	53	
Division Rate Control Register (DIV0/DIV1)		
.....	444	
DTP/interrupt Cause Register (EIRR) .....	521	
DTP/interrupt Enable Register (ENIR) .....	522	
Input Capture Control Status Register,Lower Byte		
(ICSL23) .....	307	
Input Capture Control Status Register,Lower Byte		
(PICSL01) .....	311	
Input Capture Control Status Register,Upper Byte		
(ICSH23) .....	306	
Input Capture Register (IPCP0 to IPCP3).....	305	
Input Control Lower Register (IPCLR) .....	386	
Input Control Upper Register (IPCUR) .....	384	
Input Data Register (SIDR0/SIDR1).....	482	
Interrupt Level Mask Register (ILM) .....	51	
Noise Cancellation Control Register (NCCR)		
.....	392	
Output Control Lower Register (OPCLR) .....	374	
Output Control Upper Register (OPCUR) .....	372	
Output Data Buffer Lower Register (OPDBR)		
.....	382	
Output Data Buffer Upper Register (OPDBR)		
.....	380	
Output Data Lower Register (OPDR) .....	378	
Output Data Register (OPDR).....	398	
Output Data Register (SODR0/SODR1) .....	482	
Output Data Register Block Diagram.....	397	
Output Data Upper Register (OPDR).....	376	
PPG Control Status Register,Lower Byte		
(PCNTL0 to PCNTL2).....	269	
PPG Control Status Register,Upper Byte		
(PCNTH0 to PCNTH2) .....	267	
PPG Down Counter Register (PDCR0 to PDCR2)		
.....	264	
PPG Duty Setting Buffer Register (PDUT0 to PDUT2)		
.....	266	
PPG Output Control/Input Capture Control Status		
Register,Upper Byte (PICSH01).....	309	
PPG Period Setting Buffer Register (PCSR0 to		
PCSR2) .....	265	
Program Address Detection Control Status Register		
(PACSR) .....	576	
Program Address Detection Register 0/1		
(PADR0/PADR1) .....	575	
PWC control Status Register,Lower Byte		
(PWCSL0/PWCSL1) .....	441	
PWC Control Status Register,Upper Byte		
(PWCSH0/PWCSH1) .....	439	
PWC Data Buffer Register (PWC0/PWC1) .....	443	
Request Level Setting Register (ELVR).....	524	

## INDEX

ROM Mirroring Function Selection Register (ROMM) .....	585
Serial Control Register (SCR0/SCR1) .....	476
Serial Mode Register (SMR0/SMR1) .....	478
Serial Status Register (SSR0/SSR1) .....	480
Time-base Timer Control Register (TBTC).....	209
Timer Buffer Register (TMBR) .....	389
Timer Control Status Register, Lower Byte (TCCSL) .....	297
Timer Control Status Register (TCSR) .....	390
Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1) .....	239
Timer Control Status Register, Upper Byte (TCCSH) .....	295
Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1) .....	237
Timer Data Register (TCDT).....	294
Watchdog Timer Control Register (WDTC).....	222
Waveform Control Register (SIGCR).....	318
Register Bank .....	56
Register Bank Pointer .....	
General-purpose Register Area and Register Bank Pointer .....	50
Register Bank Pointer (RP) .....	50
Registers .....	
16-bit Timer Registers (TMRR0/TMRR1/TMRR2) .....	313
Output Compare Buffer Registers (OCCPB0 to OCCPB5) .....	299
Output Compare Registers (OCCP0 to OCCP5) .....	300
Program Address Detection Registers (×2) .....	572
reload mode .....	
Operation in Internal Clock Mode (reload mode) .....	246
Reload Operation Mode .....	
Reload Operation Mode .....	453
Reload Timer .....	
16-bit Reload Timer Interrupts.....	243
16-bit Reload Timer Interrupts and EI <sup>2</sup> OS .....	243
16-bit Reload Timer Interrupts and EI <sup>2</sup> OS .....	232
16-bit Reload Timer Pins .....	235
16-bit Reload Timer Registers .....	236
16-bit Reload Timer Settings .....	244
Baud Rates determined using the Internal Timer (16-bit Reload Timer 0) .....	495
Block Diagram of the 16-bit Reload Timer .....	233
Block Diagram of the 16-bit Reload Timer Pins .....	235
EI <sup>2</sup> OS Function of the 16-bit Reload Timer .....	243
Overview of the 16-bit Reload Timer .....	230
Signal Flow Diagram for Reload Timer 0 and Position Detection by Setting OPS2 to OPS0 = 011 <sub>B</sub> or 111 <sub>B</sub> .....	403
Signal Flow Diagram for Reload Timer 0 or Position Detection by Setting OPS2 to OPS0 = 100 <sub>B</sub> or 101 <sub>B</sub> .....	404
Signal Flow Diagram for Reload Timer 0 Underflow by Setting OPS2 to OPS0=001 <sub>B</sub> .....	402
Timing Generated by Reload Timer Underflow .....	409
Timing Generated by Reload Timer Underflow (OPS2 to OPS0=001 <sub>B</sub> ) .....	410
Usage Notes on the 16-bit Reload Timer .....	252
Reload Value .....	
Timer Value and Reload Value .....	453
Request Level Setting Register .....	
Request Level Setting Register (ELVR) .....	524
Reset .....	
Block Diagrams of the External Reset Pin .....	69
Correspondence between Reset Cause Bits and Reset Causes .....	74
Notes about Reset Cause Bits .....	74
Oscillation Stabilization Wait and Reset State.....	68
Overview of Reset Operation .....	71
Reset Cause Bits .....	73
Reset Causes .....	66
Reset Sequence .....	580
Setting the Read/Reset Status .....	601
Status of Pins during a Reset .....	75
Reset Cause .....	
Correspondence between Reset Cause Bits and Reset Causes .....	74
Notes about Reset Cause Bits .....	74
Reset Cause Bits .....	73
Reset Causes .....	
Reset Causes .....	66
Reset Causes and Oscillation Stabilization Wait Intervals .....	68
Reset Sequence .....	
Reset Sequence .....	580
Reset State .....	
Oscillation Stabilization Wait and Reset State.....	68
Resistor .....	
Software Pull-up Resistor .....	109
Return .....	
Stack Operations on Return from Interrupt Processing .....	150
Returning .....	
Returning from a Hardware Interrupt .....	129
Returning from a Software Interrupt .....	137
Rising Edge .....	
Generating GATE Signal from Rising Edge of Each RTx until 16-bit Timer 0/1/2 Underflow when GTENx is active (DTCR0/DTCR1/DTCR2:TMD2 to TMD0=010 <sub>B</sub> ) .....	341
ROM .....	
ROM Area .....	30

ROM Correction	
Block Diagram of ROM Correction Function	573
Operation of the ROM Correction Function	578
ROM Correction Function Registers	574
ROM Correction Interrupts	572
ROM Mirroring Function Selection Module	
ROM Mirroring Function Selection Module Block	
Diagram	584
ROM Mirroring Function Selection Module Register	584
ROM Mirroring Function Selection Register	
ROM Mirroring Function Selection Register	
(ROMM)	585
ROMM	
ROM Mirroring Function Selection Register	
(ROMM)	585
RP	
Register Bank Pointer (RP)	50
RTO	
Output Condition of RTO0 to RTO5 and GATE	339
RUN Mode	
RUN Mode	158
<b>S</b>	
Sample	
Sample I/O Port Program	203
Sample Program	
Processing Specifications of Sample Program for	
Extended Intelligent I/O Service (EI <sup>2</sup> OS)	153
Sample Program for 16-bit Free-run Timer	351
Sample Program for 16-bit Output Compare	352
Sample Program for Continuous Conversion Mode	
using EI <sup>2</sup> OS	566
Sample Program for Single Conversion Mode using	
EI <sup>2</sup> OS	564
Sample Program for Stop Conversion Mode using	
EI <sup>2</sup> OS	568
Sample Program for the 16-bit PPG Timer	277
Sample Program for the DTP Function	533
Sample Program for the External Interrupt Function	
	532
Sample Program for the Multi-pulse Generator	
	431
Sample Program for the PWC Timer	464
Sample Program for the Time-base Timer	216
Sample Program for the Watchdog Timer	227
Sample Program for UART	510
Sample Program in Event Count Mode	254
Sample Program in Internal Clock Mode	253
Sample Programs	
Sample Programs for Interrupt Processing	152
SCR	
Serial Control Register (SCR0/SCR1)	476
Sector	
Procedure of Deleting a Sector	605
Restarting the Sector Deletion	608
Sector Configuration	589
Temporarily Stopping the Sector Deletion	607
When the Sector Deletion Temporary Stop is	
Executed	595, 597
Sector Deletion	
Restarting the Sector Deletion	608
Temporarily Stopping the Sector Deletion	607
When the Sector Deletion Temporary Stop is	
Executed	595, 597
Sector deletion	
Deleting the data (Sector deletion)	605
Sector Deletion Operation	
When the Chip/Sector Deletion Operation is Executed	
	595
When the Sector Deletion Operation is Executed	
	599
When the Write Operation or Chip/Sector Deletion	
Operation is Executed	597, 598
Sectors	
Notes on Specifying Two or More Sectors	605
Serial Control Register	
Serial Control Register (SCR0/SCR1)	476
Serial Mode	
Serial Mode Register (SMR0/SMR1)	478
Serial Mode Register	
Serial Mode Register (SMR0/SMR1)	478
Serial On-board Writing	
Standard Configuration for Fujitsu Standard Serial	
On-board Writing	616
Serial Status Register	
Serial Status Register (SSR0/SSR1)	480
Serial Writing	
Example of Connection for Serial Writing (when	
Power Supplied by User)	618
Example of Connection for Serial Writing (when	
Power Supplied from Writer)	620
SIDR	
Input Data Register (SIDR0/SIDR1)	482
SIGCR	
Waveform Control Register (SIGCR)	318
Signal Flow Diagram	
Signal Flow Diagram for OPDBR0 by Setting OPS2 to	
OPS0=000 <sub>B</sub>	401
Signal Flow Diagram for Position Detection by Setting	
OPS2 to OPS0=010 <sub>B</sub> or 110 <sub>B</sub>	403
Signal Flow Diagram for Reload Timer 0 and Position	
Detection by Setting OPS2 to OPS0 = 011 <sub>B</sub>	
or 111 <sub>B</sub>	403

## INDEX

Signal Flow Diagram for Reload Timer 0 or Position Detection by Setting OPS2 to OPS0 = 100 <sub>B</sub> or 101 <sub>B</sub> .....	404
Signal Flow Diagram for Reload Timer 0 Underflow by Setting OPS2 to OPS0=001 <sub>B</sub> .....	402
Single Conversion Mode	
Operation in Single Conversion Mode .....	557
Sample Program for Single Conversion Mode using EI <sup>2</sup> OS .....	564
Single Measurement Mode	
Single Measurement Mode and Continuous Measurement Mode .....	456
Single-chip Mode	
State of Pins in Single-chip Mode .....	109
Single-shot Mode	
Internal Clock Mode (Single-shot Mode) .....	248
Single-shot Mode (PCNTL: MDSE=1) .....	274
Sleep Mode	
Release of Sleep Mode .....	99
Switching to Sleep Mode .....	99
SMR	
Serial Mode Register (SMR0/SMR1) .....	478
SN1x/RDAx Comparison Timing Diagram	
Both Edges Detection and SN1x/RDAx Comparison Timing Diagram (CMPE=1) .....	400
SODR	
Output Data Register (SODR0/SODR1) .....	482
Software	
Software Pull-up Resistor .....	109
Software Interrupt	
Returning from a Software Interrupt .....	137
Software Interrupt Activation .....	137
Software Interrupt Operation .....	138
Software Pull-up Resistor	
Software Pull-up Resistor .....	109
SPB	
Bank Select Prefixes (PCB,DTB,ADB,SPB) .....	58
SSB	
Bank Registers (PCB,DTB,USB,SSB,ADB) .....	54
SSP	
System Stack Pointer (SSP) .....	46
SSR	
Serial Status Register (SSR0/SSR1) .....	480
Stack	
Stack Area .....	151
Stack Operations at the Start of Interrupt Processing .....	150
Stack Operations on Return from Interrupt Processing .....	150
Stack Selection .....	45
Storage of Multi-byte Data in a Stack .....	38
Stack Pointer	
System Stack Pointer (SSP) .....	46
User Stack Pointer (USP) .....	46
Standard Configuration	
Standard Configuration for Fujitsu Standard Serial On-board Writing .....	616
Standby Mode	
Notes on Standby Mode .....	110
Operating Status during Standby Mode .....	98
Standby Mode .....	91
State Change Diagram	
State Change Diagram .....	106
Status	
Setting the Read/Reset Status .....	601
Status Register	
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	143
Stop Conversion Mode	
Operation in Stop Conversion Mode .....	558
Sample Program for Stop Conversion Mode using EI <sup>2</sup> OS .....	568
Stop Mode	
Release of Stop Mode .....	104, 111
Switching to Stop Mode .....	104
Structure	
Structure of Instruction Map .....	674
Suppression	
Prefix Codes and Interrupt/hold Suppression Instructions .....	62
Synchronous Mode	
Operation in Synchronous Mode (Operation Mode 2) .....	502
System Configuration	
System Configuration .....	579
System Stack Pointer	
System Stack Pointer (SSP) .....	46
<b>T</b>	
TBTC	
Time-base Timer Control Register (TBTC) .....	209
TCCSH	
Timer Control Status Register,Upper Byte (TCCSH) .....	295
TCCSL	
Timer Control Status Register,Lower Byte (TCCSL) .....	297
TCDT	
Timer Data Register (TCDT) .....	294
TCSR	
Timer Control Status Register (TCSR) .....	390
Register	
Compare Control Register,Lower Byte (OCS0/OCS2/OCS4) .....	303
Time-base Timer	
Block Diagram of the Time-base Timer .....	208

Operation of the Interval Timer Function (Time-base Timer) .....	212
Operation of the Time-base Timer .....	215
Sample Program for the Time-base Timer .....	216
Time-base Timer Interrupts .....	211
Time-base Timer Interrupts and EI <sup>2</sup> OS .....	211
Time-base Timer Usage Notes .....	214
Time-base Timer Control Register Time-base Timer Control Register (TBTC) .....	209
Time-base Timer Mode Release of Time-base Timer Mode .....	102, 111
Switching to Time-base Timer Mode .....	102
Timer Buffer Register Timer Buffer Register (TMBR) .....	389
Timer Clear Timer Clear .....	325
Timer Control Status Register Timer Control Status Register, Lower Byte (TCCSL) .....	297
Timer Control Status Register Timer Control Status Register (TCSR) .....	390
Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1) .....	239
Timer Control Status Register, Upper Byte (TCCSH) .....	295
Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1) .....	237
Timer Data Register Timer Data Register (TCDT) .....	294
Timer Interrupts Timer Interrupts .....	329
Timer Mode Flowchart of Timer Mode Operation .....	455
Timer Mode .....	326
Rising Edge PPG0 Output Pulse from Rising Edge of RT to 16-bit Timer Underflow (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ) .....	342
TMBR Timer Buffer Register (TMBR) .....	389
TMCSRH Timer Control Status Register, Upper Byte (TMCSRH0/TMCSRH1) .....	237
TMCSRL Timer Control Status Register, Lower Byte (TMCSRL0/TMCSRL1) .....	239
TMR 16-bit Timer Register (TMR0/TMR1) .....	241
TMRD 16-bit Reload Register (TMRD0/TMRD1) .....	242
TMRR 16-bit Timer Registers (TMRR0/TMRR1/TMRR2) .....	313
Transfer Operation of Data Transfer of Output Data Register .....	407
transfer Processing Time (one transfer time) of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	147
Transition Clock Mode Transition .....	84
Transmission Interrupt Transmission Interrupt Generation and Flag Set Timing .....	489
Trigger Gate Trigger (PPG channel 0 only) .....	275
<b>U</b>	
UART Block Diagram of UART .....	470
Block Diagram of UART Pins .....	474
Notes on using UART .....	509
Operation of UART .....	498
Sample Program for UART .....	510
UART Baud Rate Selection .....	490
UART EI <sup>2</sup> OS Functions .....	487
UART Functions (×2) .....	468
UART Interrupt and EI <sup>2</sup> OS .....	469
UART Interrupts .....	486
UART Interrupts and EI <sup>2</sup> OS .....	487
UART Pins .....	473
UART Registers .....	475
Underflow Generating GATE Signal from Rising Edge of Each RTx until 16-bit Timer 0/1/2 Underflow when GTENx is active (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ) .....	341
PPG0 Output Pulse from Rising Edge of RT to 16-bit Timer Underflow (DTCR0/DTCR1/ DTCR2:TMD2 to TMD0=010 <sub>B</sub> ) .....	342
Signal Flow Diagram for Reload Timer 0 Underflow by Setting OPS2 to OPS0=001 <sub>B</sub> .....	402
Timing Generated by One-shot Position Detection and Timer Underflow (OPS2 to OPS0 = 111 <sub>B</sub> ) .....	419
Timing Generated by One-shot Position Detection or Timer Underflow (OPS2 to OPS0 = 101 <sub>B</sub> ) .....	420
Timing Generated by Position Detection and Timer Underflow .....	413
Timing Generated by Position Detection and Timer Underflow (OPS2 to OPS0=011 <sub>B</sub> ) .....	415
Timing Generated by Position Detection or Timer Underflow .....	416
Timing Generated by Position Detection or Timer Underflow (OPS2 to OPS0=100 <sub>B</sub> ) .....	417
Timing Generated by Reload Timer Underflow .....	409



## INDEX

Timing Generated by Reload Timer Underflow (OPS2 to OPS0=001 <sub>B</sub> ).....	410
When One-shot Position Detection and Timer Underflow .....	419
When One-shot Position Detection or Timer Underflow .....	420
<b>USB</b>	
Bank Registers (PCB,DTB,USB,SSB,ADB) .....	54
<b>User Stack Pointer</b>	
User Stack Pointer (USP).....	46
<b>USP</b>	
User Stack Pointer (USP).....	46
<b>W</b>	
<b>Watchdog Timer</b>	
Block Diagram of the Watchdog Timer .....	221
Sample Program for the Watchdog Timer .....	227
Usage Notes on the Watchdog Timer .....	226
Watchdog Timer Function.....	220
Watchdog Timer Operation .....	224
<b>Watchdog Timer Control Register</b>	
Watchdog Timer Control Register (WDTC).....	222
<b>Waveform</b>	
OPTx Output Waveform Timing Diagram (WTS1,WTS0=00 <sub>B</sub> ) .....	398
<b>Waveform Control Register</b>	
Waveform Control Register (SIGCR).....	318
<b>Waveform Generator</b>	
Block Diagram of Waveform Generator .....	285
Usage Notes on the Waveform Generator .....	350
Waveform Generator .....	281
Waveform Generator Interrupts .....	322
Waveform Generator Interrupts and EI <sup>2</sup> OS .....	323
Waveform Generator Registers .....	292
<b>Waveform Sequencer</b>	
Block Diagram of Waveform Sequencer .....	360
Function of Waveform Sequencer.....	356
Usage Notes on the Waveform Sequencer .....	429
<b>WDTC</b>	
Watchdog Timer Control Register (WDTC) .....	222
<b>Write</b>	
Detailed Explanation on the Flash Memory Write/ Delete .....	600
<b>Write Operation</b>	
When the Write Operation is Executed.....	595
When the Write Operation or Chip/Sector Deletion Operation is Executed.....	597, 598
<b>Writer</b>	
Example of Connection for Serial Writing (when Power Supplied from Writer) .....	620
Example of Minimum Connection with Flash Microcontroller Programmer (when Power Supplied from Writer) .....	624
<b>Writing</b>	
Notes on Writing the Data .....	602
Procedure for Writing/Deleting the data to the Flash Memory .....	588
Procedure of Writing the Data to the Flash Memory .....	602
Writing the Data .....	602
<b>WTIN</b>	
WTIN1 Output Condition and Register Setting .....	400

CM44-10120-4E

---

**FUJITSU MICROELECTRONICS • CONTROLLER MANUAL**

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

16-BIT MICROCONTROLLER

MB90460/465 Series

HARDWARE MANUAL

---

August 2008 the fourth edition

Published **FUJITSU MICROELECTRONICS LIMITED**

Edited Business & Media Promotion Dept.

---

