



---

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**  
**MB90480/485 Series**  
**HARDWARE MANUAL**



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

## **16-BIT MICROCONTROLLER**

# **MB90480/485 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/>





# PREFACE

## ■ Purpose of This Manual and Intended Readers

Thank you very much for purchasing FUJITSU products.

MB90480/485 series is a 16-bit microcontroller designed for applications such as consumer devices requiring high-speed real-time processing. MB90480/485 series functions are suitable for controlling PHS, cellular phones, CD-ROMs, and VTRs.

This manual, intended for engineers developing products using the MB90480/485 series, explains the MB90480/485 series functions and operations. Read this manual first, before using the product.

For details on the instructions, refer to the "Instruction Manual".

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

## ■ Trademark

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

## ■ License

Purchase of Fujitsu I<sup>2</sup>C components conveys a license under the Philips I<sup>2</sup>C Patent Rights to use, these components in an I<sup>2</sup>C system provided that the system conforms to the I<sup>2</sup>C Standard Specification as defined by Philips.

## ■ Composition of This Manual

This manual consists of the following 27 chapters and an appendix.

### **CHAPTER 1 "OVERVIEW OF MB90480/485 SERIES"**

This chapter gives an overview of MB90480/485 series, including its basic features and basic specifications.

### **CHAPTER 2 "CPU"**

This chapter explains CPU specifications, memory, and the functions of registers to provide readers with a better understanding of the MB90480/485 series functions.

### **CHAPTER 3 "INTERRUPT"**

This chapter explains interrupts and direct memory access (DMA).

### **CHAPTER 4 "RESET"**

This chapter explains reset for the MB90480/485 series.

### **CHAPTER 5 "CLOCKS"**

This chapter describes the clocks of the MB90480/485 series.

### **CHAPTER 6 "LOW-POWER CONSUMPTION MODE"**

This chapter explains the low-power consumption mode of the MB90480/485 series.

### **CHAPTER 7 "MODE SETTING"**

This chapter explains mode setting, mode pins, mode data, external memory access and its operation.

### **CHAPTER 8 "I/O PORT"**

This chapter explains the configuration and the functions of the registers used for the I/O port.

### **CHAPTER 9 "TIME-BASE TIMER"**

This chapter explains the functions and operations of the time-base timer.

### **CHAPTER 10 "WATCHDOG TIMER"**

This chapter provides an overview of watchdog timer, explains control register, configuration, operations and shows the precautions on use, and sample program.

### **CHAPTER 11 "WATCH TIMER"**

This chapter provides an overview of the watch timer, explains the configuration and functions of its register, and the operation of the watch timer.

### **CHAPTER 12 "16-BIT INPUT/OUTPUT TIMER"**

This chapter provides an overview of the 16-bit input/output timer, explains the configuration and functions of its registers, interrupt and its operation.

### **CHAPTER 13 "8/16-BIT UP/DOWN COUNTER/TIMER"**

This chapter provides an overview of the 8/16-bit up/down counter/timer, explains the configuration and functions of its registers, interrupt and its operation.

### **CHAPTER 14 "16-BIT RELOAD TIMER"**

This chapter provides an overview of the 16-bit reload timer, explains the configuration and functions of its registers, interrupt and its operation.

## **CHAPTER 15 "8/16-BIT PPG TIMER"**

This chapter provides an overview of the 8/16-bit PPG timer, explains the configuration and functions of its registers interrupt and its operation.

## **CHAPTER 16 "DTP/EXTERNAL INTERRUPTS"**

This chapter provides an overview of the DTP/external interrupt unit, explains configuration and functions of its registers and its operation, shows the precautions on use.

## **CHAPTER 17 "8/10-BIT A/D CONVERTER"**

This chapter provides an overview of the 8/10-bit A/D converter, explains configuration and functions of these registers, operation, conversion data protection function, and shows the precautions on use.

## **CHAPTER 18 "EXPANDED I/O SERIAL INTERFACE"**

This chapter provides an overview, of the expanded I/O serial interface, explains the configuration, interrupt, and its operation, the configuration and functions of its registers.

## **CHAPTER 19 "UART"**

This chapter provides an overview, of the UART, explains the configuration, interrupt, its operation, the configuration and functions of its registers shows the precautions on use, and program example of the UART.

## **CHAPTER 20 "CHIP SELECTION FACILITY"**

This chapter provides an overview, of the chip selection facility explains the configuration, and its operation, the configuration and functions of its registers.

## **CHAPTER 21 "ADDRESS MATCH DETECTION FUNCTION"**

This chapter explains the functions and operations of the address match detection.

## **CHAPTER 22 "ROM MIRROR FUNCTION SELECTION MODULE"**

This chapter provides an overview of the ROM mirror function selection module and explains its registers.

## **CHAPTER 23 "2M/3M BIT FLASH MEMORY"**

This chapter explains the functions and operations of the 2M/3M bit flash memory.

## **CHAPTER 24 "EXAMPLES OF MB90F481B/MB90F482B/MB90F488B/MB90F489B SERIAL PROGRAMMING CONNECTION"**

This chapter shows an example of a serial programming connection using the AF220/AF210/AF120/AF110 Flash Microcontroller Programmer by Yokogawa Digital Computer Corporation.

## **CHAPTER 25 "PWC TIMER (ONLY MB90485 SERIES)"**

This chapter provides an overview of the PWC timer, explains the configuration, the configuration and functions of its registers interrupt, shows the precautions on use.

## **CHAPTER 26 "μPG TIMER (ONLY MB90485 SERIES)"**

This chapter provides an overview, explains the configuration of the μPG timer and its timing chart, the configuration and functions of its registers.

## **CHAPTER 27 "I<sup>2</sup>C INTERFACE (ONLY MB90485 SERIES)"**

This chapter provides an overview, explains configuration, interrupt, and operation of the I<sup>2</sup>C interface, the configuration and functions of its registers.

## **APPENDIX**

The appendix provides the memory map and lists the instructions used in the F<sup>2</sup>MC-16LX.

- 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 OF MB90480/485 SERIES</b>                              | <b>1</b>  |
| 1.1              | Features of MB90480/485 Series                                     | 2         |
| 1.2              | Block Diagram of MB90480/485 Series                                | 6         |
| 1.3              | Package Dimensions   | 7         |
| 1.4              | Pin Assignment   | 9         |
| 1.5              | Pin Functions  | 11        |
| 1.6              | I/O Circuit Type   | 18        |
| 1.7              | Handling the Device  | 21        |
| <b>CHAPTER 2</b> | <b>CPU</b>   | <b>23</b> |
| 2.1              | Overview of CPU Specifications                                     | 24        |
| 2.2              | Memory Space   | 25        |
| 2.3              | CPU Registers  | 29        |
| 2.3.1            | Accumulator (A)  | 31        |
| 2.3.2            | User Stack Pointer (USP) and System Stack Pointer (SSP)            | 32        |
| 2.3.3            | Processor Status (PS)  | 33        |
| 2.3.4            | Program Counter (PC)   | 36        |
| 2.3.5            | Program Counter Bank Register (PCB)                                | 37        |
| 2.3.6            | Direct Page Register (DPR)   | 38        |
| 2.3.7            | General-Purpose Register (Register Bank)                           | 39        |
| 2.4              | Prefix Codes   | 40        |
| <b>CHAPTER 3</b> | <b>INTERRUPT</b>   | <b>43</b> |
| 3.1              | Overview of Interrupt  | 44        |
| 3.2              | Interrupt Factor and Interrupt Vector                              | 46        |
| 3.3              | Interrupt Control Register and Peripheral Function                 | 49        |
| 3.3.1            | Interrupt Control Register (ICR00 to ICR15)                        | 50        |
| 3.4              | Hardware Interrupt   | 53        |
| 3.4.1            | Hardware Interrupt Operation                                       | 56        |
| 3.4.2            | Flow of Hardware Interrupt Operation                               | 58        |
| 3.4.3            | Procedure for Using Hardware Interrupt                             | 59        |
| 3.4.4            | Multiple Interrupts  | 60        |
| 3.4.5            | Hardware Interrupt Processing Time                                 | 62        |
| 3.5              | Software Interrupt   | 64        |
| 3.6              | Interrupt by $\mu$ DMAC  | 66        |
| 3.6.1            | DMA Descriptor   | 69        |
| 3.6.2            | Individual Registers of DMA Descriptor                             | 71        |
| 3.6.3            | $\mu$ DMAC Processing Procedure                                    | 74        |
| 3.6.4            | $\mu$ DMAC Processing Time   | 75        |
| 3.7              | Interrupt by Extended Intelligent I/O Service (EI <sup>2</sup> OS) | 77        |
| 3.7.1            | EI <sup>2</sup> OS descriptor (ISD)                                | 79        |
| 3.7.2            | Each Register of EI <sup>2</sup> OS Descriptor (ISD)               | 81        |
| 3.7.3            | Operation of EI <sup>2</sup> OS                                    | 84        |

|                  |  |            |
|------------------|--|------------|
| 3.7.4            | Procedure for Use of EI <sup>2</sup> OS .....                                      | 85         |
| 3.7.5            | Processing Time of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) ..... | 86         |
| 3.8              | Exception Processing Interrupt .....   | 88         |
| 3.9              | Stack Operation of Interrupt Processing .....                                      | 89         |
| 3.10             | Sample Program of Interrupt Processing .....                                       | 91         |
| 3.11             | Delay Interrupt Generation Module .....  | 92         |
| 3.11.1           | Operation of Delay Interrupt Generation Module .....                               | 93         |
| <b>CHAPTER 4</b> | <b>RESET .....</b>   | <b>95</b>  |
| 4.1              | Overview of Reset .....  | 96         |
| 4.2              | Reset Factors and Oscillation Stabilization Wait Time .....                        | 98         |
| 4.3              | External-Reset Pin .....   | 100        |
| 4.4              | Resetting .....  | 101        |
| 4.5              | Reset-Factor Bits .....  | 103        |
| 4.6              | Condition of Pins as Result of Reset .....   | 105        |
| <b>CHAPTER 5</b> | <b>CLOCKS .....</b>  | <b>107</b> |
| 5.1              | Overview of Clocks .....   | 108        |
| 5.2              | Block Diagram of Clock Generator .....   | 110        |
| 5.3              | Clock Selection Register (CKSCR) and PLL Output Selection Register (PLLOS) .....   | 112        |
| 5.4              | Clock Modes .....  | 117        |
| 5.5              | Oscillation Stabilization Wait Time .....  | 121        |
| 5.6              | Connecting Oscillator to External Clock .....                                      | 122        |
| <b>CHAPTER 6</b> | <b>LOW-POWER CONSUMPTION MODE .....</b>  | <b>123</b> |
| 6.1              | Overview of Low-Power Consumption Mode .....                                       | 124        |
| 6.2              | Block Diagram of Low-Power Consumption Control Circuit .....                       | 126        |
| 6.3              | Low-Power Consumption Mode Control Register (LPMCR) .....                          | 128        |
| 6.4              | CPU Intermittent Operation Mode .....  | 131        |
| 6.5              | Standby Mode .....   | 132        |
| 6.5.1            | Sleep Mode .....   | 133        |
| 6.5.2            | Time-Base Timer Mode .....   | 135        |
| 6.5.3            | Watch Mode .....   | 137        |
| 6.5.4            | Stop Mode .....  | 139        |
| 6.6              | State Transition Diagram .....   | 141        |
| 6.7              | Pin State in Standby Mode, Hold, and Reset .....                                   | 143        |
| 6.8              | Caution on Using Low-Power Consumption Mode .....                                  | 148        |
| <b>CHAPTER 7</b> | <b>MODE SETTING .....</b>  | <b>153</b> |
| 7.1              | Mode Setting .....   | 154        |
| 7.2              | Mode Pins (MD2 to MD0) .....   | 155        |
| 7.3              | Mode Data .....  | 156        |
| 7.4              | External Memory Access .....   | 160        |
| 7.4.1            | Automatic ready function selection register (ARSR) .....                           | 162        |
| 7.4.2            | External address output control register (HACR) .....                              | 164        |
| 7.4.3            | Bus control signal selection register (EPCR) .....                                 | 165        |
| 7.5              | Operation of Each Mode for Mode Setting .....                                      | 167        |

|                   |  |            |
|-------------------|--|------------|
| 7.5.1             | External memory access control signals .....                           | 168        |
| 7.5.2             | Ready function .....   | 171        |
| 7.5.3             | Hold function .....  | 174        |
| <b>CHAPTER 8</b>  | <b>I/O PORT .....</b>  | <b>177</b> |
| 8.1               | Functions of I/O Port .....  | 178        |
| 8.2               | Registers for I/O Port .....   | 179        |
| 8.2.1             | Port registers (PDR0 to PDRA) .....                                    | 180        |
| 8.2.2             | Port direction registers (DDR0 to DDRA) .....                          | 181        |
| 8.2.3             | Other registers .....  | 183        |
| <b>CHAPTER 9</b>  | <b>TIME-BASE TIMER .....</b>   | <b>185</b> |
| 9.1               | Overview of Time-Base Timer .....                                      | 186        |
| 9.2               | Time-Base Timer Configuration .....                                    | 188        |
| 9.3               | Time-Base Timer Control Register (TBTC) .....                          | 190        |
| 9.4               | Time-Base Timer Interrupt .....  | 192        |
| 9.5               | Time-Base Timer Operation .....  | 193        |
| 9.6               | Notes on Using Time-Base Timer .....                                   | 196        |
| 9.7               | Sample Programs of Time-Base Timer .....                               | 197        |
| <b>CHAPTER 10</b> | <b>WATCHDOG TIMER .....</b>  | <b>199</b> |
| 10.1              | Overview of Watchdog Timer .....                                       | 200        |
| 10.2              | Watchdog Timer Control Register (WDTC) .....                           | 201        |
| 10.3              | Watchdog Timer Configuration .....                                     | 203        |
| 10.4              | Watchdog Timer Operation .....   | 205        |
| 10.5              | Notes on Using Watchdog Timer .....                                    | 207        |
| 10.6              | Sample Programs of Watchdog Timer .....                                | 208        |
| <b>CHAPTER 11</b> | <b>WATCH TIMER .....</b>   | <b>209</b> |
| 11.1              | Overview of Watch Timer .....  | 210        |
| 11.2              | Watch Timer Configuration .....  | 211        |
| 11.3              | Watch Timer Control Register (WTC) .....                               | 212        |
| 11.4              | Watch Timer Operation .....  | 214        |
| <b>CHAPTER 12</b> | <b>16-BIT INPUT/OUTPUT TIMER .....</b>                                 | <b>217</b> |
| 12.1              | Overview of 16-bit Input/Output Timer .....                            | 218        |
| 12.2              | Configuration of 16-bit Input/Output Timer .....                       | 219        |
| 12.3              | Configuration and Function of 16-bit Input/Output Timer Register ..... | 223        |
| 12.3.1            | Free-Run Timer .....   | 224        |
| 12.3.2            | Output compare .....   | 229        |
| 12.3.3            | Input capture .....  | 233        |
| 12.4              | Interrupt of 16-bit Input/Output Timer .....                           | 235        |
| 12.5              | 16-bit Input/Output Timer Operation .....                              | 237        |
| 12.5.1            | Operation of Free-Run Timer .....                                      | 238        |
| 12.5.2            | Operation of output compare .....                                      | 240        |
| 12.5.3            | Operation of input capture .....                                       | 242        |
| 12.5.4            | Free-Run Timer Timing .....  | 243        |



|                   |   |            |
|-------------------|---|------------|
| 12.5.5            | Output compare timing .....   | 244        |
| 12.5.6            | Timing of input capture .....   | 245        |
| 12.6              | Program Example of 16-bit Input/Output Timer .....                                | 246        |
| <b>CHAPTER 13</b> | <b>8/16-BIT UP/DOWN COUNTER/TIMER .....</b>                                       | <b>259</b> |
| 13.1              | Overview of 8/16-bit Up/Down Counter Timer .....                                  | 260        |
| 13.2              | Configuration of 8/16-bit Up/Down Counter/Timer .....                             | 261        |
| 13.3              | Configuration and Functions of Registers for 8/16-bit Up/Down Counter/Timer ..... | 264        |
| 13.3.1            | Counter control register (ch.0) upper (CCR0) .....                                | 265        |
| 13.3.2            | Counter control register (ch.1) upper (CCR1) .....                                | 267        |
| 13.3.3            | Counter control register (ch.0/ch.1) lower (CCRL0/CCRL1) .....                    | 269        |
| 13.3.4            | Counter status register 0/1 (CSR0/CSR1) .....                                     | 271        |
| 13.3.5            | Up/down count register (ch.0/ch.1) (UDCR0/UDCR1) .....                            | 273        |
| 13.3.6            | Reload/compare register (ch.0/ch.1) (RCR0/RCR1) .....                             | 274        |
| 13.4              | Interrupt of 8/16-bit Up/Down Counter/Timer .....                                 | 275        |
| 13.5              | 8/16-bit Up/Down Counter/Timer Operation .....                                    | 277        |
| 13.5.1            | Reload/compare function .....   | 280        |
| 13.5.2            | Writing data to up/down count register (UDCR) .....                               | 283        |
| 13.6              | Program Example of 8/16-bit Up/Down Counter/Timer .....                           | 285        |
| <b>CHAPTER 14</b> | <b>16-BIT RELOAD TIMER .....</b>  | <b>291</b> |
| 14.1              | Overview of 16-Bit Reload Timer .....   | 292        |
| 14.2              | Configuration and Functions of 16-Bit Reload Timer Registers .....                | 296        |
| 14.2.1            | Timer Control Status Register (TMCSR) .....                                       | 297        |
| 14.2.2            | 16-Bit Timer Register (TMR)/16-Bit Reload Register (TMRLR) .....                  | 301        |
| 14.3              | Interrupt of 16-Bit Reload Timer .....  | 303        |
| 14.4              | Operations of the 16-Bit Reload Timer .....                                       | 304        |
| 14.4.1            | State Transitions During Count Operation .....                                    | 305        |
| 14.4.2            | Operations of Internal Clock Mode (Reload Mode) .....                             | 306        |
| 14.4.3            | Internal Clock Mode (One-Shot Mode) .....   | 308        |
| 14.4.4            | Event Count Mode .....  | 310        |
| 14.5              | Program Example of 16-Bit Reload Timer .....                                      | 312        |
| <b>CHAPTER 15</b> | <b>8/16-BIT PPG TIMER .....</b>   | <b>317</b> |
| 15.1              | Overview of 8/16-Bit PPG Timer .....  | 318        |
| 15.2              | Configuration of 8/16-Bit PPG Timer .....   | 319        |
| 15.3              | Configuration and Functions of 8/16-Bit PPG Timer Registers .....                 | 322        |
| 15.3.1            | PPG0/2/4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4) .....                | 323        |
| 15.3.2            | PPG1/3/5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5) .....                | 325        |
| 15.3.3            | PPG0 to PPG5 Output Control Registers (PPG01, PPG23, PPG45) .....                 | 328        |
| 15.3.4            | Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5) .....                             | 330        |
| 15.4              | Interrupt of 8/16-Bit PPG Timer .....   | 331        |
| 15.5              | Operations of 8/16-Bit PPG Timer .....  | 333        |
| 15.6              | Program Example of 8/16-Bit PPG Timer .....                                       | 338        |

|   |                |
|---|----------------|
| <b>CHAPTER 16 DTP/EXTERNAL INTERRUPTS .....</b>                                   | <b>341</b>     |
| 16.1 Overview of DTP/External Interrupt Unit .....                                | 342            |
| 16.2 Configuration and Functions of DTP/External Interrupt Unit Registers .....   | 344            |
| 16.3 DTP/External Interrupt .....   | 346            |
| 16.4 Operations of DTP/External Interrupt Unit .....                              | 348            |
| 16.5 Precautions on Use of DTP/External Interrupt Unit .....                      | 350            |
| 16.6 Program Example of DTP/External Interrupt .....                              | 352            |
| <br><b>CHAPTER 17 8/10-BIT A/D CONVERTER .....</b>                                | <br><b>355</b> |
| 17.1 Overview of 8/10-Bit A/D Converter .....                                     | 356            |
| 17.2 Configuration of 8/10-Bit A/D Converter .....                                | 357            |
| 17.3 Configuration and Functions of 8/10-Bit A/D Converter Registers .....        | 359            |
| 17.3.1 Control Status Register 1 (ADCS1) .....                                    | 360            |
| 17.3.2 Control Status Register 2 (ADCS2) .....                                    | 363            |
| 17.3.3 Data Registers (ADCR2 and ADCR1) .....                                     | 366            |
| 17.4 Interrupt of 8/10-Bit A/D Converter .....                                    | 367            |
| 17.5 Operations of 8/10-Bit A/D Converter .....                                   | 368            |
| 17.5.1 Example of $\mu$ DMAC Start in Single Mode .....                           | 370            |
| 17.5.2 Example of $\mu$ DMAC Start in Continuous Mode .....                       | 372            |
| 17.5.3 Example of $\mu$ DMAC Start in Stop Mode .....                             | 374            |
| 17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter .....          | 376            |
| 17.7 Precautions on use of the 8/10-Bit A/D Converter .....                       | 378            |
| 17.8 Program Example of 8/10-Bit A/D Converter .....                              | 379            |
| <br><b>CHAPTER 18 EXPANDED I/O SERIAL INTERFACE .....</b>                         | <br><b>385</b> |
| 18.1 Overview of Expanded I/O Serial Interface .....                              | 386            |
| 18.2 Configuration of Expanded I/O Serial Interface .....                         | 387            |
| 18.3 Configuration and Functions of Expanded I/O Serial Interface Registers ..... | 389            |
| 18.3.1 Serial Mode Control Status Register 0/1 (SMCS0/SMCS1) .....                | 390            |
| 18.3.2 Serial Data Register 0/1 (SDR0/SDR1) .....                                 | 394            |
| 18.3.3 Communication Prescaler Control Register0/1 (SDCR0/SDCR1) .....            | 395            |
| 18.4 Interrupt of Expanded I/O Serial Interface .....                             | 396            |
| 18.5 Operation of Expanded I/O Serial Interface .....                             | 397            |
| 18.5.1 Shift Clock Modes .....  | 398            |
| 18.5.2 Operational States of Serial I/O Units .....                               | 399            |
| 18.5.3 Start/Stop Timing and Input/Output Timing of Shift Operation .....         | 401            |
| 18.5.4 Interrupt Function .....   | 403            |
| 18.6 Program Example of Expanded I/O Serial Interface .....                       | 404            |
| <br><b>CHAPTER 19 UART .....</b>  | <br><b>407</b> |
| 19.1 Overview of the UART .....   | 408            |
| 19.2 Configuration of UART .....  | 409            |
| 19.3 Configuration and Functions of UART Registers .....                          | 411            |
| 19.3.1 Serial Mode Register (SMR) .....   | 412            |
| 19.3.2 Serial Control Register (SCR) .....  | 414            |
| 19.3.3 Serial Input/Output Register (SIDR/SODR) .....                             | 416            |
| 19.3.4 Serial Status Register (SSR) .....   | 417            |

|                   |  |            |
|-------------------|--|------------|
| 19.3.5            | Communication Prescaler Control Register (CDCR)                  | 419        |
| 19.4              | Interrupt of UART  | 421        |
| 19.5              | UART Operations  | 423        |
| 19.5.1            | Operation in Asynchronous Mode (Operation Modes 0 and 1)         | 427        |
| 19.5.2            | Operation in Synchronous Mode (Operation Mode 2)                 | 430        |
| 19.5.3            | Two-Way Communication Function (Normal Mode)                     | 432        |
| 19.5.4            | Master/Slave Communication Function (Multiprocessor Mode)        | 434        |
| 19.6              | Precautions on use of the UART                                   | 437        |
| 19.7              | Program Example of UART  | 438        |
| <b>CHAPTER 20</b> | <b>CHIP SELECTION FACILITY</b>                                   | <b>445</b> |
| 20.1              | Overview of Chip Selection Facility                              | 446        |
| 20.2              | Configuration of Chip Selection Facility                         | 447        |
| 20.3              | Configuration and Functions of Chip Selection Facility Registers | 449        |
| 20.3.1            | Chip Select Area MASK Register (CMRx)                            | 450        |
| 20.3.2            | Chip Selection Area Register (CARx)                              | 451        |
| 20.3.3            | Chip Selection Control Register (CSCR)                           | 452        |
| 20.3.4            | Chip Selection Active Level Register (CALR)                      | 453        |
| 20.4              | Operation of the Chip Selection Facility                         | 454        |
| <b>CHAPTER 21</b> | <b>ADDRESS MATCH DETECTION FUNCTION</b>                          | <b>457</b> |
| 21.1              | Overview of Address Match Detection Function                     | 458        |
| 21.2              | Block Diagram of Address Match Detection Function                | 459        |
| 21.3              | Configuration of Registers for Address Match Detection Function  | 460        |
| 21.3.1            | Program Address Detection Control Status Register (PACSR)        | 461        |
| 21.3.2            | Program Address Detection Registers (PADR0, PADR1)               | 463        |
| 21.4              | Explanation of Operation of Address Match Detection Function     | 465        |
| 21.4.1            | Example of using Address Match Detection Function                | 466        |
| 21.5              | Program Example of Address Match Detection Function              | 471        |
| <b>CHAPTER 22</b> | <b>ROM MIRROR FUNCTION SELECTION MODULE</b>                      | <b>473</b> |
| 22.1              | Overview of ROM Mirror Function Selection Module                 | 474        |
| 22.2              | ROM Mirror Function Selection Register (ROMM)                    | 475        |
| <b>CHAPTER 23</b> | <b>2M/3M BIT FLASH MEMORY</b>                                    | <b>477</b> |
| 23.1              | Overview of 2M/3M Bit Flash Memory                               | 478        |
| 23.2              | Sector Configuration of 2M/3M Bit Flash Memory                   | 479        |
| 23.3              | Flash memory Control Status Register (FMCS)                      | 480        |
| 23.4              | Method for Starting the Flash Memory's Automatic Algorithm       | 486        |
| 23.5              | Verifying the Execution State of the Automatic Algorithm         | 487        |
| 23.5.1            | Data Polling Flag (DQ7)  | 489        |
| 23.5.2            | Toggle Bit Flag (DQ6)  | 491        |
| 23.5.3            | Timing Limit Excess Flag (DQ5)                                   | 492        |
| 23.5.4            | Sector Erase Timer Flag (DQ3)                                    | 493        |
| 23.6              | Flash Memory Write/Erase Operations                              | 494        |
| 23.6.1            | Setting the Flash Memory to Read/Reset State                     | 495        |
| 23.6.2            | Writing Data to Flash Memory                                     | 496        |

|  |  |            |
|--|--|------------|
| 23.6.3   | Erasing All Data in the Flash Memory (Chip Erase)  | 498        |
| 23.6.4   | Erasing Arbitrary Data in Flash Memory (Sector Erase)  | 499        |
| 23.6.5   | Suspending Sector Erasure for the Flash Memory   | 501        |
| 23.6.6   | Resuming the Sector Erasure of Flash Memory  | 502        |
| 23.7   | Flash Security Function  | 503        |
| <b>CHAPTER 24 EXAMPLES OF MB90F481B/MB90F482B/MB90F488B/MB90F489B SERIAL PROGRAMMING CONNECTION.....</b> |  | <b>505</b> |
| 24.1   | Basic Configuration of Serial Programming Connection with MB90F481B/MB90F482B/MB90F488B/MB90F489B..... | 506        |
| 24.2   | Example of Connection in Single-Chip Mode (When Using the User Power Supply)                           | 510        |
| 24.3   | Example of Minimum Connection with Flash Microcontroller Programmer (When Using the User Power Supply) | 512        |
| <b>CHAPTER 25 PWC TIMER (ONLY MB90485 SERIES) .....</b>  |  | <b>515</b> |
| 25.1   | Overview of PWC Timer  | 516        |
| 25.2   | Configuration of PWC Timer   | 517        |
| 25.3   | Configuration and Functions of PWC Timer Registers   | 519        |
| 25.3.1   | PWC Control/Status Register (PWCSR0 to PWCSR2)   | 520        |
| 25.3.2   | PWC Data Buffer Register (PWCR0 to PWCR2)  | 525        |
| 25.3.3   | Divide Ratio Control Register (DIVR0 to DIVR2)   | 526        |
| 25.4   | Interrupt of PWC Timer   | 527        |
| 25.5   | Operations of PWC Timer  | 529        |
| 25.5.1   | Operations of the Timer Function   | 530        |
| 25.5.2   | Operations of the Pulse Width Measurement Function   | 531        |
| 25.5.3   | Selection of Count Clock and Operation Mode  | 533        |
| 25.5.4   | Start and Stop of Timer/Pulse Width Measurement  | 535        |
| 25.5.5   | Timer Mode Operation   | 537        |
| 25.5.6   | Operation in Pulse Width Measurement Mode  | 540        |
| 25.6   | Notes on PWC Timer Usage   | 546        |
| <b>CHAPTER 26 <math>\mu</math>PG TIMER (ONLY MB90485 SERIES) .....</b>                                   |  | <b>549</b> |
| 26.1   | Overview and Configuration of $\mu$ PG Timer   | 550        |
| 26.2   | Configuration and Functions of $\mu$ PG Timer Registers  | 552        |
| 26.3   | Timing Chart of $\mu$ PG Timer   | 554        |
| <b>CHAPTER 27 I<sup>2</sup>C INTERFACE (ONLY MB90485 SERIES) .....</b>                                   |  | <b>555</b> |
| 27.1   | Overview of I <sup>2</sup> C Interface   | 556        |
| 27.2   | Configuration of I <sup>2</sup> C Interface  | 557        |
| 27.3   | Configuration and Functions of I <sup>2</sup> C Interface Registers                                    | 559        |
| 27.3.1   | Bus Status Register (IBSR)   | 560        |
| 27.3.2   | Bus Control Register (IBCR)  | 562        |
| 27.3.3   | Clock Control Register (ICCR)  | 568        |
| 27.3.4   | Address Register (IADR)  | 570        |
| 27.3.5   | Data Register (IDAR)   | 571        |
| 27.4   | Interrupt of I <sup>2</sup> C Interface  | 572        |
| 27.5   | I <sup>2</sup> C Interface Operation   | 574        |

|   |            |
|---|------------|
| <b>APPENDIX .....</b>   | <b>579</b> |
| APPENDIX A Memory Map .....   | 580        |
| APPENDIX B I/O Map .....  | 584        |
| APPENDIX C Interrupt Source, Interrupt Vector, and Interrupt Control Register ..... | 592        |
| APPENDIX D Instructions .....   | 594        |
| D.1 Instruction Types .....   | 595        |
| D.2 Addressing .....  | 596        |
| D.3 Direct Addressing .....   | 598        |
| D.4 Indirect Addressing .....   | 604        |
| D.5 Execution Cycle Count .....   | 612        |
| D.6 Effective address field .....   | 615        |
| D.7 How to Read the Instruction List .....  | 616        |
| D.8 F <sup>2</sup> MC-16LX Instruction List .....                                   | 619        |
| D.9 Instruction Map .....   | 633        |
| <b>Index .....</b>  | <b>655</b> |

# Main changes in this edition

| Page       | Changes (For details, refer to main body.)           |
|------------|--|
| 594 to 654 | Changed the entire part of "APPENDIX D Instructions" |

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

## Reference: Main changes (Rev.5 → Rev.6)

| Page | Changes (For details, refer to main body.)  |
|------|---|
| -    | Product name is changed.<br>(MB90483B → MB90483C)<br>Series name is changed.<br>(MB90480 SERIES → MB90480/485 SERIES)                             |
| i    | ■ License is added.   |
| 11   | Function, Pin name D00 to D07 and D08 to D15 in Table 1.5-1 Pin Functions (1/7).<br>(output pin → input/output pin)                               |
| 93   | Figure 3.11-2 Operation of Delay Interrupt Generation Module is changed.<br>(ICR <sub>XX</sub> → IL)<br>(ICR <sub>XX</sub> → ILM)<br>(NTA → INTA) |
| 157  | Figure 7.3-1 Relationship Between Access Areas and Physical Addresses is changed.<br>(No access → Access Inhibited)                               |
| 350  | ■ Conditions for External Connection of Peripheral Devices is changed.<br>((interim value) is deleted.)   |
| 390  | The Figure is changed in ■ Serial Mode Control Status Register0/1 (SMCS0/SMCS1).<br>(Deleted the description; *1 and *2.)                         |
| 531  | Figure 25.5-2 Pulse Width Measurement Operation (One-shot Measurement Mode/"H" Level Pulse Width Measurement) is changed.                         |
| 532  | Figure 25.5-3 Pulse Width Measurement Operation (Repeated Measurement Mode/"H" Level Pulse Width Measurement) is changed.                         |
| 574  | Note is added.  |
| 580  | Figure A-1 Memory Map is changed.<br>(No access → Access inhibited)   |
| 581  | Table A-1 Relationship Among Address #1, Address #2, and Address #3 by Product Type is changed.   |
| 582  | Figure A-2 MB90F489B Memory Map is changed.<br>(No access → Access inhibited)   |
| 583  | Figure A-3 MB90483C Memory Map is added.  |

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



# CHAPTER 1      OVERVIEW OF MB90480/485 SERIES

---

**This chapter gives an overview of MB90480/485 series, including its basic features and basic specifications.**

---

- 1.1 Features of MB90480/485 Series
- 1.2 Block Diagram of MB90480/485 Series
- 1.3 Package Dimensions
- 1.4 Pin Assignment
- 1.5 Pin Functions
- 1.6 I/O Circuit Type
- 1.7 Handling the Device



## 1.1 Features of MB90480/485 Series

---

**MB90480/485 series is a 16-bit microcontroller designed for applications such as consumer devices requiring high-speed real-time processing.**

---

### ■ MB90480/485 Series Features

The MB90480/485 series has the following features:

- **Minimum instruction execution time**
  - 40.0 ns/6.25 MHz oscillation multiplied by 4 (25 MHz/3.3 V  $\pm$  0.3 V for internal operation)
  - 62.5 ns/4 MHz oscillation multiplied by 4 (16 MHz/3.0 V  $\pm$  0.3 V for internal operation)
  - PLL clock multiply system
- **Maximum memory space: 16 Mbytes**
- **Instruction system optimized for control applications**
  - Available data types: bit, byte, word, long word
  - Standard addressing modes: 23 types
  - Improved high-precision operation using a 32-bit accumulator
  - Signed multiply and divide operations, extensive RETI instruction
- **Instruction system supporting multitasking in high-level languages (such as C)**
  - Use of a system stack pointer
  - Symmetry of instruction sets and barrel shift instructions
- **Nonmultiplex bus and multiplex bus support**
- **Improved execution speed: 4-byte queue**
- **Improved interrupt function (priority is a programmable setting of up to 8 levels): 8 external interrupts**
- **Data transfer function ( $\mu$ DMAC): maximum of 16 channels**
- **Built-in ROM: Flash version: 192 K bytes/256 K bytes/384K bytes,  
Mask version: 192 K bytes/256K bytes**
- **Built-in RAM: Flash version: 4 K bytes/6 K bytes/10 K bytes/24K bytes,  
Mask version: 10 K bytes/16K bytes**
- **General-purpose ports: maximum of 84 ports**
  - (Input pull-up resistor settings available: 16 ports)
  - (Output open-drain settings available: 16 ports)

- **A/D converter (RC step-by-step compare type): 8 channels**  
(resolution: 10 bits; conversion time: 3.68  $\mu$ s (25 MHz operation))
- **UART: 1 channel**
- **Extensive I/O serial interface (SIO): 2 channels**
- **8/16-bit PPG: 3 channels**  
(8 bits  $\times$  6 channels and 16 bits  $\times$  3 channels with mode switching function)
- **8/16-bit up/down timer: 1 channel**  
(8 bits  $\times$  2 channels or 16 bits  $\times$  1 channel with mode switching function)
- **16-bit reload timer: 1 channel**
- **16-bit input/output timer**  
(input capture  $\times$  2 channels; output compare  $\times$  6 channels; free-run timer  $\times$  1 channel)
- **Built-in dual-system clock generator**
- **Power-saving mode**  
(stop mode, sleep mode, CPU intermittent operation mode, watch mode/time-base timer mode)
- **Package: QFP100/LQFP100**
- **CMOS technology**
- **3V single power supply**  
(On only MB90485 series, some port can be operated on 5V power supply.)
- **I<sup>2</sup>C interface\* 1 channel (only for MB90485 series)**  
Only MB90487B has built-in I<sup>2</sup>C interface, and its pins (P77/P76) are N-ch open drain pins (not P-ch).
- **16-bits PWC: 3 channel (only for MB90485 series)**  
2 channels of the 3 channels have the function of input compare.
- **$\mu$ PG: 1 channel (only for MB90485 series)**

\*: I<sup>2</sup>C license

Purchase of Fujitsu I<sup>2</sup>C components conveys a license under the Philips I<sup>2</sup>C Patent Rights to use, these components in an I<sup>2</sup>C system provided that the system conforms to the I<sup>2</sup>C Standard Specification as defined by Philips.

## ■ Product Configuration

Table 1.1-1 is an outline of the MB90480 series product configuration and Table 1.1-2 is an outline of the MB90485 series product configuration.

**Table 1.1-1 MB90480 Series Product Configuration**

|   | MB90V480B                                 | MB90F481B  | MB90F482B  |
|---|---|--|--|
| Product configuration                         | EVA function                              | Flash memory product                                       |  |
| ROM capacity                                  | -   | 192K bytes   | 256K bytes   |
| RAM capacity                                  | 16K bytes                                 | 4K bytes   | 6K bytes   |
| Description                                   | 3-V/5-V version of user pin <sup>*1</sup> | f=25MHz<br>3V version of user pins<br>No security function | f=25MHz<br>3V version of user pins<br>No security function |
| Emulator-dedicated power supply <sup>*2</sup> | Provided                                  | -  | -  |

\*1: User pins: P20 to P27, P30 to P37, P40 to P47, P70 to P77

\*2: It is setting of jumper switch (TOOL VCC) when Emulator (MB2147-01) is used. Please refer to the MB2147-01 or MB2147-20 hardware manual (3.3 Emulator-dedicated Power Supply Switching) about details.

**Table 1.1-2 MB90485 Series Product Configuration**

|   | MB90V485B  | MB90487B   | MB90F488B  | MB90488B   | MB90F489B  | MB90483C   |
|---|--|--|--|--|--|--|
| Product configuration                         | EVA function   | Mask ROM product   | Flash memory product   | Mask ROM product   | Flash memory product   | Mask ROM product   |
| ROM capacity                                  | -  | 192K bytes   | 256K bytes   | 256K bytes   | 384K bytes   | 256K bytes   |
| RAM capacity                                  | 16K bytes  | 10K bytes  | 10K bytes  | 10K bytes  | 24K bytes  | 16K bytes  |
| Description                                   | f=25MHz<br>3-V/5-V power supply <sup>*1</sup><br>Built-in PWC,<br>$\mu$ PG, I <sup>2</sup> C <sup>*2</sup> | f=25MHz<br>3-V/5-V power supply <sup>*1</sup><br>Built-in PWC,<br>$\mu$ PG, I <sup>2</sup> C <sup>*2</sup> | f=25MHz<br>3-V/5-V power supply <sup>*1</sup><br>Built-in PWC,<br>$\mu$ PG, I <sup>2</sup> C <sup>*2</sup><br>Provided security function | f=25MHz<br>3-V/5-V power supply <sup>*1</sup><br>Built-in PWC,<br>$\mu$ PG, I <sup>2</sup> C <sup>*2</sup> | f=25MHz<br>3-V/5-V power supply <sup>*1</sup><br>Built-in PWC,<br>$\mu$ PG, I <sup>2</sup> C <sup>*2</sup><br>Provided security function | f=25MHz<br>3-V/5-V power supply <sup>*1</sup><br>Built-in PWC,<br>$\mu$ PG, I <sup>2</sup> C <sup>*2</sup> |
| Emulator-dedicated power supply <sup>*3</sup> | Provided   | -  | -  | -  | -  | -  |

\*1: 3-V/5-V: I/F pins (P20 to P27, P30 to P37, P40 to P47, and P70 to P77)

The power supply for the other pins is 3 V.

\*2: If I<sup>2</sup>C is set, P76/P77 pins are N-ch open drain pin (without P-ch).

\*3: It is setting of jumper switch (TOOL VCC) when Emulator (MB2147-01) is used. Please refer to the MB2147-01 or MB2147-20 hardware manual (3.3 Emulator-dedicated Power Supply Switching) about details.

## ■ Package of Corresponding Products

### ○ Package

Differences among packages are shown below.

**Table 1.1-3 MB90480/485 Series Package and Correspondence of Product**

| Product<br>Package | MB90487B/488B<br>MB90483C | MB90F481B/F482B<br>MB90F488B/F489B | MB90V480B<br>MB90V485B |
|--------------------|---------------------------|------------------------------------|------------------------|
| FPT-100P-M06       | ○                         | ○                                  | ×                      |
| FPT-100P-M05       | ○                         | ○                                  | ×                      |
| PGA-299C           | ×                         | ×                                  | ○                      |

○:Usable

×:No usable

### ○ Memory space

Differences among memory space are shown in memory map in appendix.

### ○ Power supply consumption

Differences among power supply consumption, refer to ELECTRICAL CHARACTERISTICS in datasheet.

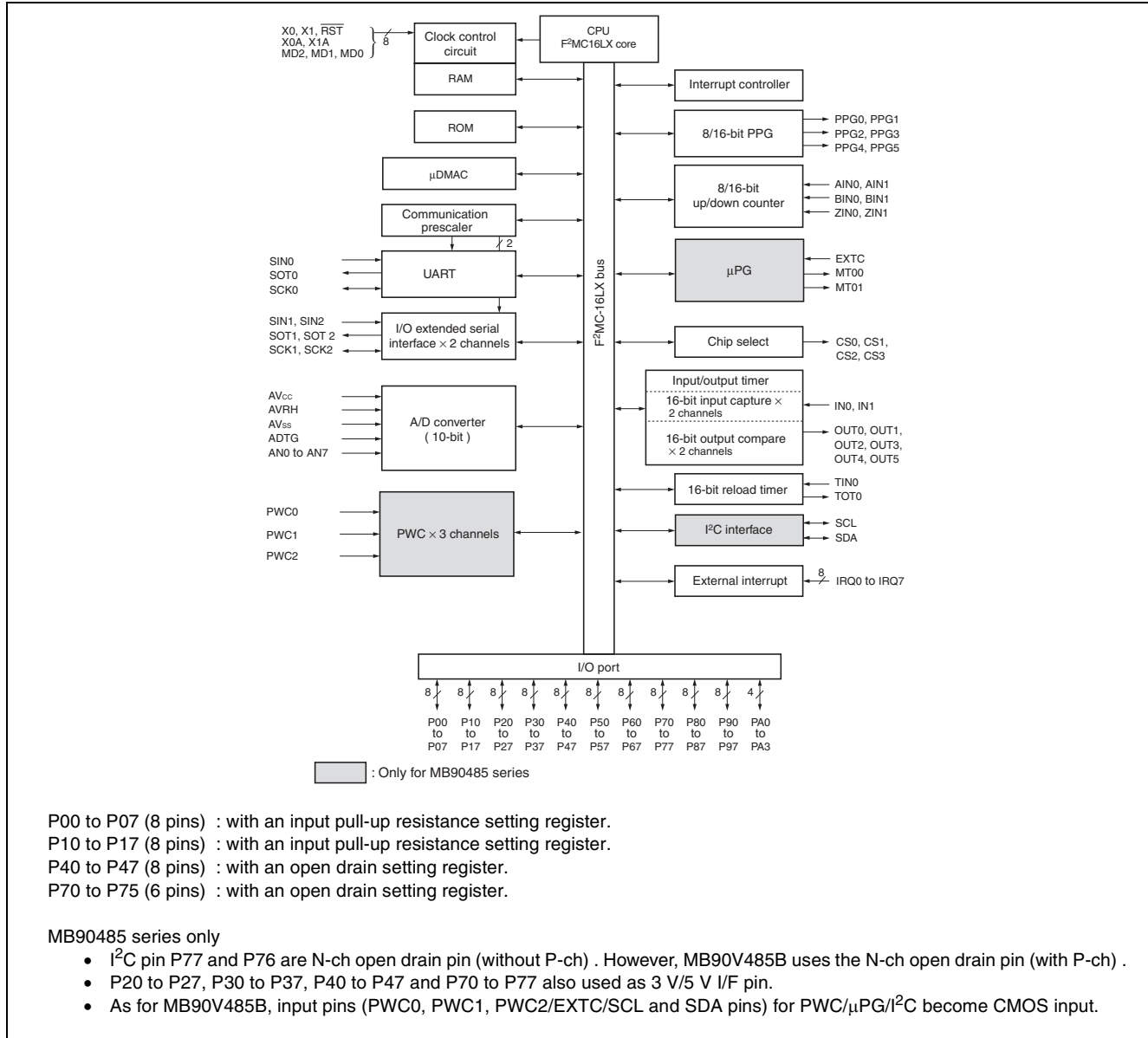
## 1.2 Block Diagram of MB90480/485 Series

This section has a block diagram of the MB90480/485 series.

### ■ Block Diagram of MB90480/485 Series

Figure 1.2-1 is a block diagram of the MB90480/485 series.

Figure 1.2-1 Block Diagram of MB90480/485 Series



Note:

In the Figure 1.2-1, the I/O port shares a pin with each built-in function block. The pin cannot be used as an I/O port if it is used as a built-in module pin.

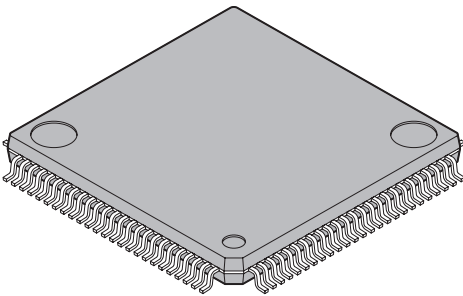
## 1.3 Package Dimensions

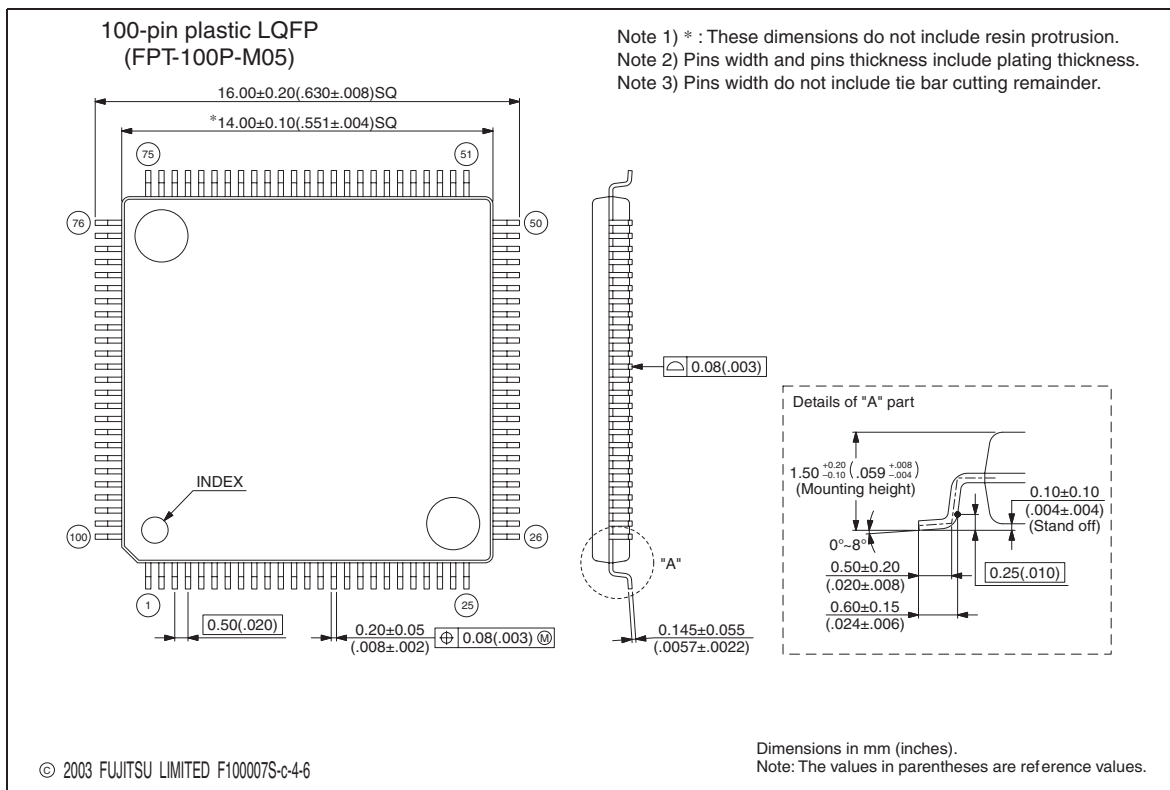
MB90480/485 series has two types of packages.

### ■ Package Dimensions (LQFP-100)

Figure 1.3-1 is a diagram for the package dimensions of the LQFP-100 type.

**Figure 1.3-1 Package Dimensions of LQFP-100 Type**

|   |                                |                       |
|---|--------------------------------|-----------------------|
|  <p>100-pin plastic LQFP</p> <p>(FPT-100P-M05)</p> | Lead pitch                     | 0.50 mm               |
|   | Package width × package length | 14.0 × 14.0 mm        |
|   | Lead shape                     | Gullwing              |
|   | Sealing method                 | Plastic mold          |
|   | Mounting height                | 1.70 mm MAX           |
|   | Weight                         | 0.65g                 |
|   | Code (Reference)               | P-LFQFP100-14×14-0.50 |



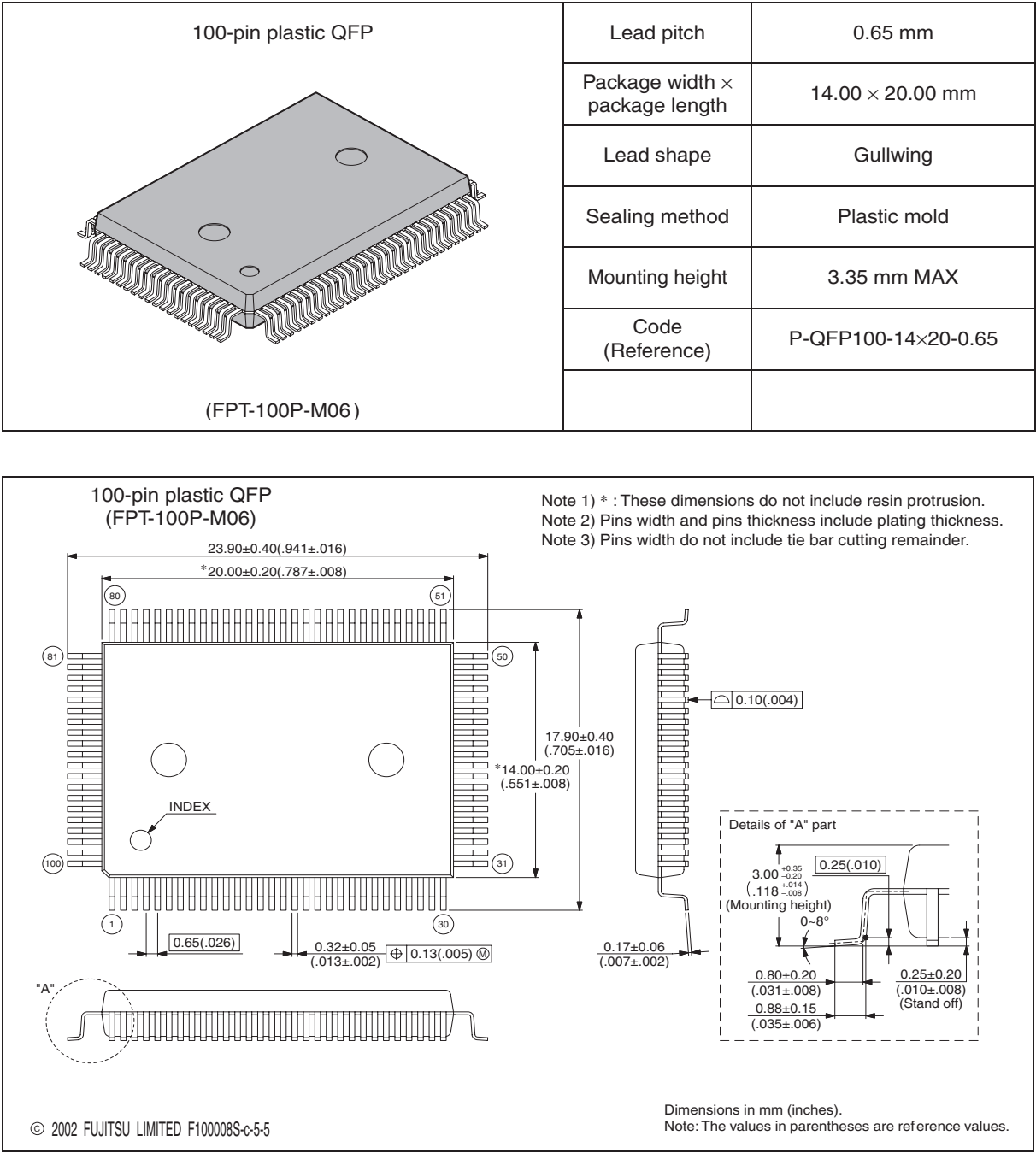
Please confirm the latest Package dimension by following URL.

<http://edevic.fujitsu.com/fj/DATASHEET/ef-ovpkiv.html>

■ Package Dimensions (QFP-100)

Figure 1.3-2 is a diagram for the package dimensions of the QFP-100 type.

Figure 1.3-2 Package Dimensions of QFP-100 Type



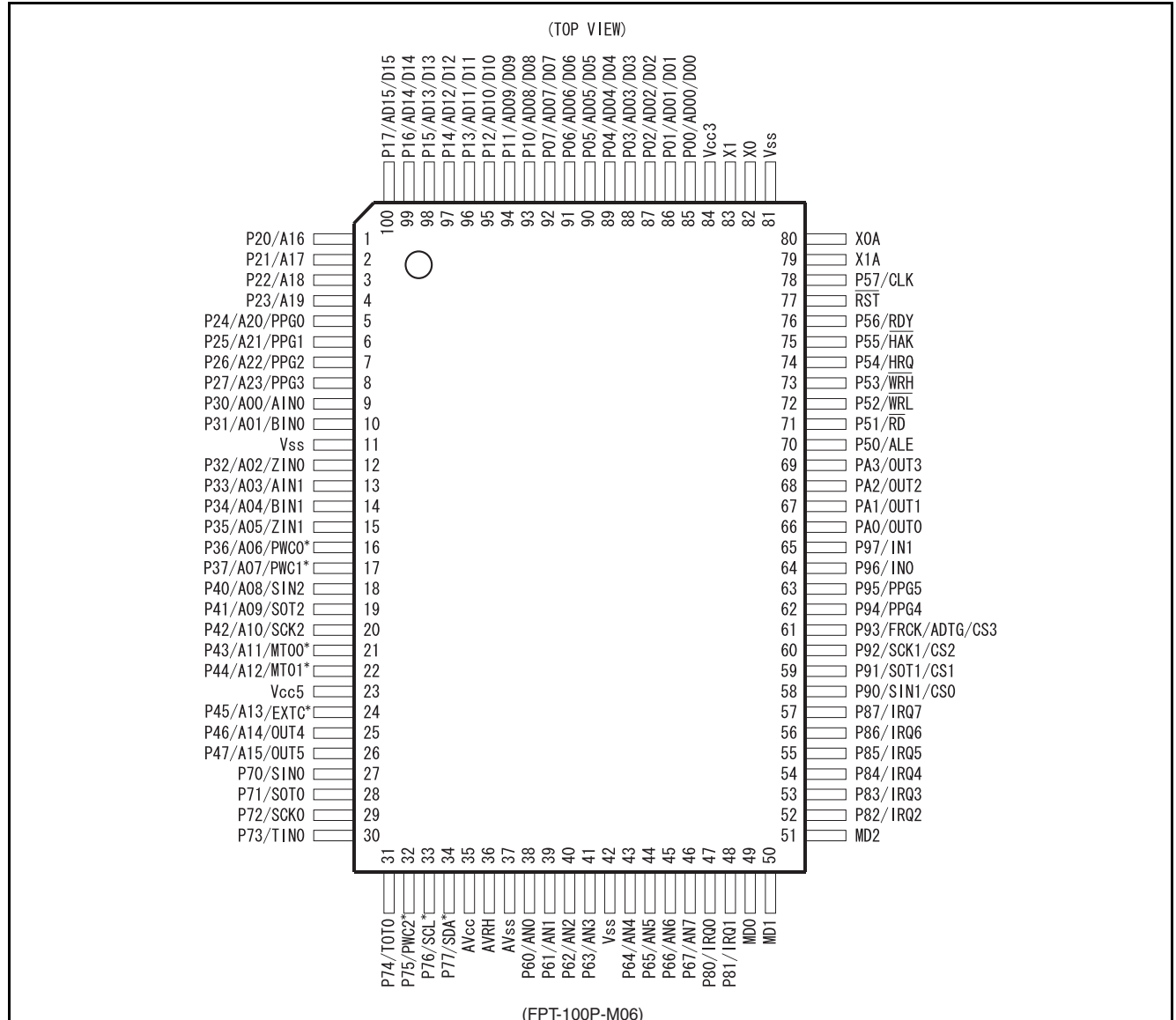
## 1.4 Pin Assignment

This section shows the MB90480/485 series pin assignments for two types of packages.

### ■ Pin Assignment Diagram (QFP-100)

Figure 1.4-1 is a pin assignment diagram for the QFP-100 type.

Figure 1.4-1 Pin Assignment Diagram of MB90480/485 Series (QFP-100)



\*: These pins are only for MB90485 series and correspond to P36/A06, P37/A07, P43/A11, P44/A12, P45/A13, P75, P76, and P77 for MB90480 series.

MB90480 series only

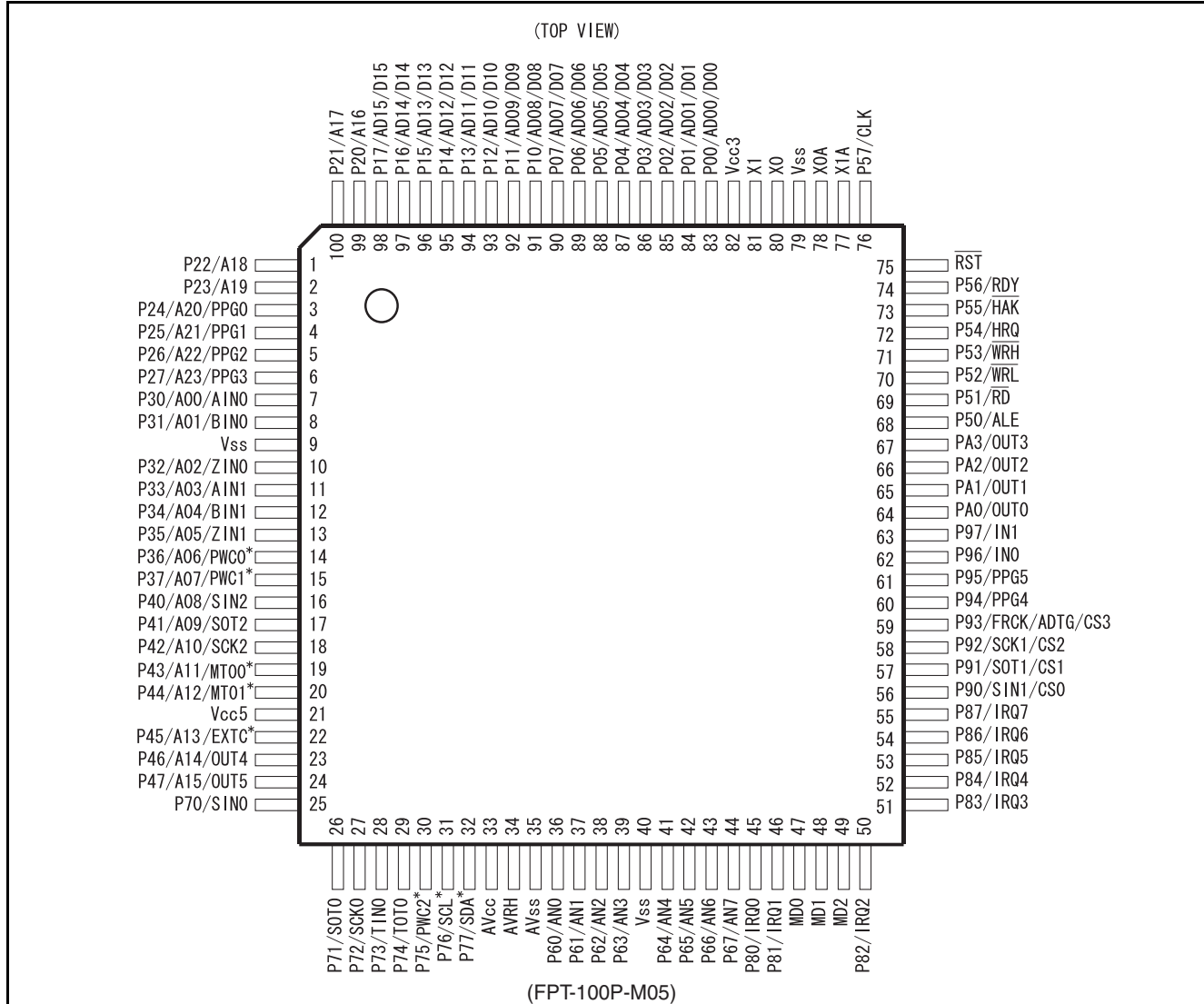
- I<sup>2</sup>C pin P77 and P76 are N-ch open drain pin (without P-ch). However, MB90V485B uses the N-ch open drain pin (with P-ch).
- P20 to P27, P30 to P37, P40 to P47 and P70 to P77 also used as 3 V/5 V I/F pin.
- As for MB90V485B, input pins (PWC0, PWC1, PWC2/EXTC/SCL and SDA pins) for PWC/μPG/I<sup>2</sup>C become CMOS input.



## ■ Pin Assignment Diagram (LQFP-100)

Figure 1.4-2 is a pin assignment diagram for the LQFP-100 type.

**Figure 1.4-2 Pin Assignment Diagram of MB90480/485 Series (LQFP-100)**



\*: These pins are only for MB90485 series and correspond to P36/A06, P37/A07, P43/A11, P44/A12, P45/A13, P75, P76, and P77 for MB90480 series.

MB90480 series only

- I<sup>2</sup>C pin P77 and P76 are N-ch open drain pin (without P-ch). However, MB90V485B uses the N-ch open drain pin (with P-ch) .
- P20 to P27, P30 to P37, P40 to P47 and P70 to P77 also used as 3 V/5 V I/F pin.
- As for MB90V485B, input pins (PWC0, PWC1, PWC2/EXTC/SCL and SDA pins) for PWC/μPG/I<sup>2</sup>C become CMOS input.

## 1.5 Pin Functions

This section explains the pin functions of the MB90480/485 series.

### ■ Pin Functions

Table 1.5-1 explains the pin functions of MB90480/485 series.

**Table 1.5-1 Pin Functions (1/7)**

| Pin number                 |                            | Pin name                | Circuit <sup>3</sup> | Function   |
|----------------------------|----------------------------|-------------------------|----------------------|--|
| FPT-100P-M06 <sup>*1</sup> | FPT-100P-M05 <sup>*2</sup> |                         |                      |  |
| 82                         | 80                         | X0                      | A                    | Oscillation pin  |
| 83                         | 81                         | X1                      | A                    | Oscillation pin  |
| 80                         | 78                         | X0A                     | A                    | 32-kHz oscillation pin   |
| 79                         | 77                         | X1A                     | A                    | 32-kHz oscillation pin   |
| 77                         | 75                         | $\overline{\text{RST}}$ | B                    | Reset input pin  |
| 85<br>to<br>92             | 83<br>to<br>90             | P00 to P07              | C<br>(CMOS)          | General-purpose input/output port.<br>With a register for Port 0 input register settings (RDR0), a pull-up resistor can be set to the enabled state (RD00-RD07="1") (disabled for the output setting).   |
|                            |                            | AD00 to AD07            |                      | Functions as the lower input/output pin of an external address and data bus in the multiplex mode.   |
|                            |                            | D00 to D07              |                      | Functions as the lower input/output pin of an external data bus in the non-multiplex mode.   |
| 93<br>to<br>100            | 91<br>to<br>98             | P10 to P17              | C<br>(CMOS)          | General-purpose input/output port.<br>With a register for Port 1 input register settings (RDR1), a pull-up resistor can be set to the enabled state (RD10-RD17="1") (disabled for the output setting).   |
|                            |                            | AD08 to AD15            |                      | Functions as the upper input/output pin of an external address and data bus in the multiplex mode.   |
|                            |                            | D08 to D15              |                      | Functions as the upper input/output pin of an external data bus in the non-multiplex mode.   |
| 1<br>to<br>4               | 99<br>100<br>1<br>2        | P20 to P23              | E<br>(CMOS/H)        | General-purpose input/output port.<br>Functions as the general-purpose input/output port in the external bus mode if the bit corresponding to external address output control register (HACR) is set to "1".   |
|                            |                            | A16 to A19              |                      | Functions as the upper output pin of an address (A16 to A19) in the multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0".<br>Functions as the upper output pin of an address (A16 to A19) in the non-multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0". |

**Table 1.5-1 Pin Functions (2/7)**

| Pin number     |                | Pin name     | Circuit*3     | Function   |
|----------------|----------------|--------------|---------------|--|
| FPT-100P-M06*1 | FPT-100P-M05*2 |              |               |  |
| 5<br>to<br>8   | 3<br>to<br>6   | P24 to P27   | E<br>(CMOS/H) | General-purpose input/output port.<br>Functions as the general-purpose input/output port in the external bus mode if the bit corresponding to external address output control register (HACR) is set to "1".   |
|                |                | A20 to A23   |               | Functions as the upper output pin of an address (A20 to A23) in the multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0".<br>Functions as the upper output pin of an address (A20 to A23) in the non-multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0". |
|                |                | PPG0 to PPG3 |               | Functions as the PPG-timer output pin.   |
| 9              | 7              | P30          | E<br>(CMOS/H) | General-purpose input/output port.   |
|                |                | A00          |               | Functions as an external address pin in the non-multi-bus mode.  |
|                |                | AIN0         |               | 8/16-bit up-down timer input pin (ch.0)  |
| 10             | 8              | P31          | E<br>(CMOS/H) | General-purpose input/output port  |
|                |                | A01          |               | Functions as an external address pin in the non-multiplex mode.  |
|                |                | BIN0         |               | 8/16-bit up-down timer input pin (ch.0)  |
| 12             | 10             | P32          | E<br>(CMOS/H) | General-purpose input/output port  |
|                |                | A02          |               | Functions as an external address pin in the non-multiplex mode.  |
|                |                | ZIN0         |               | 8/16-bit up-down timer input pin (ch.0)  |
| 13             | 11             | P33          | E<br>(CMOS/H) | General-purpose input/output port  |
|                |                | A03          |               | Functions as an external address pin in the non-multiplex mode.  |
|                |                | AIN1         |               | 8/16-bit up-down timer input pin (ch.1)  |
| 14             | 12             | P34          | E<br>(CMOS/H) | General-purpose input/output port  |
|                |                | A04          |               | Functions as an external address pin in the non-multiplex mode.  |
|                |                | BIN1         |               | 8/16-bit up-down timer input pin (ch.1)  |
| 15             | 13             | P35          | E<br>(CMOS/H) | General-purpose input/output port  |
|                |                | A05          |               | Functions as an external address pin in the non-multiplex mode.  |
|                |                | ZIN1         |               | 8/16-bit up-down timer input pin (ch.1)  |

Table 1.5-1 Pin Functions (3/7)

| Pin number                 |                            | Pin name                 | Circuit <sup>3</sup> | Function  |   |
|----------------------------|----------------------------|--------------------------|----------------------|---|---|
| FPT-100P-M06 <sup>*1</sup> | FPT-100P-M05 <sup>*2</sup> |                          |                      |   |   |
| 16<br>17                   | 14<br>15                   | P36, P37                 | D<br>(CMOS)          | MB90480<br>series   | General-purpose input/output port                               |
|                            |                            | A06, A07                 |                      |   | Functions as an external address pin in the non-multiplex mode. |
|                            |                            | P36, P37                 | E<br>(CMOS/H)        | MB90485<br>series   | General-purpose input/output port                               |
|                            |                            | A06, A07                 |                      |   | Functions as an external address pin in the non-multiplex mode. |
|                            |                            | PWC0, PWC1 <sup>*4</sup> |                      |   | Functions as a PWC input pin.                                   |
| 18                         | 16                         | P40                      | G<br>(CMOS/H)        | General-purpose input/output port                               |   |
|                            |                            | A08                      |                      | Functions as an external address pin in the non-multiplex mode. |   |
|                            |                            | SIN2                     |                      | Simple serial I/O input pin                                     |   |
| 19                         | 17                         | P41                      | F<br>(CMOS)          | General-purpose input/output port                               |   |
|                            |                            | A09                      |                      | Functions as an external address pin in the non-multiplex mode. |   |
|                            |                            | SOT2                     |                      | Simple serial I/O output pin                                    |   |
| 20                         | 18                         | P42                      | G<br>(CMOS/H)        | General-purpose input/output port                               |   |
|                            |                            | A10                      |                      | Functions as an external address pin in the non-multiplex mode. |   |
|                            |                            | SCK2                     |                      | Simple serial I/O clock input/output pin                        |   |
| 21<br>22                   | 19<br>20                   | P43, P44                 | F<br>(CMOS)          | MB90480<br>series   | General-purpose input/output port                               |
|                            |                            | A11, A12                 |                      |   | Functions as an external address pin in the non-multiplex mode. |
|                            |                            | P43, P44                 | F<br>(CMOS)          | MB90485<br>series   | General-purpose input/output port                               |
|                            |                            | A11, A12                 |                      |   | Functions as an external address pin in the non-multiplex mode. |
|                            |                            | MT00, MT01               |                      |   | μPG output pin.   |
| 24                         | 22                         | P45                      | F(CMOS)              | MB90480<br>series   | General-purpose input/output port                               |
|                            |                            | A13                      |                      |   | Functions as an external address pin in the non-multiplex mode. |
|                            |                            | P45                      | G<br>(CMOS/H)        | MB90485<br>series   | General-purpose input/output port                               |
|                            |                            | A13                      |                      |   | Functions as an external address pin in the non-multiplex mode. |
|                            |                            | EXTC <sup>*4</sup>       |                      |   | μPG input pin.  |

**Table 1.5-1 Pin Functions (4/7)**

| Pin number                 |                            | Pin name         | Circuit <sup>3</sup> | Function  |
|----------------------------|----------------------------|------------------|----------------------|---|
| FPT-100P-M06 <sup>*1</sup> | FPT-100P-M05 <sup>*2</sup> |                  |                      |   |
| 25<br>26                   | 23<br>24                   | P46, P47         | F(CMOS)              | General-purpose input/output port   |
|                            |                            | A14, A15         |                      | Functions as an external address pin in the non-multiplex mode.   |
|                            |                            | OUT4, OUT5       |                      | Functions as the output pin for output compare events.  |
| 70                         | 68                         | P50              | D<br>(CMOS)          | General-purpose input/output port. Functions as the ALE pin in the external bus mode.   |
|                            |                            | ALE              |                      | Functions as the address read permission signal (ALE) pin in the external bus mode.   |
| 71                         | 69                         | P51              | D<br>(CMOS)          | General-purpose input/output port. Functions as the $\overline{RD}$ pin in the external bus mode.   |
|                            |                            | $\overline{RD}$  |                      | Functions as the read strobe output ( $\overline{RD}$ ) pin in the external bus mode.   |
| 72                         | 70                         | P52              | D<br>(CMOS)          | General-purpose input/output port. Functions as the $\overline{WRL}$ pin in the external bus mode if the WRE bit of the EPCR register is set to "1".  |
|                            |                            | $\overline{WRL}$ |                      | Functions as the write strobe output ( $\overline{WRL}$ ) pin of data on the lower side in the external bus mode. Functions as a general-purpose input/output port if the WRE bit of the EPCR register is set to "0".                       |
| 73                         | 71                         | P53              | D<br>(CMOS)          | General-purpose input/output port. Functions as the $\overline{WRH}$ pin in the external bus mode of a 16-bit bus width if the WRE bit of the EPCR register is set to "1".  |
|                            |                            | $\overline{WRH}$ |                      | Functions as the write strobe output ( $\overline{WRH}$ ) pin of data on the upper side in the external bus mode of a 16-bit bus width. Functions as a general-purpose input/output port if the WRE bit of the EPCR register is set to "0". |
| 74                         | 72                         | P54              | D<br>(CMOS)          | General-purpose input/output port. Functions as the HRQ pin in the external bus mode if the HDE bit of the EPCR register is set to "1".   |
|                            |                            | HRQ              |                      | Functions as the hold request input (HRQ) pin in the external bus mode. Functions as a general-purpose input/output port if the HDE bit of the EPCR register is set to "0".   |
| 75                         | 73                         | P55              | D<br>(CMOS)          | General-purpose input/output port. Functions as the $\overline{HAK}$ pin in the external bus mode if the HDE bit of the EPCR register is set to "1".  |
|                            |                            | $\overline{HAK}$ |                      | Functions as the hold acknowledge output ( $\overline{HAK}$ ) pin in the external bus mode. Functions as a general-purpose input/output port if the HDE bit of the EPCR register is set to "0".   |
| 76                         | 74                         | P56              | D<br>(CMOS)          | General-purpose input/output port. Functions as the RDY pin in the external bus mode if the RYE bit of the EPCR register is set to "1".   |
|                            |                            | RDY              |                      | Functions as the external ready input (RDY) pin in the external bus mode. Functions as a general-purpose input/output port if the RYE bit of the EPCR register is set to "0".   |

Table 1.5-1 Pin Functions (5/7)

| Pin number                 |                            | Pin name           | Circuit <sup>3</sup> | Function  |   |
|----------------------------|----------------------------|--------------------|----------------------|---|---|
| FPT-100P-M06 <sup>*1</sup> | FPT-100P-M05 <sup>*2</sup> |                    |                      |   |   |
| 78                         | 76                         | P57                | D<br>(CMOS)          | General-purpose input/output port. Functions as the CLK pin in the external bus mode if the CKE bit of the EPCR register is set to "1".   |   |
|                            |                            | CLK                |                      | Functions as the machine cycle clock output (CLK) pin in the external bus mode. Functions as a general-purpose input/output port if the CKE bit of the EPCR register is set to "0". |   |
| 38<br>to<br>41             | 36<br>to<br>39             | P60 to P63         | H<br>(CMOS)          | General-purpose input/output port   |   |
|                            |                            | AN0 to AN3         |                      | Functions as an analog input pin.   |   |
| 43<br>to<br>46             | 41<br>to<br>44             | P64 to P67         | H<br>(CMOS)          | General-purpose input/output port   |   |
|                            |                            | AN4 to AN7         |                      | Functions as an analog input pin.   |   |
| 27                         | 25                         | P70                | G<br>(CMOS/H)        | General-purpose input/output port   |   |
|                            |                            | SIN0               |                      | Functions as the UART data input pin.   |   |
| 28                         | 26                         | P71                | F<br>(CMOS)          | General-purpose input/output port   |   |
|                            |                            | SOT0               |                      | Functions as the UART data output pin.  |   |
| 29                         | 27                         | P72                | G<br>(CMOS/H)        | General-purpose input/output port   |   |
|                            |                            | SCK0               |                      | Functions as the UART clock input/output pin.   |   |
| 30                         | 28                         | P73                | G<br>(CMOS/H)        | General-purpose input/output port   |   |
|                            |                            | TIN0               |                      | Functions as the event input pin of a 16-bit reload timer.  |   |
| 31                         | 29                         | P74                | F<br>(CMOS)          | General-purpose input/output port   |   |
|                            |                            | TOT0               |                      | Functions as the output pin of a 16-bit reload timer.   |   |
| 32                         | 30                         | P75                | F(CMOS)              | MB90480 series  | General-purpose input/output port   |
|                            |                            | P75                | G<br>(CMOS/H)        | MB90485 series  | General-purpose input/output port   |
|                            |                            | PWC2 <sup>*4</sup> |                      |   | Functions as a PWC input pin.   |
| 33                         | 31                         | P76                | F(CMOS)              | MB90480 series  | General-purpose input/output port   |
|                            |                            | P76                | I<br>(NMOS/H)        | MB90485 series  | General-purpose input/output port   |
|                            |                            | SCL <sup>*4</sup>  |                      |   | Functions as I <sup>2</sup> C interface data I/O pin. When I <sup>2</sup> C interface is operating, the port output should be set Hi-Z. |
| 34                         | 32                         | P77                | F(CMOS)              | MB90480 series  | General-purpose input/output port   |
|                            |                            | P77                | I<br>(NMOS/H)        | MB90485 series  | General-purpose input/output port   |
|                            |                            | SDA <sup>*4</sup>  |                      |   | Functions as I <sup>2</sup> C interface data I/O pin. When I <sup>2</sup> C interface is operating, the port output should be set Hi-Z. |

## CHAPTER 1 OVERVIEW OF MB90480/485 SERIES

**Table 1.5-1 Pin Functions (6/7)**

| Pin number                 |                            | Pin name         | Circuit <sup>3</sup> | Function   |
|----------------------------|----------------------------|------------------|----------------------|--|
| FPT-100P-M06 <sup>*1</sup> | FPT-100P-M05 <sup>*2</sup> |                  |                      |  |
| 47<br>48                   | 45<br>46                   | P80, P81         | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | IRQ0, IRQ1       |                      | Functions as the external interrupt input pin.                             |
| 52<br>to<br>57             | 50<br>to<br>55             | P82 to P87       | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | IRQ2 to IRQ7     |                      | Functions as the external interrupt input pin.                             |
| 58                         | 56                         | P90              | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | SIN1             |                      | Functions as the input pin of simple serial I/O data.                      |
|                            |                            | CS0              |                      | Chip select 0  |
| 59                         | 57                         | P91              | D<br>(CMOS)          | General-purpose input/output port  |
|                            |                            | SOT1             |                      | Functions as the output pin of simple serial I/O data.                     |
|                            |                            | CS1              |                      | Chip select 1  |
| 60                         | 58                         | P92              | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | SCK1             |                      | Functions as the input/output pin of a simple serial I/O clock.            |
|                            |                            | CS2              |                      | Chip select 2  |
| 61                         | 59                         | P93              | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | FRCK             |                      | Functions as the external clock input pin when a free-run timer is used.   |
|                            |                            | ADTG             |                      | Functions as the external trigger input pin when an A/D converter is used. |
|                            |                            | CS3              |                      | Chip select 3  |
| 62                         | 60                         | P94              | D<br>(CMOS)          | General-purpose input/output port  |
|                            |                            | PPG4             |                      | Functions as the output pin of the PPG timer.                              |
| 63                         | 61                         | P95              | D<br>(CMOS)          | General-purpose input/output port.   |
|                            |                            | PPG5             |                      | Functions as the output pin of the PPG timer.                              |
| 64                         | 62                         | P96              | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | IN0              |                      | Captured as trigger input of input capture ch.0.                           |
| 65                         | 63                         | P97              | E<br>(CMOS/H)        | General-purpose input/output port  |
|                            |                            | IN1              |                      | Captured as trigger input of input capture ch.1.                           |
| 66<br>to<br>69             | 64<br>to<br>67             | PA0 to PA3       | D<br>(CMOS)          | General-purpose input/output port  |
|                            |                            | OUT0 to OUT3     |                      | Functions as the output pin of output compare events.                      |
| 35                         | 33                         | AV <sub>CC</sub> | -                    | Power supply pin of A/D converter  |
| 36                         | 34                         | AVRH             | -                    | External reference power supply pin of A/D converter                       |
| 37                         | 35                         | AV <sub>SS</sub> | -                    | Power supply pin of A/D converter  |

Table 1.5-1 Pin Functions (7/7)

| Pin number                 |                            | Pin name         | Circuit <sup>*3</sup> | Function  |   |
|----------------------------|----------------------------|------------------|-----------------------|---|---|
| FPT-100P-M06 <sup>*1</sup> | FPT-100P-M05 <sup>*2</sup> |                  |                       |   |   |
| 49<br>to<br>51             | 47<br>to<br>49             | MD0 to MD2       | J<br>(CMOS/H)         | Input pin name to specify the operation mode.       |   |
| 84                         | 82                         | V <sub>CC3</sub> | -                     | Power supply pin (V <sub>CC3</sub> ) of 3.3V ± 0.3V |   |
| 23                         | 21                         | V <sub>CC5</sub> | -                     | MB90480 series                                      | Power supply pin for 3.3V (its tolerance is -0.3V to +0.3V)<br>This pin is connected with V <sub>CC</sub> and V <sub>CC3</sub> and used for 3V power supply.  |
|                            |                            |                  |                       | MB90485 series                                      | Power supply pin for 3V or 5V<br>5V power supply pin:<br>In case that P20 to P27, P30 to P37, P40 to P47 and P70 to P77 are used as a 5V I/F pin.<br>On single 3V power supply voltage, this pin is connected with V <sub>CC</sub> and V <sub>CC3</sub> , and used for 3V power supply. |
| 11<br>42<br>81             | 9<br>40<br>79              | V <sub>SS</sub>  | -                     | Power supply input (GND)                            |   |

\*1: QFP : FPT-100P-M06

\*2: LQFP: FPT-100P-M05

\*3: Refer to "1.6 I/O Circuit Type" about I/O circuit type.

\*4: CMOS input for only MB90V485B



# 1.6 I/O Circuit Type

This section explains the I/O circuit type of MB90480/485 series pins.

## ■ I/O Circuit Type

Table 1.6-1 summarizes the I/O circuit type of MB90480/485 series pins.

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

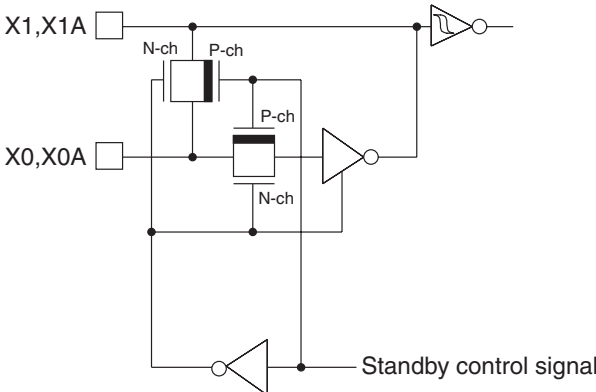
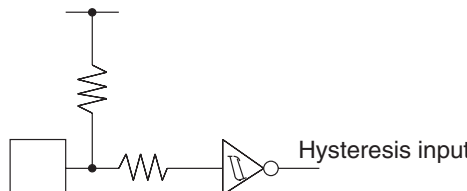
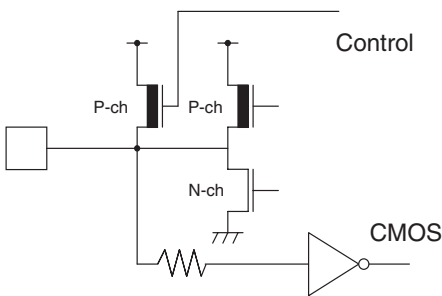
| Class | Circuit   | Description  |
|-------|---|--|
| A     |   | <ul style="list-style-type: none"><li>• Oscillation feedback resistor:<br/>X1, X0: about 1 MΩ<br/>X1A, X0A: about 10 MΩ</li><li>• Use of standby control</li></ul> |
| B     |  | <ul style="list-style-type: none"><li>• Hysteresis input with pull-up</li></ul>  |
| C     |  | <ul style="list-style-type: none"><li>• Use of input pull-up resistor control</li><li>• CMOS level input/output</li></ul>  |

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

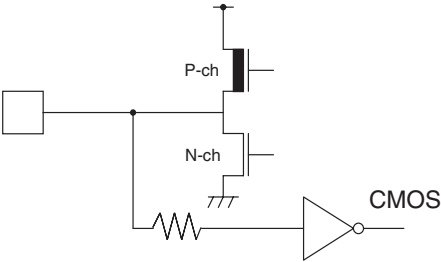
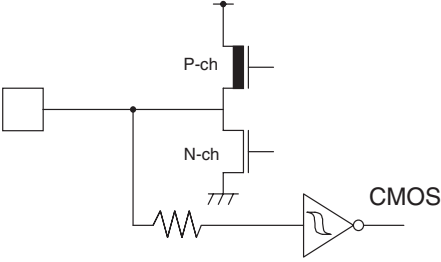
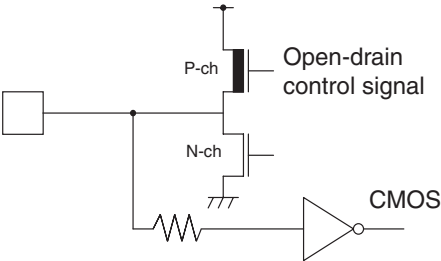
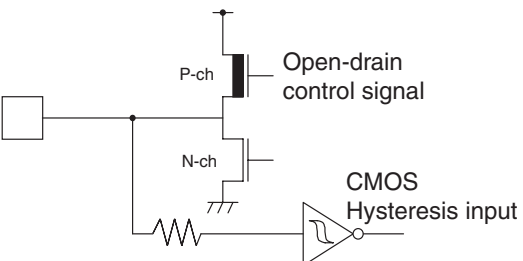
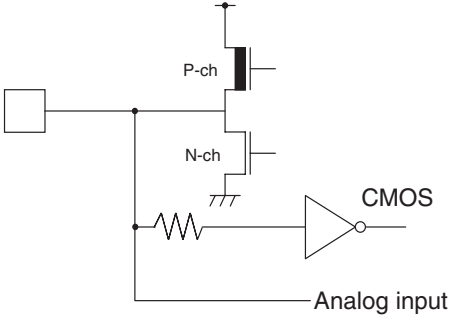
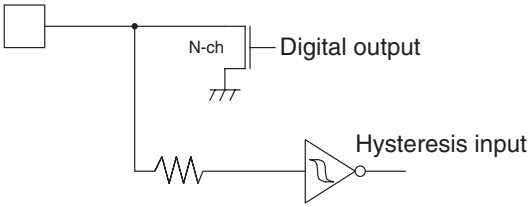
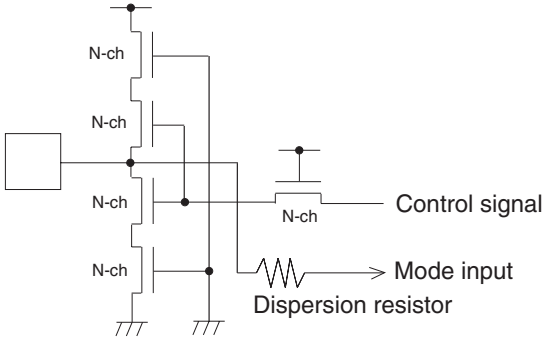
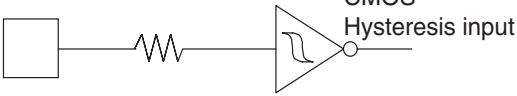
| Class | Circuit   | Description  |
|-------|---|--|
| D     |    | <ul style="list-style-type: none"><li>• CMOS level input/output</li></ul>  |
| E     |    | <ul style="list-style-type: none"><li>• Hysteresis input</li><li>• CMOS level output</li></ul>                                     |
| F     |   | <ul style="list-style-type: none"><li>• CMOS level input/output</li><li>• Use of open-drain control</li></ul>                      |
| G     |  | <ul style="list-style-type: none"><li>• CMOS level output</li><li>• Hysteresis input</li><li>• Use of open-drain control</li></ul> |

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

| Class | Circuit   | Description  |
|-------|---|--|
| H     |                          | <ul style="list-style-type: none"><li>• CMOS level input/output</li><li>• Analog input</li></ul>   |
| I     |                          | <ul style="list-style-type: none"><li>• Hysteresis input</li><li>• N-ch open drain output</li></ul>  |
| J     | <p>(Flash product)</p>  | <p>(Flash product)</p> <ul style="list-style-type: none"><li>• CMOS level input</li><li>• High-voltage control provided for Flash test</li></ul> |
|       | <p>(Mask product)</p>  | <p>(Mask product)</p> <ul style="list-style-type: none"><li>• CMOS Hysteresis input</li></ul>  |

## 1.7 Handling the Device

---

This section gives notes on handling the MB90480/485 series.

---

### ■ Notes on Handling the Device

#### ○ Latch-up prevention and power-on sequence

Some CMOS ICs may cause latch-up symptoms as described below:

- If voltages higher than  $V_{CC}$  or lower than  $V_{SS}$  are applied to the input and output pins.
- If the applied voltage ranging between  $V_{CC}$  and  $V_{SS}$  is higher than the rated voltage.
- If voltage  $V_{CC}$  is applied following the  $AV_{CC}$  power supply

Once a latch-up occurs, the power supply current increases rapidly, possibly causing heat damage in the element. Be sure to prevent such damage during use.

Analog voltage must be supplied at the same time as  $V_{CC}$ , or it must be applied after the digital power supply is turned on (Turn off the analog power supply before or at the same time as the power supply is turned off).

#### ○ Processing unused input pins

Keeping an unused input pin open may cause an error in operation or apply pull-up or pull-down to such pins with 2 k $\Omega$  or more resistor as necessary. For an unused A/D converter, connect it so that  $AV_{CC} = AVR_H = V_{CC}$ , and  $AV_{SS} = V_{SS}$ .

#### ○ Handling a power supply pin ( $V_{CC}/V_{SS}$ )

If multiple  $V_{CC}$  and/or  $V_{SS}$  are used, all power supply pins must be connected with a power supply or ground externally in consideration of device design in order to decrease latch-up and unnecessary radiation and to prevent the malfunction of the strobe signal due to a rise of ground level. Be sure to connect all power supply pins to the power supply or ground so that the total output current specification is not exceeded.

As much as possible, the power supply source must be connected with  $V_{CC}/V_{SS}$  of this device at the lowest impedance.

Fujitsu recommends placing a bypass condenser of 0.1  $\mu$ F between  $V_{CC}$  and  $V_{SS}$ .

#### ○ Crystal oscillation circuit

Noise around the X0/X1 or X0A/X1A pins may cause an error during operation on this device. X0/X1, X0A/X1A and a crystal oscillator (or ceramic oscillator), or a bypass condenser to ground must be arranged as close to each other as possible to prevent crossover between them.

To use the printed board artwork is strongly recommended since it is expected to provide stable operation.

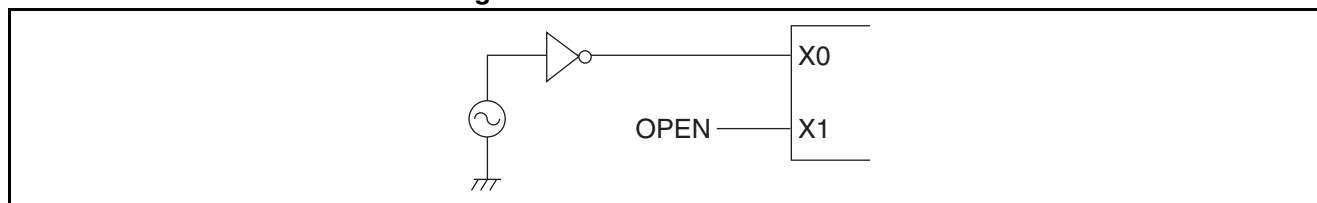
Please ask the crystal maker to evaluate the oscillational characteristics of the crystal and this device.

### ○ Notes on using external clock

To use external clock, drive only pin X0. Be sure to set up pin X1 to be open.

Figure 1.7-1 shows a usage of the external clock ( $f=25$  MHz or below).

**Figure 1.7-1 Use of External Clock**



### ○ Note on operations during PLL clock mode

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

## ■ Notes on Handling the Power Supply

### ○ Stabilizing the power supply

Even in the range of  $V_{CC}$  power supply voltage, a rapid change in the power supply may cause a misoperation. Fujitsu recommends that, as a reference for stabilization, the  $V_{CC}$  ripple variation (P-P value) in the commercial frequency (50/60 MHz) must be 10% of the standard  $V_{CC}$  value or lower, or the transient variation must be 0.1 V/ms in instantaneous variation including power supply switching.

### ○ To use this product as a single system

To use this product, which is two system products, as a single system, use it under the conditions of  $X0A = V_{SS}$  and  $X1A = OPEN$ .

### ○ Writing to flash memory

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

For normal writing to flash memory, always make sure that the operating voltage  $V_{CC}$  is between 3 V and 3.6 V.

### ○ Note on using dual power supply

The MB90485 series usually uses 3V power supply. However, if  $V_{cc3}=3V/V_{cc5}=5V$ , P20 to P27, P30 to P37, P40 to P47, and P70 to P77 are used as the interface of 5V power supply.

Note that analog power supply ( $AV_{CC}$ ,  $AVRH$ ) is used only as 3V during A/D conversion.

### ○ Handling of P90/CS0 pin

P90/CS0 pin outputs "L" during flash serial writing. Do not input from outside.

## CHAPTER 2 CPU

---

**This chapter explains CPU specifications, memory, and the functions of registers to provide readers with a better understanding of the MB90480/485 series functions.**

---

2.1 Overview of CPU Specifications

2.2 Memory Space

2.3 CPU Registers

2.4 Prefix Codes

## 2.1 Overview of CPU Specifications

---

This section gives an overview of the CPU specifications.

---

### ■ Overview of the CPU Specifications

The F<sup>2</sup>MC-16LX CPU core is a 16-bit CPU designed for devices such as consumer devices that requires high-speed real-time processing. The F<sup>2</sup>MC-16LX instruction set is designed for controller applications, providing high-speed and high-efficiency control processes.

In addition to 16-bit data processing, the F<sup>2</sup>MC-16LX CPU core can provide 32-bit data processing with an installed internal 32-bit accumulator (some instructions perform 32-bit data processing). Memory spaces are a maximum of 16 M bytes (expandable) and can be accessed by using a linear pointer or bank. Based on the F<sup>2</sup>MC-8L AT architecture, its instruction system is improved because of increasing the instructions supporting high-level languages, expanding addressing modes, improving multiply and divide operation instructions, and enhancing bit processing. The followings are features of F<sup>2</sup>MC-16LX CPU.

#### ○ Minimum instruction execution time

- 40.0 ns/6.25 MHz oscillation multiplied by 4 (25 MHz/3.3 V  $\pm$  0.3 V for internal operation)
- 62.5 ns/4 MHz oscillation multiplied by 4 (16 MHz/3.0 V  $\pm$  0.3 V for internal operation)
- PLL clock multiply scheme

#### ○ Maximum memory space: 16 M bytes, accessing by using a linear pointer or bank

#### ○ Instruction system optimized for control applications

- Data types available: bit, byte, word, long word
- Standard addressing mode: Use of 23-type, 32-bit accumulator for enhancing high-precision operation
- Signed multiply and divide operations, expanded RETI instruction

#### ○ Enhanced interrupt function: 8 priority levels (programmable)

#### ○ CPU independent automatic transfer function: Up to 16 channels $\mu$ DMAC

#### ○ Multitasking-compatible instruction system in high-level language (C)

Use of system stack pointers, symmetrical instruction set, and barrel shift instruction

#### ○ Improved execution speed: 4-byte queue

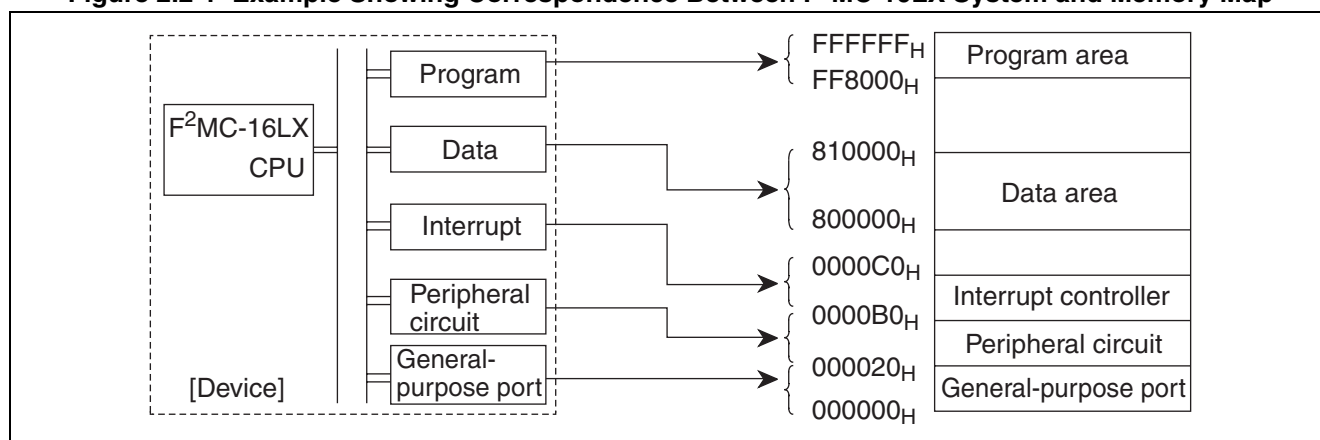
## 2.2 Memory Space

The F<sup>2</sup>MC-16LX CPU has a 16M bytes memory space, to which all input to and output from the F<sup>2</sup>MC-16LX CPU controlled data program is allocated. CPU has a 24-bit address bus to access each resource.

### ■ Memory Map

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

**Figure 2.2-1 Example Showing Correspondence Between F<sup>2</sup>MC-16LX System and Memory Map**



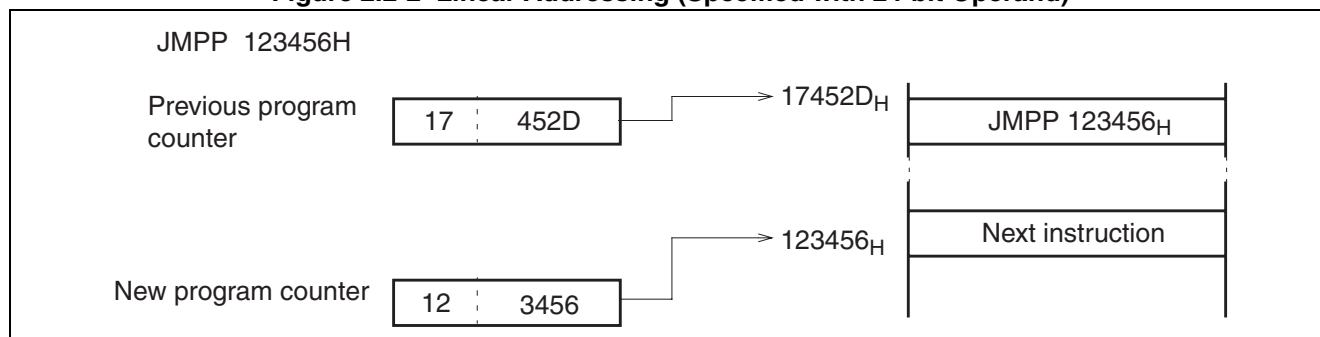
### ■ Address Generation Type

The F<sup>2</sup>MC-16LX CPU has two types of address generation. One is linear addressing that specifies all 24-bit addresses with instructions. The other is bank addressing that specifies upper 8-bit addresses with appropriate bank registers and lower 16-bit addresses with instructions. Linear addressing has two types: one uses operands to directly specify 24-bit addresses; the other refers to contents of the lower 24 bits in a 32-bit general-purpose register as addresses.

#### ○ Linear addressing (specified with 24-bit operand)

Figure 2.2-2 shows an example of linear addressing scheme specified with 24-bit operands.

**Figure 2.2-2 Linear Addressing (Specified with 24-bit Operand)**

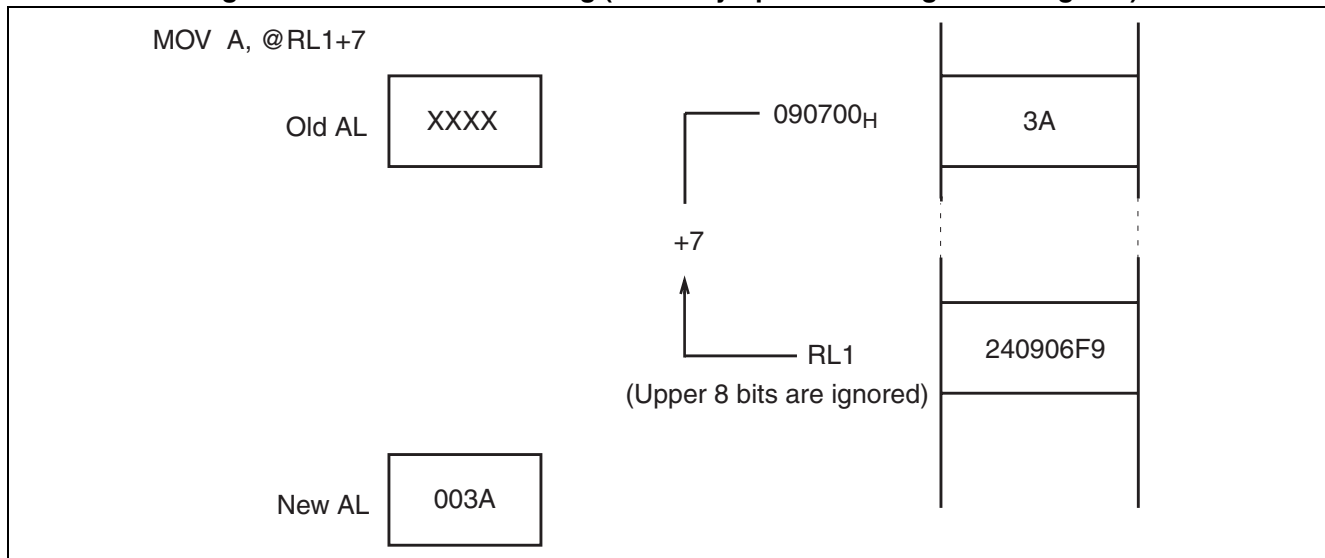




○ **Linear addressing (indirectly specified using 32-bit register)**

Figure 2.2-3 shows an example of linear addressing scheme indirectly specified using a 32-bit register.

**Figure 2.2-3 Linear Addressing (Indirectly Specified Using 32-bit Register)**



■ **Addressing Type by Bank**

Bank addressing divides a 16M bytes space into 256 banks of 64K bytes each, using five bank registers to specify banks for each space.

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

A 64K bytes bank specified with PCB is called the program (PC) space. The PC space includes such information as instruction codes, vector tables, and immediate data.

A 64K bytes bank specified with DTB is called the data (DT) space. The DT space includes writable data, and internal and external resource control/data registers.

A 64K bytes bank specified with USB or SSB is called the stack (SP) space. The SP space is accessed if a stack access occurs by saving the push/pop instruction or interrupt register. The stack space to be accessed is determined by the S-flag in the condition code register.

A 64K bytes bank specified with ADB is called the additional (AD) space. The AD space includes, for example, the data that cannot be included in the DT space.

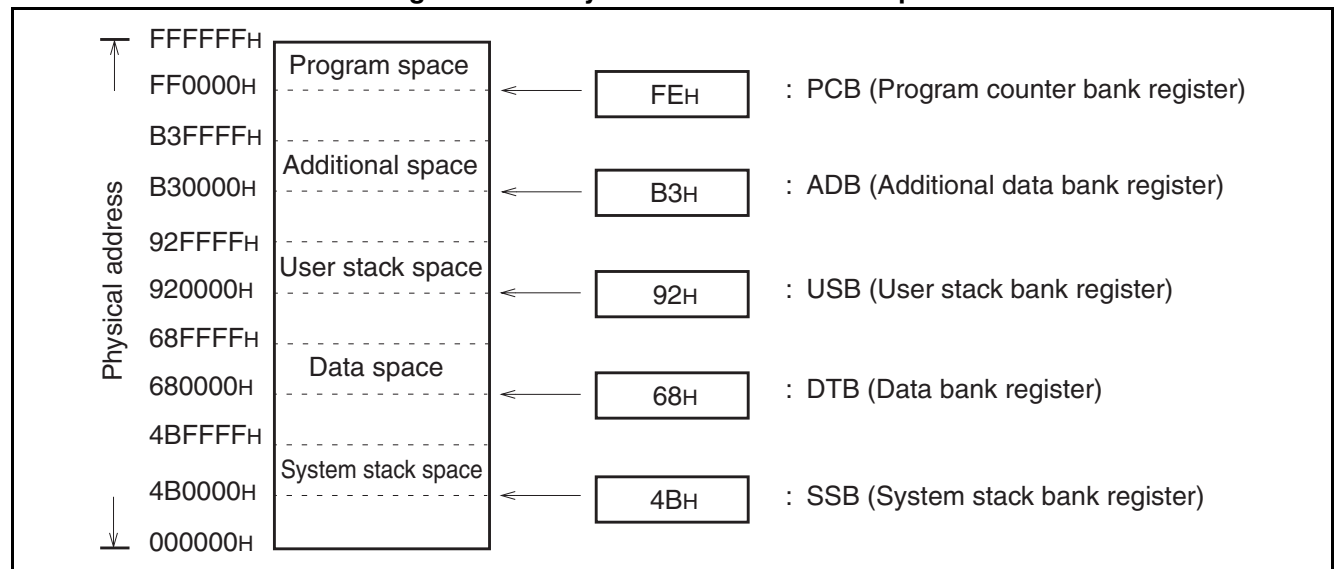
As shown in Table 2.2-1, each addressing mode uses a default space defined in advance to improve the efficiency of coding instructions. If an addressing mode uses a space other than the default space, a prefix code corresponding to the bank must be specified prior to the instruction code, enabling access to any bank space corresponding to the prefix code.

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

**Table 2.2-1 Default Space**

| Default space    | Addressing mode   |
|------------------|---|
| Program space    | PC indirect, program access, branch instruction                   |
| Data space       | Addressing mode using @RW0, @RW1, @RW4, and @RW5; @A; addr16; dir |
| Stack space      | Addressing mode using PUCHW, POPW, @RW3, and @RW7                 |
| Additional space | Addressing mode using @RW2 and @RW6                               |

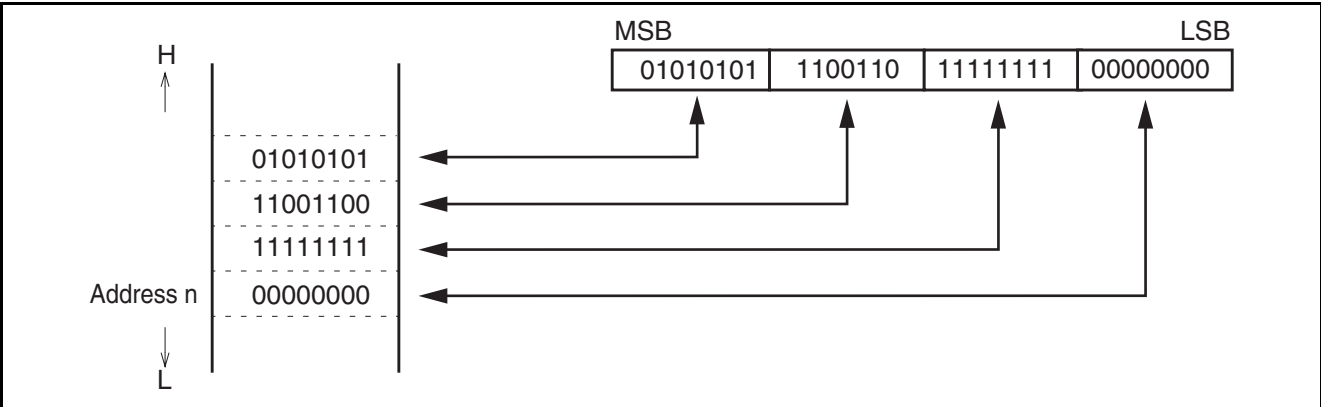
Figure 2.2-4 shows an example of a memory space divided for a register bank.

**Figure 2.2-4 Physical Address in Each Space**

■ Allocation for Data of Multi-byte Length in Memory Space

Figure 2.2-5 shows the configuration of data of a multi-byte length in memory. The lower 8 bits of a data item are stored at address n, then address n+1, address n+2, address n+3, etc.

Figure 2.2-5 Example for Allocating Data of Multi-byte Length in Memory

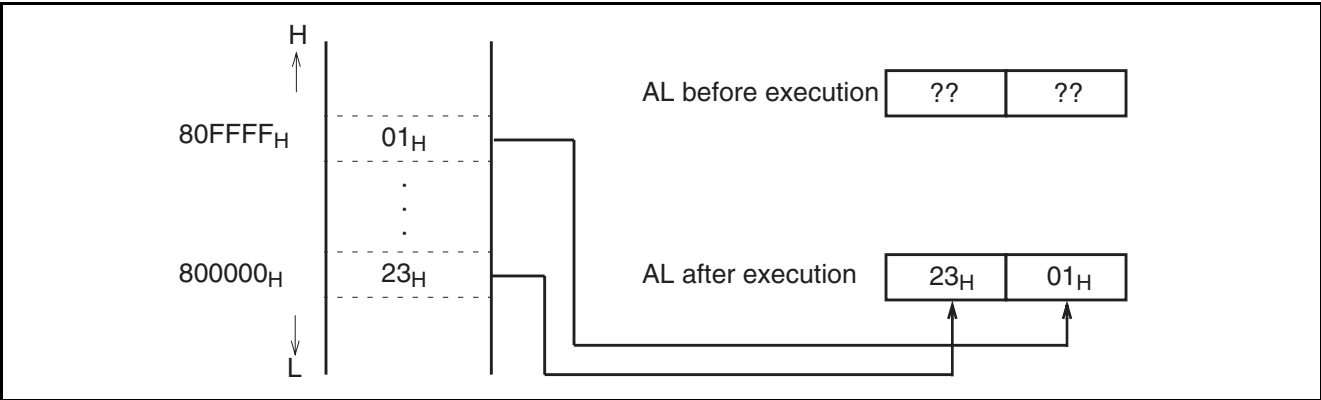


Data is written to memory in sequence starting from the lower addresses. Thus, the lower 16 bits of a 32-bit data item is transferred first, followed by the upper 16 bits. If a reset signal is input immediately after writing the lower bit, writing the upper bit may fail.

■ Access to Data of Multi-byte Length

Figure 2.2-6 shows an example for accessing data of a multi-byte length. In this example, MOVW A, 030FFFFH is executed.

Figure 2.2-6 Example for Accessing Data of Multi-byte Length



## 2.3 CPU Registers

The F<sup>2</sup>MC-16LX registers are divided into special registers inside CPU and general-purpose registers on memory. The former is dedicated hardware inside the CPU, and its use is limited because of the CPU architecture. The latter shares CPU address spaces with RAM. A general-purpose register can also be accessed without specifying an address, but a user can specify the use of a general-purpose register, which is the same as for memory spaces.

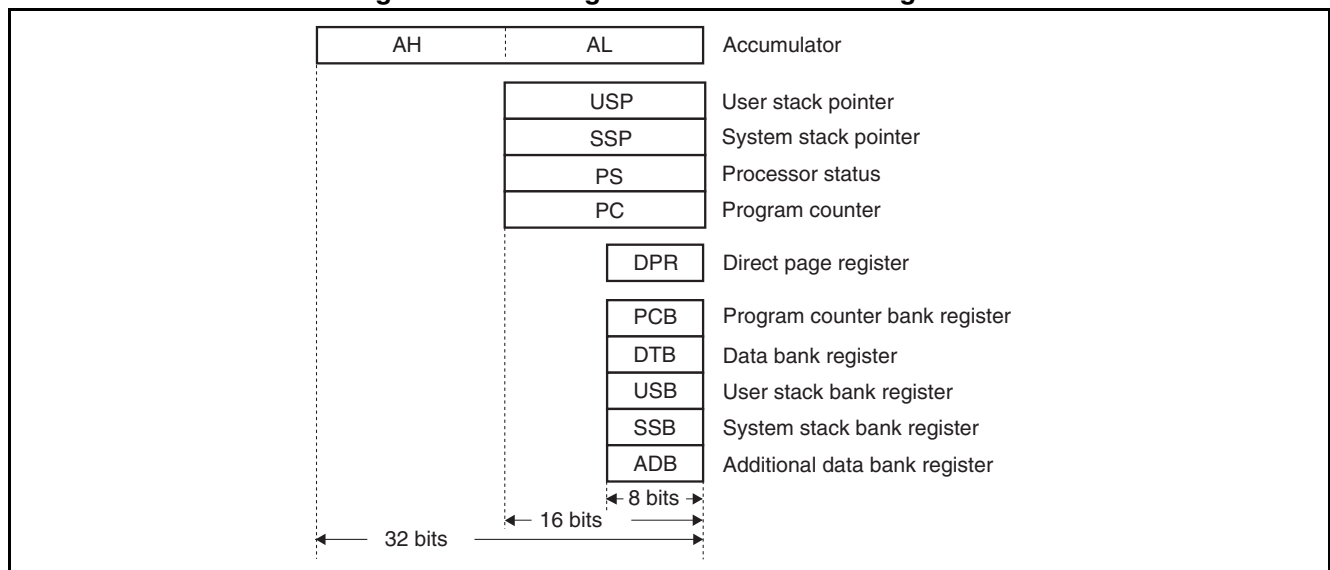
### ■ Dedicated Registers

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

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

Figure 2.3-1 shows the configuration of the dedicated registers.

**Figure 2.3-1 Configuration of Dedicated Registers**



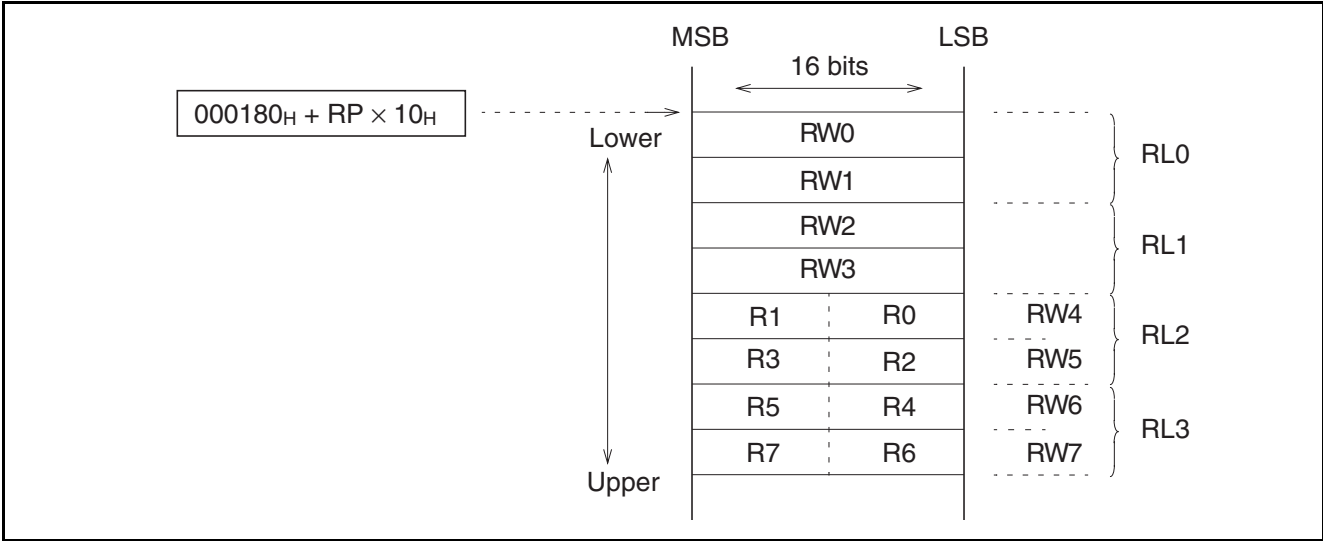
■ General-purpose Register

The F<sup>2</sup>MC-16LX general-purpose register resides on the main memory addresses: 000180<sub>H</sub> to 00037F<sub>H</sub> (maximum configuration). It uses a register bank register (RP) to indicate which part of addresses are currently used for register banks. Each bank has the three types of registers listed below. They are dependent on one another, as shown in Figure 2.3-2.

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

Figure 2.3-2 shows the configuration of a general-purpose register.

Figure 2.3-2 Configuration of General-purpose Register



The relationship between upper and lower bytes in a byte register and word register is represented with the following formula:

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

The relationship of upper and lower bytes in RL<sub>i</sub> is represented with the following formula:

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

## 2.3.1 Accumulator (A)

This section explains the accumulator (A) functions.

### ■ Accumulator (A)

An accumulator (A) consists of two 16-bit arithmetic operation registers (AH/AL) that are used to store operation results and temporarily store data transfer results. For 32-bit data processing, AH is connected with AL. For word processing in the 16-bit data processing mode and for byte processing in the 8-bit data processing mode, only AL is used. Data stored in an accumulator (A) is used together with that in memory and registers (Ri, Rwi, Rli); and similar to F<sup>2</sup>MC-8L operations, the data item with a smaller word length is transferred to AL. This enables data items in AL before the transfer to be automatically transferred to AH (data hold function). The data hold function and operation between AL-AH support improvements in processing efficiency.

During a transfer of a data item with a lower byte length to AL, a sign extension or zero extension is added to the data, and the data is saved in AL as a 16-bit data item. Also, data in AL is handled in either word lengths or byte lengths.

If an arithmetic operation instruction of byte processing is executed in AL, the upper 8 bits in AL before the operation is ignored, and the upper 8 bits of operation results are reset to zero. Resetting an accumulator (A) does not initialize it, and it has an undefined value after the reset.

Figure 2.3-3 shows 32-bit data transfer processing, and Figure 2.3-4 shows AL-AH transfer processing.

Figure 2.3-3 32-bit Data Transfer

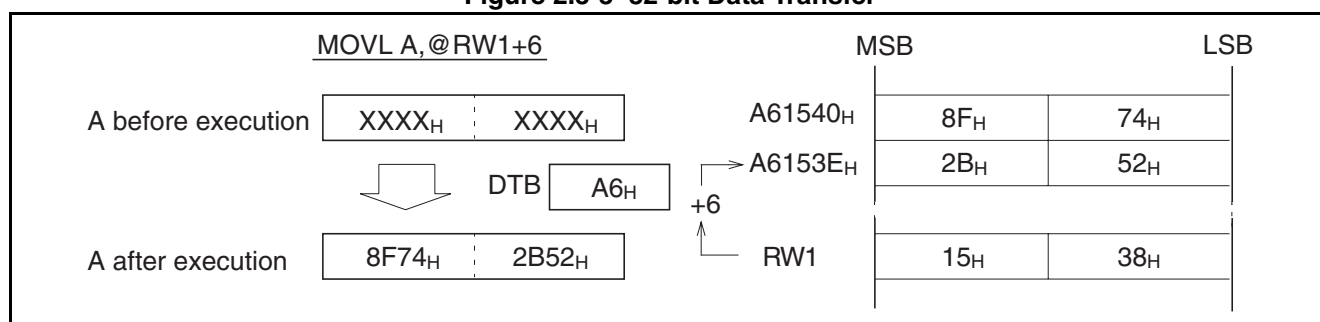
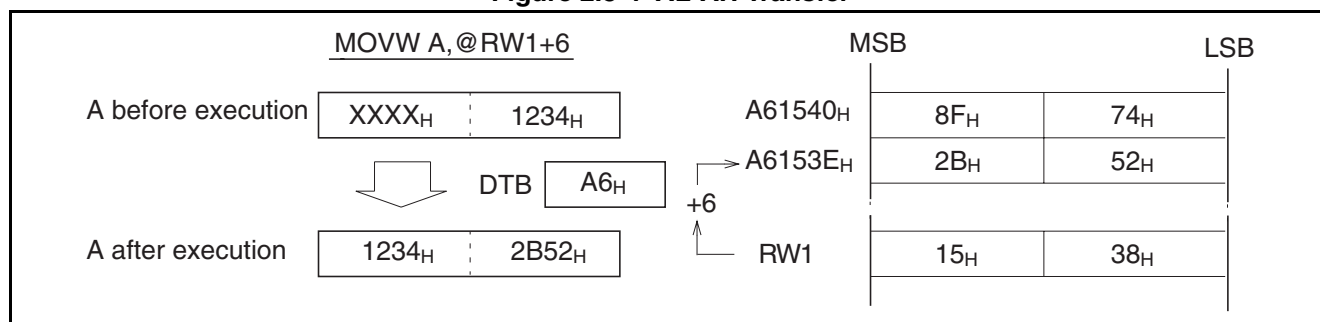


Figure 2.3-4 AL-AH Transfer



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

This section explains the functions of the user stack pointer (USP) and system stack pointer (SSP).

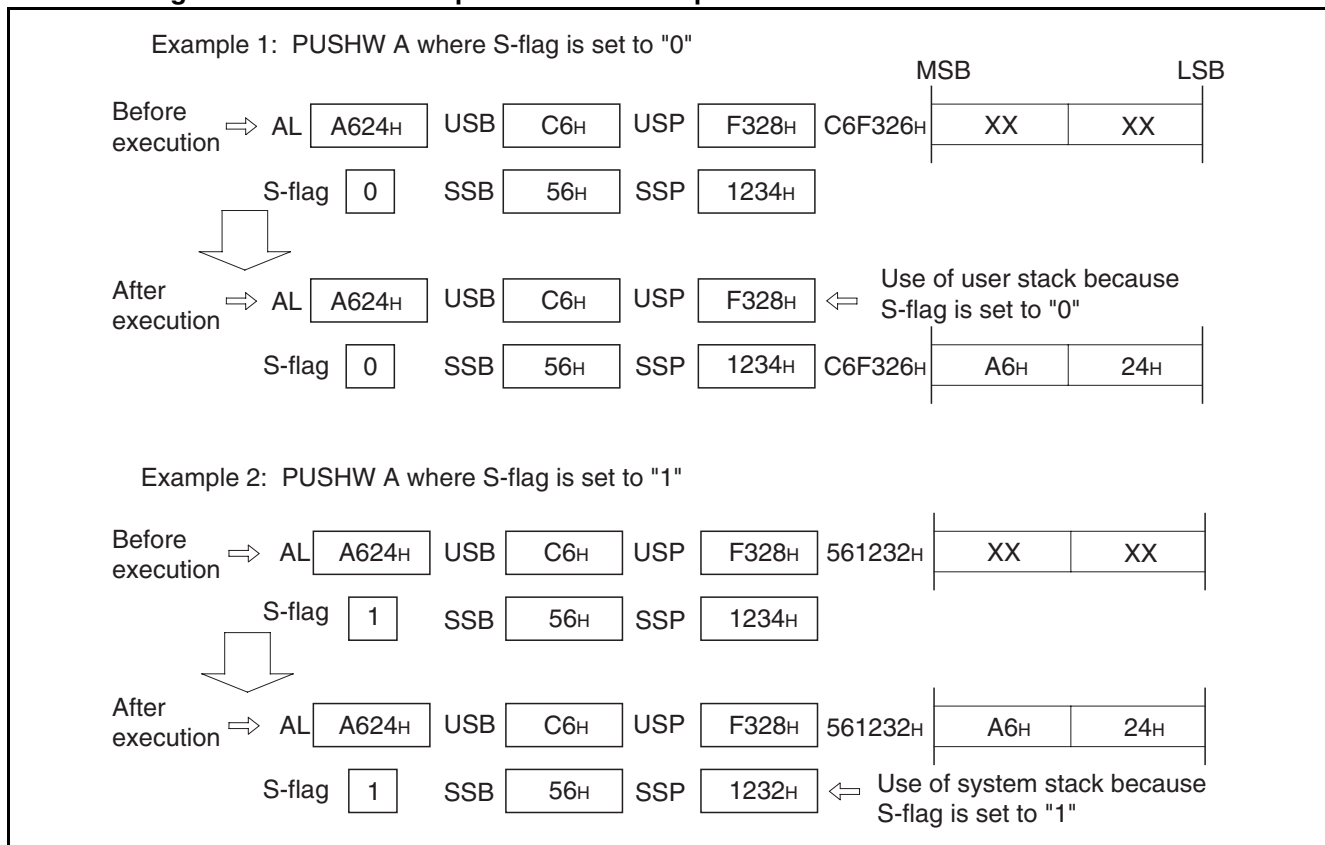
### ■ User Stack Pointer (USP) and System Stack Pointer (SSP)

The user stack pointer (USP) and system stack pointer (SSP) are 16-bit registers indicating the push/pop instruction or the memory address to which data is saved or restored at subroutine execution. The USP register and SSP register are used in stack-type instructions. If the S-flag in the processor status register is set to "0", the USP register is enabled. If the S-flag is set to "1", SSP register is enabled (see Figure 2.3-5). If an interrupt is accepted, the S-flag is set and then the register value is saved in the memory area indicated by SSP in interrupt processing. SSP is used to execute stack processing of interrupt routines, and USP is used to execute stack processing other than interrupt routines. Only SSP is used if stack space is not divided.

In stack processing, the address of upper 8 bits is indicated with SSP -> SSB and USP -> USB. Resetting USP and SSP does not initialize them, but each then has an undefined value.

Figure 2.3-5 shows the relationship between stack operation instructions and the stack pointer where the S-flag is set to "0" and "1".

**Figure 2.3-5 Relationship Between Stack Operation Instructions and Stack Pointer**



Note:

Use an even-numbered address for a stack pointer, in principle.

## 2.3.3 Processor Status (PS)

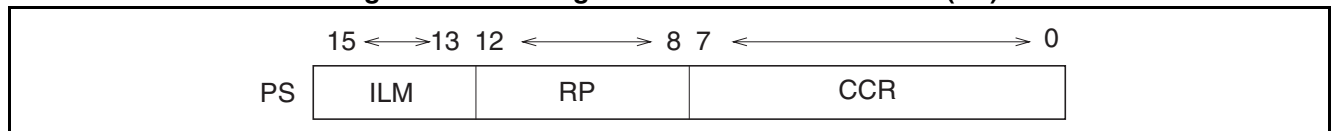
This section explains the processor status (PS) functions.

### ■ Processor Status (PS)

Processor status (PS) consists of bits used to execute CPU operations and bits indicating the CPU state. As shown in Figure 2.3-6, the upper byte in the PS register consists of a register bank pointer (RP) and interrupt level mask register (ILM). RP indicates the header address of a register bank. The lower byte of PS register is the condition code register (CCR) that includes a flag that is set and reset depending on execution results or interrupt events.

Figure 2.3-6 shows the configuration of processor status (PS).

**Figure 2.3-6 Configuration of Processor Status (PS)**



### ■ Condition Code Register (CCR)

Figure 2.3-7 shows the configuration of the condition code register.

**Figure 2.3-7 Configuration of Condition Code Register**



#### ○ I: Interrupt permission flag

An interrupt other than software interrupt is permitted if the I-flag is set to "1" and masked if set to "0".

The I-flag is cleared if reset.

#### ○ S: Stack flag

If the S-flag is set to "0", USP is enabled as the stack operation pointer, and if it is set to "1", SSP is enabled. The S-flag is set if an interrupt or reset occurs.

#### ○ T: Sticky bit flag

If there is one or more "1" in the data shifted out by a carry operation after the logic right shift instruction or arithmetic right shift instruction, the T-flag is set to "1". Otherwise, it is set to "0". If the shift amount is zero, it is set to "0".

#### ○ N: Negative flag

If MSB in operation results indicates "1", the N-flag is set. Otherwise, it is cleared.



○ **Z: Zero flag**

If all operation results indicate "0", the Z-flag is set. Otherwise, it is cleared.

○ **V: Overflow flag**

If an overflow with a signed figure occurs as an operation execution result, the V-flag is set. Otherwise, it is cleared.

○ **C: Carry flag**

If a shift-in/shift-out operation occurs from MSB as operation execution results, the C-flag is set. Otherwise, it is cleared.

■ **Register Bank Pointer (RP)**

The register bank pointer (RP) shows the relationship between the F<sup>2</sup>MC-16LX general-purpose register and internal RAM addresses. RP indicates the header memory address in the currently used register bank with the conversion formula  $[00180_H + RP \times 10_H]$ .

RP consists of 5 bits, with an address ranging from 00<sub>H</sub> to 1F<sub>H</sub>.

A register bank can be allocated to a memory address in a range of 000180<sub>H</sub> to 00037F<sub>H</sub>. Even in this range, however, a register bank cannot be used as a general-purpose register if a register bank is not in internal RAM. An instruction transfers an immediate value of 8 bits to RP, but only the lower 5 bits are actually used.

Figure 2.3-8 Configuration of Register Bank Pointer (RP)

|               |    |    |    |    |    |
|---------------|----|----|----|----|----|
|               | B4 | B3 | B2 | B1 | B0 |
| Initial value | 0  | 0  | 0  | 0  | 0  |

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

The interrupt level mask register (ILM) consists of 3 bits indicating the level of the CPU interrupt mask. Only interrupt request of an interrupt level higher than that represented with the 3 bits is accepted. The highest level is indicated with "0", the lowest level is indicated with "7" (see Table 2.3-1). Thus, to accept an interrupt, its level must be lower than the current ILM value. If an interrupt is accepted, its interrupt level value is set to ILM, and then any interrupts with the same or lower level of the interrupt priority are not accepted. ILM is initialized to zero by a reset. An instruction can transfer an 8-bit immediate value to the ILM register, but only the lower 3 bits are actually used.

Figure 2.3-9 shows the configuration of the interrupt level mask register. Table 2.3-1 has explanations of the level indicated in the interrupt level mask register (ILM).

Figure 2.3-9 Configuration of Interrupt Level Mask Register

|               |      |      |      |
|---------------|------|------|------|
|               | ILM2 | ILM1 | ILM0 |
| Initial value | 0    | 0    | 0    |

**Table 2.3-1 Level Indicated by Interrupt Level Mask Register (ILM)**

| ILM2 | ILM1 | ILM0 | Level value | Permitted interrupt level |
|------|------|------|-------------|---------------------------|
| 0    | 0    | 0    | 0           | Interrupt prohibited      |
| 0    | 0    | 1    | 1           | "0" only                  |
| 0    | 1    | 0    | 2           | Level value less than 1   |
| 0    | 1    | 1    | 3           | Level value less than 2   |
| 1    | 0    | 0    | 4           | Level value less than 3   |
| 1    | 0    | 1    | 5           | Level value less than 4   |
| 1    | 1    | 0    | 6           | Level value less than 5   |
| 1    | 1    | 1    | 7           | Level value less than 6   |

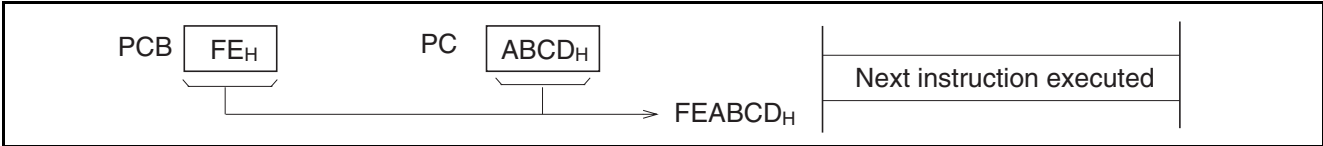
### 2.3.4 Program Counter (PC)

This section explains the program counter (PC) functions.

■ Program Counter (PC)

PC is a 16-bit counter indicating the lower 16 bits in the memory address of an instruction code to be executed by CPU. An upper 8-bit address is indicated with the program count bank register (PCB). PC contents are updated by condition branch instructions, subroutine call instructions, interrupts, or resets. It may also be used as a base pointer for operand access. Figure 2.3-10 explains the program counter (PC) functions.

Figure 2.3-10 Program Counter (PC) Functions



## 2.3.5 Program Counter Bank Register (PCB)

---

This section explains the program counter bank register (PCB) functions.

---

### ■ Program Counter Bank Register (PCB) <Initial Value: Value in Reset Vector>

The program counter bank register (PCB) consists of the following registers:

- Data bank register (DTB) < Initial value: 00<sub>H</sub> >
- User stack bank register (USB) < Initial value: 00<sub>H</sub> >
- System stack bank register (SSB) < Initial value: 00<sub>H</sub> >
- Additional data bank register (ADB) < Initial value: 00<sub>H</sub> >

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

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

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

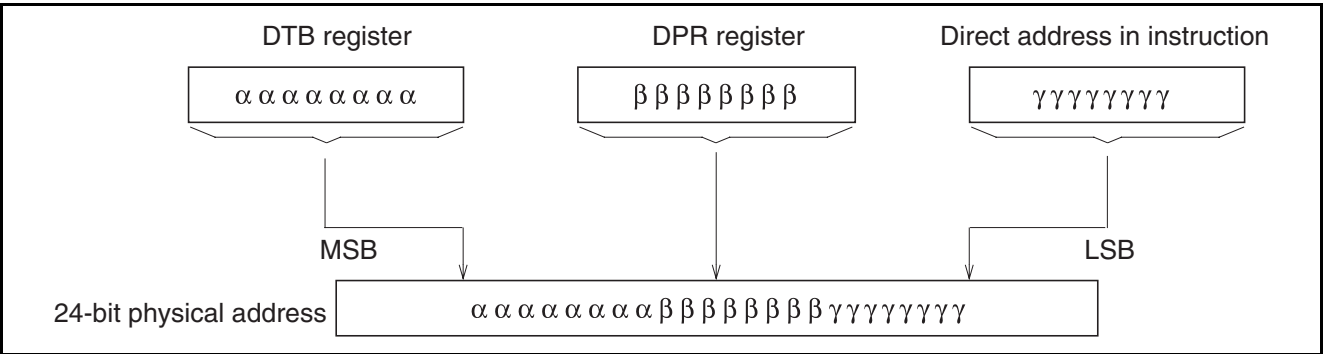
### 2.3.6 Direct Page Register (DPR)

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

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

The direct page register (DPR) specifies, as shown in Figure 2.3-11, addresses 8 to 15 of an instruction operand in the direct addressing mode. DPR has a length of 8 bits, and is initialized to 01<sub>H</sub> by a reset. It also allows reading and writing by instructions. Figure 2.3-11 illustrates the generation of a physical address in the direct addressing mode.

Figure 2.3-11 Generating a Physical Address in Direct Addressing Mode



## 2.3.7 General-Purpose Register (Register Bank)

This section explains the general-purpose register (register bank) functions.

### ■ General-purpose Register (Register Bank)

A register bank consists of 8 words and is used as a general-purpose register for arithmetic operation in the byte register (R0 to R7), word register (RW0 to RW7), and long-word register (RL0 to RL3). A register bank is also used as an instruction pointer. Table 2.3-2 lists the register functions and Figure 2.3-12 shows the relationship among registers.

Register bank values are not initialized by a reset, the same as for RAM spaces, but the state before resetting is kept.

At power-on, however, the values are undefined.

**Table 2.3-2 Register Functions**

|            |   |
|------------|---|
| R0 to R7   | Used as operand in different instructions<br><b>Note:</b><br>R0 is used as the barrel shift counter or normalization instruction counter. |
| RW0 to RW7 | Used as a pointer or operand in different instructions<br><b>Note:</b><br>RW0 is used as a string instruction counter.                    |
| RL0 to RL3 | Used as a long pointer or operand in different instructions   |

**Figure 2.3-12 Relationship Among Registers**

|    |     |     |  |    |     |     |
|----|-----|-----|--|----|-----|-----|
|    | RW0 | RL0 |  | R4 | RW6 | RL3 |
|    | RW1 |     |  |    |     |     |
|    | RW2 | RL1 |  | R5 |     |     |
|    | RW3 |     |  |    |     |     |
| R0 |     | RL2 |  | R6 | RW7 |     |
| R1 |     |     |  |    |     |     |
| R2 |     |     |  | R7 |     |     |
| R3 |     |     |  |    |     |     |

## 2.4 Prefix Codes

By inserting a prefix code before an instruction, part of an instruction operation may change. Three types of prefix codes are provided: bank select prefixes, common register bank prefixes, and flag change suppress prefixes.

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

Memory space used in data access is determined according to the addressing mode.

By inserting a bank select prefix before an instruction, the instruction selects the memory space used for data access regardless of the addressing mode in use.

Table 2.4-1 shows the relationship between the bank select prefix and a selected space.

**Table 2.4-1 Bank Select Prefix**

| Bank select prefix | Selected space   |
|--------------------|--|
| PCB                | PC space   |
| DTB                | Data space   |
| ADB                | AD space   |
| SPB                | Either SSB or USB space is used depending on the stack flag value. |

Be careful when you are using the following instructions:

○ **String instruction (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)**

This uses bank registers specified with an operand regardless of a prefix.

○ **Stack operation instruction (PUSHW, POPW)**

SSB or USB is used depending on the S-flag, regardless of a prefix.

○ **I/O access instruction**

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

The I/O space of a bank is used regardless of whether a prefix is in an instruction.

○ **Flag change instruction (AND CCR,#imm8,OR CCR,#imm8)**

An instruction operation is normal, but a prefix affects the next instruction.

○ **POPW PS**

Regardless of prefix, SSB or USB is used depending on the S-flag. A prefix affects the next instruction.

○ **MOV ILM, #imm8**

If an instruction operation is normal, but a prefix affects the next instruction.

- **RETI**

SSB is used regardless of prefix.

## ■ Common Register Bank Prefix (CMR)

To facilitate data exchange between multiple tasks, the same register bank needs to be easily accessed regardless of each register bank pointer (RP) value. If CMR is inserted before an instruction that accesses a register bank, the instruction accesses the common bank with addresses ranging from 000180<sub>H</sub> to 00018F<sub>H</sub> (register bank selected if RP = 0) regardless of the current RP value.

However, be careful when you are using the following instructions:

- **String instruction (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)**

If an interrupt request is issued while a string instruction with a prefix code is executed, the prefix code is disabled when the string instruction is returned after the interrupt is processed. Thus, with interrupt processing, the string instruction causes an error. Do not add a CMR prefix to such a string instruction.

- **Flag change instruction (AND CCR, #imm8, OR CCR, #imm8)**

An instruction operation is normal, but a prefix affects the next instruction.

- **MOV ILM, #imm8**

An instruction operation is normal, but a prefix affects the next instruction.

## ■ Flag Change Suppress Prefix (NCC)

To suppress a flag change, specify the flag change suppress prefix code (NCC). By inserting NCC before an instruction, the flag change caused by an instruction is suppressed. However, be careful if you use instructions listed below.

- **String instruction (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)**

If an interrupt request occurs while a string instruction with prefix code is executed, the prefix code is disabled when the string instruction is returned after the interrupt is processed. Thus, with interrupt processing, the string instruction causes an error. Do not add a CMR prefix to the above string instruction.

- **Flag change instruction (AND CCR, #imm8, OR CCR, #imm8)**

An instruction operation is normal, but a prefix affects the next instruction.

- **Interrupt instruction (INT #vct8, INT9, INT addr16, INTO addr24, POPW PS)**

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

- **JCTX@A**

CCR change according to the instruction specification regardless of the prefix.

- **MOV ILM, #imm8**

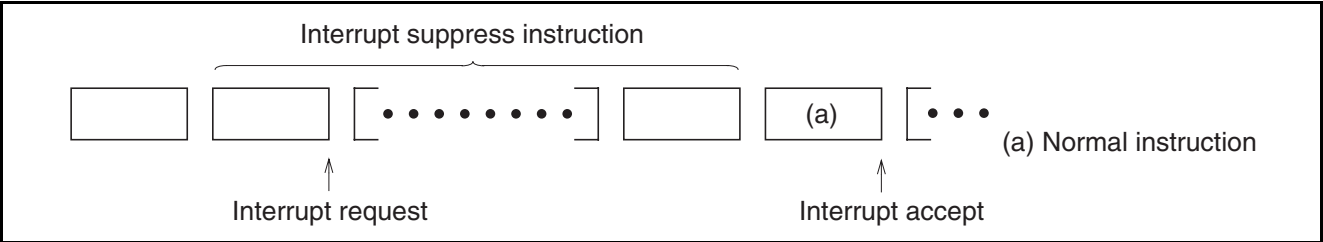
An instruction operation is normal, but a prefix affects the next instruction.



■ Interrupt Suppress Instruction

No interrupt requests are sampled on ten types of instruction as follows.  
MOV ILM, #imm8/PCB/SPB/OR CCR, #imm8/NCC  
AND CCR, #imm8/ADB/CMR/POPW PS/DTB  
If an effective interrupt request is issued when any of above instructions is executed, an interrupt may be processed only if instructions other than the above are executed. For more information, see Figure 2.4-1.

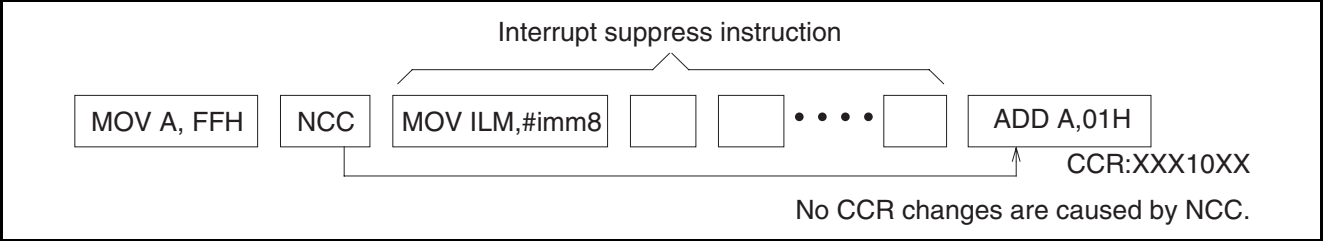
Figure 2.4-1 Interrupt Suppress Instruction



■ Restrictions on Interrupt Suppress Instruction and Prefix Instruction

If a prefix code is inserted before an interrupt is suppressed, the prefix code affects up to the first instruction that appears after any code other than interrupt suppress instructions, as shown in Figure 2.4-2.

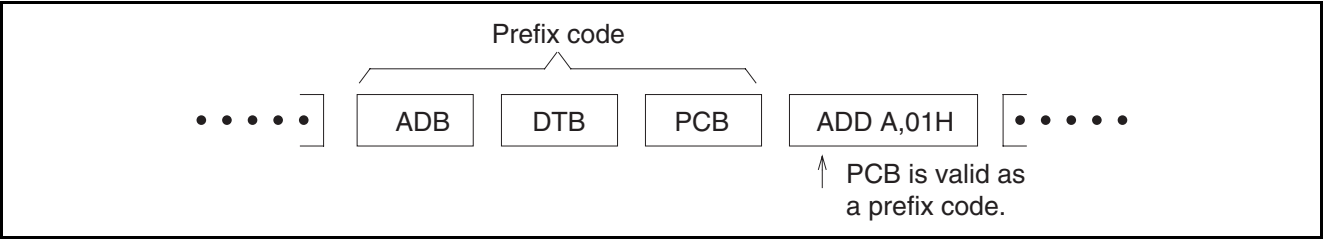
Figure 2.4-2 Interrupt Suppress Instruction and Prefix Code



■ Continuous Prefix Codes

If continuous prefix codes conflict, the latest ones are valid, as shown in Figure 2.4-3. Such conflicting prefix codes mean PCB, ADB, DTB, and SPB, as shown in Figure 2.4-3.

Figure 2.4-3 Continuous Prefix Codes



# CHAPTER 3    INTERRUPT

---

**This chapter explains interrupts and direct memory access (DMA).**

---

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

## 3.1 Overview of Interrupt

---

F<sup>2</sup>MC-16LX has the following four interrupt functions that temporarily stop processing currently being performed and make the control move to programs defined separately when certain events occur:

- **Hardware interrupt**
  - **Software interrupt**
  - **Interrupt by  $\mu$ DMAC**
  - **Exception processing**
- 

### ■ Types and Functions of Interrupts

#### ○ **Hardware interrupt**

Control is moved to the user-defined interrupt processing program in response to an interrupt request from a peripheral function.

#### ○ **Software interrupt**

Control is moved to the user-defined interrupt processing program by execution of a dedicated instruction for software interrupts (e.g., INT instruction).

#### ○ **Interrupt by $\mu$ DMAC**

$\mu$ DMAC is a function used to automatically transfer data between peripheral functions and memory. Previous data transfers by an interrupt processing program is provided in the same way as the direct memory access (DMAC). When a transfer for data of a specified count is completed, an interrupt processing program is automatically executed. Interrupt by  $\mu$ DMAC is a type of hardware interrupts.

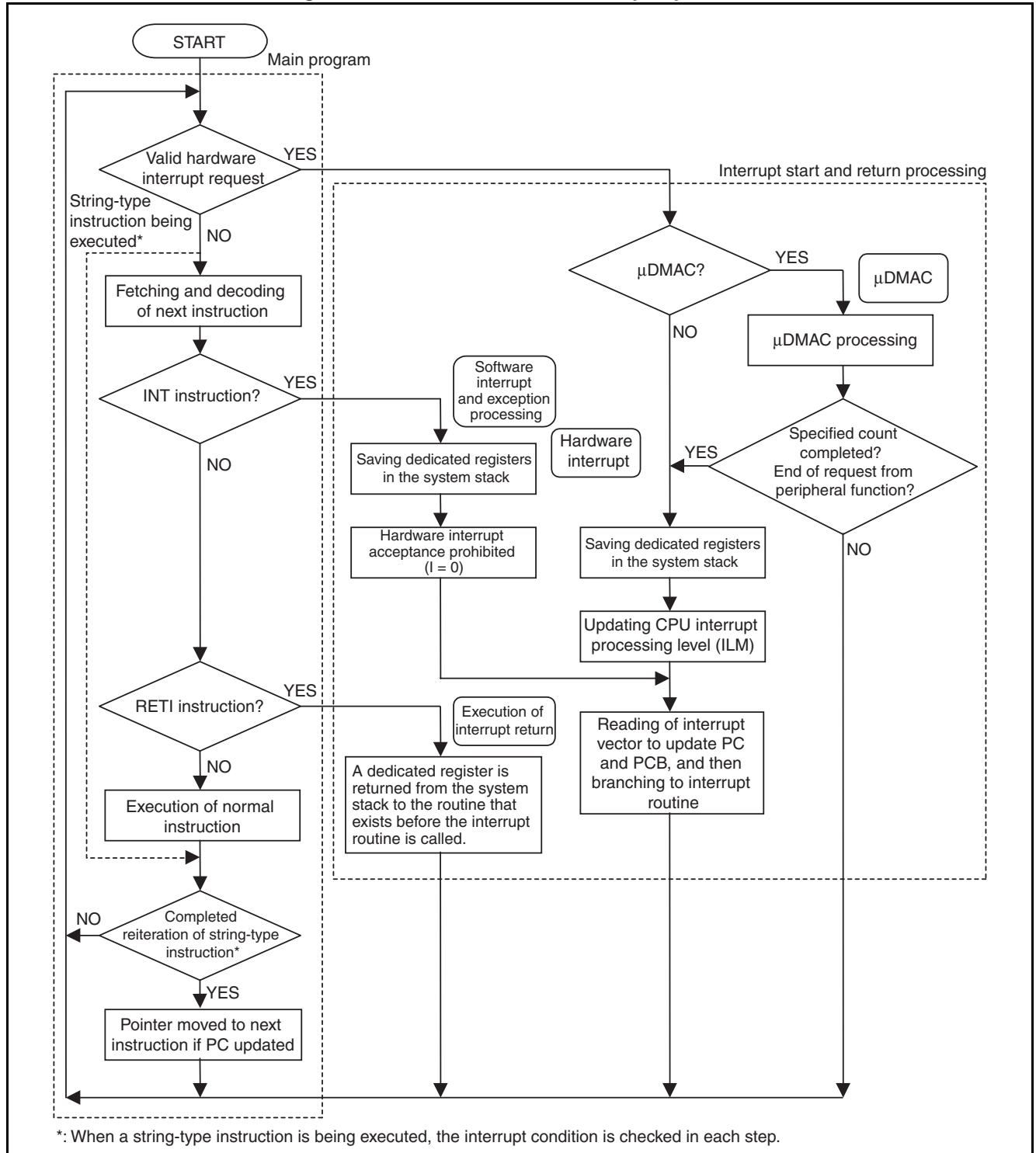
#### ○ **Exception processing**

Exception processing is basically the same with interrupts. Exceptions are handled by interrupting normal processing at the instruction boundary if exception event (execution of undefined instruction) generation is detected. Exception processing is equivalent to software interrupt instruction "INT10".

## ■ Overall Flow of Interrupt Operation

Four types of interrupt functions provide start and return processing, as shown in Figure 3.1-1.

### Figure 3.1-1 Overall Flow of Interrupt Operation



## 3.2 Interrupt Factor and Interrupt Vector

The F<sup>2</sup>MC-16LX has functions corresponding to 256 types of interrupt factors, and 256 interrupt vector tables are assigned to the highest address in the memory. The interrupt vector is shared by all interrupts.

A software interrupt may use all of the above interrupts (INT0 to INT255), although parts of interrupt vectors are shared by hardware interrupt and exception processing interrupt. A hardware interrupt has a specific interrupt vector and interrupt control register (ICR) for each peripheral function.

### ■ Interrupt Vector

Interrupt vector tables referred during interrupt processing are assigned to the memory area of the highest address (FFFC00<sub>H</sub> to FFFFFFF<sub>H</sub>). The interrupt vectors for  $\mu$ DMAC, exception processing, hardware interrupt, and software interrupt share the same area. Table 3.2-1 lists the assignment of interrupt numbers and interrupt vectors.

Table 3.2-1 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> | Unused              | #0            | None                     |
| :                              | :                   | :                   | :                   | :                   | :             | :                        |
| INT7                           | FFFFE0 <sub>H</sub> | FFFFE1 <sub>H</sub> | FFFFE2 <sub>H</sub> | Unused              | #7            | None                     |
| INT8                           | FFFFDC <sub>H</sub> | FFFFDD <sub>H</sub> | FFFFDE <sub>H</sub> | FFFFDF <sub>H</sub> | #8            | (RESET vector)           |
| INT9                           | FFFFD8 <sub>H</sub> | FFFFD9 <sub>H</sub> | FFFFDA <sub>H</sub> | Unused              | #9            | None                     |
| INT10                          | FFFFD4 <sub>H</sub> | FFFFD5 <sub>H</sub> | FFFFD6 <sub>H</sub> | Unused              | #10           | < Exception processing > |
| INT11                          | FFFFD0 <sub>H</sub> | FFFFD1 <sub>H</sub> | FFFFD2 <sub>H</sub> | Unused              | #11           | Hardware interrupt #0    |
| INT12                          | FFFFCC <sub>H</sub> | FFFFCD <sub>H</sub> | FFFFCE <sub>H</sub> | Unused              | #12           | Hardware interrupt #1    |
| INT13                          | FFFFC8 <sub>H</sub> | FFFFC9 <sub>H</sub> | FFFFCA <sub>H</sub> | Unused              | #13           | Hardware interrupt #2    |
| INT14                          | FFFFC4 <sub>H</sub> | FFFFC5 <sub>H</sub> | FFFFC6 <sub>H</sub> | Unused              | #14           | Hardware interrupt #3    |
| :                              | :                   | :                   | :                   | :                   | :             | :                        |
| INT254                         | FFFC04 <sub>H</sub> | FFFC05 <sub>H</sub> | FFFC06 <sub>H</sub> | Unused              | #254          | None                     |
| INT255                         | FFFC00 <sub>H</sub> | FFFC01 <sub>H</sub> | FFFC02 <sub>H</sub> | Unused              | #255          | None                     |

#### Reference:

For interrupt vectors that are not used, Fujitsu recommends specifying such vectors for the address for exception processing.

## ■ Interrupt Factors, Interrupt Vector, and Interrupt Control Register

Table 3.2-2 shows the relationship among interrupt factors excluding software interrupts, interrupt vectors, and interrupt control registers.

**Table 3.2-2 Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers (Sheet 1 of 2)**

| Interrupt factor   | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|--|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|  |                          |                      | Number           | Address             | Number                     | Address             |
| Reset  | ×                        | -                    | #08              | FFFFDC <sub>H</sub> | -                          | -                   |
| INT9 instruction   | ×                        | -                    | #09              | FFFFD8 <sub>H</sub> | -                          | -                   |
| Exception  | ×                        | -                    | #10              | FFFFD4 <sub>H</sub> | -                          | -                   |
| INT0   | ○                        | 0                    | #11              | FFFFD0 <sub>H</sub> | ICR00                      | 0000B0 <sub>H</sub> |
| INT1   | ○                        | ×                    | #12              | FFFFCC <sub>H</sub> |                            |                     |
| INT2   | ○                        | ×                    | #13              | FFFFC8 <sub>H</sub> | ICR01                      | 0000B1 <sub>H</sub> |
| INT3   | ○                        | ×                    | #14              | FFFFC4 <sub>H</sub> |                            |                     |
| INT4   | ○                        | ×                    | #15              | FFFFC0 <sub>H</sub> | ICR02                      | 0000B2 <sub>H</sub> |
| INT5   | ○                        | ×                    | #16              | FFFFBC <sub>H</sub> |                            |                     |
| INT6   | ○                        | ×                    | #17              | FFFFB8 <sub>H</sub> | ICR03                      | 0000B3 <sub>H</sub> |
| INT7   | ○                        | ×                    | #18              | FFFFB4 <sub>H</sub> |                            |                     |
| PWC1 (Only MB90485 series)   | ○                        | ×                    | #19              | FFFFB0 <sub>H</sub> | ICR04                      | 0000B4 <sub>H</sub> |
| PWC2 (Only MB90485 series)   | ○                        | ×                    | #20              | FFFFAC <sub>H</sub> |                            |                     |
| PWC0 (Only MB90485 series)   | ○                        | 1                    | #21              | FFFFA8 <sub>H</sub> | ICR05                      | 0000B5 <sub>H</sub> |
| PPG0/PPG1 counter borrow   | ×                        | ×                    | #22              | FFFFA4 <sub>H</sub> |                            |                     |
| PPG2/PPG3 counter borrow   | ×                        | ×                    | #23              | FFFFA0 <sub>H</sub> | ICR06                      | 0000B6 <sub>H</sub> |
| PPG4/PPG5 counter borrow   | ×                        | ×                    | #24              | FFFF9C <sub>H</sub> |                            |                     |
| 8/16-bit U/D counter / timer (ch.0,1) / compare / underflow / overflow / up-down reverse | ○                        | ×                    | #25              | FFFF98 <sub>H</sub> | ICR07                      | 0000B7 <sub>H</sub> |
| Input capture (ch.0) load  | ○                        | 5                    | #26              | FFFF94 <sub>H</sub> |                            |                     |
| Input capture (ch.1) load  | ○                        | 6                    | #27              | FFFF90 <sub>H</sub> | ICR08                      | 0000B8 <sub>H</sub> |
| Output compare (ch.0) match  | ○                        | 8                    | #28              | FFFF8C <sub>H</sub> |                            |                     |
| Output compare (ch.1) match  | ○                        | 9                    | #29              | FFFF88 <sub>H</sub> | ICR09                      | 0000B9 <sub>H</sub> |
| Output compare (ch.2) match  | ×                        | 10                   | #30              | FFFF84 <sub>H</sub> |                            |                     |
| Output compare (ch.3) match  | ×                        | ×                    | #31              | FFFF80 <sub>H</sub> | ICR10                      | 0000BA <sub>H</sub> |
| Output compare (ch.4) match  | ×                        | ×                    | #32              | FFFF7C <sub>H</sub> |                            |                     |

## CHAPTER 3 INTERRUPT

**Table 3.2-2 Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers (Sheet 2 of 2)**

| Interrupt factor   | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|--|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|  |                          |                      | Number           | Address             | Number                     | Address             |
| Output compare (ch.5) match  | ○                        | ×                    | #33              | FFFF78 <sub>H</sub> | ICR11                      | 0000BB <sub>H</sub> |
| UART transmit completed  | ○                        | 11                   | #34              | FFFF74 <sub>H</sub> |                            |                     |
| 16-bit free-run timer overflow<br>16-bit reload timer underflow *2 | ○                        | 12                   | #35              | FFFF70 <sub>H</sub> | ICR12                      | 0000BC <sub>H</sub> |
| UART receive completed   | ◎                        | 7                    | #36              | FFFF6C <sub>H</sub> |                            |                     |
| SI01(ch.0)   | ○                        | 13                   | #37              | FFFF68 <sub>H</sub> | ICR13                      | 0000BD <sub>H</sub> |
| SI02(ch.1)   | ○                        | 14                   | #38              | FFFF64 <sub>H</sub> |                            |                     |
| I <sup>2</sup> C interface (Only MB90485 series)                   | ×                        | ×                    | #39              | FFFF60 <sub>H</sub> | ICR14                      | 0000BE <sub>H</sub> |
| A/D converter  | ○                        | 15                   | #40              | FFFF5C <sub>H</sub> |                            |                     |
| FLASH write/delete, time-base timer, watch timer*1                 | ○                        | ×                    | #41              | FFFF58 <sub>H</sub> | ICR15                      | 0000BF <sub>H</sub> |
| Delay interrupt generation module                                  | ×                        | ×                    | #42              | FFFF54 <sub>H</sub> |                            |                     |

x: The interrupt request flag cannot be cleared by the interrupt clear signal.

○: The interrupt request flag is cleared.

◎: The interrupt request flag is cleared. The stop request is provided.

\*1: Caution: The FLASH write/erase, time-base timer, and watch timer cannot be used at the same time.

\*2: Please write "0" in the INTE bit, after prohibiting interrupt by setting the IL2 bit to IL0 bit of the interrupt control register to "111<sub>B</sub>", if the reload timer underflow interrupt setting is changed from enable (INTE bit of TMCSR registers =1) to prohibit (INTE bit of TMCSR registers =0).

### Note:

If there are two interruption factors to the same interruption number, both interrupt request flag is cleared by interrupt clear signal on the resource. Therefore, when one of two factors uses the EI<sup>2</sup>OS function or the μDMAC function, the other interrupt function cannot be used. Set the interruption request permission bit of the corresponding resource to "0" and handle with software polling processing.

### 3.3 Interrupt Control Register and Peripheral Function

Interrupt control registers (ICR00 to ICR15) are located in the interrupt controller, and they correspond to every peripheral function that has an interrupt function. This register controls interrupts.

#### ■ List of Interrupt Control Registers

Table 3.3-1 lists interrupt control registers and the corresponding peripheral functions.

**Table 3.3-1 List of Interrupt Control Registers**

| Address             | Register                      | Abbreviation | Corresponding peripheral functions  |
|---------------------|-------------------------------|--------------|---|
| 0000B0 <sub>H</sub> | Interrupt control register 00 | ICR00        | INT0, 1   |
| 0000B1 <sub>H</sub> | Interrupt control register 01 | ICR01        | INT2, 3   |
| 0000B2 <sub>H</sub> | Interrupt control register 02 | ICR02        | INT4, 5   |
| 0000B3 <sub>H</sub> | Interrupt control register 03 | ICR03        | INT6, 7   |
| 0000B4 <sub>H</sub> | Interrupt control register 04 | ICR04        | PWC1,2 (Only MB90485 series)  |
| 0000B5 <sub>H</sub> | Interrupt control register 05 | ICR05        | 8/16-bit PPG timer 0, 1<br>PWC0 (Only MB90485 series)                             |
| 0000B6 <sub>H</sub> | Interrupt control register 06 | ICR06        | 8/16-bit PPG timer 2, 3, 4, 5   |
| 0000B7 <sub>H</sub> | Interrupt control register 07 | ICR07        | 8/16UD counter 0, 1, input capture 0  |
| 0000B8 <sub>H</sub> | Interrupt control register 08 | ICR08        | Input capture 1, output compare 0   |
| 0000B9 <sub>H</sub> | Interrupt control register 09 | ICR09        | Output compare 1, 2   |
| 0000BA <sub>H</sub> | Interrupt control register 10 | ICR10        | Output compare 3, 4   |
| 0000BB <sub>H</sub> | Interrupt control register 11 | ICR11        | Output compare 5, UART transmit   |
| 0000BC <sub>H</sub> | Interrupt control register 12 | ICR12        | UART receive, 16-bit free-run timer,<br>16-bit reload timer                       |
| 0000BD <sub>H</sub> | Interrupt control register 13 | ICR13        | SIO0, 1   |
| 0000BE <sub>H</sub> | Interrupt control register 14 | ICR14        | A/D, I <sup>2</sup> C interface (Only MB90485 series)                             |
| 0000BF <sub>H</sub> | Interrupt control register 15 | ICR15        | FLASH write, time-base timer, watch timer,<br>delayed interrupt generation module |

**Note:**

Avoid accessing the interrupt control register (ICR) with read-modify-write instructions since they may cause incorrect operation.



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

The interrupt control register (ICR00 to ICR15) corresponds to every peripheral function that has interrupt functions for controlling processing during interrupt request generation. This register has different functions between write and read operations.

#### ■ Interrupt Control Register (ICR00 to ICR15) Function

The interrupt control register (ICR00 to ICR15) consists of bit having the following four functions:

- Interrupt level setting bit (IL2 to IL0)
- Extended intelligent I/O service (EI<sup>2</sup>OS) permission bit (ISE3)
- 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 bits (S1, S0)

#### ■ Configuration of Interrupt Control Register (ICR00 to ICR15)

Figure 3.3-1 shows the bit configuration of the interrupt control register (ICR00 to ICR15).

**Figure 3.3-1 Bit Configuration of Interrupt Control Register (ICR00 to ICR15)**

|  |      |      |      |      |      |      |      |      |
|--|------|------|------|------|------|------|------|------|
| Interrupt control register (ICR)                 |      |      |      |      |      |      |      |      |
| Address  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
| 0000B0 <sub>H</sub><br>to<br>0000BF <sub>H</sub> | ICS3 | ICS2 | ICS1 | ICS0 | ISE  | IL2  | IL1  | IL0  |
|  | W    | W    | W    | W    | R/W  | R/W  | R/W  | R/W  |
| Initial value                                    |      |      |      |      |      |      |      |      |
| 00000111 <sub>B</sub>                            |      |      |      |      |      |      |      |      |
| Interrupt control register (ICR)                 |      |      |      |      |      |      |      |      |
| Address  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
| 0000B0 <sub>H</sub><br>to<br>0000BF <sub>H</sub> | —    | —    | S1   | S0   | ISE  | IL2  | IL1  | IL0  |
|  | -    | -    | R    | R    | R/W  | R/W  | R/W  | R/W  |
| Initial value                                    |      |      |      |      |      |      |      |      |
| XX000111 <sub>B</sub>                            |      |      |      |      |      |      |      |      |
| R/W: Readable/Writable                           |      |      |      |      |      |      |      |      |
| R : Read only                                    |      |      |      |      |      |      |      |      |
| W : Write only                                   |      |      |      |      |      |      |      |      |
| - : Undefined                                    |      |      |      |      |      |      |      |      |

Notes:


- Only when extended intelligent I/O service (EI<sup>2</sup>OS) is started, the ICS3 to ICS0 bits are valid. Please set "1" to the ISE bit when EI<sup>2</sup>OS is started. Please set "0" to the ISE bit when EI<sup>2</sup>OS is not started. When EI<sup>2</sup>OS is not started, the setting of the ICS3 to ICS0 bits are unnecessary.
- ICS1, ICS0 bits can be only write. S1, S0 bits can be only read.

## ■ Function of Each Bit in Interrupt Control Register (ICR00 to ICR15)

### ○ Interrupt level setting bit (IL2 to IL0)

This specifies the corresponding interrupt level in the peripheral function. A reset initializes the bit to level 7 (no interrupts). Table 3.3-2 lists the relationship between interrupt level setting bits and every interrupt level.

**Table 3.3-2 Relationship Between Interrupt Level Setting Bits and Interrupt Levels**

| IL2 | IL1 | IL0 | Interrupt level   |
|-----|-----|-----|---|
| 0   | 0   | 0   | 0 (Highest interrupt)   |
| 0   | 0   | 1   |  |
| 0   | 1   | 0   |   |
| 0   | 1   | 1   |   |
| 1   | 0   | 0   |   |
| 1   | 0   | 1   |   |
| 1   | 1   | 0   | 6 (Lowest interrupt)  |
| 1   | 1   | 1   | 7 (No interrupt)  |

### ○ Extended intelligent I/O service (EI<sup>2</sup>OS) permission bit (ISE)

When this bit is "1" at the generating interrupt request, EI<sup>2</sup>OS is started. When this bit is "0" at the generating interrupt request, the interruption sequence is started. When the EI<sup>2</sup>OS end requirement is satisfied (without S1, S0=00<sub>B</sub>), the ISE bit is cleared to "0". Please set "0" to the ISE bit with software when the corresponding resource doesn't have the EI<sup>2</sup>OS function. The ISE bit is initialized to "0" by reset.

### ○ Extended intelligent I/O service (EI<sup>2</sup>OS) channel selection bit (ICS3 to ICS0)

The ICS3 to ICS0 bits are write only bits and specify the channel of EI<sup>2</sup>OS descriptor address is determined depending on the value set to the ICS3 to ICS0 bits. The ICS3 to ICS0 bits are initialized to "0000<sub>B</sub>" by reset.

Table 3.3-3 shows the relation between EI<sup>2</sup>OS channel selection bit and the descriptor address.

**Table 3.3-3 Relationship between EI<sup>2</sup>OS Channel Selection Bits and Descriptor Address**

| 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> |
| 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 bit (S1, S0)

S1, S0 bits are read-only bits. Whether the state is operating or terminated can be read by confirming the S1, S0 bits value at the end of EI<sup>2</sup>OS. S1, S0 bits are initializes to "00<sub>B</sub>" by reset. Table 3.3-4 shows the relation between S1, S0 bit and EI<sup>2</sup>OS status.

**Table 3.3-4 Relationship Between EI<sup>2</sup>OS Status Bit and EI<sup>2</sup>OS State**

| S1 | S0 | EI <sup>2</sup> OS state                        |
|----|----|---|
| 0  | 0  | Operating EI <sup>2</sup> OS or inactive        |
| 0  | 1  | Stop due to end of counting                     |
| 1  | 0  | Unused  |
| 1  | 1  | Stop due to generation of request from resource |

## 3.4 Hardware Interrupt

---

Hardware interrupt is a function to temporarily stop the execution of program being executed by the CPU in response to an interrupt request signal from the peripheral function. It then moves control to the interrupt processing program defined by a user. Also,  $\mu$ DMAC and external interrupt may be executed as a kind of hardware interrupt.

---

### ■ Hardware Interrupt Function

#### ○ Hardware interrupt function

A hardware interrupt compares the interrupt level of an interrupt request signal that is output by the peripheral function with interrupt level mask register (ILM) in the CPU processor status (PS). It then refers to the I-flag in the processor status (PS) to determine whether or not the interrupt is acceptable.

If the hardware interrupt is accepted, registered contents in the CPU are automatically saved to the system stack, and the interrupt level currently requested is stored in the interrupt level mask register (ILM). In this event, control then branches to the corresponding interrupt vector.

#### ○ Multiple interrupts

Multiple hardware interrupts can start at one time.

#### ○ $\mu$ DMAC

$\mu$ DMAC is an automatic transfer function between memory and I/O, and if the transfer is completed, a hardware interrupt starts.  $\mu$ DMAC does not start in a multiplex manner, and if some  $\mu$ DMAC process is executed, other interrupt requests and all  $\mu$ DMAC requests wait temporarily.

#### ○ External interrupt

An external interrupt (including wake-up interrupt) is accepted as a hardware interrupt via the peripheral function (interrupt request detect circuit).

#### ○ Interrupt vector

Interrupt processing refers to the interrupt vector table assigned in memory addresses ranging from  $\text{FFFC00}_H$  to  $\text{FFFFFF}_H$  and shared with software interrupts.

For the assignment of interrupt number and interrupt vector, see Section "3.2 Interrupt Factor and Interrupt Vector".

## ■ Configuration of Hardware Interrupt

The hardware-interrupt mechanism is divided into four parts, as shown in Table 3.4-1. To use hardware interrupts, a program must contain settings for the four locations.

**Table 3.4-1 Hardware-interrupt Mechanism**

|   | Hardware-interrupt mechanism                | Function   |
|---|---|--|
| Peripheral function                                   | Interrupt enable bit, interrupt request bit | Control of interrupt request by peripheral function            |
| Interrupt controller                                  | Interrupt control register (ICR)            | Setting for interrupt level and controlling $\mu$ DMAC         |
| CPU   | Interrupt enable flag (I)                   | Identifying interrupt enable state                             |
|   | Interrupt level mask register (ILM)         | Comparing request interrupt level with current interrupt level |
|   | Micro-code                                  | Executing interrupt processing routine                         |
| FFFC00 <sub>H</sub> to FFFFFFF <sub>H</sub> in memory | Interrupt vector table                      | Storing branch address for interrupt processing                |

## ■ Suppressing Hardware Interrupt

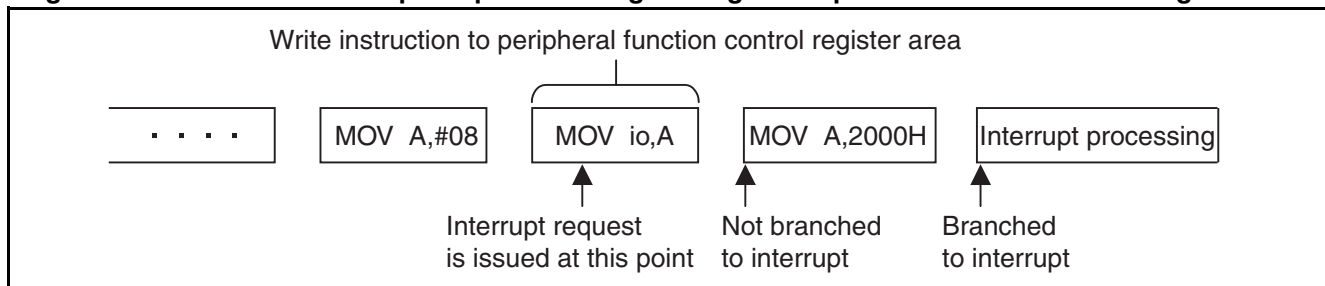
Accepting a hardware interrupt request is suppressed under the following conditions:

- **Suppressing hardware interrupts during writing to peripheral function control register area**

During writing to the peripheral function control register area, no hardware interrupt requests are accepted. This prevents incorrect interrupt-related operations by the CPU during its rewriting of the interrupt control registers with each peripheral function. Peripheral function control register area refers not to a range of 000000<sub>H</sub> to 0000FF<sub>H</sub> for the I/O addressing area, but to the areas assigned to the control register within peripheral function control registers and data register.

Figure 3.4-1 shows hardware interrupt operations during writing to the peripheral function control register area.

**Figure 3.4-1 Hardware Interrupt Requests During Writing to Peripheral Function Control Register Area**



○ **Suppressing hardware interrupts in the interrupt suppress instruction**

Of the ten types of hardware interrupt suppress instruction listed in Table 3.4-2, none can detect whether or not hardware interrupt requests are present, and none can ignore an interrupt request. If a valid hardware interrupt request is generated during execution of these instructions, the interrupt request is not executed until execution of a subsequent instruction is completed. In this case, the subsequent instruction is other than the instructions mentioned above.

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

|  | Prefix code                            | Interrupt and hold suppress instruction<br>(instruction to delay the effect of prefix code) |
|--|--|---|
| Instruction that rejects<br>interrupt and hold<br>requests | PCB<br>DTB<br>ADB<br>SPB<br>CMR<br>NCC | MOV ILM, #imm8<br>OR CCR, #imm8<br>AND CCR, #imm8<br>POPW PS                                |

○ **A hardware interrupt is suppressed during execution of software interrupt**

When a software interrupt starts, other interrupt requests are not accepted so that the I-flag is cleared to "0".

### 3.4.1 Hardware Interrupt Operation

---

This section explains an operation starting from hardware interrupt request generation until completion of interrupt processing.

---

#### ■ Starting Hardware Interrupt

##### ○ Operation of peripheral function (generating an interrupt request)

The peripheral functions including hardware interrupt request functions have the "interrupt request flag" that indicates whether or not to generate an interrupt request and "interrupt enable flag" that selects whether to enable or disable an interrupt request to the CPU. The interrupt request flag is set when a peripheral function specific event is generated, and it issues an interrupt request to the interrupt controller if the interrupt enable flag is set to "enable".

##### ○ Operation of interrupt controller (control of interrupt request)

The interrupt controller compares interrupt levels (IL) in interrupt requests that are received at one time, and selects the request of the highest level (the lowest value of IL) and notifies it to CPU. If multiple requests have the same level, the one with the lower interrupt number has a priority.

##### ○ CPU operation (acceptance of interrupt requests and interrupt processing)

CPU compares the levels (ICR: IL2 to IL0) of received interrupts with the interrupt level mask register (ILM). If  $IL < ILM$  and the interrupt is permitted ( $I = 1$ , in PC: CCR), the interrupt processing microcode starts and interrupt processing is executed, after the instruction currently being executed is completed.

Interrupt processing first saves the contents of a dedicated register (12 bytes of A, DPR, ADB, DTB, PCB, PC and PS) in the system stack (system stack space indicated by SSB and SSP). Then, it loads the interrupt vector to the program counter (PCB, PC), updates ILM, and sets the stuffing (S) flag (i.e., sets the CCR S-flag to "1" and enables the system stack).

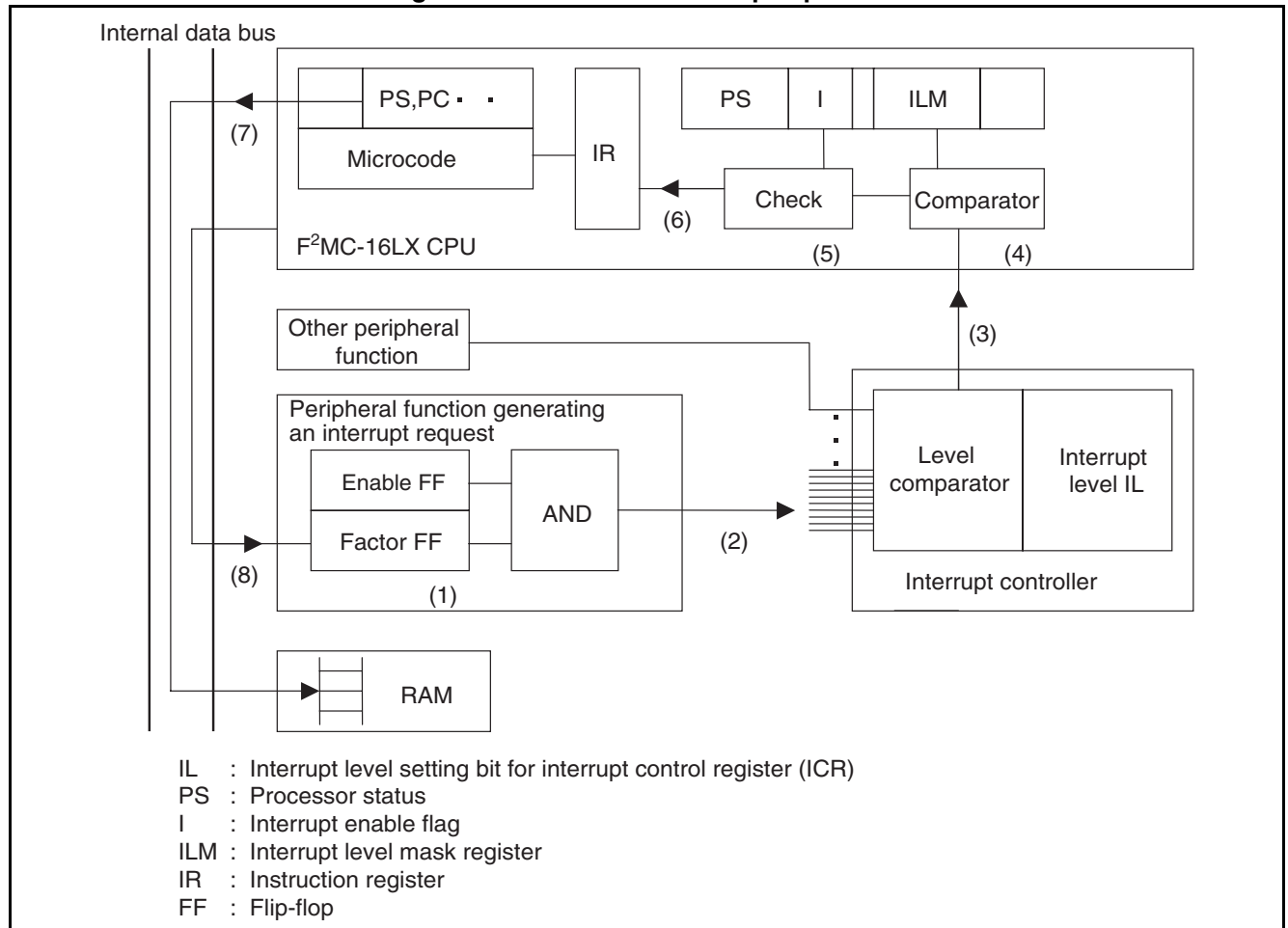
#### ■ Return from Hardware Interrupt

In the interrupt processing program, if the RETI instruction is executed and the interrupt request flag of a peripheral function that is an interrupt factor is cleared, 12 bytes of the data saved in the system stack is returned to the dedicated register to restart the processing in progress before the interrupt branch. By clearing the interrupt request flag, the interrupt request that the peripheral function output to the interrupt controller is automatically removed.

## ■ Hardware Interrupt Operation

Figure 3.4-2 shows the operation from the generation of hardware interrupt until the completion of interrupt processing.

**Figure 3.4-2 Hardware Interrupt Operation**





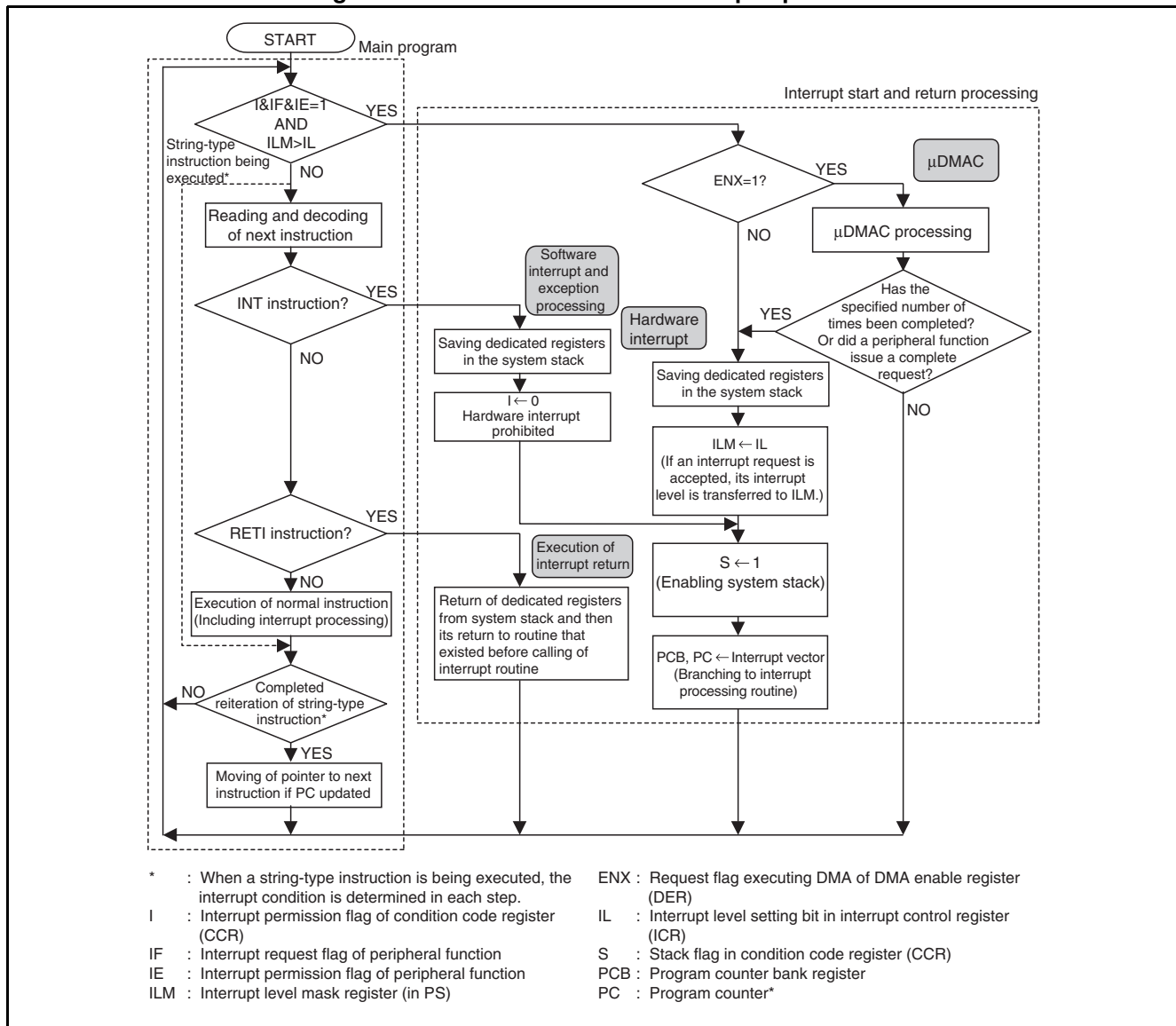
### 3.4.2 Flow of Hardware Interrupt Operation

If an interrupt request is generated by a peripheral function, the interrupt controller transfers its interrupt level to the CPU. If the CPU accepts the interrupt request, the instruction currently being executed is temporarily suspended to execute the interrupt processing routine or to start  $\mu$ DMAC. If a software interrupt is generated by the INT instruction, the interrupt processing routine is executed regardless of the CPU state. Moreover, if a software interrupt is generated by the INT instruction, the hardware interrupt is prohibited.

#### ■ Hardware Interrupt Operation Flow

Figure 3.4-3 shows the hardware interrupt operation flow.

Figure 3.4-3 Flow of Hardware Interrupt Operation



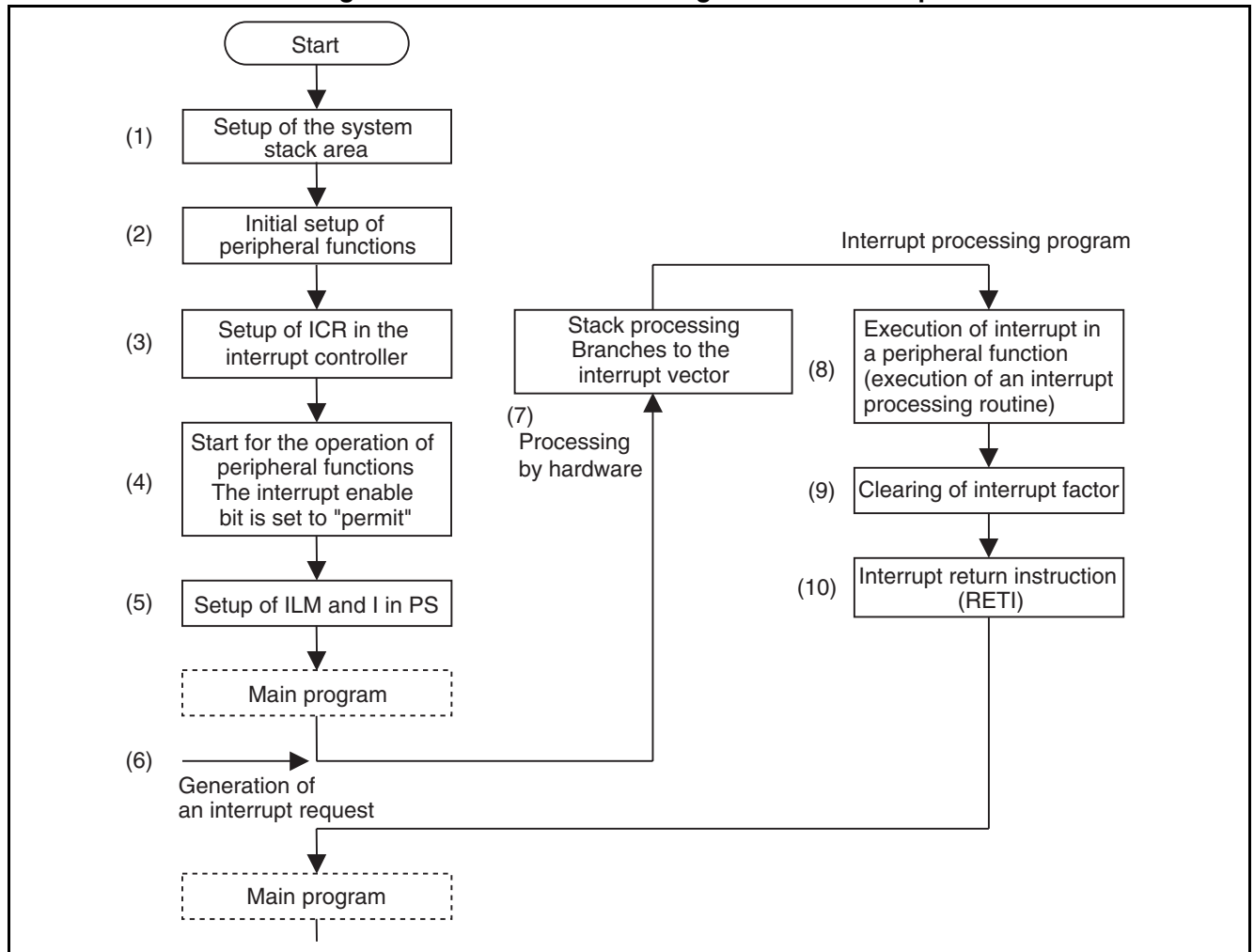
### 3.4.3 Procedure for Using Hardware Interrupt

To use hardware interrupts, necessary setup including the system stack area, peripheral functions, and interrupt control registers (ICR) must be performed.

#### ■ Procedure for Using Hardware Interrupt

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

**Figure 3.4-4 Procedure for Using Hardware Interrupt**



- (1) The system stack area is set up.
- (2) The initial setup of peripheral functions for which interrupt requests can be generated is performed.
- (3) The interrupt control register (ICR) is set up in the interrupt controller.
- (4) The peripheral function is set to the operation start state, and the interrupt enable bit is set to "permit".
- (5) The interrupt level mask register (ILM) and interrupt enable flag (I) are set to "interrupt acceptable".
- (6) A hardware interrupt request is generated by generation of a peripheral function interrupt.
- (7) Interrupt processing hardware saves registers to branch to the interrupt processing program.
- (8) The interrupt processing program processes peripheral functions because of interrupt generation.
- (9) The interrupt request from peripheral function is canceled.
- (10) The interrupt return instruction is executed, and the program is restored to what it was before branching.

### 3.4.4 Multiple Interrupts

Multiple hardware interrupt can be executed by specifying a different interrupt level for each interrupt level setting bit (IL0 to IL2) in the interrupt control register (ICR) in response to multiple interrupt request from the peripheral function.  $\mu$ DMACs cannot be started in duplicate, however.

#### ■ Multiple Interrupt Operations

While an interrupt processing routine is executed, if an interrupt request with a higher level is generated, the interrupt processing is interrupted and then the higher interrupt request is accepted. In this case, after execution of the interrupt with the higher level is completed, the interrupt processing being stopped is restarted. The interrupt level can be set in a range of 0 to 7, but a CPU does not accept the interrupt request when the level is set to 7.

While an interrupt is executed, if another interrupt request with the same or lower level occurs, that interrupt request waits until the current interrupt is completed, unless ILM is changed by the I-flag. In the interrupt processing routine, if the I-flag in the condition code register (CCR) is set to "interrupt prohibited" (I in CCR set to "0") or the interrupt level mask register (ILM) is set to "interrupt prohibited" (ILM set to "000"), the starting of multiple interrupts within the interrupt can be temporarily prohibited.

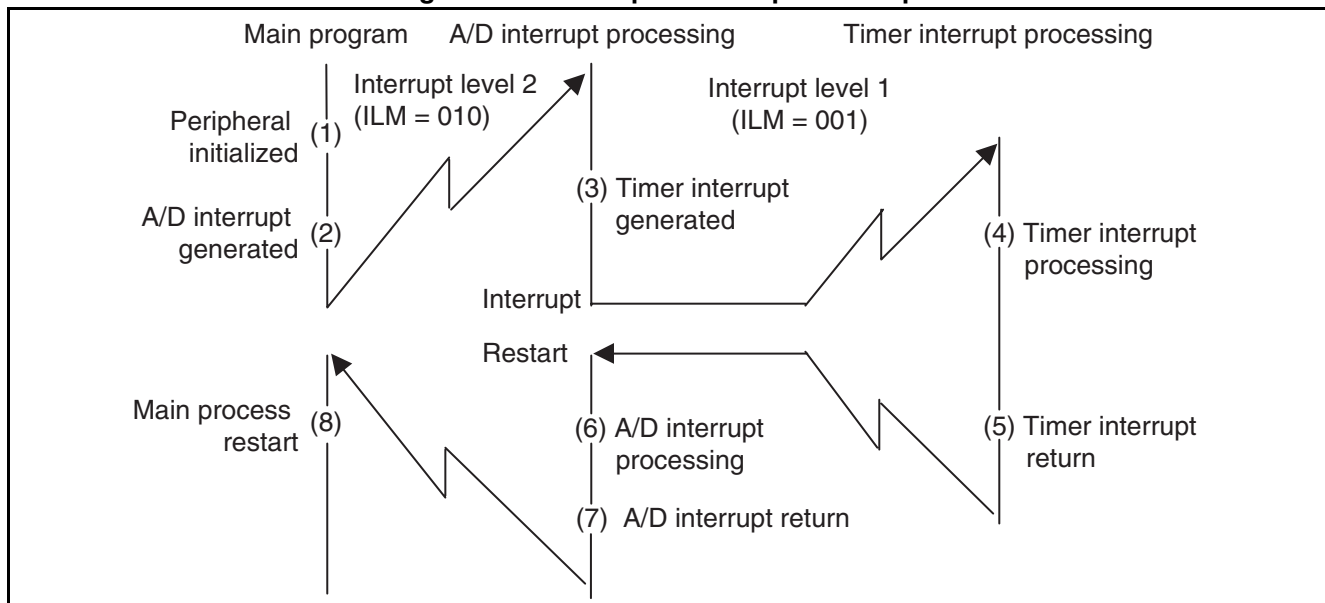
Note:

$\mu$ DMAC cannot be started in duplicate. All other interrupt requests and  $\mu$ DMAC requests have to wait during execution of  $\mu$ DMAC,

#### ■ Example of Multiple Interrupts

Suppose a timer interrupt has priority over the A/D converter. In this case, the interrupt level of the A/D converter is "2" whereas that of timer interrupt is "1". If a timer interrupt is generated while an A/D converter interrupt is executed, the processing shown in the Figure 3.4-5 is performed.

Figure 3.4-5 Example of Multiple Interrupts



- **A/D interrupt generation**

When the A/D converter interrupt processing starts, the interrupt level mask register (ILM) is automatically set to the same interrupt level (i.e., 2 in this example) as that for the A/D converter (IL2 to IL0 in ICR). In this example, if an interrupt request of level 1 or 0 is generated, the interrupt with higher priority is executed first.

- **End of interrupt processing**

If interrupt processing is completed and a return instruction (RETI) is then executed, the values of the dedicated registers (A, DPR, ADB, DTB, PCB, PC, PS) saved in the stack are returned and the values of the interrupt level mask register (ILM) are specified to those defined before the interrupt.

### 3.4.5 Hardware Interrupt Processing Time

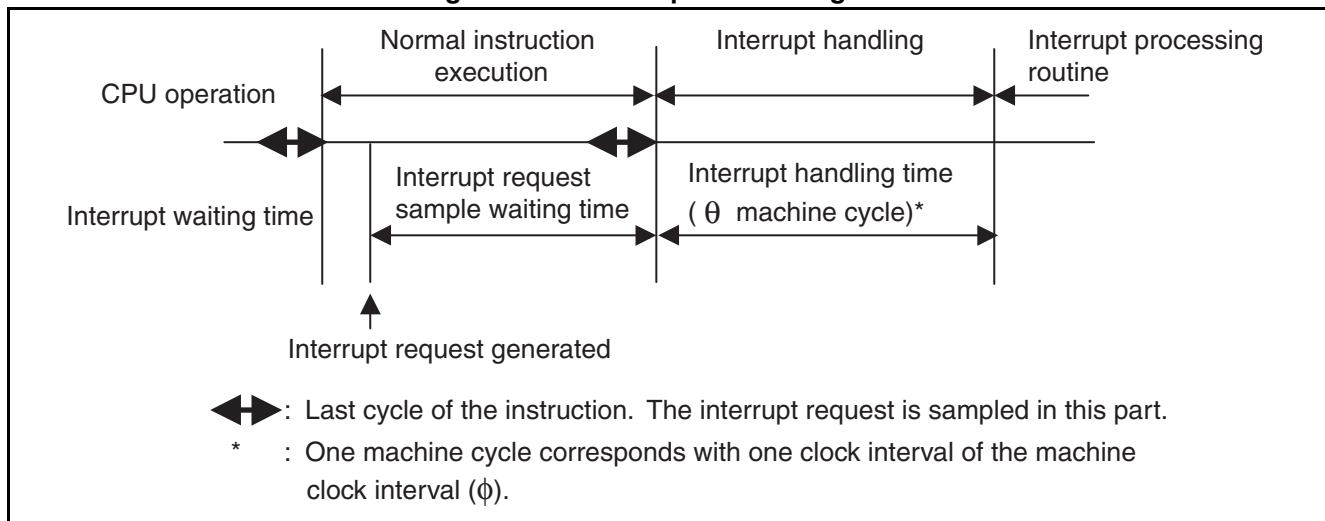
The time period starting from generation of a hardware interrupt request until the execution of interrupt handling routine requires the time until the instruction currently being executed is completed plus the interrupt processing time.

#### ■ Hardware Interrupt Processing Time

The time period starting from generation of a hardware interrupt request and acceptance of the interrupt until the execution of interrupt handling routine requires the interrupt request sample waiting time and interrupt handling time (time required for preparing interrupt processing).

Figure 3.4-6 illustrates the interrupt processing time.

**Figure 3.4-6 Interrupt Processing Time**



#### ○ Interrupt request sample waiting time

Refers to a time period starting after an interrupt request is generated until the instruction currently being executed is completed. Sampling is performed in the last cycle of each instruction to determine whether an interrupt request is generated or not. Thus, during execution of each instruction, the CPU is unable to recognize an interrupt request, resulting in a waiting time.

The interrupt request sample waiting time reaches the maximum value if an interrupt request occurs immediately after the start of PCPW, RW0, ... RW7 instructions (45 machine cycles), which have the longest execution cycle.

○ **Interrupt processing time ( $\theta$  machine cycles)**

After the CPU accepts an interrupt request, the CPU saves the dedicated registers in the system stack and fetches the interrupt vector. The interrupt processing time is thus derived from the following formula:

- At interrupt start:  $\theta = 24 + 6 \times Z$  machine cycles
- At interrupt return:  $\theta = 11 + 6 \times Z$  machine cycles (RETI instruction)

The interrupt processing time differs depending on the address indicated by the stack pointer. Table 3.4-3 lists correction values (Z) for interrupt processing times.

One machine cycle corresponds to a clock interval of machine clocks ( $\phi$ ).

**Table 3.4-3 Correction Values (Z) for Interrupt Processing Times**

| Address pointed by stack pointer | Correction value (Z) |
|----------------------------------|----------------------|
| External 8 bits                  | +4                   |
| External even address            | +1                   |
| External odd address             | +4                   |
| Internal even address            | 0                    |
| Internal odd address             | +2                   |

## 3.5 Software Interrupt

---

Software interrupt is a function used to move control to the user-defined program for interrupt processing from a program that the CPU is being executed if a software interrupt instruction (INT instruction) is executed. A hardware interrupt is stopped while a software interrupt is executed.

---

### ■ Start of Software Interrupt

#### ○ Software interrupt start

To start a software interrupt, execute the INT instruction. A software interrupt request has neither the interrupt request flag nor enable flag. It always generates an interrupt request if the INT instruction is executed.

#### ○ Hardware interrupt suppressed

Because the INT instruction has no interrupt levels, the interrupt level mask register (ILM) is not updated. During INT instruction execution, the I-flag in the condition code register (CCR) is set to "0" to mask hardware interrupts. To permit a hardware interrupt during software interrupt processing, set the I-flag to "1" in the software interrupt processing routine.

#### ○ Software interrupt operation

If the CPU obtains the INT instruction and execute it, a microcode for software interrupt processing starts. This microcode is used to save the registers inside the CPU in the system stack and to mask hardware interrupts (set the I-flag in CCR to "0"), leading to branch to the corresponding interrupt vector.

For the assignment of interrupt numbers and interrupt vectors, see Section "3.2 Interrupt Factor and Interrupt Vector".

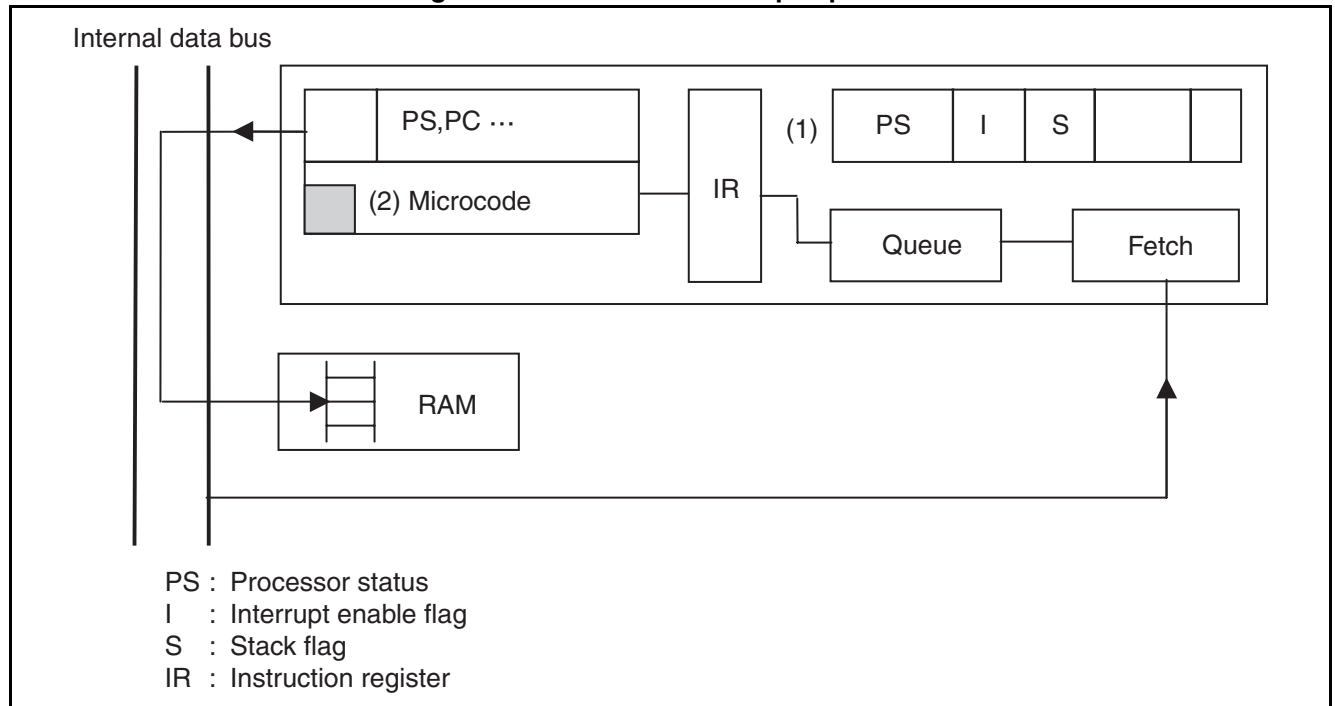
### ■ Return from Software Interrupt

If an interrupt return instruction (RETI instruction) is executed in the interrupt processing program, the 12-byte data saved in the system stack is restored to the dedicated registers, returning control to the processing that was executed before the interrupt processing.

## ■ Software Interrupt Operation

Figure 3.5-1 shows the operation starting from software interrupt generation until interrupt processing completion.

**Figure 3.5-1 Software Interrupt Operation**



- (1) Run a software interrupt instruction.
- (2) Based on the microcode corresponding to the software interrupt instruction, the necessary processes are performed, such as save of the dedicated registers. The branch processing is then executed.
- (3) The RETI instruction is executed in user's interrupt processing routine to end interrupt processing.

## ■ Notes on Software Interrupts

If the program counter bank register (PCB) is set to "FF<sub>H</sub>", the CALLV instruction vector area is duplicated with the table for INT #vct8 instructions. When creating software, make sure that the CALLV instruction and INT #vct8 instruction have no address duplication.



## 3.6 Interrupt by $\mu$ DMA

The  $\mu$ DMA controller is a simplified DMA that has the same function as EI<sup>2</sup>OS. DMA transfers are set up using the DMA descriptor.

### ■ $\mu$ DMA Functions

$\mu$ DMA has the functions listed below.

- Provides an automatic data transfer between a peripheral resource (I/O) and memory.
- CPU program execution stops during the DMA start sequence.
- The DMA transfer channel has 16 channels (a smaller channel number is assigned a higher DMA transfer priority)
- Allow selection of whether or not to increment the transfer source and transfer destination addresses.
- The DMA transfer starts with an interrupt factor of the peripheral resource (I/O).
- DMA transfers are controlled with the (a)  $\mu$ DMA enable register, (b)  $\mu$ DMA stop status register, (c)  $\mu$ DMA status register and (e) descriptor (assigned to a range of 000100<sub>H</sub> to 00017F<sub>H</sub> in RAM).
- STOP requests are issued as a means to stop DMA transfers from a resource.
- After the end of the DMA transfer, a flag is set to the bit corresponding to the transfer end channel of the DMA status register, and an end interrupt is then output to the interrupt controller.

### ■ List of $\mu$ DMA Registers

#### ○ $\mu$ DMA enable register (DER)

$\mu$ DMA enable register (DER) has the bit configuration shown in the diagram below.

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

μDMAC enable register (DER) has the bit functions listed below.

| ENx bit              | Function  |
|----------------------|---|
| 0<br>(Initial value) | Outputs an interrupt request from a resource to the interrupt controller.<br>(An interrupt request from a resource is not used as a DMA start request). |
| 1                    | An interrupt request output from a resource is used as a DMA start request.<br>Cleared to "0" when the DMA transfer byte count reaches "0".             |

Note:

Please shift to the mode after setting "0000<sub>H</sub>" to DMA enable register (DER) whenever shifting to stand-by mode (sleep mode, stop mode, watch mode, time-base timer mode) or (main clock intermittent operation mode, PLL clock intermittent operation mode, sub clock intermittent operation mode).

#### ○ μDMAC stop status register (DSSR)

The bit configuration of the μDMAC stop status register (DSSR) is shown below.

|                         |      |      |      |      |      |      |      |      |                                     |
|-------------------------|------|------|------|------|------|------|------|------|-------------------------------------|
| bit                     | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |                                     |
| 0000A4 <sub>H</sub>     | STP7 | STP6 | STP5 | STP4 | STP3 | STP2 | STP1 | STP0 | DSSR                                |
|                         | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | Initial value 00000000 <sub>B</sub> |
| R/W : Readable/Writable |      |      |      |      |      |      |      |      |                                     |

The function of each bit in the μDMAC stop status register (DSSR) is shown below.

| STPx bit             | Function  |
|----------------------|---|
| 0<br>(Initial value) | No STOP request is accepted in a DMA transfer.  |
| 1                    | STOP request is accepted in a DMA transfer to stop DMA operation.<br>STOP request is accepted the UART receive (ch.7) only.<br>The bits other than the bit7 are not valid.<br>Writing "1" by running software is not valid. |

#### ○ μDMAC status register (DSR)

The bit configuration of the μDMAC status register (DSR) is shown below.

|                         |      |      |      |      |      |      |     |     |                                     |
|-------------------------|------|------|------|------|------|------|-----|-----|-------------------------------------|
| bit                     | 15   | 14   | 13   | 12   | 11   | 10   | 9   | 8   |                                     |
| 00009D <sub>H</sub>     | DE15 | DE14 | DE13 | DE12 | DE11 | DE10 | DE9 | DE8 | DSRH                                |
|                         | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W | R/W | Initial value 00000000 <sub>B</sub> |
| bit                     | 7    | 6    | 5    | 4    | 3    | 2    | 1   | 0   |                                     |
| 00009C <sub>H</sub>     | DE7  | DE6  | DE5  | DE4  | DE3  | DE2  | DE1 | DE0 | DSRL                                |
|                         | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W | R/W | Initial value 00000000 <sub>B</sub> |
| R/W : Readable/Writable |      |      |      |      |      |      |     |     |                                     |

The functions of each bit in the  $\mu$ DMAC status register (DSR) is shown in the table below.

| DEx bit              | Function  |
|----------------------|---|
| 0<br>(Initial value) | No DMA transfer has ended.  |
| 1                    | If the DMA transfer ends, an interrupt request is output to the interrupt controller. |

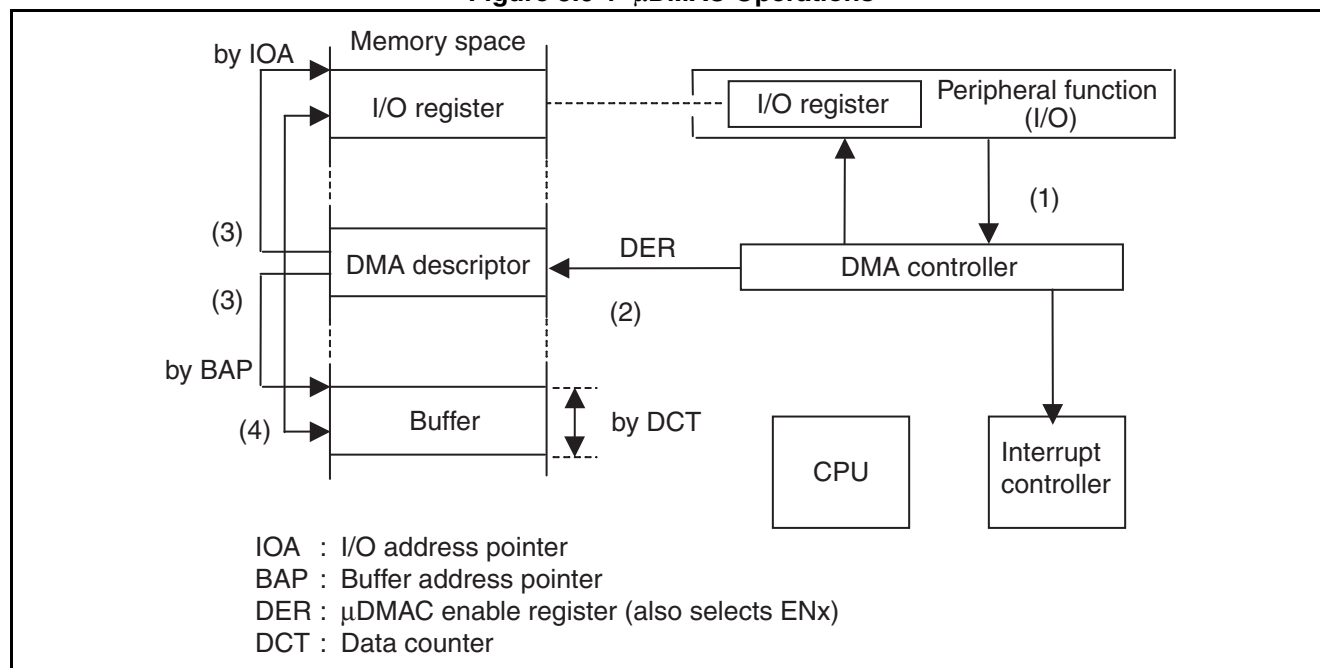
Note:

If "1" is written, DMA transfer does not end. An interrupt is output to the interrupt controller.

## ■ $\mu$ DMAC Operations

Figure 3.6-1 shows  $\mu$ DMAC operations. Data transfer using DMA is performed as described below.

### Figure 3.6-1 $\mu$ DMAC Operations



- (1) A peripheral resource (I/O) requests the DMA transfer.
- (2) The DMA controller reads a descriptor.
- (3) The transfer source, transfer destination, and transfer data count are read from the descriptor.
- (4) The DMA transfer between I/O and memory starts.
- (5) For no transfer end: The interrupt request of a resource is cleared.

For transfer end: After the DMA transfer ends, the  $\mu$ DMAC status register is set to the transfer end flag, thereby causing output of an interrupt request to the interrupt controller.

### 3.6.1 DMA Descriptor

The DMA descriptor is located in internal RAM within a range from "000100<sub>H</sub>" to "00017F<sub>H</sub>" consisting of 8 bytes × 16 channels.

#### ■ DMA Descriptor Configuration

The DMA descriptor consists of 8 bytes × 16 channels. Each DMA descriptor has the configuration shown in the Figure 3.6-2. Table 3.6-1 lists the relationship between channel number and DMA descriptor address.

**Figure 3.6-2 Configuration of DMA Descriptor**

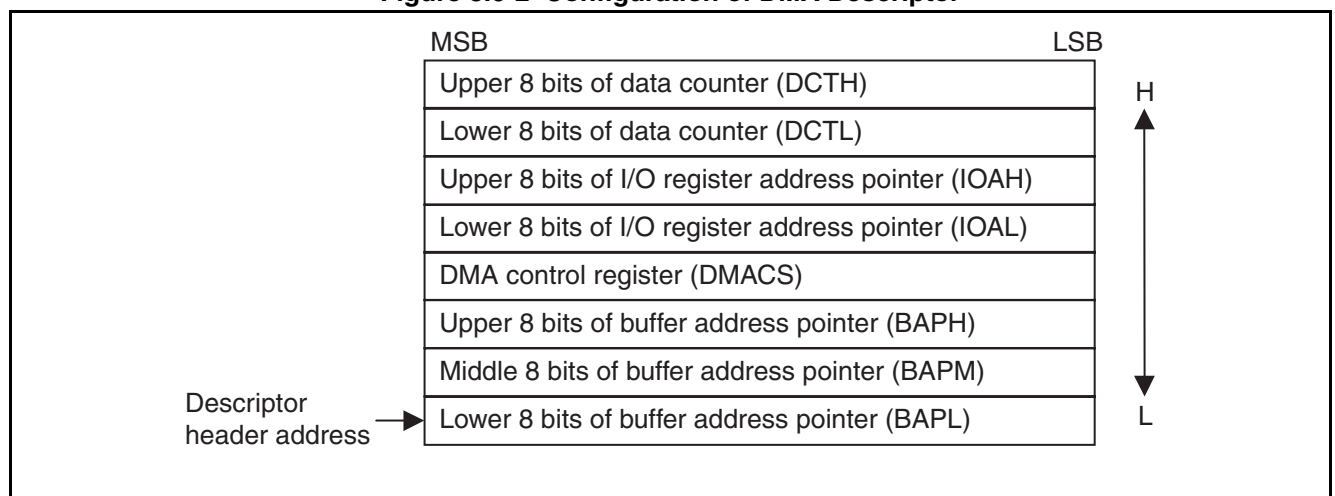


Table 3.6-1 Relationship between Channel Number and Descriptor Address

| $\mu$ DMAC enable register | Channel | Descriptor address  | Resource interrupt request              |
|----------------------------|---------|---------------------|---|
| EN0                        | 0       | 000100 <sub>H</sub> | INT0                                    |
| EN1                        | 1       | 000108 <sub>H</sub> | PWC0 (Only MB90485 series)              |
| EN2                        | 2       | 000110 <sub>H</sub> | PPG0/PPG1 counter borrow                |
| EN3                        | 3       | 000118 <sub>H</sub> | PPG2/PPG3 counter borrow                |
| EN4                        | 4       | 000120 <sub>H</sub> | PPG4/PPG5 counter borrow                |
| EN5                        | 5       | 000128 <sub>H</sub> | Input capture (ch.0) load               |
| EN6                        | 6       | 000130 <sub>H</sub> | Input capture (ch.1) load               |
| EN7                        | 7       | 000138 <sub>H</sub> | UART receive completed                  |
| EN8                        | 8       | 000140 <sub>H</sub> | Output compare (ch.0) match             |
| EN9                        | 9       | 000148 <sub>H</sub> | Output compare (ch.1) match             |
| EN10                       | 10      | 000150 <sub>H</sub> | Output compare (ch.2) match             |
| EN11                       | 11      | 000158 <sub>H</sub> | UART transmit completed                 |
| EN12                       | 12      | 000160 <sub>H</sub> | 16-bit FRT/16-bit reload timer overflow |
| EN13                       | 13      | 000168 <sub>H</sub> | SIO1                                    |
| EN14                       | 14      | 000170 <sub>H</sub> | SIO2                                    |
| EN15                       | 15      | 000178 <sub>H</sub> | A/D converter                           |

## 3.6.2 Individual Registers of DMA Descriptor

Each DMA descriptor consists of the following registers:

- Data counter (DCT)
- I/O register address pointer (IOA)
- DMA control status register (DMACS)
- Buffer address pointer (BAP)

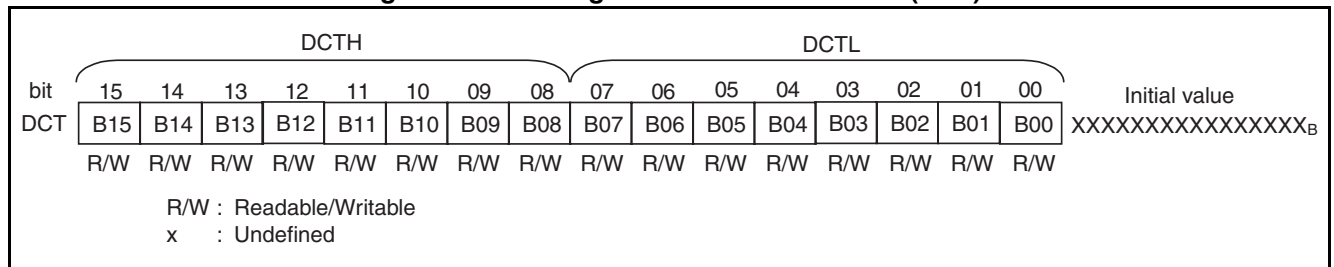
The registers must be initialized because their initial values become undefined when they are reset.

### ■ Data Counter (DCT)

The data counter (DCT) is a register with a length of 16 bits and corresponds to the transfer data count. After each item of data is transferred, the counter decrements by one. If this counter reaches "0", DMA ends. Figure 3.6-3 shows the configuration of the data counter (DCT).

If the data counter (DCT) is set to "0", the maximum data transfer count (i.e., 65536) is defined.

Figure 3.6-3 Configuration of Data Counter (DCT)

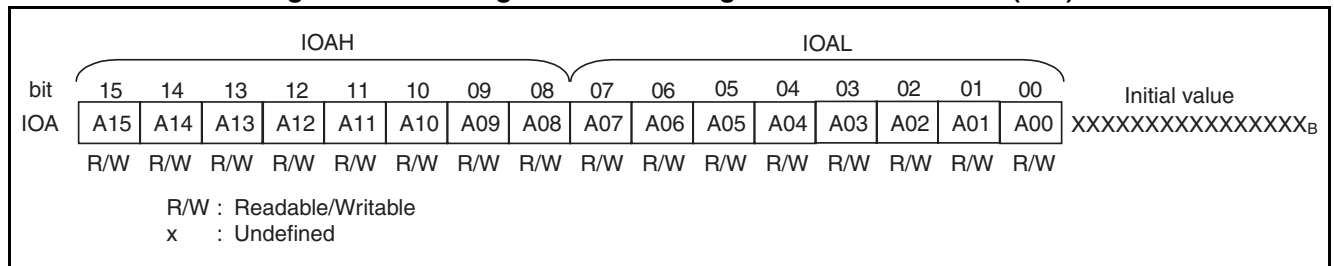


### ■ I/O Register Address Pointer (IOA)

The I/O register address pointer (IOA) is a register with a length of 16 bits, and it indicates the lower address (A15 to A0) of the I/O register providing a buffer for data transfers. All of the upper addresses (A23 to A16) are set to "0". Any I/O in a range of 000000<sub>H</sub> to 00FFFF<sub>H</sub> can be specified with the address.

Figure 3.6-4 shows the configuration of the I/O register address pointer (IOA).

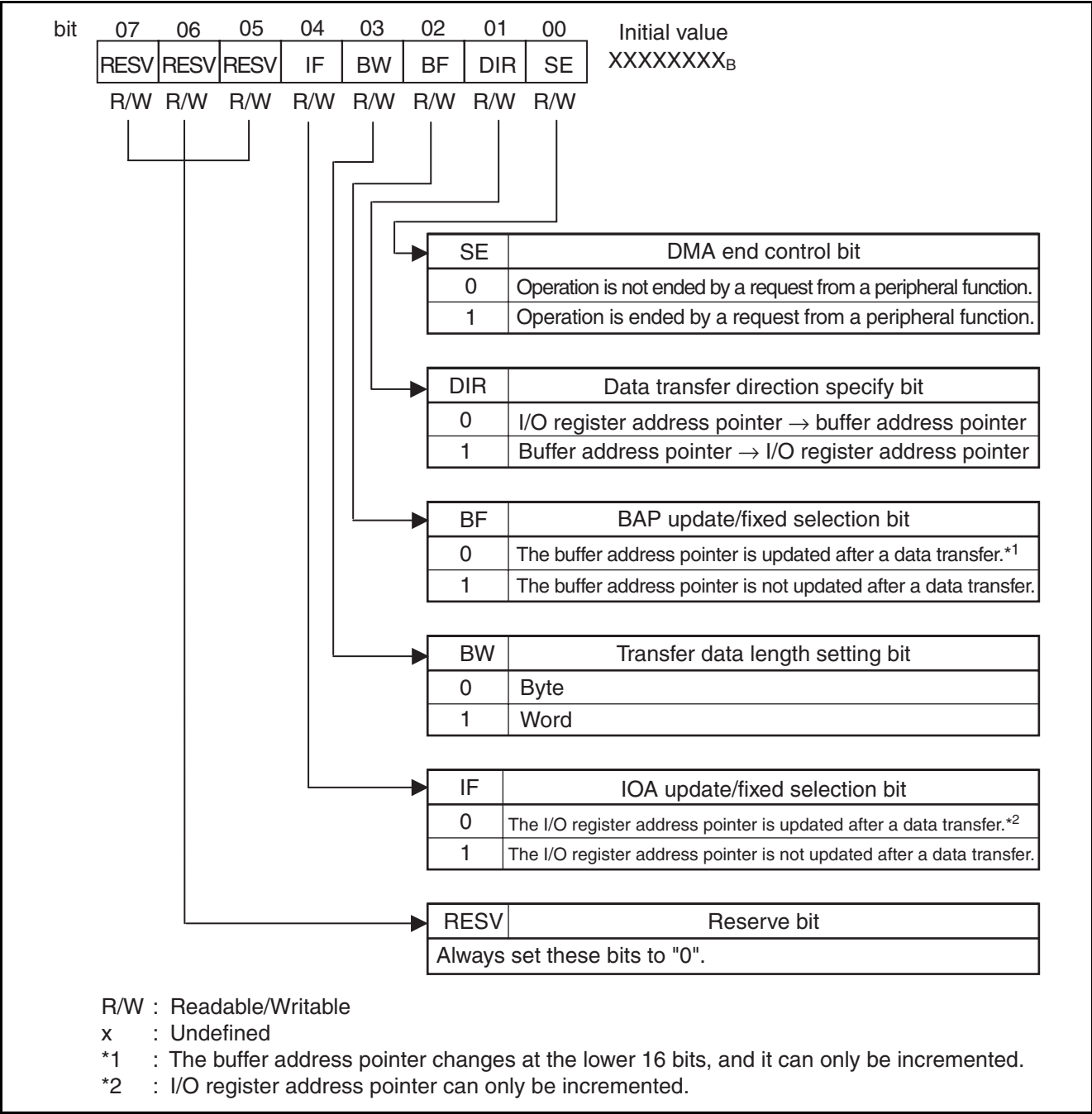
Figure 3.6-4 Configuration of I/O Register Address Pointer (IOA)



■ DMA Control Status Register (DMACS)

The DMA control status register (DMACS) has a length of 8 bits that indicate the update or fixed state, transfer data format (byte/word), and transfer directions for the buffer address pointer (BAP) and I/O register address pointer (IOA). Figure 3.6-5 shows the configuration of the DMA control status register (DMACS).

Figure 3.6-5 Configuration of DMA Control Status Register (DMACS)



## ■ Buffer Address Pointer (BAP)

The buffer address pointer (BAP) has a length of 24 bits, containing the address used in the next DMA transfer. BAP is independent from each DMA channel, so each DMA channel can transfer data between any of 16M bytes addresses and I/O. If the BF bit (BAP update/fixed selection bit) in the DMA control status register (DMACS) is set to "update provided", BAP changes at the lower 16 bits (BAPM and BAPL), but the upper 8 bits (BAPH) do not change. Figure 3.6-6 shows the configuration of BAP.

**Figure 3.6-6 Configuration of Buffer Address Pointer (BAP)**

|                         |       |    |    |   |       |   |               |                       |
|-------------------------|-------|----|----|---|-------|---|---------------|-----------------------|
|                         | 23    | 16 | 15 | 8 | 7     | 0 | Initial value |                       |
| BAP                     | BAPH  |    |    |   | BAPM  |   | BAPL          | XXXXXXXX <sub>H</sub> |
|                         | (R/W) |    |    |   | (R/W) |   | (R/W)         |                       |
| R/W : Readable/Writable |       |    |    |   |       |   |               |                       |
| X : Undefined           |       |    |    |   |       |   |               |                       |

### Notes:

- The I/O register address pointer (IOA) can be used to specify an area ranging from 000000<sub>H</sub> to 00FFFF<sub>H</sub>.
- The buffer address pointer (BAP) can be used to specify an area ranging from 000000<sub>H</sub> to FFFFFFF<sub>H</sub>.



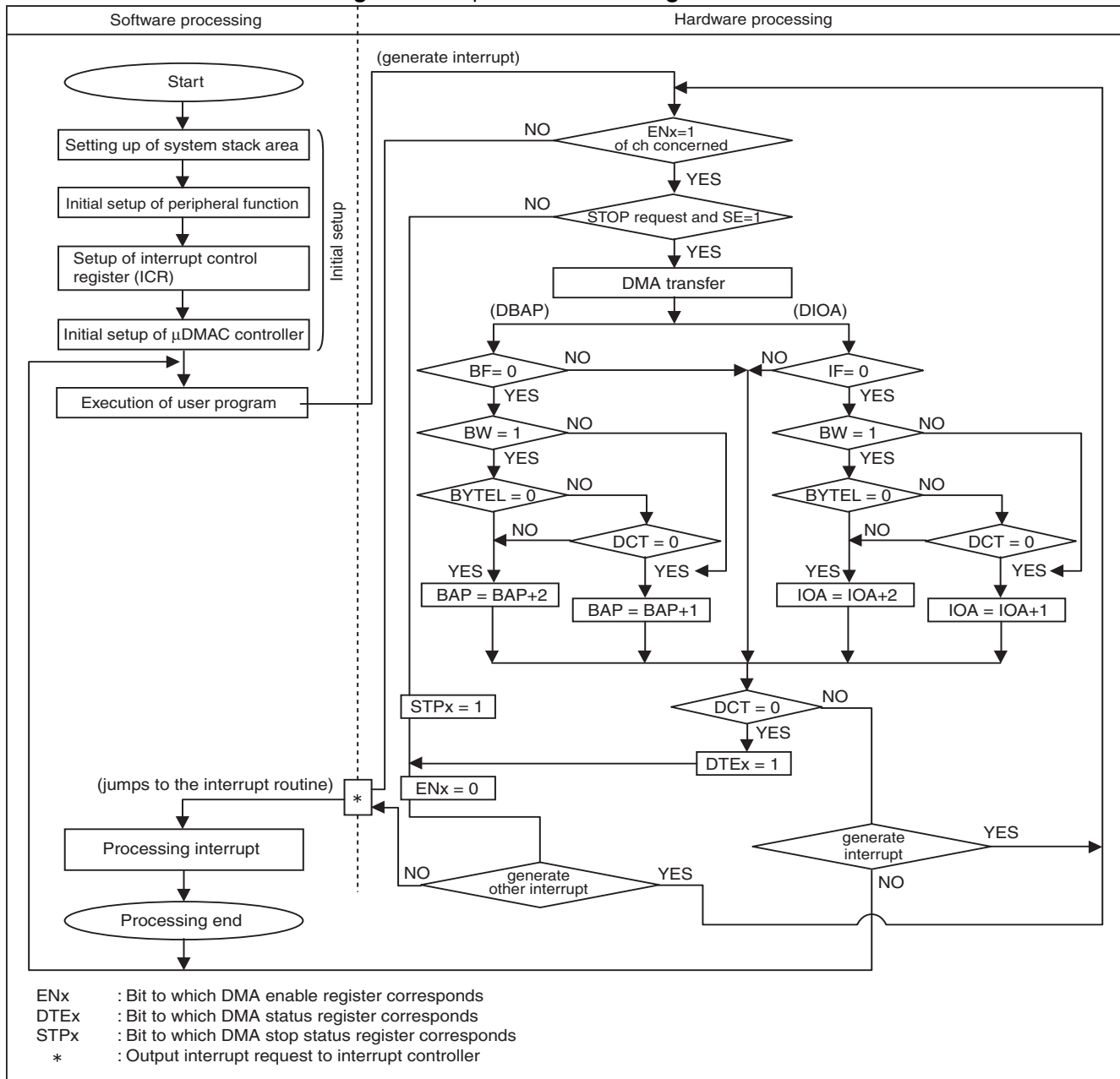
### 3.6.3 $\mu$ DMAC Processing Procedure

If an interrupt request is generated by a peripheral resource (I/O) and the corresponding  $\mu$ DMAC enable register (DER) has a setting of DMA start, then a DMA transfer is performed. If a data transfer ends at the specified count, an interrupt request is output to the interrupt controller.

#### ■ $\mu$ DMAC Processing Procedure

Figure 3.6-7 shows a simple  $\mu$ DMAC processing procedure.

Figure 3.6-7  $\mu$ DMAC Processing Procedure



### 3.6.4 $\mu$ DMAC Processing Time

Time consumed in  $\mu$ DMAC processing varies with the following factors:

- Settings of DMA control status register (DMACS)
- Address (area) indicated by the I/O register address pointer (IOA)
- Address (area) indicated by the buffer address pointer (BAP)
- External data bus width for external access
- Data length of transfer data

When the  $\mu$ DMAC data transfer ends, a hardware interrupt starts, and then the interrupt processing time is added.

#### ■ $\mu$ DMAC Processing Time (Time Per One-time Transfer)

##### ○ If data transfer continues

The  $\mu$ DMAC processing time during a continuation of a data transfer depends on the setting of DMA control status register (DMACS), as shown in Table 3.6-2.

**Table 3.6-2  $\mu$ DMAC Execution Time**

| Setting of IOA update/fixed selection bit (IF)            |        | Fixed | Update |
|---|--------|-------|--------|
| BAP address update/fixed<br>Setting of selection bit (BF) | Fixed  | 17    | 19     |
|   | Update | 19    | 21     |

Note:

In units of machine cycles. One machine cycle corresponds to one clock interval of the machine clock ( $\phi$ ).

Correction is required depending on the condition at  $\mu$ DMAC execution, as shown in Table 3.6-3.

**Table 3.6-3 Correction Values of Data Transfer for  $\mu$ DMAC 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    |

Note:

B indicates a byte data transfer, 8 indicates a word transfer with an external bus width of 8 bits, even indicates word transfer of an even-numbered address, and odd indicates a word transfer of an odd-numbered address.

## ○ Transfer performance

### Minimum transfer speed

1.7  $\mu$ s/10 MHz (machine clock)

1.07  $\mu$ s/16 MHz (machine clock)

- Built-in I/O  $\rightarrow$  built-in RAM; or built-in RAM  $\rightarrow$  built-in I/O without address increment
- Even-numbered address  $\rightarrow$  even-numbered address or 8-bit access

### Maximum transfer speed

2.8  $\mu$ s/10 MHz (machine clock)

1.75  $\mu$ s/16 MHz (machine clock)

Table 3.6-4 indicates the correction values for interrupt handling time.

**Table 3.6-4 Correction Values (Z) for Interrupt Handling Time**

| Address indicated by stack pointer | Correction value (Z) |
|------------------------------------|----------------------|
| External 8 bits                    | +4                   |
| External even-numbered address     | +1                   |
| External odd-numbered address      | +4                   |
| Internal even-numbered address     | 0                    |
| Internal odd-numbered address      | +2                   |

## ○ If a transfer is ended with an end request from a peripheral function (I/O)

If the  $\mu$ DMAC data transfer ends partway ( $DEx = 1$ ) because of an end request by a peripheral function (I/O), the data transfer fails and a hardware interrupt starts. The  $\mu$ DMAC processing time in this case is calculated with the following formula. Z in the formula indicates a correction value for interrupt processing time (see Table 3.6-4).

The  $\mu$ DMAC processing time if a transfer ends partway is:

$36 + 6 \times Z$  machine cycle

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

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

---

**Extended Intelligent I/O Services (EI<sup>2</sup>OS) are a function that automatically transfers data between the peripheral function (I/O) and RAM. After completion of the data transfer, hardware interruptions will occur.**

---

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

Extended Intelligent I/O Service (EI<sup>2</sup>OS) is a kind of the hardware interrupt. Extended Intelligent I/O Service is a function that transfers data between the I/O area and RAM. Customer can have data transfers performed just by creating in advance a completion program and a setting program for the EI<sup>2</sup>OS activation.

#### ● Advantages of EI<sup>2</sup>OS

The advantages over interruption processing routine-based data transfers are as follows:

- Since the creation of transfer program is not required, the program size can be reduced.
- Because the transfer is activated by the interrupt source of peripheral function (resource), the data transfer source needs not to be set for polling.
- Transferring address increment can be set.
- Increment and no update of I/O register address can be set.

#### ● Interrupt by EI<sup>2</sup>OS termination

Upon completion of the EI<sup>2</sup>OS data transfer(s), the completion condition branches to the interrupt routine.

The factor for an EI<sup>2</sup>OS completion can be confirmed by checking the EI<sup>2</sup>OS status bit (ICR:S1,S0) by the interruption processing program.

#### **Reference:**

Interrupt number and interrupt vector are fixed by each peripheral function. For details, see "3.2 Interrupt Factor and Interrupt Vector".

#### ● Interrupt control register (ICR)

EI<sup>2</sup>OS activation, EI<sup>2</sup>OS channel can be set. And EI<sup>2</sup>OS status at EI<sup>2</sup>OS end can be confirmed.

#### ● EI<sup>2</sup>OS descriptor (ISD)

This is an 8-byte × 16-channel register that is deployed in the "000100<sub>H</sub>" to "00017F<sub>H</sub>" area of built-in RAM and used to specify the transfer mode, address of peripheral function (resource), number of bytes to be transferred and destination address. A channel number is allocated to each of these channels by the interrupt control register (ICR: ICS3 to ICS0).

---

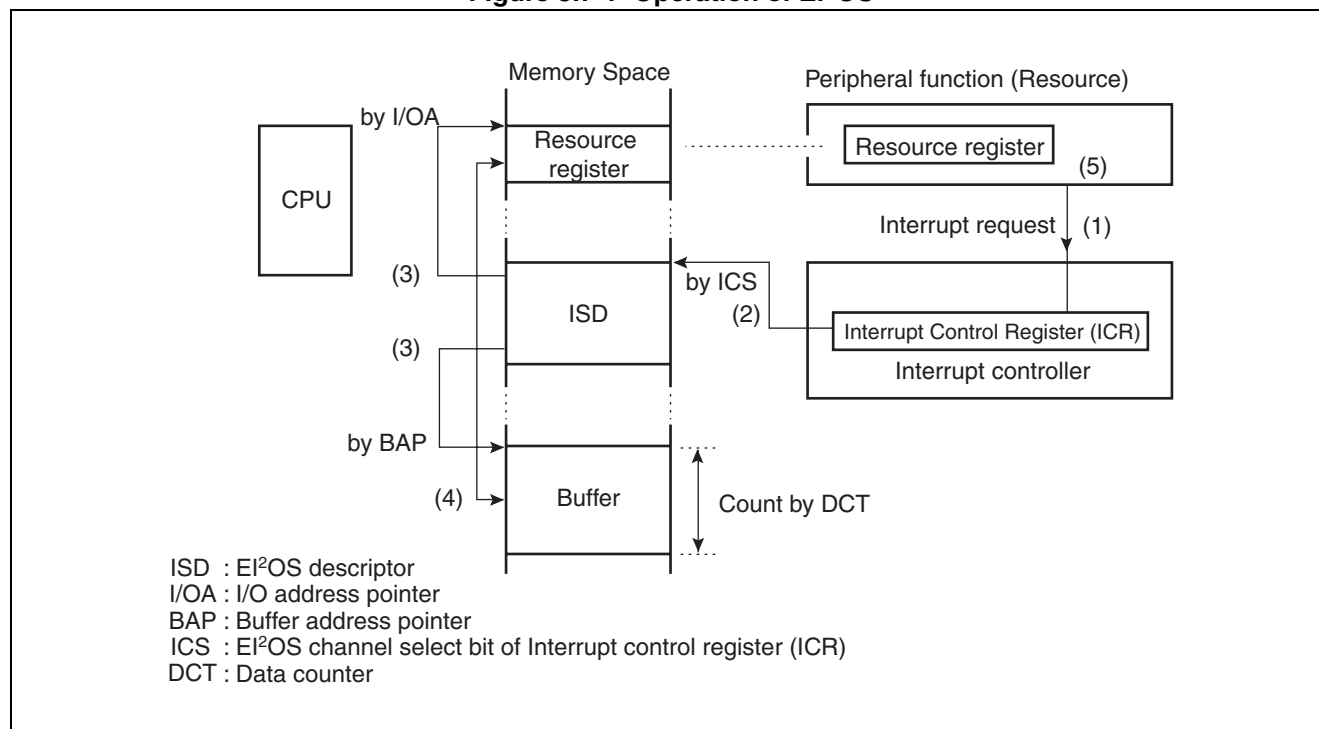
#### Note:

The CPU stops while the EI<sup>2</sup>OS is in operation.

---

## ■ Operation of EI<sup>2</sup>OS

Figure 3.7-1 Operation of EI<sup>2</sup>OS



- (1) The EI<sup>2</sup>OS will be activated from the peripheral function (resource).
- (2) The interruption controller sets the EI<sup>2</sup>OS descriptor according to the interrupt control register (ICR) setting.
- (3) The address pointers for the transfer origination or destination are read from the EI<sup>2</sup>OS descriptor.
- (4) The data is transferred based on the address pointers for the transfer origination and destination.
- (5) The interrupt request flag bit of the peripheral function is cleared after the data transferring completed.

### 3.7.1 EI<sup>2</sup>OS descriptor (ISD)

EI<sup>2</sup>OS descriptor (ISD) which is in 000100<sub>H</sub> to 00017F<sub>H</sub> of built-in RAM is consists of 8-byte × 16 channels.

#### ■ Configuration of EI<sup>2</sup>OS Descriptor (ISD)

ISD comprises 8-byte × 16 channels.

**Figure 3.7-2 Configuration of EI<sup>2</sup>OS Descriptor (ISD)**

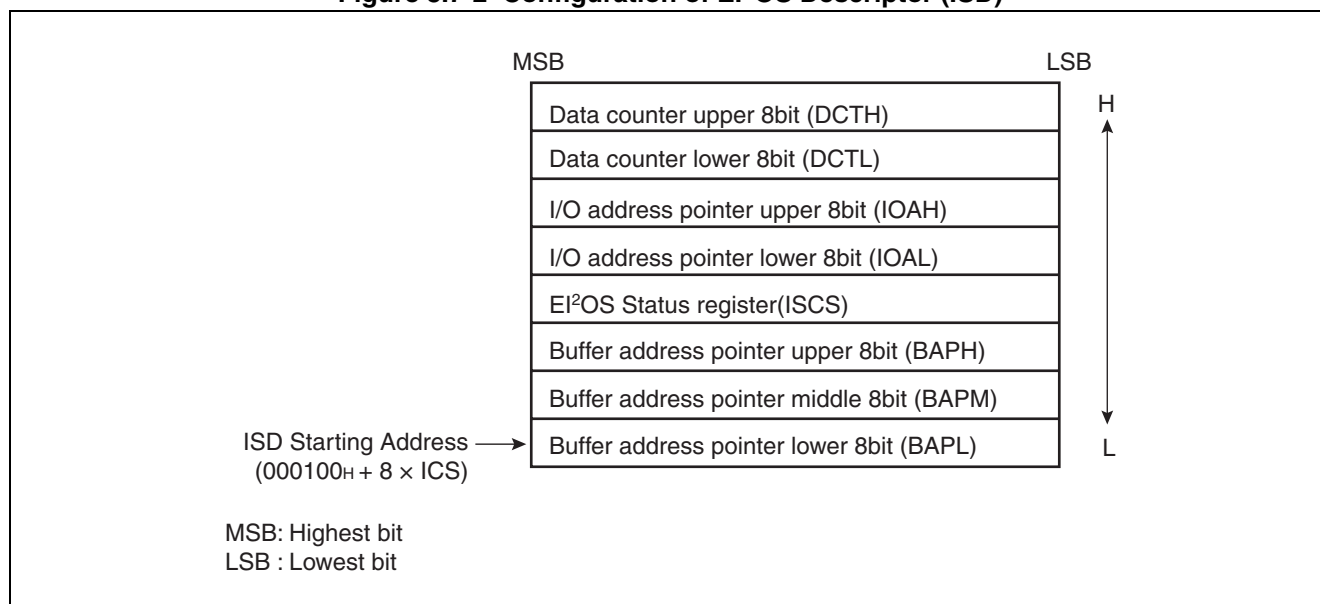


Table 3.7-1 Relation Between Channel Number and Descriptor Address

| 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> |

\*:The address of ISD indicates the first address of 8-byte.

### 3.7.2 Each Register of EI<sup>2</sup>OS Descriptor (ISD)

Extended intelligent I/O service (EI<sup>2</sup>OS) descriptor (ISD) is configured by following 4 types of 8-byte registers.

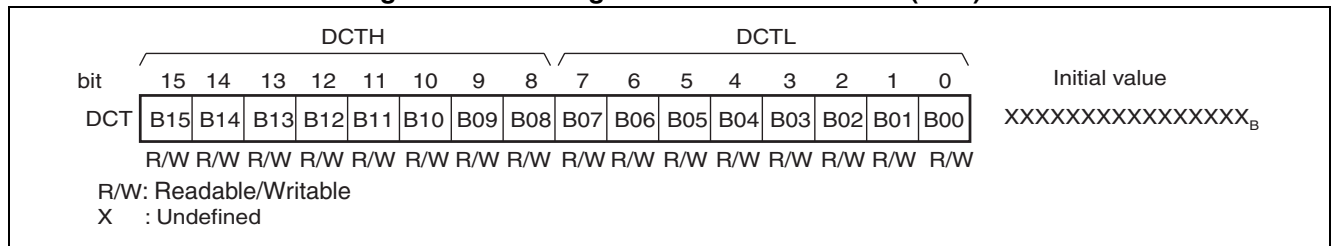
- Data counter (DCT: 2 bytes)
- I/O register address pointer (IOA: 2 bytes)
- EI<sup>2</sup>OS status register (ISCS: 1 byte)
- Buffer address pointer (BAP: 3 bytes)

The register reset values will become indeterminate.

#### ■ Data Counter (DCT)

Data counter (DCT) is 16-bit register. Specifies the number of bytes of transfer data. When a data transfer is made, 1 is decremented. When the data counter (DCT) value becomes “0000<sub>H</sub>”, the EI<sup>2</sup>OS is completed.

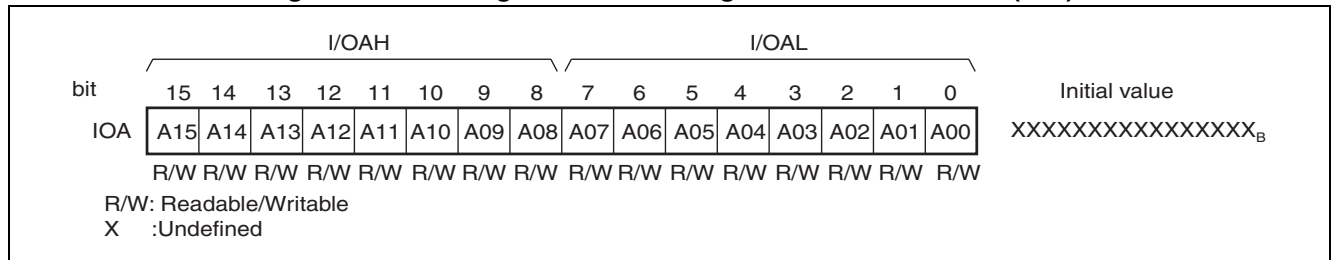
Figure 3.7-3 Configuration of Data Counter (DCT)



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

I/O register address pointer (IOA) is 16-bit register. Specifies the lower-order address (A15 to A0) for the data transfer. The higher-order address (A23 to A16) is set to “00<sub>H</sub>”. The area from 000000<sub>H</sub> to 00FFFF<sub>H</sub> can be used when specifying an address.

Figure 3.7-4 Configuration of I/O Register Address Pointer (IOA)

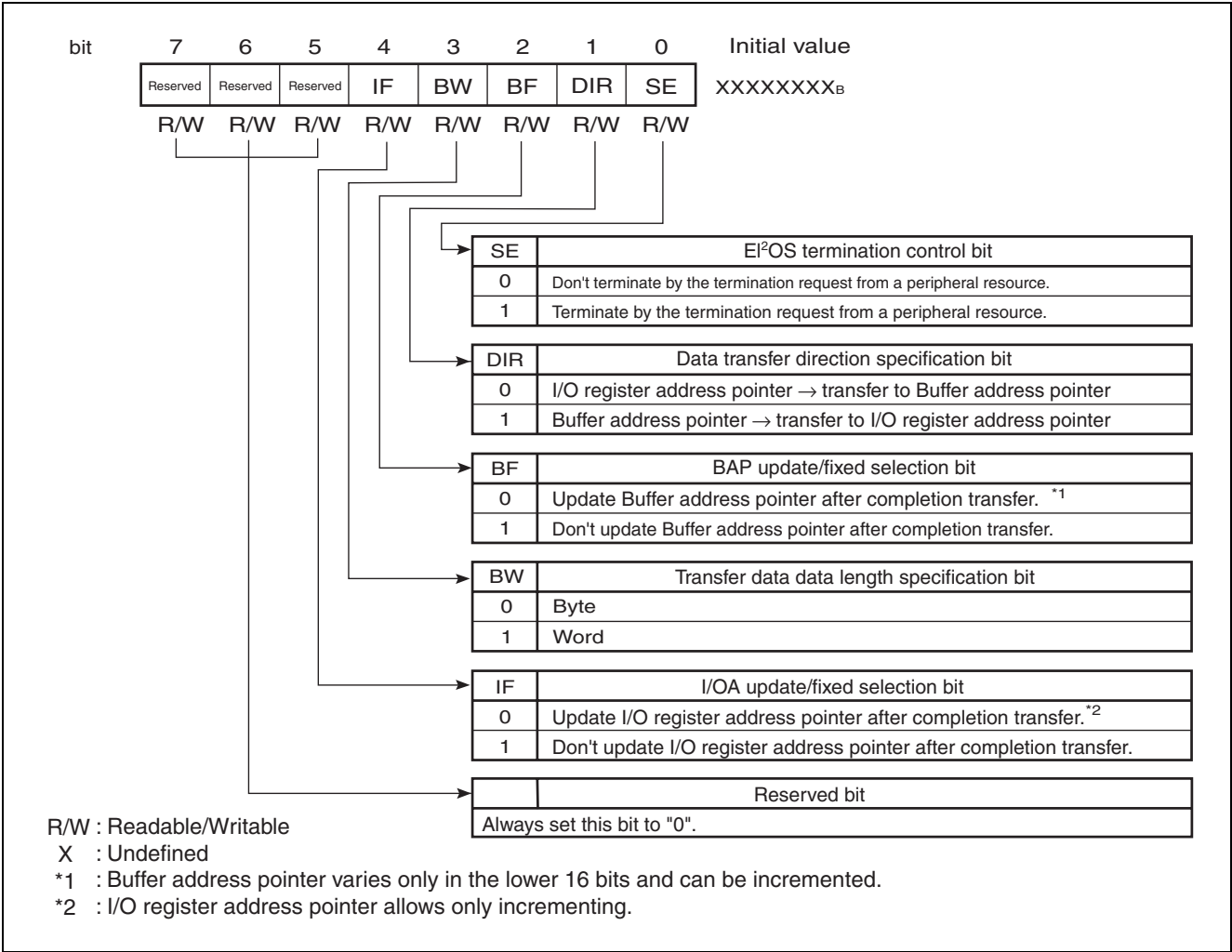




■ EI<sup>2</sup>OS Status Register (ISCS)

EI<sup>2</sup>OS status register (ISCS) is 8-bit register. The methods to renew the buffer address pointer and I/O address pointer, the transfer data type (byte/word) and the transfer direction can be specified.

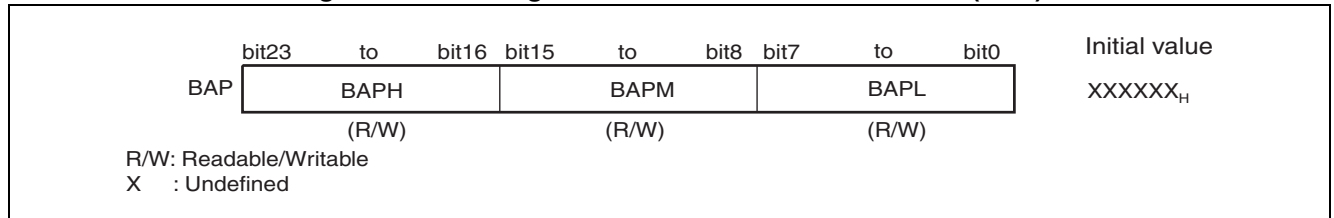
Figure 3.7-5 Configuration of EI<sup>2</sup>OS Status Register (ISCS)



## ■ Buffer Address Pointer (BAP)

Buffer address pointer (BAP) is 24-bit register. EI<sup>2</sup>OS operation set the memory address of the data transferring source. Buffer address pointer exists in each channel. So, the data can be transferring between the 16M-byte memory address and the peripheral function (resource) address. If the BAP updated/fixed setting bit (BF) is set to "0", the lower 16-bit (BAPM, BAPL) is incremented, and the higher 8-bit (BAPH) is not incremented.

**Figure 3.7-6 Configuration of Buffer Address Pointer (BAP)**



### References:

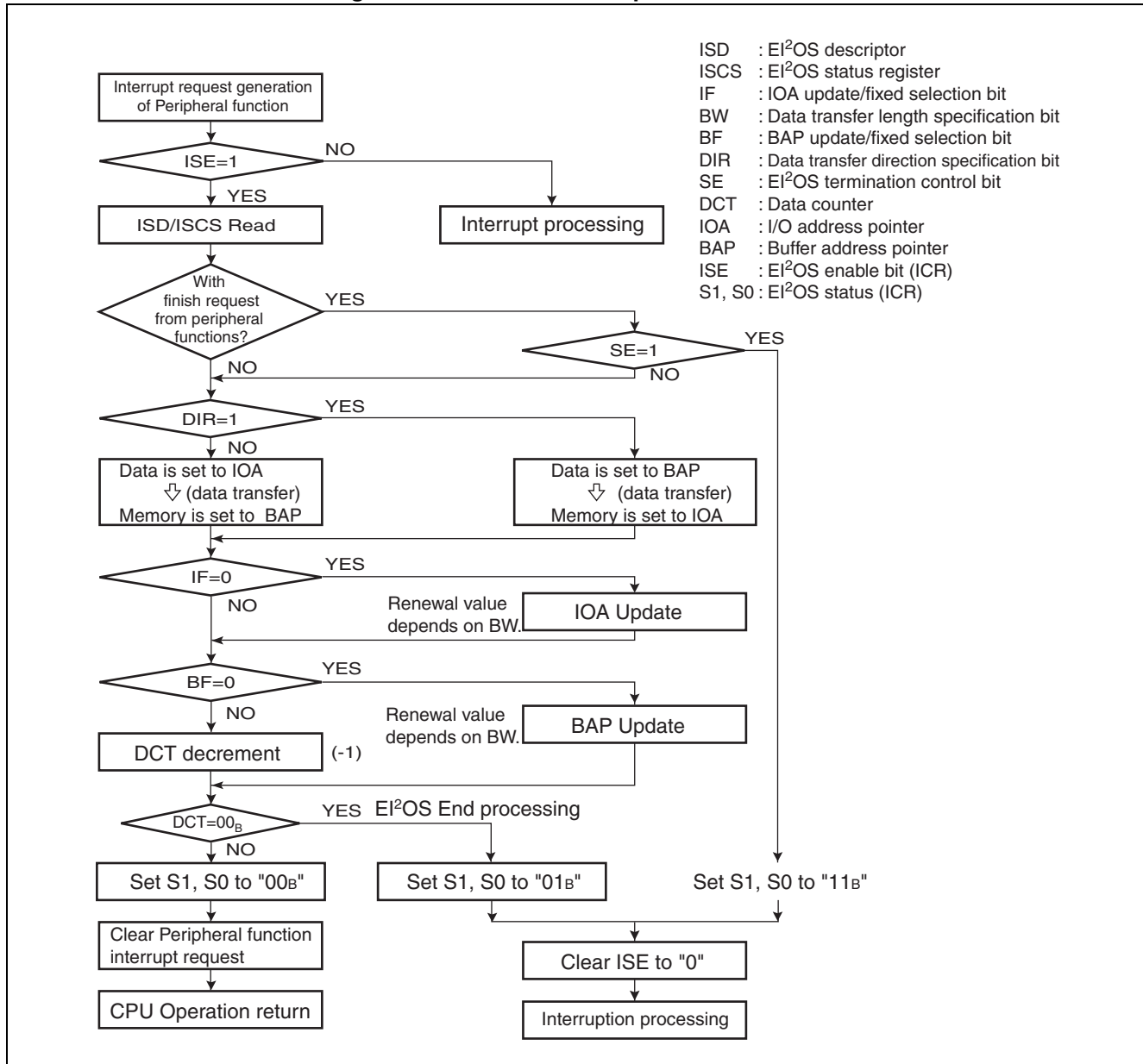
- The maximum transfer count that can be set by the data counter (DCT) is 65,536 (64K bytes).
- The area that can be set by the I/O address pointer (IOA) is 000000<sub>H</sub> to 00FFFF<sub>H</sub>.
- The area that can be set by the buffer address pointer (BAP) is 000000<sub>H</sub> to FFFFFF<sub>H</sub>.

### 3.7.3 Operation of EI<sup>2</sup>OS

CPU transfers the data by EI<sup>2</sup>OS, when the interrupt request is output from the peripheral function (resource) and the interrupt control register has been set to the start of EI<sup>2</sup>OS. When the EI<sup>2</sup>OS operation ends, hardware interrupt is done.

#### ■ Flowchart of Operation of EI<sup>2</sup>OS

Figure 3.7-7 Flowchart of Operation of EI<sup>2</sup>OS

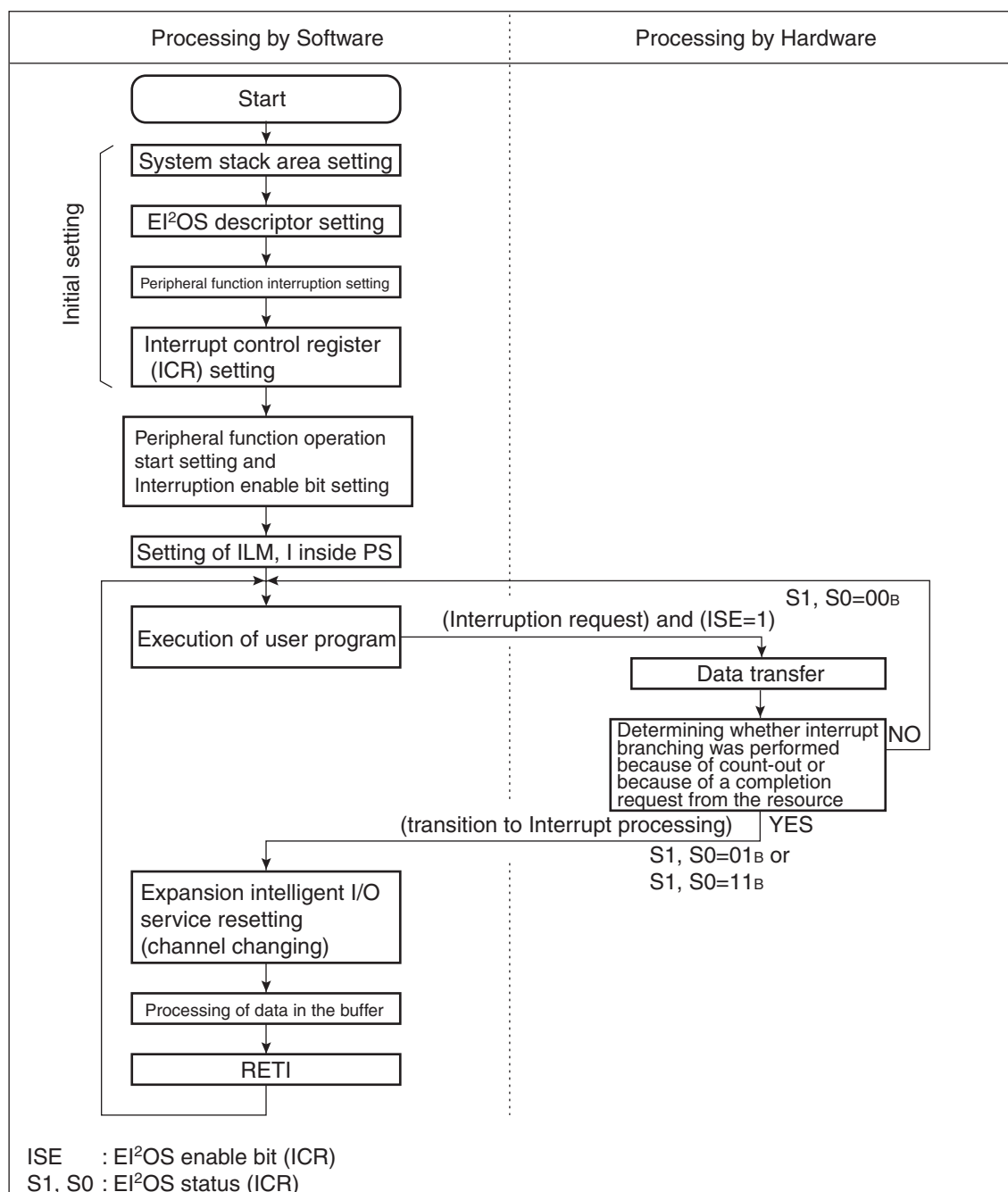


### 3.7.4 Procedure for Use of EI<sup>2</sup>OS

The setting of extended intelligent I/O service (EI<sup>2</sup>OS) is set by the system stack area, the extended intelligent I/O service (EI<sup>2</sup>OS) descriptor, the peripheral function (resource), and the interrupt control register (ICR).

#### ■ Procedure for Use of EI<sup>2</sup>OS

Figure 3.7-8 Procedure for Use of EI<sup>2</sup>OS



### 3.7.5 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) depends on setting of the extended intelligent I/O service descriptor (ISD).

- EI<sup>2</sup>OS status register (ISCS) setting
- Address (setting) pointed to by the I/O register address pointer (IOA)
- Address (setting) pointed to by the buffer address pointer (BAP)
- External data bus width 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 3.7-2 based on the EI<sup>2</sup>OS status register (ISCS) setting.

Table 3.7-2 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     |
|   | Updated | 34  | 36     | 35  | 37     |

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

As shown in Table 3.7-3, interpolation is necessary for the EI<sup>2</sup>OS processing time when the data transfer is continued depending on the EI<sup>2</sup>OS execution condition.

Table 3.7-3 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 width 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:

$$\text{EI}^2\text{OS processing time when counting terminates} = \text{EI}^2\text{OS processing time when data is transferred} + \underbrace{(21 + 6 \times Z)}_{\substack{\uparrow \\ \text{Interrupt handling time}}} \text{ machine cycle}$$

The interrupt handling time is different for the address stored by the stack pointer.

**Table 3.7-4 Interpolation Value (Z) for the Interrupt Handling Time**

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

- For termination by a termination request from the peripheral function (resource)

When data transfer by EI<sup>2</sup>OS is terminated before completion due to a termination request from the peripheral function (resource) (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 3.7-4).

$$\text{EI}^2\text{OS processing time for termination before completion} = 36 + 6 \times Z \text{ machine cycle}$$

**Reference:**

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

## 3.8 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.

Because exception processing occurs as the result of an unexpected operation, it should be used only to activate recovery software required for debugging or an emergency.

---

### ■ Exception Processing Due to Execution of Undefined Instruction

- 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 suppressed.
- The S flag of the condition code register (CCR) is set to "1", and the system stack is enabled.

The program counter (PC) value saved to the stack is an 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 the saved PC value.

- Return from exception processing

When returning by the RETI instruction from exception processing, exception processing is performed again because the PC is pointing to the undefined instruction. Take measures such as resetting software.

### 3.9 Stack Operation of Interrupt Processing

If an interrupt is accepted, contents of the dedicated registers are automatically saved in the system stack before branching to interrupt processing. Return from the stack is also automatically performed when interrupt processing is completed.

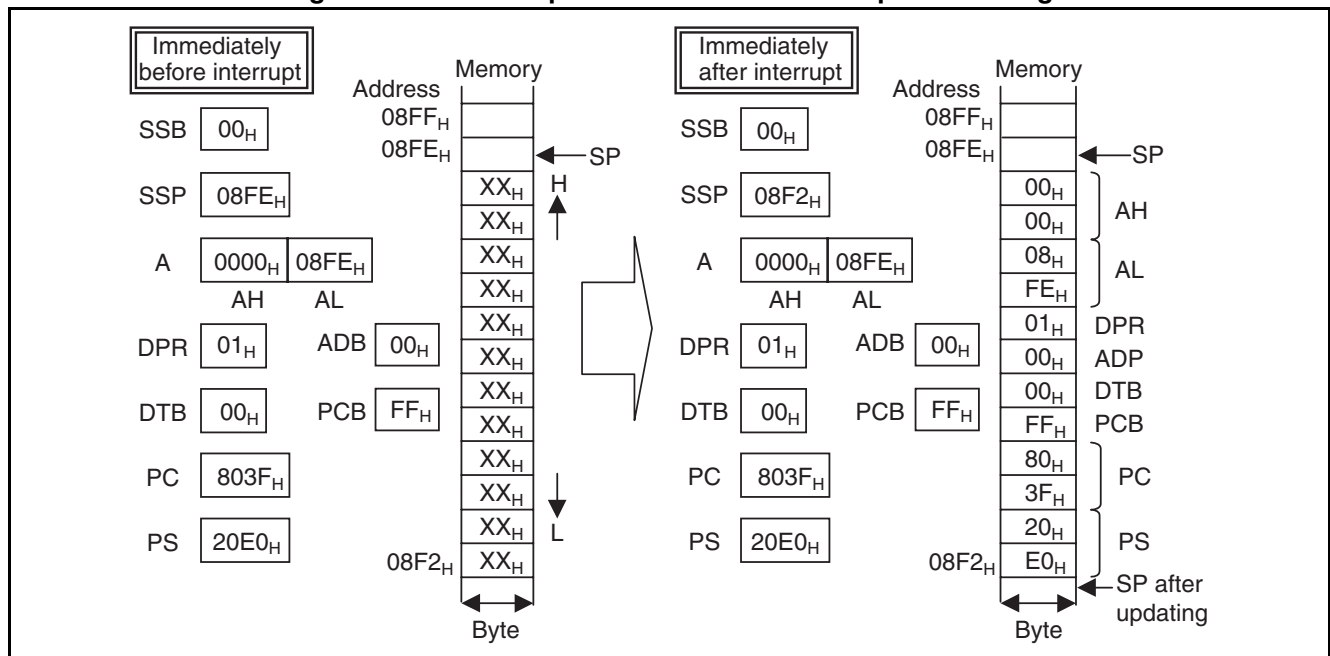
#### ■ Stack Operation When Interrupt Processing Starts

With an accepted interrupt, CPU automatically saves the contents of the current dedicated registers in the system stack in the following sequence:

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

Figure 3.9-1 shows the stack operation when interrupt processing starts.

**Figure 3.9-1 Stack Operation at Start of Interrupt Processing**



#### ■ Stack Operation During Return from Interrupt Processing

At the end of interrupt processing, if the interrupt return instruction (RSTI) is executed, PS, PC, PCB, DTB, ADB, DPR, and A values are returned from the stack in the reverse order of the start of the interrupt processing. The dedicated registers are then restored to their previous state (i.e., immediately before the interrupt started).



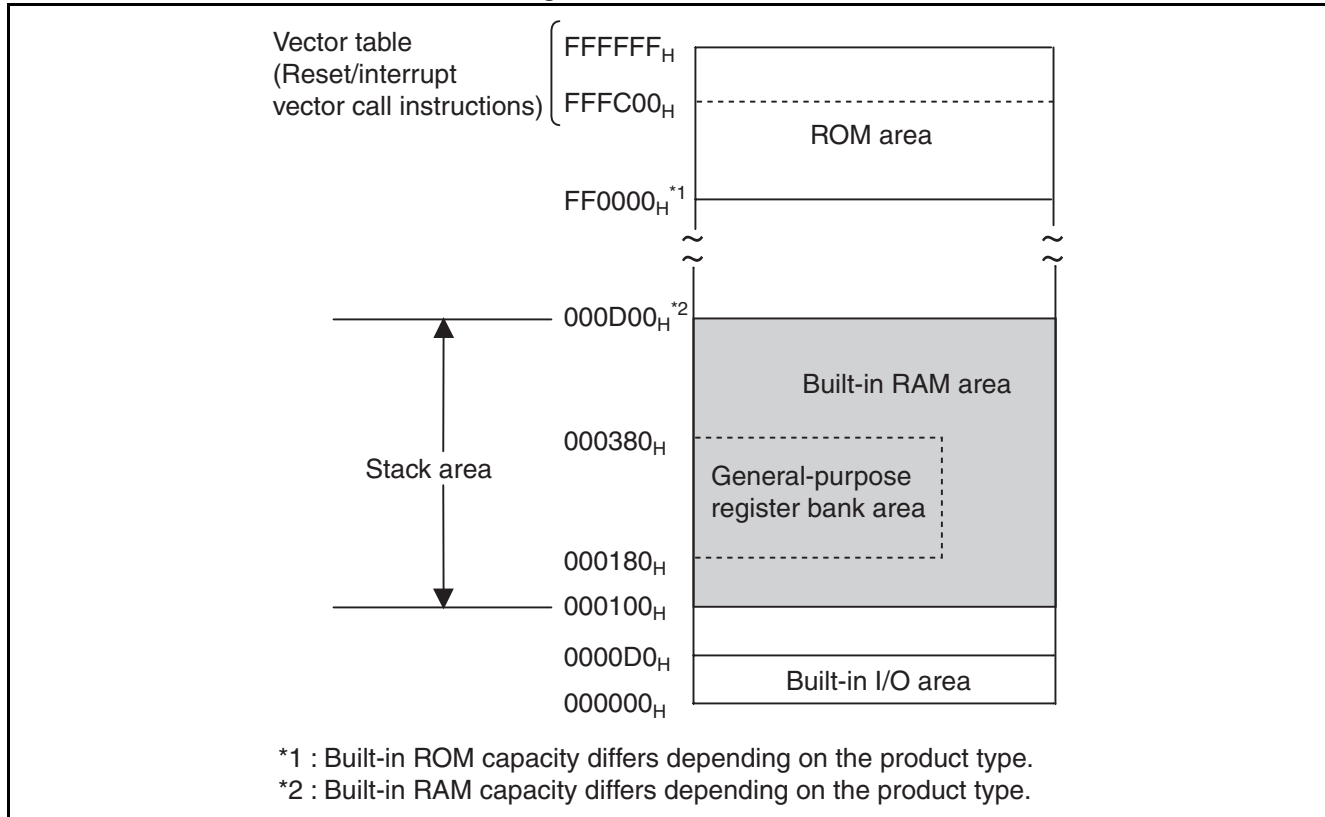
## ■ Stack Area

### ○ Assigning the stack area

The stack area is used for storage and return of the program counter (PC) required for executing interrupt processing, subroutine call instruction (CALL) and vector call instruction (CALLV), as well as temporary save and return of registers executed by using the PUSHW and POPW instructions. The stack area is assigned in RAM in addition to the data area.

Figure 3.9-2 shows the stack area.

**Figure 3.9-2 Stack Area**



**Notes:**

- If specifying addresses of the stack pointers (SSP and USP), specify them with even numbers.
- Assign the system stack area, user stack area, and data area while avoiding duplication with one another.

### ○ System stack and user stack

Interrupt processing uses the system stack area. Even if the user stack area is being used when an interrupt occurs, it is forcibly switched to the system stack. Thus, the system that primarily uses the user stack area must also correctly prepare the system stack area. Use only the system stack unless the stack space must be separated.

## 3.10 Sample Program of Interrupt Processing

A sample program for interrupt processing is shown below.

### ■ Sample Program for Interrupt Processing

#### ○ Processing specification

An example of an interrupt program using external interrupt 0 (INT0) is shown.  
Sample coding from the program is shown below.

#### [Coding example]

```

DDR1    EQU    000011H           ;Port 1 direction register
ENIR    EQU    028H             ;Interrupt/DTP enable register
EIRR    EQU    029H             ;Interrupt/DTP flag
ELVR    EQU    02AH             ;Request level set register
ICR00    EQU    0B0H            ;Interrupt control register
STACK   SSEG
        RW     100              ;Stack
STACK_T  RW     1
STACK   ENDS
;-----Main program-----
CODE    CSEG
START:
        MOV     RP,#0            ;General-purpose register use of
                                ;header bank
        MOV     ILM,#07H         ;PS:ILM is set to level 7
        MOV     A,#!STACK_T      ;Setting of system stack
        MOV     SSB,A
        MOVW    A,#STACK_T       ;Setting of stack pointer, where
        MOVW    SP,A            ;it is set to SSP since S-flag = 1
        MOV     DDR1,#00000000B  ;P10/INT0 pin is set to input
        OR      CCR,#40H         ;I-flag in PS:CCR is set for interrupt enable
        MOV     I:ICR00,#00H     ;Interrupt set to level 0 (highest)
        MOV     I:ELVR,#00000001B ;INT0 is set to H-level request
        MOV     I:EIRR,#00H      ;INT0 interrupt factor cleared
        MOV     I:ENIR,#01H      ;INT0 input enable
        :
LOOP:    NOP                     ;Dummy loop
        NOP
        NOP
        NOP
        BRA     LOOP            ;Unconditional jump
;-----Interrupt program-----
ED_INT1:
        MOV     I:EIRR,#00H      ;Prohibition of acceptance of new
                                ;INT0
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        RETI                     ;Return from interrupt
CODE    ENDS
;-----Vector setting-----
VECT    CSEG
        ABS=OFFH
        ORG     OFFD0H           ;Vector is set to interrupt #11(0BH)
        DSL     ED_INT1
        ORG     OFFDCH           ;Reset vector setting
        DSL     START
        DB      00H              ;Set to single chip mode
VECT    ENDS
END     START

```

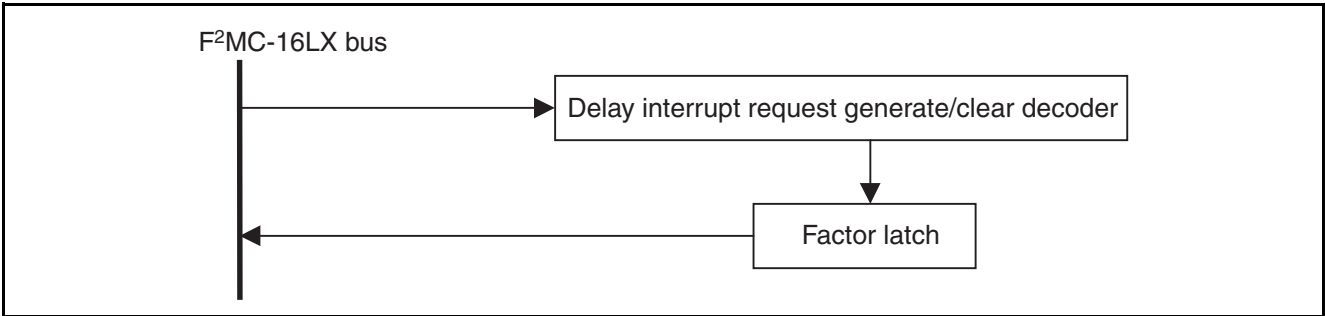
### 3.11 Delay Interrupt Generation Module

The delay interrupt generation module is a module that generates an interrupt for task switching. Using this module allows generation and clearing of an interrupt request to the F<sup>2</sup>MC-16LX CPU by software.

■ Block Diagram of Delay Interrupt Generation Module

Figure 3.11-1 is a block diagram of the delay interrupt generation module.

Figure 3.11-1 Block Diagram of Delay Interrupt Event Module



■ List of Registers in Delay Interrupt Generation Module

The delay interrupt generation module, delay interrupt factor originate/clear register (DIRR: delayed interrupt request register), has the register configuration shown below.

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

The delay interrupt factor originate/clear register (DIRR) is a register used to generate/clear the delay interrupt factor. Writing "1" to the register results in a request to delay an interrupt, and writing "0" clears the delay interrupt request. Resetting causes the factor clear state. Either "0" or "1" can be written to the reserve bit area. For future expansion, however, Fujitsu recommends using the set bit or clear bit instructions to access this register.

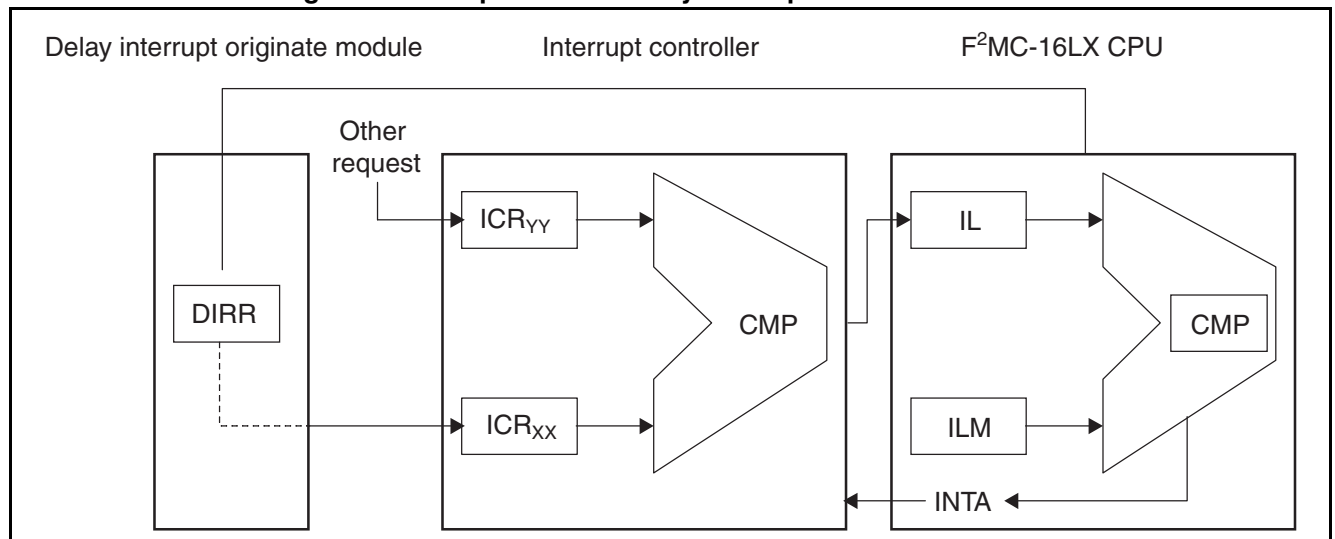
### 3.11.1 Operation of Delay Interrupt Generation Module

If CPU writes "1" to the relevant DIRR bit with software, the request latch in the delay interrupt generation module is set to generate an interrupt request to the interrupt controller.

#### ■ Operation of Delay Interrupt Generation Module

If CPU writes "1" to the relevant DIRR bit with software, the request latch in the delay interrupt generation module is set to generate an interrupt request to the interrupt controller. If other interrupt requests have a priority lower than this interrupt or there are no other interrupt requests, the interrupt controller generates an interrupt to the F<sup>2</sup>MC-16LX CPU. The F<sup>2</sup>MC-16LX CPU compares the interrupt request with the ILM bit in the internal CCR register, and if the request level is higher than that of the ILM bit, the hardware interrupt processing micro-program starts immediately after the instruction currently being executed is completed. As a result, the interrupt routine for this interrupt is executed. By writing "0" to the relevant DDIR bit within the interrupt processing routine, this interrupt factor is cleared and the task is switched. The above operation flow is illustrated in the Figure 3.11-2.

**Figure 3.11-2 Operation of Delay Interrupt Generation Module**



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

This latch is set by writing "1" to the relevant DIRR bit, and cleared by writing "0" to the same bit. Be sure to create software so that a factor is cleared in the interrupt processing routine. Otherwise, interrupt processing starts soon after the system returns from interrupt factor processing.



# CHAPTER 4    RESET

---

**This chapter explains reset for the MB90480/485 series.**

---

4.1 Overview of Reset

4.2 Reset Factors and Oscillation Stabilization Wait Time

4.3 External-Reset Pin

4.4 Resetting

4.5 Reset-Factor Bits

4.6 Condition of Pins as Result of Reset

## 4.1 Overview of Reset

If a reset factor occurs, the CPU immediately stops the processing currently in progress and stands by for cancellation of the reset. After the reset is canceled, processing starts at the address specified by the reset vector.

A reset is triggered by the following four factors:

- Power-on reset
- Watchdog timer overflow
- External reset request from  $\overline{\text{RST}}$  pin
- Software reset request

### ■ Reset Factors

Table 4.1-1 summarizes the reset factors.

Table 4.1-1 Reset Factors

| Reset          | Reset factor   | Machine clock     | Watchdog timer | Waits until oscillation is stabilized? |
|----------------|--|-------------------|----------------|--|
| Power on       | When power is turned on  | Main clock (MCLK) | Stopped        | Yes                                    |
| Watchdog timer | Watchdog timer overflow  | Main clock (MCLK) | Stopped        | No                                     |
| External pin   | "L" level input to pin $\overline{\text{RST}}$   | Main clock (MCLK) | Stopped        | No                                     |
| Software       | "0" is written in internal reset signal bit (RST) of low-power consumption mode control register (LPMCR) | Main clock (MCLK) | Stopped        | No                                     |

Main clock: clock of oscillation clock divided by two

#### ○ Power-on reset

A power-on reset occurs when the power is turned on. The oscillation stabilization wait time for evaluation devices and flash memory devices is  $2^{18}/\text{HCLK}$  (about 65.54 ms where the oscillation clock is 4 MHz). The oscillation stabilization wait time for mask ROM devices is fixed at  $2^{17}/\text{HCLK}$  (about 32.77 ms where the oscillation clock is 4 MHz). However, oscillation stabilization wait time of MB90F488B/F489B becomes  $(2^{18}+2^{15})/\text{HCLK}$  (about 73.73 ms when the oscillation clock is 4MHz). A reset is performed after the end of the oscillation stabilization wait time.

#### ○ Watchdog reset

A watchdog reset is triggered by a watchdog timer overflow if "0" is not written in the watchdog control bit (WTE) of the watchdog timer control register (WDTC) within a preset time after the watchdog timer is activated. The oscillation stabilization wait time can be specified in the clock selection register (CKSCR).

### ○ External reset

An external reset is triggered by input of the "L" level to the external-reset pin (pin  $\overline{\text{RST}}$ ). More than 16 machine cycles ( $16/\phi$ ) is required for the "L" level input time to pin  $\overline{\text{RST}}$ .

An external reset (pin  $\overline{\text{RST}}$  input reset) does not require the oscillation stabilization wait time.

#### Reference:

After an instruction processing ends, the reset cancellation waiting state is set only when a reset request is issued via pin  $\overline{\text{RST}}$  because of the event where a reset factor is triggered during writing (such as the MOV instruction while a transfer instruction is being executed). Writing thus ends normally even if a reset is input during writing.

However, string instructions (such as the MOVS instruction) accept a reset before a transfer completes at the specified count, so the transfer of all data cannot be assured. Reset requests are also accepted when a bus cycle extension with pin RDY continues for more than 16 machine cycles during external bus access.

### ○ Software reset

In a software reset, an internal reset is triggered by writing "0" in the internal reset signal bit (RST) of the low-power consumption mode control register (LPMCR). A software reset does not require the oscillation stabilization wait time.

#### Reference:

Definition of clock

HCLK: Oscillation clock (clock supplied via high-speed oscillation pin)

MCLK: Main clock (clock of HCLK divided by two)

SCLK: Sub clock (clock divided by four, supplied via low-speed oscillation pin)

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

$1/\phi$ : Machine cycle (CPU operation clock period)

Refer to Section "5.1 Overview of Clocks", for a detailed information on machine clocks.

---

#### Note:

The oscillation stabilization wait time of  $2^{17}/\text{HCLK}$  (about 32.77 ms where the oscillation clock is 4 MHz) is required if a reset is triggered in the stop mode or the sub clock mode. Refer to Section "5.4 Clock Modes", for a detailed information on clock modes.

---



## 4.2 Reset Factors and Oscillation Stabilization Wait Time

The four types of reset factors can occur in the MB90480/485 series devices. The oscillation stabilization wait time during a reset varies depending on the reset factor.

### ■ Reset Factors and Oscillation Stabilization Wait Time

Table 4.2-1 summarizes the reset factors and the oscillation stabilization wait time.

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

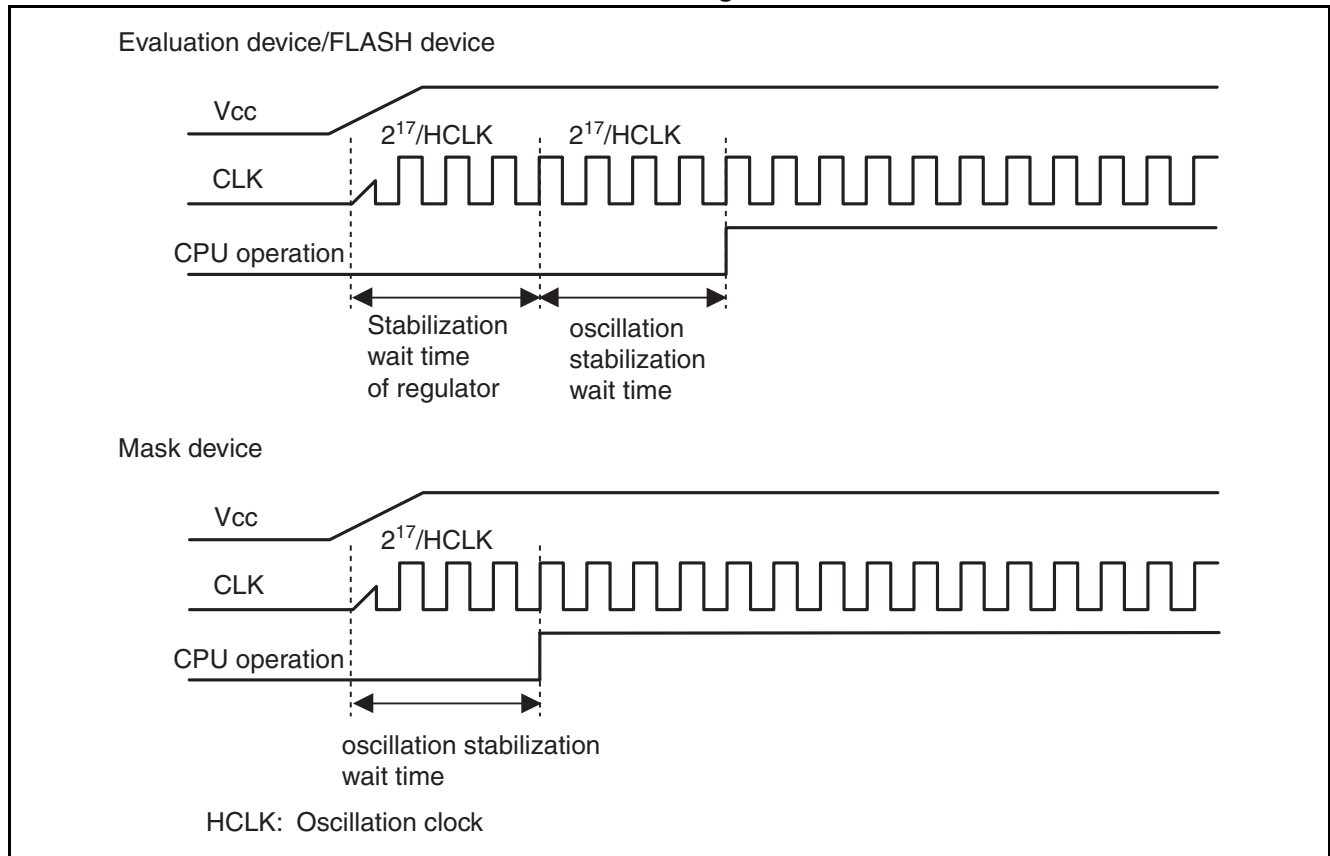
| Reset factors                                  | Oscillation stabilization wait time<br>The value in parentheses ( ) is a period when oscillation clock is 4 MHz                 |
|--|---|
| Power-on reset                                 | Evaluation devices/FLASH devices: $2^{18}/\text{HCLK}$ (about 65.54 ms).<br>Mask devices: $2^{17}/\text{HCLK}$ (about 32.77 ms) |
| Watchdog timer                                 | None: Bits WS1 and WS0 are initialized to "11 <sub>B</sub> ".   |
| External reset via pin $\overline{\text{RST}}$ | None: Bits WS1 and WS0 are initialized to "11 <sub>B</sub> ".   |
| Software reset                                 | None: Bits WS1 and WS0 are initialized to "11 <sub>B</sub> ".   |

HCLK: Oscillation clock

WS1, WS0: Bits for selecting oscillation stabilization wait time of clock selection register (CKSCR)

Figure 4.2-1 shows the oscillation stabilization wait time for evaluation devices, flash memory devices, and mask ROM devices during a power-on reset.

**Figure 4.2-1 Waiting Times to Stable Oscillation for Evaluation Devices/Flash Memory Devices and Mask ROM Devices During Power-on Reset**



**Note:**

Ceramic and crystal oscillators generally require an oscillation stabilization wait time ranging from several milliseconds to several ten milliseconds after the start of oscillation until oscillation stabilizes at a specific frequency. Therefore, specify a oscillation stabilization wait time suitable for the oscillator used.

Refer to Section "5.5 Oscillation Stabilization Wait Time", for more information.

## ■ Reset State Waiting for Stable Oscillation

A reset during the power-on sequence and a reset in response to a reset request in the stop and sub clock modes is performed after the end of the oscillation stabilization wait time created by the time-base timer. In this event, a reset is performed after an external reset is canceled unless external reset input is cleared.

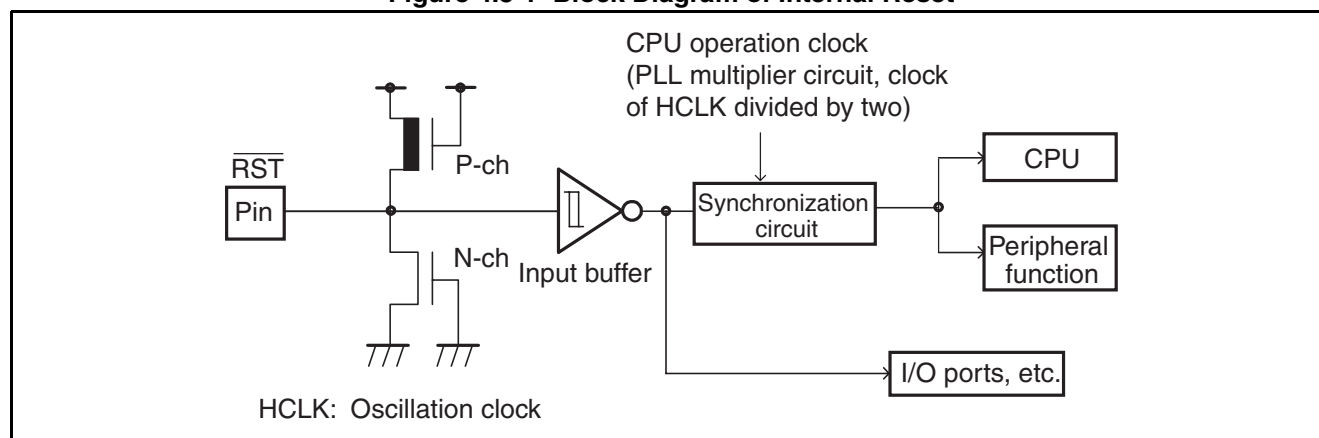
## 4.3 External-Reset Pin

The external-reset pin (pin  $\overline{\text{RST}}$ ) is a pin dedicated for the input of resets, and it triggers an internal reset by input of the "L" level. The MB90480/485 series devices have resets synchronized to the CPU operation clock. However, only external pins (e.g., ports) change asynchronously to a reset state.

### ■ Block Diagram of External-Reset Pin

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

**Figure 4.3-1 Block Diagram of Internal Reset**



#### Note:

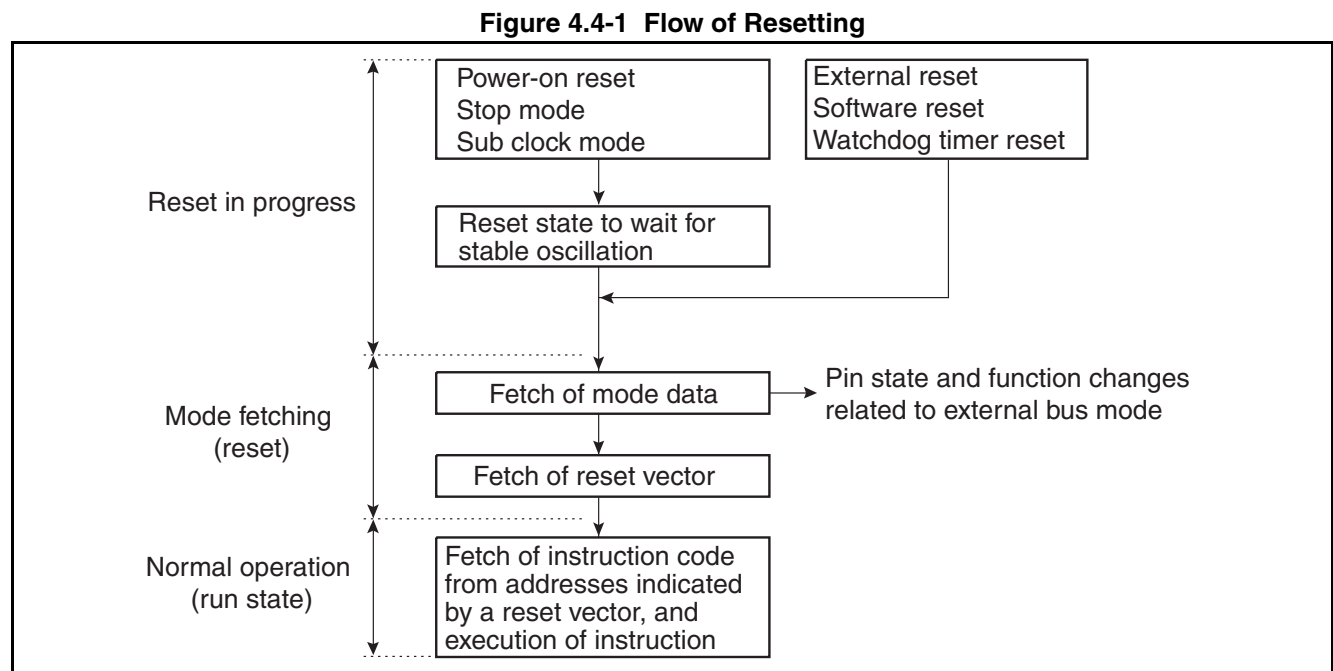
To prevent damage to the memory contents by a reset during writing, input to pin  $\overline{\text{RST}}$  is accepted in a cycle that precludes damage to memory contents. A clock is required to initialize internal circuits. Input of a clock is required during input of a reset when an external clock is used for operation.

## 4.4 Resetting

After the cancellation of a reset, a read from operation of mode data and the reset vector can be selected by setting the mode pin to perform mode fetching. Mode fetching determines the CPU operation mode and the start address of execution after the end of a reset. When the power is turned on or when the system is returned from the stop-mode by a reset, perform mode fetching after the end of the oscillation stabilization wait time.

### ■ Overview of Resetting

Figure 4.4-1 shows the flow of resetting.



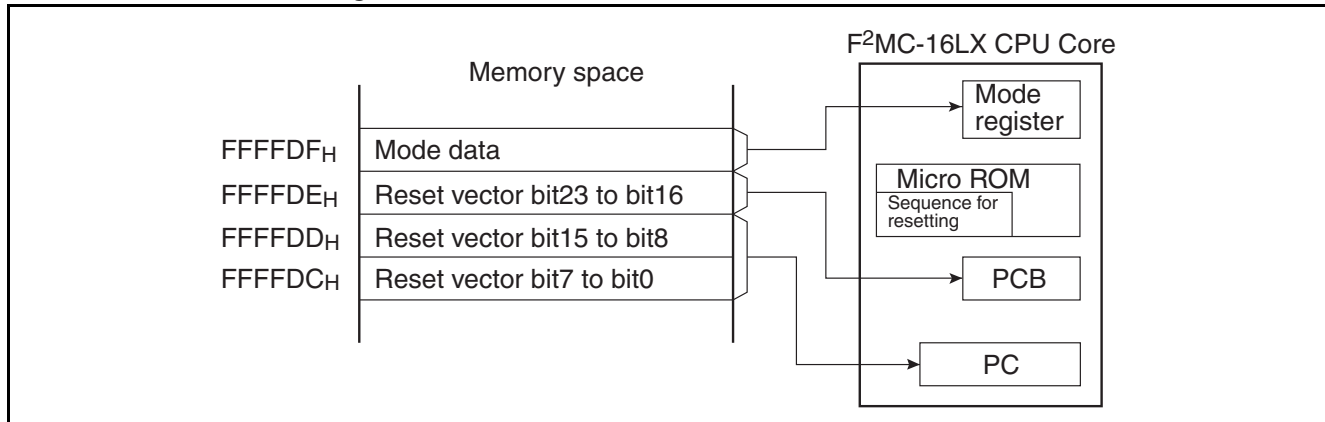
### ■ Mode Pins

Mode pins (MD2 to MD0) specify a method to fetch reset vectors and mode data. A reset vector and mode data are fetched in a sequence for resetting. Refer to Section "7.2 Mode Pins (MD2 to MD0)", for details of the mode pins.

### ■ Mode Fetch

After a reset is canceled, the CPU transfers the reset vectors and mode data to the applicable registers in the CPU core. The reset vectors and mode data are allocated to four bytes, namely  $\text{FFFFDC}_\text{H}$  to  $\text{FFFFDF}_\text{H}$ . Upon a reset cancellation, the CPU immediately outputs these addresses to a bus and fetches reset vectors and mode data. During mode fetching, the CPU starts processing from the address specified by the reset vector.

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

**Figure 4.4-2 Transfer of Reset Vectors and Mode Data****Reference:**

Use a mode pin, from which reset vectors and mode data are read, to specify either internal ROM or external memory. If the external vector mode is specified with a mode pin, however, external memory and not internal ROM is accessed to read reset vectors and mode data. Fujitsu recommends specifying the internal vector mode with a mode pin when the single-chip mode and internal ROM external bus mode are used.

○ **Mode data (Address: FFFFDF<sub>H</sub>)**

The data in the mode register can be modified only by a reset, and the mode register settings become effective after a reset. Refer to Section "7.3 Mode Data", for details on mode data.

○ **Reset vector (Address: FFFFDC<sub>H</sub> to FFFFDE<sub>H</sub>)**

Write the execution start address after the end of a reset. Execution starts from this address.

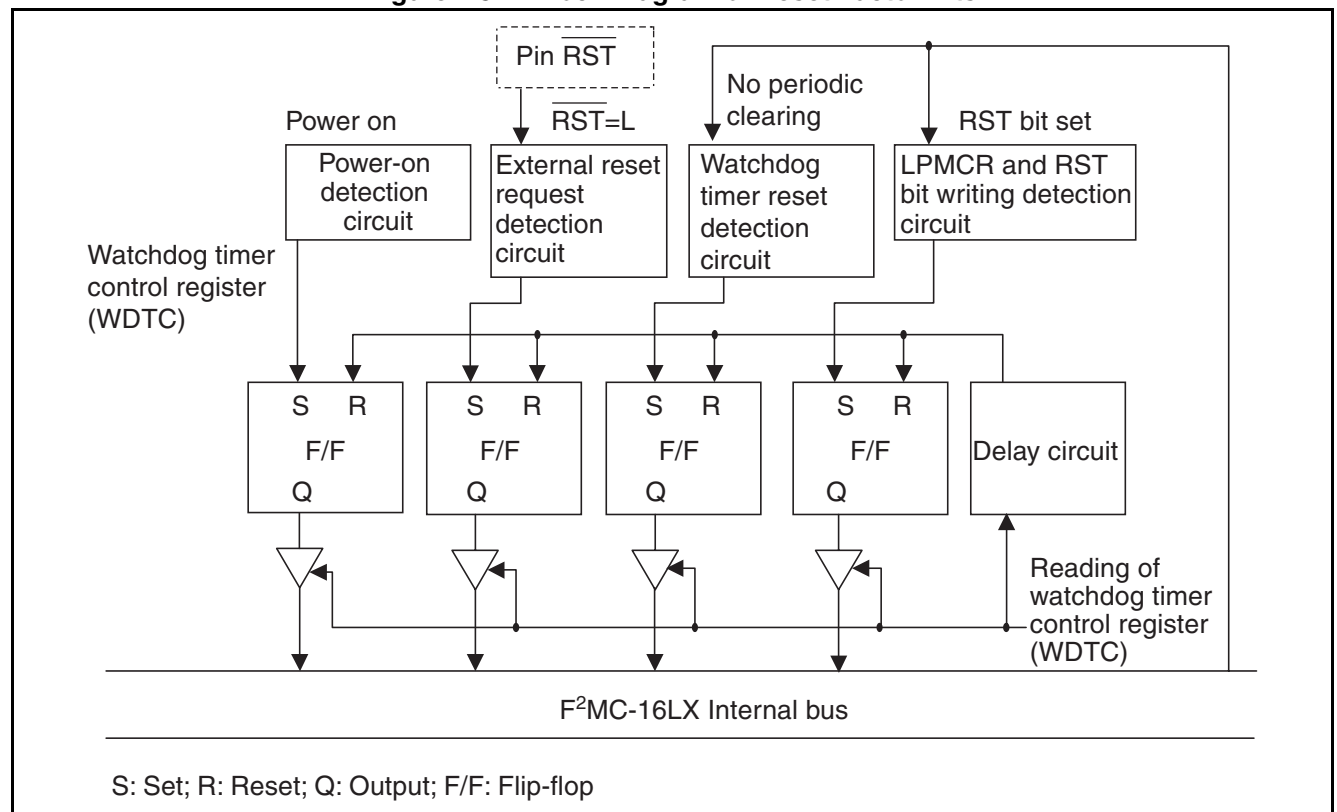
## 4.5 Reset-Factor Bits

Reset factors can be determined by reading the watchdog timer control register (WDTC).

### ■ Reset-factor Bits

As shown in the Figure 4.5-1, each reset factor has a corresponding flip-flop assigned to it. This information can be obtained by reading the watchdog timer control register (WDTC). If a reset factor must be determined after a reset cancellation, run software to process the read value of the WDTC register, and branch to an appropriate program.

**Figure 4.5-1 Block Diagram of Reset-Factor Bits**



## ■ Correspondence Between Reset-Factor Bits and Reset Factors

Figure 4.5-2 shows the configuration of the reset-factor bits for the watchdog timer control register (WDTC). Table 4.5-1 shows the correspondence between reset-factor bits and reset factors.

For details, refer to Section "10.2 Watchdog Timer Control Register (WDTC)".

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

| bit                 | 15     | 8 | 7    | 6        | 5    | 4    | 3    | 2   | 1   | 0   |               |
|---------------------|--------|---|------|----------|------|------|------|-----|-----|-----|---------------|
| 0000A8 <sub>H</sub> | (TBTC) |   | PONR | Reserved | WRST | ERST | SRST | WTE | WT1 | WT0 |               |
|                     |        |   | X    | X        | X    | X    | X    | 1   | 1   | 1   | Initial value |
|                     |        |   | R    | R        | R    | R    | R    | W   | W   | W   | R/W           |

R: Read only; W: Write only

**Table 4.5-1 Correspondence Between Reset-Factor Bits and Reset Factors**

| Reset factor   | PONR | WRST | ERST | SRST |
|--|------|------|------|------|
| Power-on reset   | 1    | X    | X    | X    |
| Watchdog timer overflow                                | *    | 1    | *    | *    |
| External reset request via pin $\overline{\text{RST}}$ | *    | *    | 1    | *    |
| Software reset request                                 | *    | *    | *    | 1    |

\* : Retains the state before

X: Undefined bit

## ■ Cautions about Reset-Factor Bits

### ○ If more than one reset factor occurs

If more than one reset factor occurs, the individual reset-factor bits of the WDTC register are set to "1". For example, if an external reset via pin  $\overline{\text{RST}}$  is requested at the same time as a watchdog timer overflow occurs, bits ERST and WRST of the reset-factor bits are set to "1".

### ○ Power-on reset

During a power-on reset, bit PONR of the reset-factor bits is set to "1". However, the reset-factor bits other than bit PONR are undefined. Therefore, if bit PONR is "1", create software so that reset-factor bits other than bit PONR are ignored.

### ○ Clearing reset-factor bits

The reset-factor bits is cleared only if the data in the WDTC register is read. Bits corresponding to reset factors that have occurred once are not cleared even if a reset is triggered (remains "1").

Note:

The values of the WDTC register may not be assured if the power is turned on under a condition that precludes a power-on reset.

## 4.6 Condition of Pins as Result of Reset

---

This section explains the states of pins after a reset.

---

### ■ Pin States During a Reset

States of the pins during a reset are determined by the settings of mode pins (MD2 to MD0).

- **If the internal vector mode is set (MD2 to MD0 = 011<sub>B</sub>)**

All I/O pins (resource pins) become set at the high-impedance state, and mode data is read from internal ROM.

Refer to Section "6.7 Pin State in Standby Mode, Hold, and Reset", for the states of pins during a reset.

### ■ Pin States after Mode Data Is Read

The states of the pins after mode data is read are determined by mode data (M1, M0).

- **If the single-chip mode is set (M1, M0 = 00<sub>B</sub>)**

All I/O pins (resource pins) become set at the high-impedance state, and mode data is read from the internal ROM.

---

Note:

Take care with the pins that have the high-impedance state during a reset so that equipment connected to the pins do not malfunction.

---





# CHAPTER 5    CLOCKS

---

**This chapter describes the clocks of the MB90480/485 series.**

---

5.1 Overview of Clocks

5.2 Block Diagram of Clock Generator

5.3 Clock Selection Register (CKSCR) and PLL Output Selection Register (PLLOS)

5.4 Clock Modes

5.5 Oscillation Stabilization Wait Time

5.6 Connecting Oscillator to External Clock

## 5.1 Overview of Clocks

---

The clock generator controls the operations of internal clocks, which are the operation clocks of the CPU and peripheral functions. In this document, the clocks are called as follows according to clock type:

- **Machine clock:** Defined as an internal clock.
  - **Machine cycle:** Defined as one period of a machine clock.
  - **Oscillation clock:** Defined as a clock supplied via a high-speed oscillation pin.
  - **PLL clock:** Defined as a clock using internal PLL oscillation.
  - **Sub clock:** Clock divided by four, provided from a low-speed oscillation pin.
- 

### ■ Overview of Clocks

The clock generator contains an oscillation circuit and generates an oscillation clock and sub clock by using an external connection to an oscillator. The generator generates an oscillation clock by inputting a clock generated externally. The generator contains a PLL clock multiplier circuit and generates four multiplication clocks of an oscillation clock. The clock generator controls the oscillation stabilization wait time, PLL clock multiplication, and operations of internal clocks by changing the clock of the clock selector.

#### ○ Oscillation clock (HCLK)

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

#### ○ Sub clock (SCLK)

This clock operates the watch timer. It can also be used as a low-speed machine clock.

This clock is divided by four and created by connecting an oscillator to the low-speed oscillation pin or by inputting an external clock.

#### ○ Main clock (MCLK)

This is a clock of the oscillation clock divided by two, and is used as an input clock to the time-base timer and clock selector.

#### ○ PLL clock (PCLK)

This clock is a clock obtained by multiplying with built-in PLL clock multiplier circuit (PLL oscillation circuit). Four types of the clocks can be selected.

#### ○ Machine clock ( $\phi$ )

This clock is an operation clock of the CPU and peripheral functions. One period of this clock is used as a machine cycle ( $1/\phi$ ). One clock can be selected from among the main clock (clock of oscillation clock divided by two), sub clock, and four types of multiplication clocks.

---

#### Note:

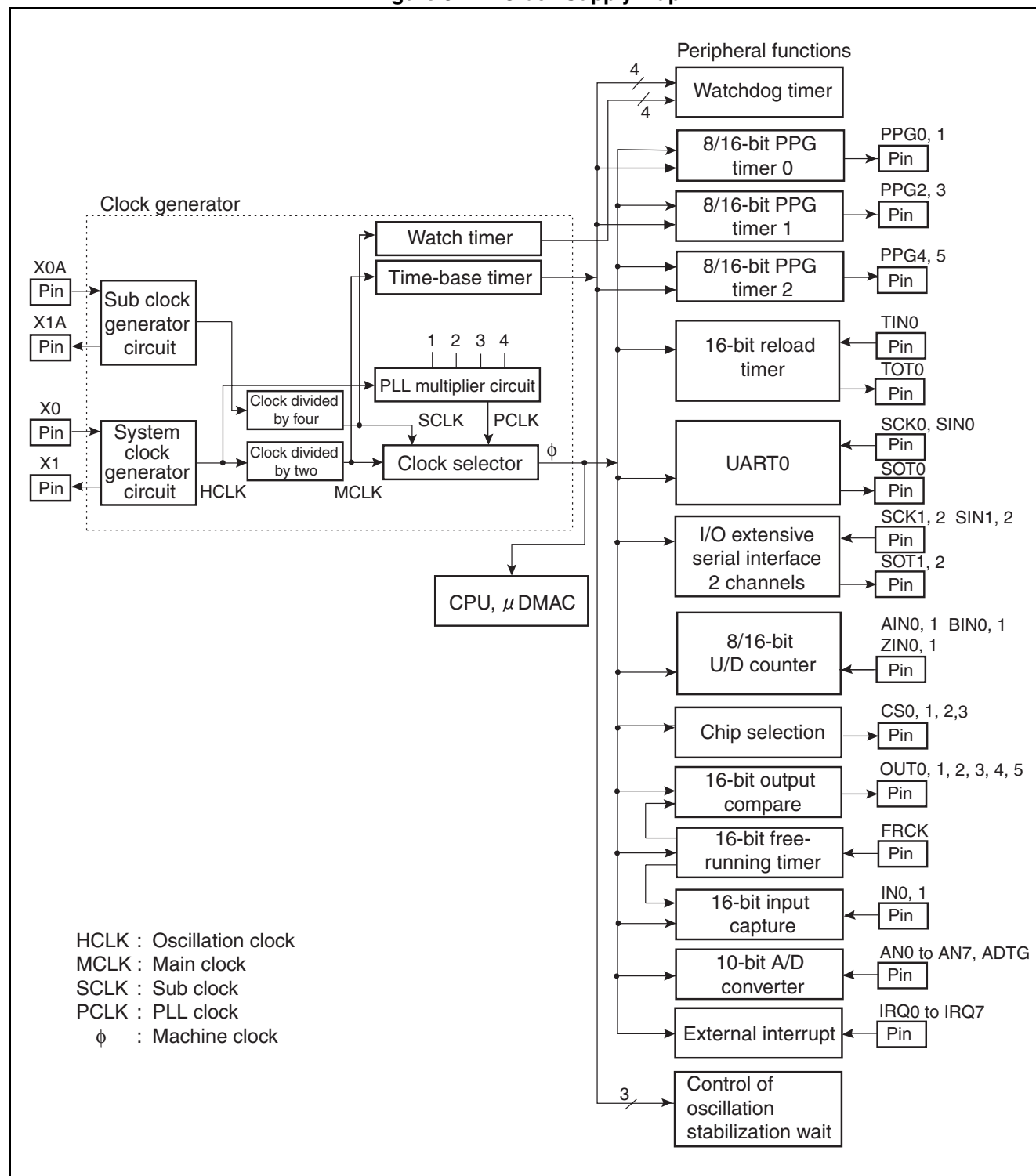
Oscillation clocks have an oscillation frequency ranging from 4.5 to 25 MHz. At using PLL, a machine clock from 20 to 25 MHz is used, set the PLL2 bit of the PLL0S register to 1. The maximum operating frequency of the CPU and peripheral functions is 25 MHz. If a multiply-by rate exceeding the maximum operating frequency is specified, the device will not operate correctly. PLL oscillation can be between 4.5 and 25 MHz. This oscillation range varies depending on operating voltage and the multiplication rate.

---

## ■ Clock Supply Map

Machine clocks generated by the clock generator are supplied as operation clocks of the CPU and peripheral functions. Therefore, operations of the CPU and peripheral functions are affected by changes between the main clock and PLL clock (clock mode) and by changes in the PLL clock multiplication rate. The clock-divided outputs of the time-base timer are supplied to some peripheral functions, and the peripheral functions can select their own operation clocks. Figure 5.1-1 shows a clock supply map.

Figure 5.1-1 Clock Supply Map



## 5.2 Block Diagram of Clock Generator

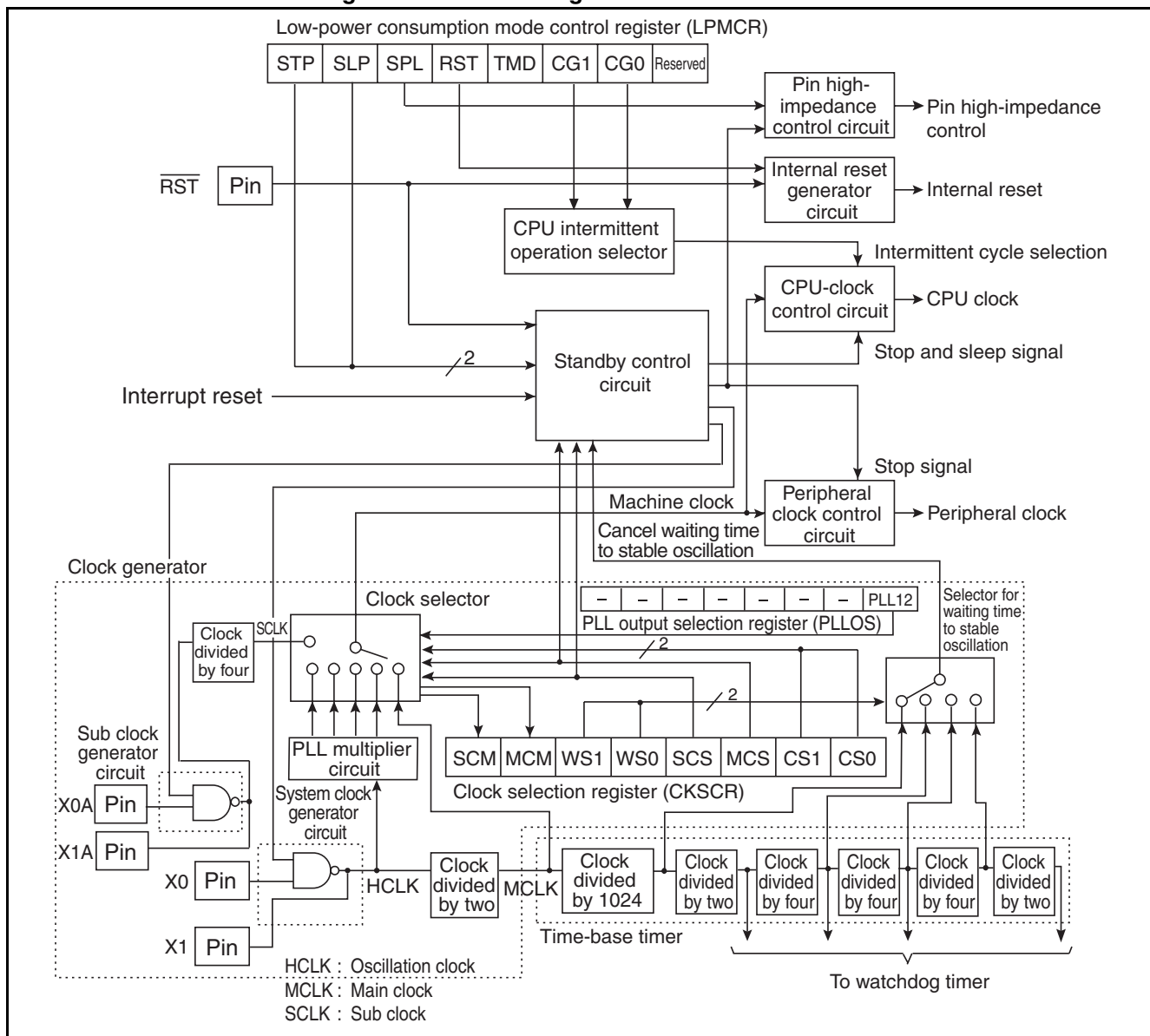
The clock generator consists of the following five blocks:

- System clock generator circuit/sub clock generator circuit
- PLL multiplier circuit
- Clock selector
- Clock Selection Register (CKSCR) and PLL Output Selection Register (PLLOS)
- Selector for oscillation stabilization wait time

### ■ Block Diagram of Clock Generator

Figure 5.2-1 is a block diagram of the clock generator. Figure 5.2-1 also includes the standby control circuits and time-base timer circuit.

**Figure 5.2-1 Block Diagram of Clock Generator**



- **System clock generator circuit**

This circuit generates an oscillation clock (HCLK) by using an oscillator connected to the high-speed oscillation pin. Also, an external clock can be input to it.

- **Sub clock generator circuit**

This circuit generates a sub clock (SCLK) by using an oscillator connected to the low-speed oscillation pin. Also, an external clock can be input to it.

- **PLL multiplier circuit**

This circuit multiplies an oscillation clock by using PLL oscillation and supplies it to the CPU clock selector.

- **Clock selector**

This circuit selects clocks from among the main clock, sub clock, and four PLL clocks supplied to the CPU clock control circuit and peripheral clock control circuit.

- **Clock selection register (CKSCR)**

This register changes between the oscillation clock and PLL clocks, selects the oscillation stabilization wait time, and selects the multiplication rate of the PLL clocks.

- **PLL output selection register (PLLOS)**

Use this register to specify doubling of the multiply-by rate specified in the CKSCR register for the PLL to be used when a machine clock is used at a frequency of 20 to 25 MHz.

- **Selector for the oscillation stabilization wait time**

This circuit selects the oscillation stabilization wait time of the oscillation clock when the stop mode is reset and during watchdog reset. Four types of the time-base timer output are selected.

## 5.3 Clock Selection Register (CKSCR) and PLL Output Selection Register (PLLOS)

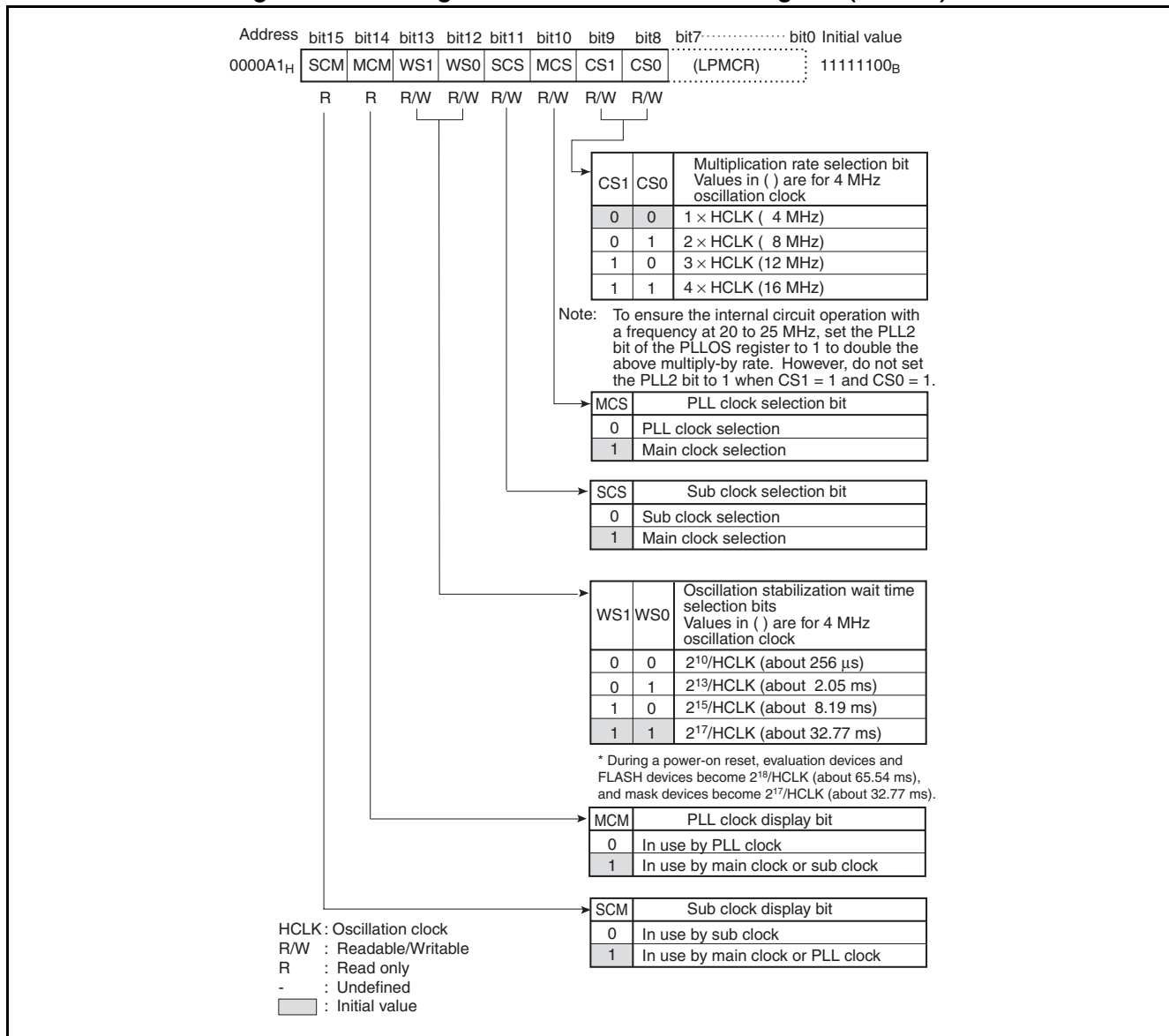
The clock selection register (CKSCR) switches among the main clock, sub clock, and PLL clock, and it selects the oscillation stabilization wait time and PLL clock multiplication rate.

The PLL output selection register (PLLOS) must be set for the PLL to be used when a machine clock is used at a frequency of 20 to 25 MHz.

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

Figure 5.3-1 shows the configuration of the clock selection register (CKSCR). Table 5.3-1 has explanations for the functions of bits in the clock selection register.

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



Note:

When reset, the machine clock selection (MCS) bit is initialized to the main clock selection.

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

| Bit name        |  | Function  |
|-----------------|--|---|
| bit15           | SCM:<br>Sub clock display bit                                      | <p>This bit displays whether the main clock or sub clock is selected as a machine clock.</p> <ul style="list-style-type: none"> <li>If the bit is "0", the sub clock is selected. If "1", the main clock or PLL clock is selected.</li> <li>If SCS = 1 and SCM = 0, the mode is the waiting time for stable oscillation of the main clock.</li> <li>Write doesn't affect operation.</li> </ul>  |
| bit14           | MCM:<br>PLL clock display bit                                      | <p>This bit displays whether the main clock or PLL clock is selected as a machine clock.</p> <ul style="list-style-type: none"> <li>If this bit is "0", the PLL clock is selected. If "1", the main clock or sub clock is selected.</li> <li>If PLL clock selection bit (MCS) = 0 and MCM = 1, the mode is the waiting time for stable oscillation of the PLL clock.</li> <li>Write doesn't affect operation.</li> </ul>  |
| bit13,<br>bit12 | WS1, WS0:<br>Oscillation stabilization<br>wait time selection bits | <p>Selects the waiting time for stable oscillation of oscillation clock in a change from the sub clock mode to the main clock mode or from the sub clock mode to the PLL clock mode if the stop mode is canceled. Initialized to "11<sub>B</sub>" by all reset factors.</p> <p><b>Note:</b></p> <p>The specified value of the waiting time for stable oscillation must be suitable for the oscillator used. Refer to Section "4.2 Reset Factors and Oscillation Stabilization Wait Time", for more information. Specify "00<sub>B</sub>" only if the mode is the main clock mode.</p> <p>When the main clock is switched to PLL clock mode, the PLL clock oscillation stabilization wait time is fixed at <math>2^{14}/\text{HCLK}</math>.</p> <p>When sub clock mode is switched to PLL clock mode or when PLL stop mode is returned to PLL clock mode, the oscillation stabilization wait time uses the specified values in the WS1 and WS0 bits. For PLL clock oscillation stabilization wait time, at least <math>2^{14}/\text{HCLK}</math> is required. Accordingly, when sub clock mode is switched to PLL clock mode, or when PLL clock mode is switched to PLL stop mode, set WS1 and WS0 bits to "10<sub>B</sub>" or "11<sub>B</sub>".</p> |



Table 5.3-1 Functions of Bits in Clock Selection Register (CKSCR) (2/2)

| Bit name      |  | Function  |
|---------------|--|---|
| bit11         | SCS:<br>Sub clock selection bit                | <p>This bit specifies selection of the main clock or sub clock as a machine clock.</p> <ul style="list-style-type: none"> <li>If this bit is "0", the sub clock is selected. If "1", the main clock is selected.</li> <li>When this bit is rewritten from "1" to "0", the mode is switched to the sub clock mode synchronizing the sub clock (approx. 130 <math>\mu</math>s.).</li> <li>Writing "1" when this bit is "0" generates a standby period for stable oscillation of the main clock. The time-base timer is automatically cleared.</li> <li>Initialized to "1" by all reset factors.</li> </ul> <p><b>Note:</b><br/>Use the sub clock as an operation clock when the sub clock is selected. (The machine clock changes to a frequency of 8 kHz during low-speed oscillation at 32 kHz)<br/>If both SCS and MCS are "0", SCS is assigned with priority, and the sub clock is selected.</p>  |
| bit10         | MCS:<br>PLL clock selection bit                | <p>This bit specifies selection of the main clock or PLL clock as a machine clock.</p> <ul style="list-style-type: none"> <li>If this bit is "0", the PLL clock is selected. If "1", the main clock is selected.</li> <li>Writing "0" when this bit is "1" generates a waiting time for stable oscillation of the PLL clock. The time-base timer is automatically cleared. The interrupt request flag bit (TBOF) of the time-base timer control register (TBTC) is also cleared.</li> <li>When the main clock is switched to PLL clock mode, the oscillation stabilization wait time is fixed at <math>2^{14}/\text{HCLK}</math>. (The oscillation stabilization wait time is about 4.1 ms if the oscillation clock has a frequency of 4 MHz.) When sub clock mode is switched to PLL clock, the oscillation stabilization wait time uses the specified values in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0).</li> <li>When the main clock is selected, the operation clock is the oscillation clock divided by 2. (The operation clock is 2 MHz if the oscillation clock is 4 MHz.)</li> <li>Initialized to "1" by all reset factors.</li> </ul> <p><b>Note:</b><br/>Writing "0" when the MCS bit is "1", write while the time-base timer interrupt is masked by using the interrupt request enable bit (TBIE) of the TBTC register or the interrupt level register (ILM).</p> |
| bit9,<br>bit8 | CS1, CS0:<br>Multiplication rate selection bit | <p>This bit selects the multiplication rate of the PLL clocks. Selection is from four multiplication rates. All reset factors initialize it to "00<sub>B</sub>".</p> <p><b>Note:</b><br/>Writing is disabled if the MCS bit or MCM bit is "0". Rewrite the CS1 and CS0 bits after setting the MCS bit to "1" (main clock mode).</p>   |

HCLK: Oscillation clock

■ Configuration of PLL Output Selection Register (PLLOS)

Figure 5.3-2 shows the configuration of the PLL output selection register (PLLOS). Table 5.3-2 explains the functions of the bits for the PLL output selection register.

Figure 5.3-2 Configuration of PLL Output Selection Register (PLLOS)

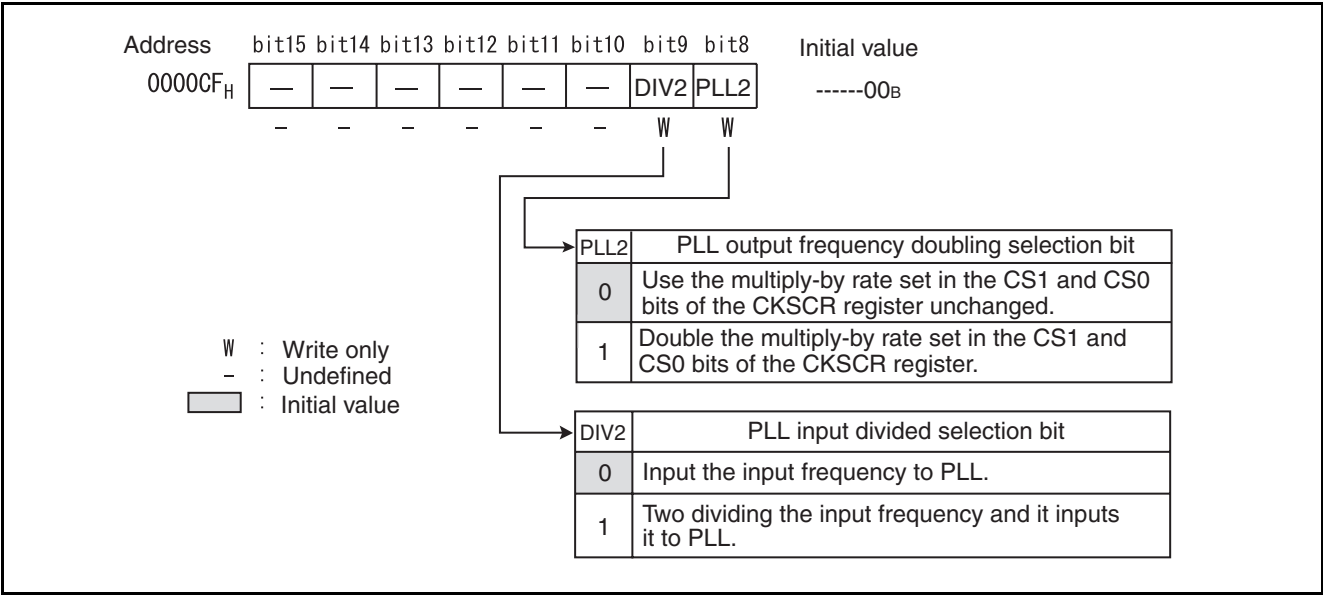


Table 5.3-2 Functions of Bits for PLL Output Selection Register (PLLOS)

| Bit name       |   | Function   |
|----------------|---|--|
| bit15 to bit10 | Undefined bits  | Not used   |
| bit9           | DIV2:<br>PLL input<br>divided<br>selection bit                | <ul style="list-style-type: none"> <li>This bit selects dividing of input clock to PLL or input as it is.</li> <li>It is initialized to "0" by all reset sources.</li> <li>Read value is undefined.</li> <li>Please do not change this bit when you use the clock of PLL.</li> <li>Please set DIV2 bit of PLLOS register = 1 and PLL2 bit = 1 during PLL1, 2, 3, 4 multiplication setting and internal clock with <math>20\text{ MHz} &lt; f_{CP} \leq 25\text{MHz}</math>. (During PLL 4 multiplication setting, please set input frequency is 60MHz or more.)</li> </ul> <p>Example: If sending frequency = 24 MHz and PLL1 multiplication setting:<br/>CKSCR register: CS1 bit = 0, CS0 bit = 0<br/>PLLOS register: DIV2 bit = 1, PLL2 bit = 1</p> <p>Example: If sending frequency = 8 MHz and PLL3 multiplication setting:<br/>CKSCR register: CS1 bit = 1, CS0 bit = 0<br/>PLLOS register: DIV2 bit = 1, PLL2 bit = 1</p> <ul style="list-style-type: none"> <li>It is possible to set the following during PLL 2, 4 multiplication setting and internal clock with <math>20\text{ MHz} &lt; f_{CP} \leq 25\text{MHz}</math>;</li> </ul> <p>PLL 2 multiplication:<br/>CKSCR register: CS1 bit = 0, CS0 bit = 0<br/>PLLOS register: DIV2 bit = 0, PLL2 bit = 1</p> <p>PLL 4 multiplication:<br/>CKSCR register: CS1 bit = 0, CS0 bit = 1<br/>PLLOS register: DIV2 bit = 0, PLL2 bit = 1</p> <ul style="list-style-type: none"> <li>Please set DIV2 bit of PLLOS register = 0 and PLL2 bit = 1 during PLL6, 8 multiplication setting.</li> </ul> <p>Example: If sending frequency = 4 MHz and PLL6 multiplication setting:<br/>CKSCR register: CS1 bit = 1, CS0 bit = 0<br/>PLLOS register: DIV2 bit = 0, PLL2 bit = 1</p> <p>Example: If sending frequency = 3 MHz and PLL8 multiplication setting:<br/>CKSCR register: CS1 bit = 1, CS0 bit = 1<br/>PLLOS register: DIV2 bit = 0, PLL2 bit = 1</p> |
| bit8           | PLL2: PLL<br>output<br>frequency<br>doubling<br>selection bit | <ul style="list-style-type: none"> <li>This bit specifies doubling of the multiply-by rate for the PLL to be used when a machine clock is used at a frequency of 20 to 25 MHz.</li> <li>Initialized to "0" by all reset sources.</li> <li>The readout value is undefined.</li> </ul> <p>Do not change this bit when a PLL clock is being used.</p>   |

## 5.4 Clock Modes

---

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

---

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

#### ○ Main clock mode

The main clock mode uses a clock obtained by dividing the oscillation clock by two as the operation clock of the CPU and peripheral resources. This mode stops the PLL clock.

#### ○ PLL clock mode

The PLL clock mode uses the PLL clock obtained as the operation clock of the CPU and peripheral functions. The multiplication rate of the PLL clock can be selected with the clock selection register (CKSCR).

#### ○ Sub clock mode

The sub clock mode uses a sub clock as the operation clock of the CPU and peripheral resources. This mode stops the main and PLL clocks.

### ■ Change of Clock Mode

The clock mode changes to the main clock, PLL clock, or sub clock mode according to the writing of the PLL clock selection bit (MCS) and sub clock selection bit (SCS) in the CKSCR register.

#### ○ Change from the main clock mode to the PLL clock mode

Rewriting the MCS bit in the CKSCR register from "1" to "0" in the main clock mode changes the main clock to the PLL clock after the end of the oscillation stabilization wait time of the PLL clock ( $2^{14}/HCLK$ ).

#### ○ Change from the PLL clock mode to the main clock mode

Rewriting the MCS bit in the CKSCR register from "0" to "1" in the PLL clock mode changes the PLL clock to the main clock adjusted to the timing where the edges of the PLL clock and main clock match (after 1 to 8 PLL clocks).

#### ○ Change from the main clock mode to the sub clock mode

Rewriting the SCS bit in the CKSCR register from "1" to "0" in the main clock mode changes the main clock to the sub clock synchronizing the sub clock (approx. 130  $\mu$ s.).

#### ○ Change from the sub clock mode to the main clock mode

Rewriting the SCS bit in the CKSCR register from "0" to "1" in the sub clock mode changes the sub clock to the main clock after end of the oscillation stabilization wait time of the main clock. Select the oscillation stabilization wait time by using selection bits (WS1, WS0) for the oscillation stabilization wait time of the CKSCR register.

### ○ Change from the PLL clock mode to the sub clock mode

Rewriting the sub clock selection bit (SCS) of the clock selection register (CKSCR) from "1" to "0" in the PLL clock mode changes the PLL clock to the sub clock.

### ○ Change from the sub clock mode to the PLL clock mode

Rewriting the SCS bit in the CKSCR register from "0" to "1" in the sub clock mode changes the sub clock to the PLL clock after the end of the oscillation stabilization wait time of the main clock. Select the oscillation stabilization wait time by using selection bits (WS1, WS0) for the oscillation stabilization wait time of the CKSCR register.

---

#### Note:

Rewriting the PLL clock selection bit (MCS) or SCS bit in the CKSCR register does not change the machine clock immediately. When operating a resource that depends on a machine clock, make sure that the intended machine clock change has completed by checking the PLL clock display bit (MCM) and sub clock display bit (SCM) in the CKSCR register. Then operate the resource.

If both of the SCS and MCS bits are "0", SCS is assigned with priority, and the sub clock mode is set.

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.

---

## ■ Selection of PLL Clock Multiplication Rate

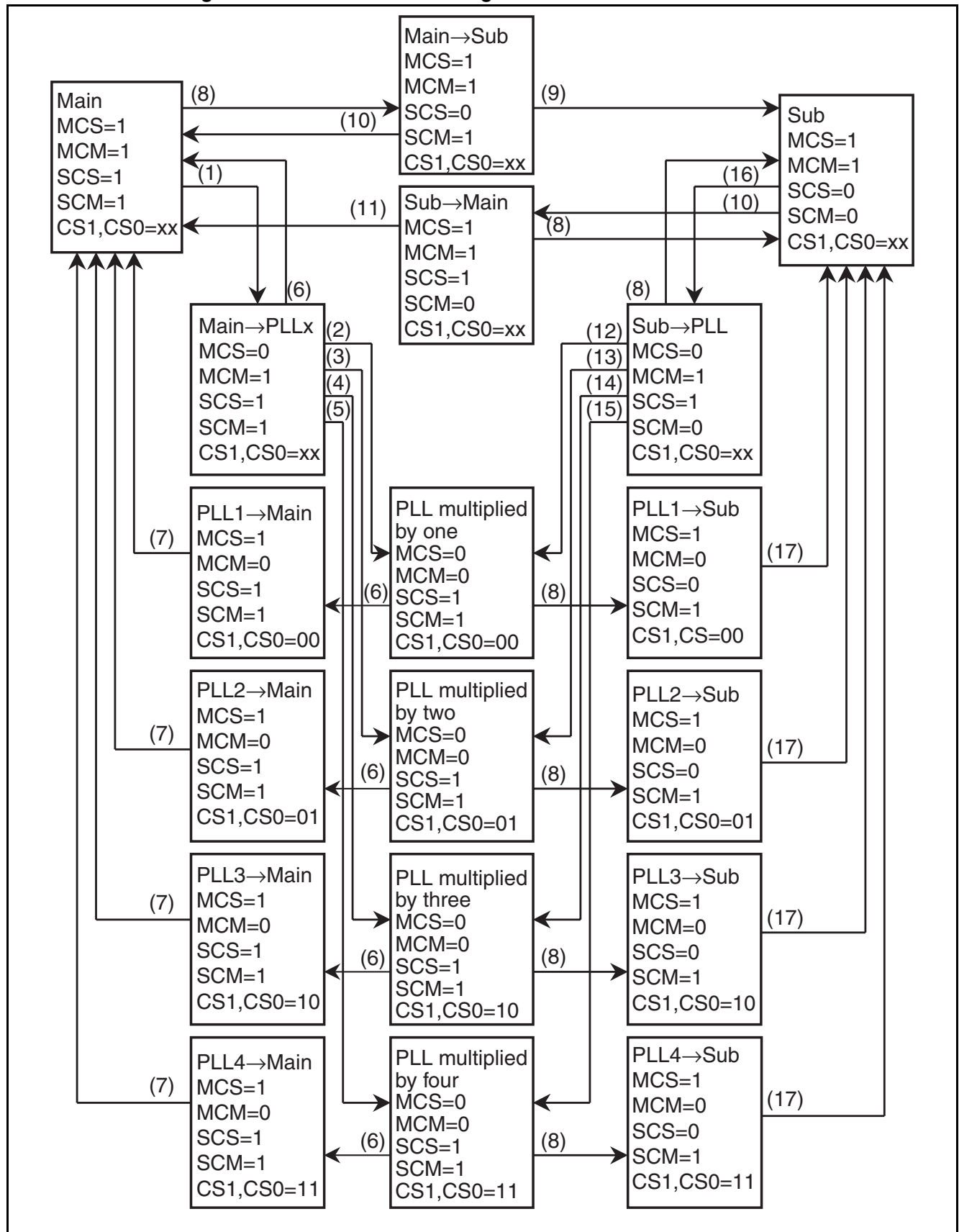
If 00<sub>B</sub> to 11<sub>B</sub> are written to the CS1 and CS0 bits of the CKSCR register, four types of PLL clock multiply-by rates can be selected: multiply-by 1 to 4.

## ■ Machine Clock

The PLL clock, main clock, and sub clock output by the PLL multiplier circuit are machine clocks, which are supplied to the CPU and peripheral functions. The main clock, PLL clock, and sub clock can be selected by writing in the SCS or MCS bit of the CKSCR register.

Figure 5.4-1 is a state transition diagram of machine clock selection.

Figure 5.4-1 State Transition Diagram of Machine Clock Selection



- (1) MCS bit "0" write
  - (2) Waiting for PLL clock oscillation stability is complete. &CS1, CS0 = 00
  - (3) Waiting for PLL clock oscillation stability is complete. &CS1, CS0 = 01
  - (4) Waiting for PLL clock oscillation stability is complete. &CS1, CS0 = 10
  - (5) Waiting for PLL clock oscillation stability is complete. &CS1, CS0 = 11
  - (6) MCS bit "1" write (includes watchdog reset)
  - (7) Synchronization timing of PLL and main clocks
  - (8) SCS bit "0" write
  - (9) End of waiting time for sub clock oscillation stability (maximum  $2^{14}/\text{SCLK}$ )
  - (10) SCS bit "1" write
  - (11) Waiting for main clock oscillation stability is complete.
  - (12) Waiting for main clock oscillation stability is complete. &CS1, CS0 = 00
  - (13) Waiting for main clock oscillation stability is complete. &CS1, CS0 = 01
  - (14) Waiting for main clock oscillation stability is complete. &CS1, CS0 = 10
  - (15) Waiting for main clock oscillation stability is complete. &CS1, CS0 = 11
  - (16) SCS bit "1" write, MCS bit "0" write
  - (17) Synchronization timing of PLL and sub clocks
- MCS: PLL clock selection bit of clock selection register (CKSCR)  
MCM: PLL clock display bit of clock selection register (CKSCR)  
SCS: Sub clock selection bit of clock selection register (CKSCR)  
SCM: Sub clock display bit of clock selection register (CKSCR)  
CS1, CS0: Multiplication rate selection bit of clock selection register (CKSCR)

---

### Note:

The initial value of the machine clocks is the main clock (MCS = 1, SCS = 1).  
If both of the SCS and MCS bits are "0", SCS is assigned with priority and the sub clock is set.  
When sub clock mode is switched to PLL clock mode, set "10<sub>B</sub>" or "11<sub>B</sub>" in the oscillation stabilization wait time selection bits (WS1, WS0) of the CKSCR register.

---

## 5.5 Oscillation Stabilization Wait Time

When the power is turned on, when stop mode is released, or switching from the sub clock to the main clock or from sub clock to the PLL clock occurs, an oscillation stabilization wait time is required after oscillation begins because the oscillation clock is stopped. When switching from the main clock to the PLL clock or from the main clock to the sub clock occurs, an oscillation stabilization wait time is required.

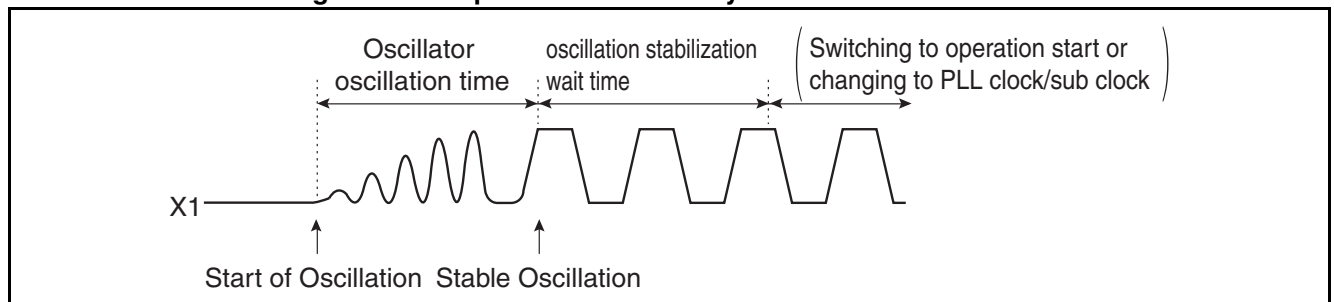
### ■ Oscillation Stabilization Wait Time

Ceramic and crystal oscillators generally require a waiting time ranging from several milliseconds to several ten milliseconds after the start of oscillation until oscillation stabilizes to a natural frequency (oscillation frequency). Therefore, disable CPU operation immediately after the start of oscillation, and supply a clock to the CPU when oscillation completely stabilizes following the elapse of the oscillation stabilization wait time. Specify a oscillation stabilization wait time suitable for the oscillator used because the time required for oscillation to stabilize varies depending on the type of oscillator (crystal, ceramic, or other material). The oscillation stabilization wait time can be selected by defining the clock selection register (CKSCR).

When the clock mode is switched from the main clock to the PLL clock, the main clock to the sub clock, the sub clock to the main clock, or the sub clock to the PLL clock, the CPU runs in the clock mode set before switching. After the oscillation stabilization wait time, the CPU changes to the selected clock mode.

Figure 5.5-1 illustrates operation immediately after the start of oscillation.

**Figure 5.5-1 Operation Immediately after Start of Oscillation**





## 5.6 Connecting Oscillator to External Clock

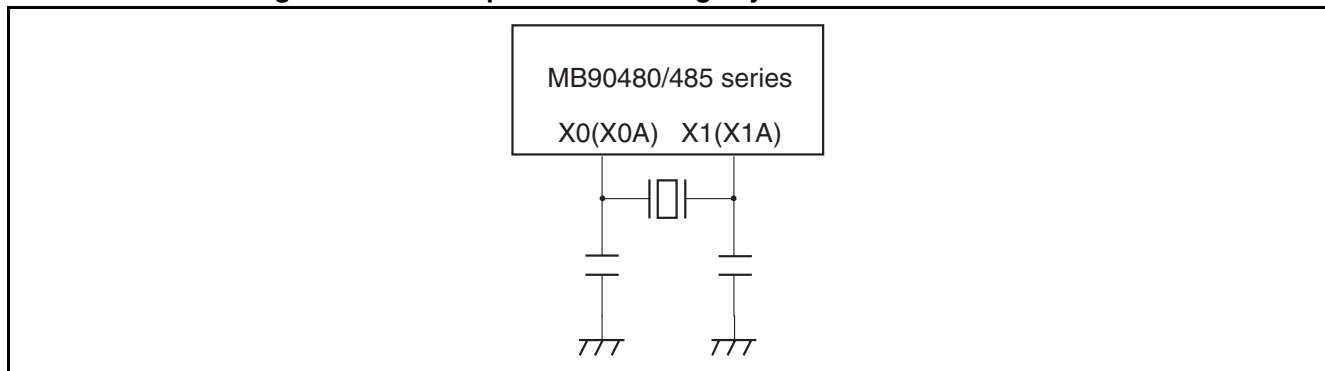
Devices in the MB90480/485 series contain a system clock generator circuit and generate clocks using an externally connected oscillator. Also, an external clock can be input to it.

### ■ Connection of Oscillator and External Clock

#### ○ Example of connecting crystal or ceramic oscillator

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

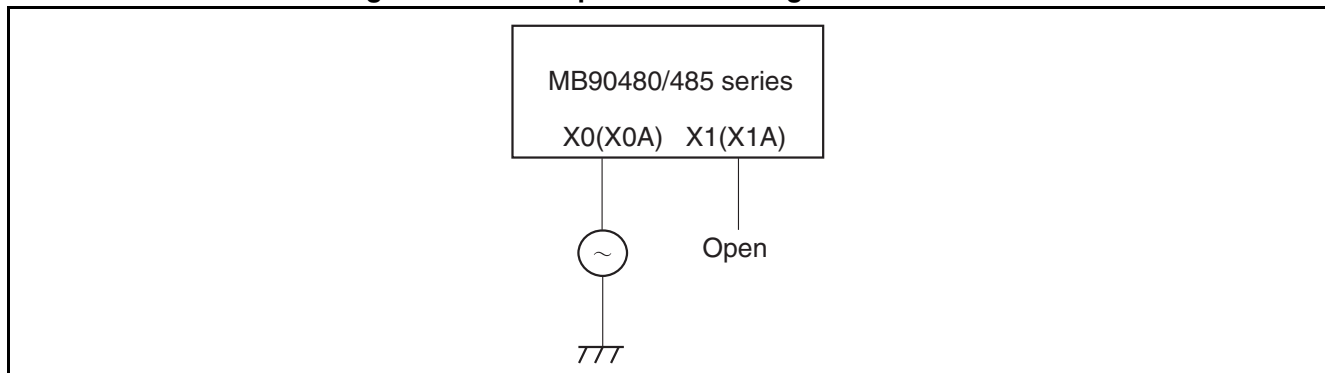
**Figure 5.6-1 Example of Connecting Crystal or Ceramic Oscillator**



#### ○ Example of connecting external clock

Connect an external clock to pin X0 and set up pin X1 to be open as shown in the example in the Figure 5.6-2.

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



# CHAPTER 6    LOW-POWER CONSUMPTION MODE

---

**This chapter explains the low-power consumption mode of the MB90480/485 series.**

---

- 6.1 Overview of Low-Power Consumption Mode
- 6.2 Block Diagram of Low-Power Consumption Control Circuit
- 6.3 Low-Power Consumption Mode Control Register (LPMCR)
- 6.4 CPU Intermittent Operation Mode
- 6.5 Standby Mode
- 6.6 State Transition Diagram
- 6.7 Pin State in Standby Mode, Hold, and Reset
- 6.8 Caution on Using Low-Power Consumption Mode

## 6.1 Overview of Low-Power Consumption Mode

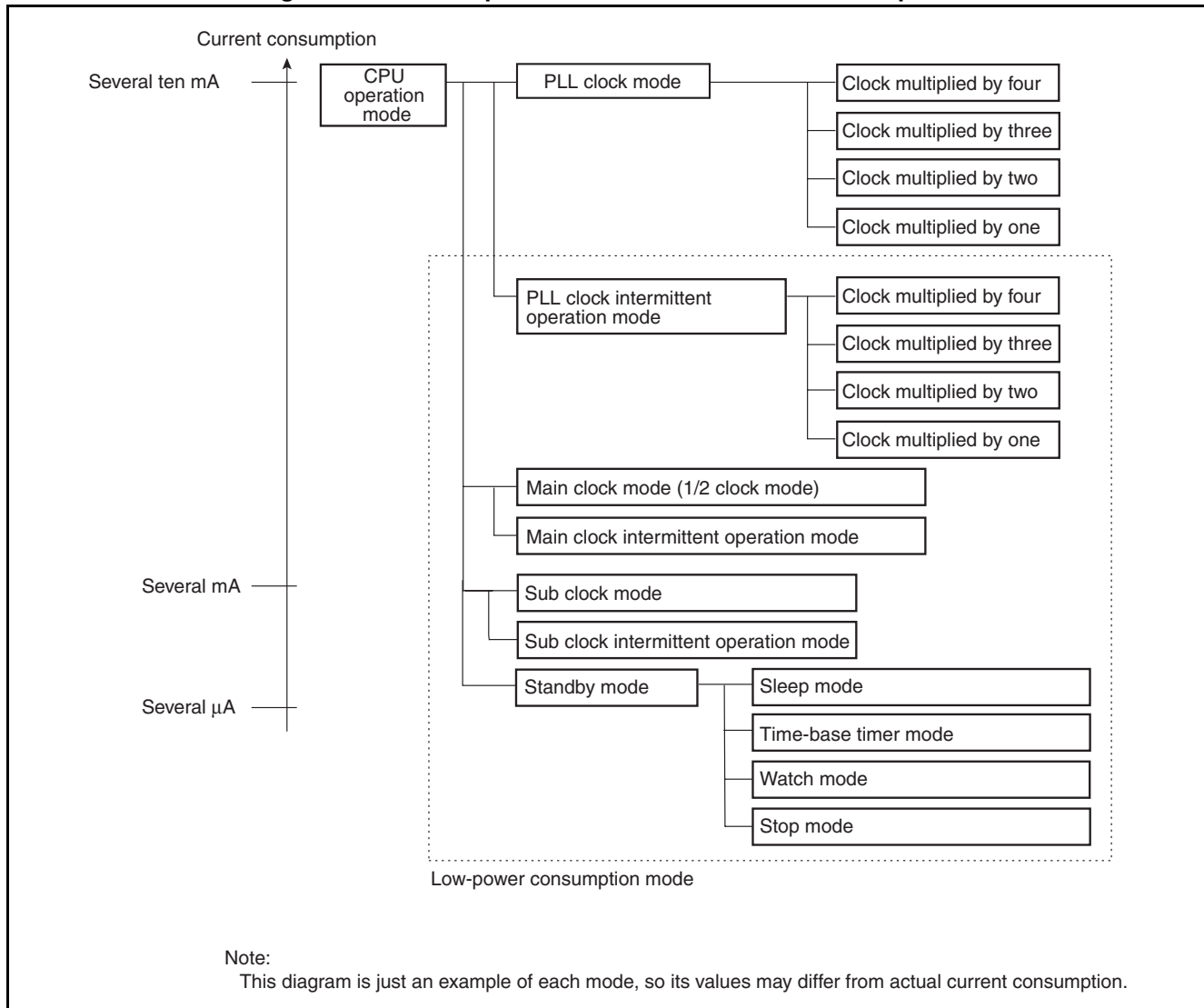
The following CPU operation modes are available on the MB90480/485 series devices by selecting a suitable operation clock and by controlling clock operation.

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

### ■ CPU Operation Mode and Current Consumption

Figure 6.1-1 illustrates the relationship between CPU operation mode and current consumption.

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



## ■ Clock Modes

### ○ Main clock mode

This mode operates the CPU and peripheral functions by using the clock of the oscillation clock (HCLK) divided by two. The PLL multiplier circuit stops its operation in the main clock mode.

### ○ Sub clock mode

This mode operates the CPU and peripheral functions by using the sub clock (SCLK). The main clock and the PLL multiplier circuit stop their operation in the sub clock mode.

#### Reference:

The clock mode includes the PLL clock mode except the low power consumption mode. Refer to Section "5.4 Clock Modes", for more information on clock modes.

## ■ CPU Intermittent Operation Mode

The CPU intermittent operation mode operates the CPU intermittently while supplying a high-speed clock to the peripheral functions, thereby reducing power consumption. This mode inputs an intermittent clock only to the CPU while the CPU accesses registers, built-in memory, peripheral functions, and external devices.

## ■ Standby Mode

The standby mode reduces current consumption by stopping supply of a clock to the CPU by using the low-power consumption control circuit (sleep mode), stopping supply of a clock to the CPU and peripheral functions (time-base timer mode), and stopping oscillation clocks (stop mode).

### ○ Sleep mode

The sleep mode stops the CPU operation clock in each clock mode. The CPU stops, and the peripheral functions operates with the clock before the sleep mode shifts.

The clock mode when shifting to sleep mode divides into the main sleep mode, PLL sleep mode, and the sub sleep mode.

### ○ Time-base timer mode

The time-base timer mode stops clocks and operations other than the oscillation clock, time-base timer, and watch timer. Functions other than the time-base timer and watch timer are stopped.

### ○ Watch mode

The watch mode operates only the watch timer. In this mode, only the sub clock operates. The main clock and PLL multiplier circuit are stopped.

### ○ Stop mode

The stop mode stops source oscillation, and all functions are stopped. In the stop mode, the oscillation clock stops and data can be retained with the lowest consumption of power.

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

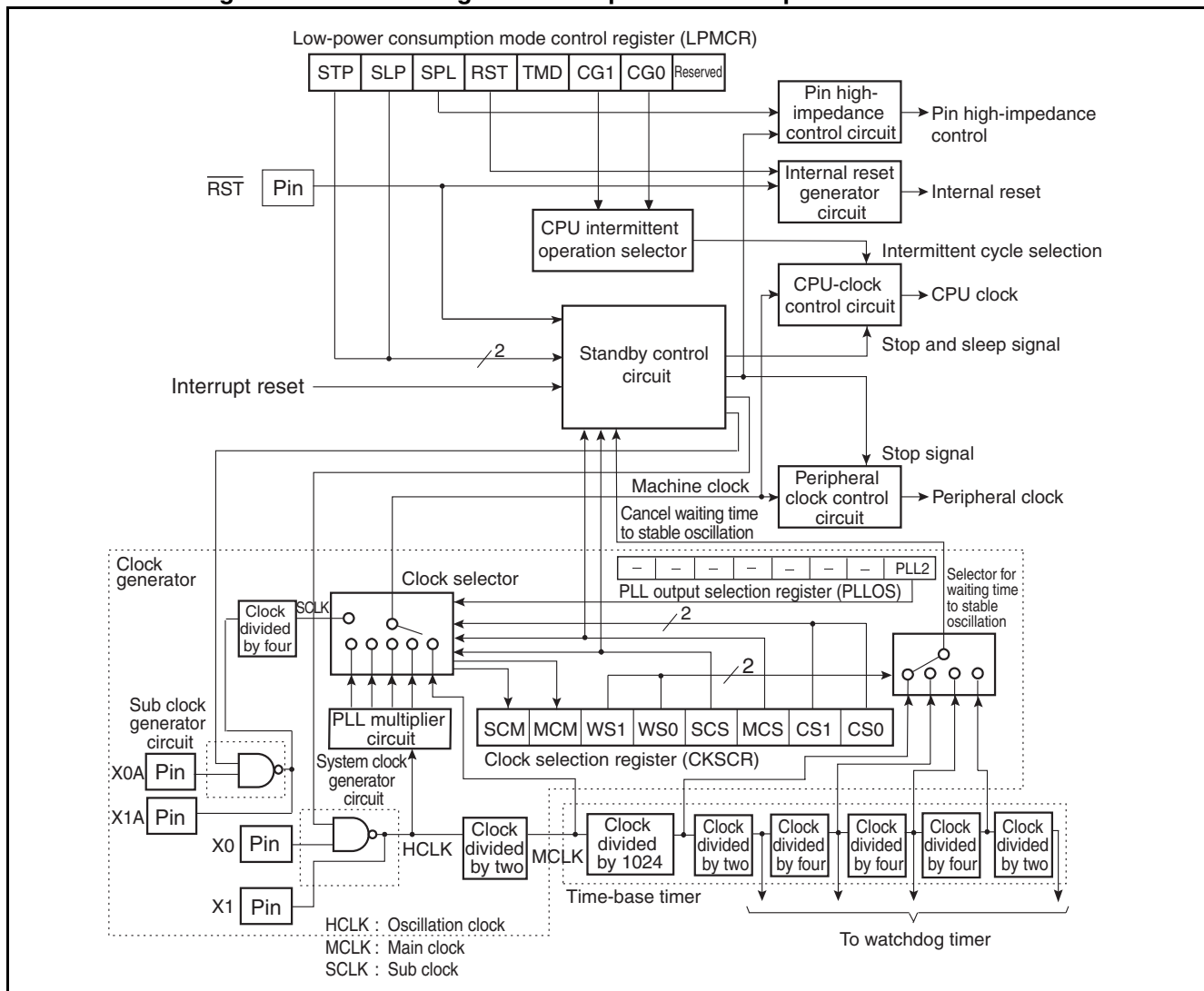
The low-power consumption control circuit is composed of the following seven blocks:

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

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

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

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



- **CPU intermittent operation selector**

The CPU intermittent operation selector selects the number of pause clocks in the CPU intermittent operation mode.

- **Standby control circuit**

The standby control circuit controls the CPU-clock control circuit and peripheral clock control circuit for resetting and changing to the low-power consumption mode.

- **CPU-clock control circuit**

The CPU-clock control circuit controls clocks supplied to the CPU.

- **Peripheral clock control circuit**

The peripheral clock control circuit controls clocks supplied to peripheral functions.

- **Pin high-impedance control circuit**

The pin high-impedance control circuit changes the states of external pins to high impedance in the time-base timer mode and stop mode. In the stop mode, the circuit isolates pull-up resistance with pins for which the pull-up option is selected.

- **Internal reset generator circuit**

The internal reset generator circuit generates an internal reset signal.

- **Low-power consumption mode control register (LPMCR)**

The low-power consumption mode control register (LPMCR) performs functions, such as resetting and changing to the standby mode and defining the CPU intermittent operation function.

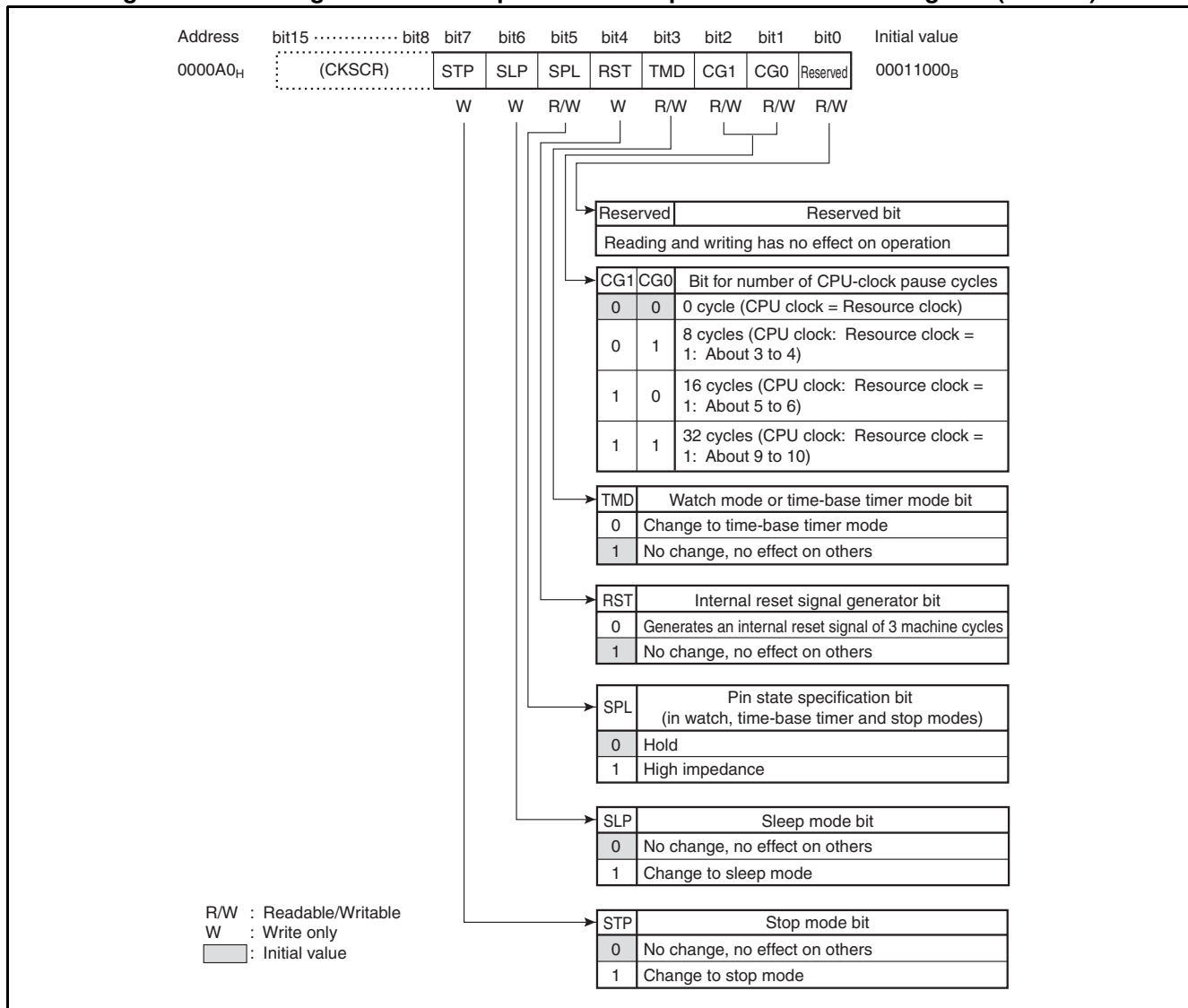
## 6.3 Low-Power Consumption Mode Control Register (LPMCR)

The low-power consumption mode control register (LPMCR) performs functions including changing the current mode to the low-power consumption mode, canceling from the low-power consumption mode, and specifying the number of CPU-clock pause cycles in the CPU intermittent operation mode.

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

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

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



**Table 6.3-1 Functions of Bits in Low-power Consumption Mode Control Register (LPMCR)**

| Bit name      |   | Function   |
|---------------|---|--|
| bit7          | STP:<br>Stop mode bit   | <p>This bit instructs a change to the stop mode.</p> <ul style="list-style-type: none"> <li>• Write "1" in this bit to change the mode to the stop mode.</li> <li>• Writing "0" in this bit does not affect operation.</li> <li>• Cleared to "0" by a reset or if an interrupt request is generated.</li> <li>• "0" is always read when this bit is read.</li> </ul>   |
| bit6          | SLP:<br>Sleep mode bit  | <p>This bit instructs a change to the sleep mode.</p> <ul style="list-style-type: none"> <li>• Write "1" in this bit to change the mode to the sleep mode.</li> <li>• Writing "0" in this bit does not affect operation.</li> <li>• Cleared to "0" by a reset or if an interrupt request is generated.</li> <li>• "0" is always read when this bit is read.</li> </ul>   |
| bit5          | SPL:<br>Pin state specification bit (in watch, time-base timer, and stop modes) | <p>This bit is effective only in the watch, time-base timer, and stop modes.</p> <ul style="list-style-type: none"> <li>• If this bit is "0", the levels of external pins are retained.</li> <li>• If this bit is "1", the levels of external pins are changed to high impedance.</li> <li>• Initialized to "0" when reset.</li> </ul>   |
| bit4          | RST:<br>Internal reset signal generator bit                                     | <p>This bit generates the software reset.</p> <ul style="list-style-type: none"> <li>• Write "0" in this bit to generate an internal reset signal of 3 machine cycles.</li> <li>• Writing "1" in this bit does not affect operation.</li> <li>• "1" is always read when this bit is read.</li> </ul>   |
| bit3          | TMD:<br>watch and time-base timer mode bit                                      | <p>This bit instructs a change to the watch or time-base timer mode.</p> <ul style="list-style-type: none"> <li>• Write "0" in this bit at the main clock or PLL clock mode to change the mode to the time-base timer mode.</li> <li>• Write "0" in this bit at the sub clock mode to change the mode to the watch mode.</li> <li>• Initialized to "1" by a reset or if an interrupt request is generated.</li> <li>• "1" is always read when this bit is read.</li> </ul> |
| bit2,<br>bit1 | CG1, CG0:<br>Bit for selecting number of CPU-clock pause cycles                 | <p>This bit specifies the number of pause cycles of the CPU clock in the CPU intermittent operation function.</p> <ul style="list-style-type: none"> <li>• Stops supply of CPU clocks for the specified number of cycles per instruction.</li> <li>• Capable selected from four clock numbers.</li> <li>• Initialized to "00<sub>B</sub>" by a reset.</li> </ul>   |
| bit0          | Reserved:<br>Reserved bit   | Reading and writing has no effect on operation.  |



## ■ Accessing Low-power Consumption Mode Control Register

Writing in the low-power consumption mode control register executes a change to the standby mode (stop, sleep, time-base timer and watch modes). Use the instructions listed in Table 6.3-2.

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

MOV LPMCR,#H'xx ; The low-power consumption mode transition instruction in Table 6.3-2.

|     |      |                            |
|-----|------|----------------------------|
| NOP |      |                            |
| NOP |      |                            |
| JMP | \$+3 | ; Jump to next instruction |

MOV A,#H'10 ; Any instruction

The devices does not guarantee its operation after releasing from the standby mode if you place an array of instructions other than the one enclosed in the line.

To access the low-power consumption mode control register (LPMCR) with C language, refer to "6.8 Caution on Using Low-Power Consumption Mode". When writing to the low-power consumption mode control register with a length of words, use even addresses only. Performing transition by using an odd address for writing may result in operation errors.

Any instruction may be used to control functions not listed in Table 6.3-1.

**Table 6.3-2 Instructions Used for Change to Low-power Consumption Mode**

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

## ■ Priority of STP, SLP, and TMD Bits

Requests are processed with the following order of priority in the event that stop mode, sleep mode, and time-base timer mode requests are issued simultaneously.

Stop-mode request > Time-base timer mode request > Sleep mode request

## 6.4 CPU Intermittent Operation Mode

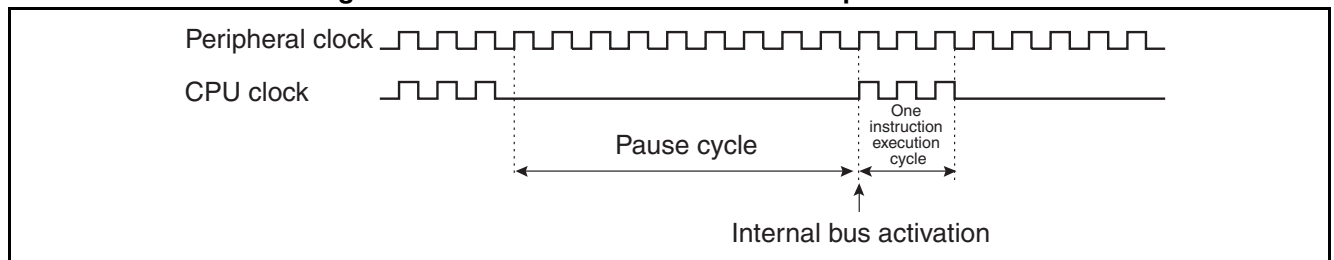
The CPU intermittent operation mode reduces power consumption by intermittently operating the CPU while operating external buses and peripheral functions at high speeds.

### ■ CPU Intermittent Operation Mode

To delay activation of the internal bus cycle, the CPU intermittent operation mode stops clocks supplied to the CPU for a preset period for each instruction during access to registers, embedded memory (ROM or RAM), I/O, peripheral functions, and external buses. Low-power consumption processing is possible by lowering the CPU execution speed while high-speed peripheral clocks are supplied to peripheral functions.

- Select the number of clock pause cycles supplied to the CPU using a bit for selecting the number of CPU-clock pause cycles (CG1 or CG0) of the low-power consumption mode control register (LPMCR).
- Use the same clock as that for the peripheral functions when operating external buses.
- The instruction execution time when the CPU intermittent operation mode is set can be calculated by dividing the number of instruction executions for accessing registers, embedded memory, embedded peripheral functions, and external buses by the number of pauses cycles. The correction value thus obtained is added to the usual execution time. Figure 6.4-1 illustrates operation clocks in the CPU intermittent operation mode.

**Figure 6.4-1 Clocks in CPU Intermittent Operation Mode**



## 6.5 Standby Mode

The standby mode is divided into four modes, namely, the sleep (PLL sleep, main sleep, and sub sleep), time-base timer, watch, and stop modes.

### ■ Operational States in Standby Mode

Table 6.5-1 lists operational states in the standby mode.

**Table 6.5-1 Operational States in Standby Mode**

| Standby mode         |                                | Change condition              | Main clock   | Sub clock    | Machine clock | CPU        | Peripheral   | Pin          | Cancellation method |
|----------------------|--------------------------------|-------------------------------|--------------|--------------|---------------|------------|--------------|--------------|---------------------|
| Sleep mode           | PLL sleep mode                 | SCS = 1<br>MCS = 0<br>SLP = 1 | In operation |              | In operation  | Stopped    | In operation | In operation | Reset or interrupt  |
|                      | Main sleep mode                | SCS = 1<br>MCS = 1<br>SLP = 1 |              |              |               |            |              |              |                     |
|                      | Sub sleep mode                 | SCS = 0<br>SLP = 1            | Stopped      |              |               |            |              |              |                     |
| Time-base timer mode | Time-base timer mode (SPL = 0) | SCS = 1<br>TMD = 0            | In operation | In operation | Stopped       | Stopped *1 | Hold         |              |                     |
|                      | time-base timer mode (SPL = 1) | SCS = 1<br>TMD = 0            |              |              |               |            | Hi-Z         |              |                     |
| Watch mode           | Watch mode (SPL = 0)           | SCS = 0<br>TMD = 0            |              |              |               | Stopped    | Stopped *2   | Hold         |                     |
|                      | Watch mode (SPL = 1)           | SCS = 0<br>TMD = 0            |              | Hi-Z         |               |            |              |              |                     |
| Stop mode            | Stop mode (SPL = 0)            | STP = 1                       | Stopped      |              | Stopped       |            | Hold         |              |                     |
|                      | Stop mode (SPL = 1)            | STP = 1                       |              |              |               | Hi-Z       |              |              |                     |

\*1: The time-base timer and watch timer are operating.

\*2: The watch timer is operating

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

SLP: Sleep mode bit of low-power consumption mode control register (LPMCR)

STP: Stop mode bit of low-power consumption mode control register (LPMCR)

TMD: Watch/time-base timer mode bit of low-power consumption mode control register (LPMCR)

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

SCS: Machine clock selection bit (Sub) of the clock selection register (CKSCR)

Hi-Z: High impedance

RST: External-reset pin

## 6.5.1 Sleep Mode

---

The sleep mode stops CPU operation clocks, allowing devices other than the CPU to continue operation.

---

### ■ Change to Sleep Mode

Writing "1" in the sleep mode bit (SLP), "1" in the watch/time-base timer mode bit (TMD), and "0" in the stop mode bit (STP) of the low-power consumption mode control register (LPMCR) changes the mode to the sleep mode.

---

Note:

If "1" is simultaneously written in the SLP and STP bits of the LPMCR register, the STP bit has the priority and the device is changed to the stop mode.

If writing "1" in the SLP bit and writing "0" in the TMD bit of the low-power consumption mode control register are performed at the same time, the TMD bit has the priority and the device is changed to the time-base timer mode or watch mode.

---

#### ○ Data hold function

This function in the sleep mode holds data of the internal RAM and dedicated registers such as an accumulator.

#### ○ Hold function

The external bus hold function operates in the sleep mode. A hold state is set if a hold request is issued.

#### ○ Operation during interrupt request

The sleep mode is not set if an interrupt request is issued while "1" is written in the SLP bit of the LPMCR register. The CPU executes a next instruction if an interrupt request is not accepted. If the CPU can accept an interrupt request, the request is immediately branched to an interrupt processing routine.

#### ○ Pin state

In the sleep mode, the previous states are maintained except for pins used for bus input and output or for bus control.

## ■ Canceling the Sleep Mode

The low-power consumption control circuit cancels the sleep mode by input of a reset or by an interrupt.

### ○ Restore by a reset

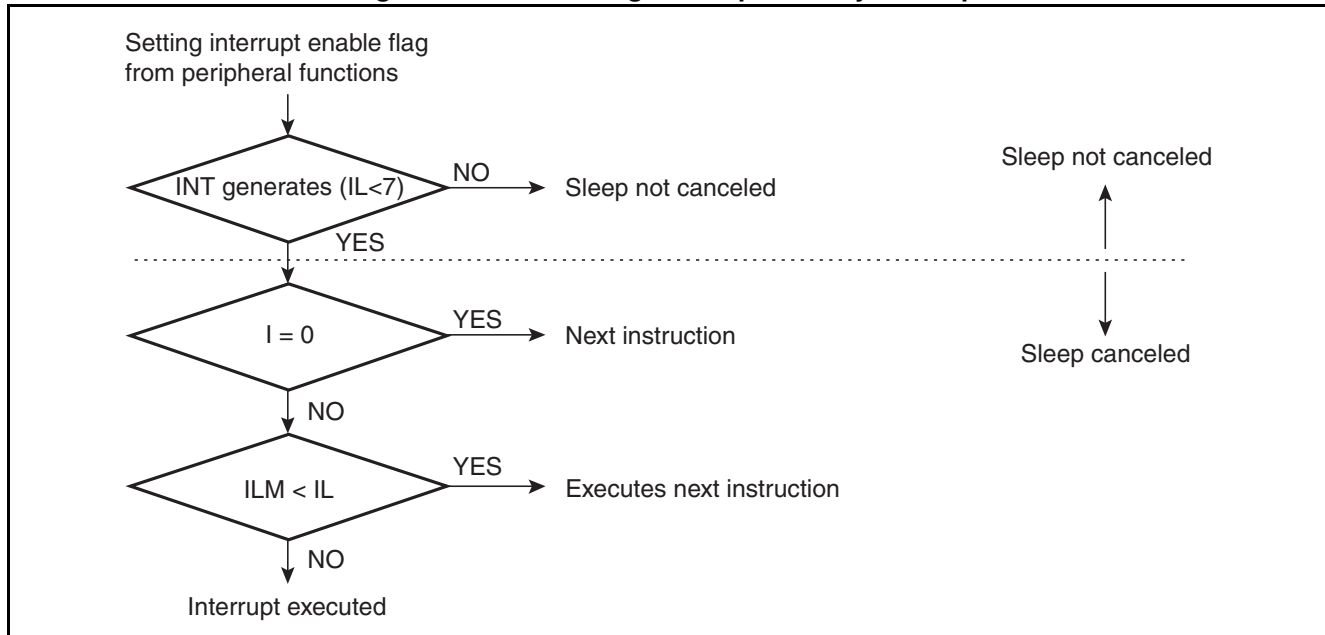
Reset initializes to the main clock mode.

### ○ Restore by interrupt

The sleep mode is canceled if an interrupt request whose interrupt level is higher than 7 is generated in a peripheral circuit, etc., in the sleep mode. After the sleep mode is canceled, the interrupt is processed with the same method as for ordinary interrupt processing. If interrupts are accepted by setting the I-flag of the condition code register (CCR), interrupt level mask register (ILM), or the interrupt control register (ICR), then the CPU executes the interrupts. If the interrupts cannot be accepted, the CPU continues processing beginning from an instruction next to the instruction specifying the sleep mode.

Figure 6.5-1 illustrates the canceling of the sleep mode by an interrupt.

**Figure 6.5-1 Canceling of Sleep Mode by Interrupt**



#### Note:

When executing an interrupt, an instruction next to the instruction that specified the sleep mode is normally executed first before an interrupt request is processed. If a change to the sleep mode occurs at the same time as an external bus hold request is received, an interrupt may be executed first before the next instruction is executed.

## 6.5.2 Time-Base Timer Mode

---

**The time-base timer mode stops operations except for source oscillation, time-base timer and watch timer. All functions except the time-base timer and watch timer are stopped.**

---

### ■ Change to Time-base Timer Mode

To change the mode to the time-base timer mode, write "0" in watch/time-base timer mode bit (TMD) of the low-power consumption mode control register (LPMCR) in the PLL clock mode or the main clock mode (sub clock display bit (SCM) = 1 of the clock selection register (CKSCR)).

#### ○ Data hold function

This function in the time-base timer mode holds data of the internal RAM and dedicated registers such as an accumulator.

#### ○ Hold function

In the time-base timer mode, the external bus hold function is stopped and hold requests cannot be accepted even if they are input. If a hold request is input during a change to the time-base timer mode, the level of the HAK signal may not change to "L" while the bus is set to the high-impedance state.

#### ○ Operation during interrupt request

The time-base timer mode is not set if an interrupt request is issued while "0" is written to the TMD bit of the low-power consumption mode control register (LPMCR).

#### ○ Pin state

Pin state specification bit (SPL) of the LPMCR register can control whether to maintain the state of an external pin in the time-base timer mode in the previous state or in the high-impedance state.

### ■ Canceling the Time-base Timer Mode

The low-power consumption control circuit cancels the time-base timer mode by input of a reset or by an interrupt.

#### ○ Return by external reset

External reset initializes to the main clock mode.

#### ○ Return by interrupt

The time-base timer mode is canceled by the low-power consumption control circuit if an interrupt request whose interrupt level is higher than 7 (other than IL2, IL1, and IL0=111<sub>B</sub> of the interrupt control register (ICR)) is generated in a peripheral circuit, etc., in the time-base timer mode. After the time-base timer mode is canceled, interrupts are processed with the same method as for ordinary interrupt processing. If interrupts are accepted by setting the I-flag of the condition code register (CCR), interrupt level mask register (ILM), or the interrupt control register (ICR), then the CPU executes the interrupts. If an interrupt cannot be accepted, the CPU continues processing beginning from an instruction that was processed before the time-base timer mode was set.

Note:

When executing an interrupt, an instruction next to the instruction specifying the time-base timer mode is normally executed first before an interrupt request is processed. If a change to the time-base timer mode occurs at the same time as an external bus hold request is received, an interrupt may be executed first before the next instruction is executed.

---

### 6.5.3 Watch Mode

---

**The watch mode stops operations other than those of the sub clock and watch timer. Almost all functions on the chip are stopped.**

---

#### ■ Change to Watch Mode

To change the mode to the watch mode, write "0" in watch/time-base timer mode bit (TMD) of the low-power consumption mode control register (LPMCR) in the sub clock mode (sub clock display bit (SCS) = 0 of the clock selection register (CKSCR)).

##### ○ Data hold function

This function in the watch mode holds data of the internal RAM and dedicated registers such as an accumulator.

##### ○ Hold function

In the watch mode, the external bus hold function is stopped and hold requests cannot be accepted even if they are input.

---

Note:

If a hold request is input during a change to the watch mode, the level of the  $\overline{HAK}$  signal may not change to "L" while the bus is set to the high-impedance state.

---

##### ○ Operation during interrupt request

The watch mode is not set if an interrupt request is issued while "0" is set in the TMD bit of the LPMCR register.

##### ○ Pin state setting

Pin state specification bit (SPL) of the LPMCR register can control whether to maintain the state of an external pin in the watch mode in the previous state or in the high-impedance state.

#### ■ Canceling the Watch Mode

The low-power consumption control circuit cancels the watch mode by input of a reset or by an interrupt.

##### ○ Return by a reset

When the watch mode is canceled by a reset factor, the watch mode is canceled first, and a reset state for standing by for stable oscillation is set. The sequence for resetting is executed after the end of the oscillation stabilization wait time.



○ Return by interrupt

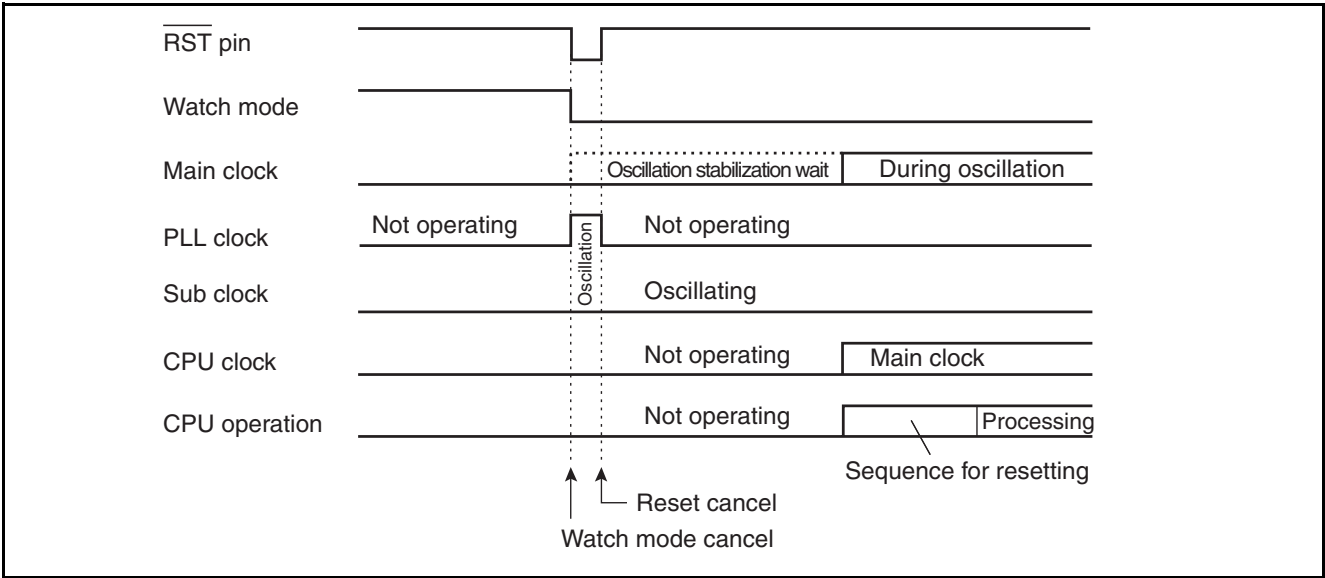
The watch mode is canceled by the low-power consumption control circuit if an interrupt request whose interrupt level is higher than 7 (other than IL2, IL1, and IL0=111<sub>B</sub> of the interrupt control register (ICR)) is generated in a peripheral circuit, etc., in the watch mode. The mode immediately changes to the sub clock mode. After the change to the sub clock mode, interrupts are processed with the same method as for ordinary interrupt processing. If interrupts are accepted by setting the I-flag of the condition code register (CCR), interrupt level mask register (ILM), or the interrupt control register (ICR), then the CPU executes the interrupts. If an interrupt cannot be accepted, the CPU continues processing beginning from an instruction next to the instruction that was processed before the watch mode was set.

Note:

When executing an interrupt, an instruction next to the instruction specifying the watch mode is normally executed first before an interrupt request is processed. If a change to the watch mode occurs at the same time as an external bus hold request is received, an interrupt may be executed first before the next instruction is executed.

Figure 6.5-2 illustrates the cancel operation of the watch mode.

Figure 6.5-2 Cancel Operation of Watch Mode (External Reset)



## 6.5.4 Stop Mode

---

The stop mode stops source oscillation and stops all functions, thereby enabling retention of data with the lowest consumption of power.

---

### ■ Change to Stop Mode

Write "1" in the stop mode bit (STP) of the low-power consumption mode control register (LPMCR) to change the mode to the stop mode.

#### ○ Data hold function

This function in the stop mode holds data of the internal RAM and dedicated registers such as an accumulator.

#### ○ Hold function

In the stop mode, the external bus hold function is stopped and hold requests cannot be accepted even if they are input. If a hold request is input during a change to the stop mode, the level of the  $\overline{\text{HAK}}$  signal may not change to "L" while the bus is set to the high-impedance state.

#### ○ Operation during interrupt request

The stop mode is not set if an interrupt request is issued while "1" is set in the STP bit of the LPMCR register.

#### ○ Pin state setting

Pin state specification bit (SPL) of the LPMCR register can specify whether to maintain the state of an external pin in the stop mode in the previous state or in the high-impedance state.

### ■ Canceling the Stop Mode

The low-power consumption control circuit releases the stop mode when a reset is input or an interrupt occurs. Because the oscillation clock (HCLK) and sub clock (SCLK) are halted, the stop mode is released after the oscillation stabilization wait interval of the main clock or sub clock.

#### ○ Restore by a reset

When the stop mode is canceled by a reset factor, the stop mode is canceled first and the reset state standing by for stable oscillation is set. The sequence for resetting is executed after the end of the oscillation stabilization wait time.

○ Restore by interrupt

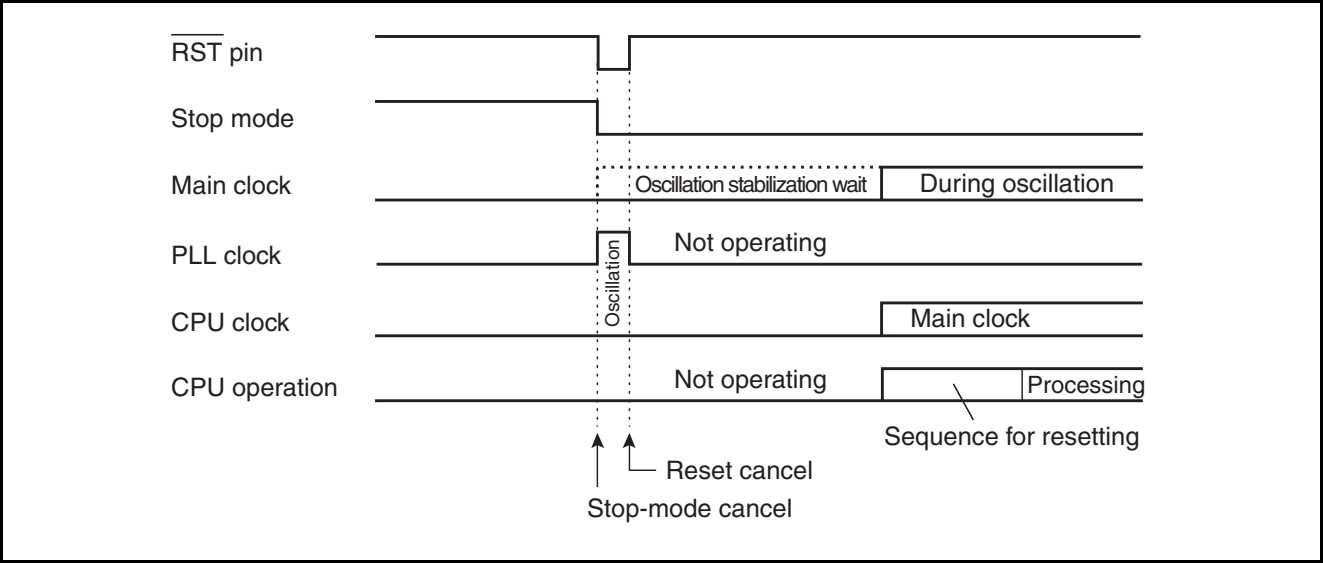
The stop mode is canceled by the low-power consumption control circuit if an interrupt request whose interrupt level is higher than 7 (other than IL2, IL1, and IL0=111<sub>B</sub> of the interrupt control register (ICR)) is generated in a peripheral circuit, etc., in the stop mode. After the stop mode is canceled, interrupts are processed with the same method as for ordinary interrupt processing, following the elapse of the oscillation stabilization wait time for the main clock specified by the selection bits (WS1, WS0) for the oscillation stabilization wait time of the clock selection register (CKSCR). If the interrupts are accepted by setting the I-flag of the condition code register (CCR), interrupt level mask register (ILM) or the interrupt control register (ICR), the CPU executes interrupts. If an interrupt cannot be accepted, the CPU continues processing beginning from an instruction next to the instruction that was processed before the stop mode was set.

Notes:

- When executing an interrupt, an instruction next to the instruction that specified the stop mode is normally executed first before an interrupt request is processed. If a change to the stop mode occurs at the same time as an external bus hold request is received, an interrupt may be executed first before the next instruction is executed.
- In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires 2<sup>14</sup>/HCLK or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10<sub>B</sub>" or "11<sub>B</sub>".

Figure 6.5-3 shows the cancel operation of the stop mode.

Figure 6.5-3 Cancel Operation of Stop Mode (External Reset)



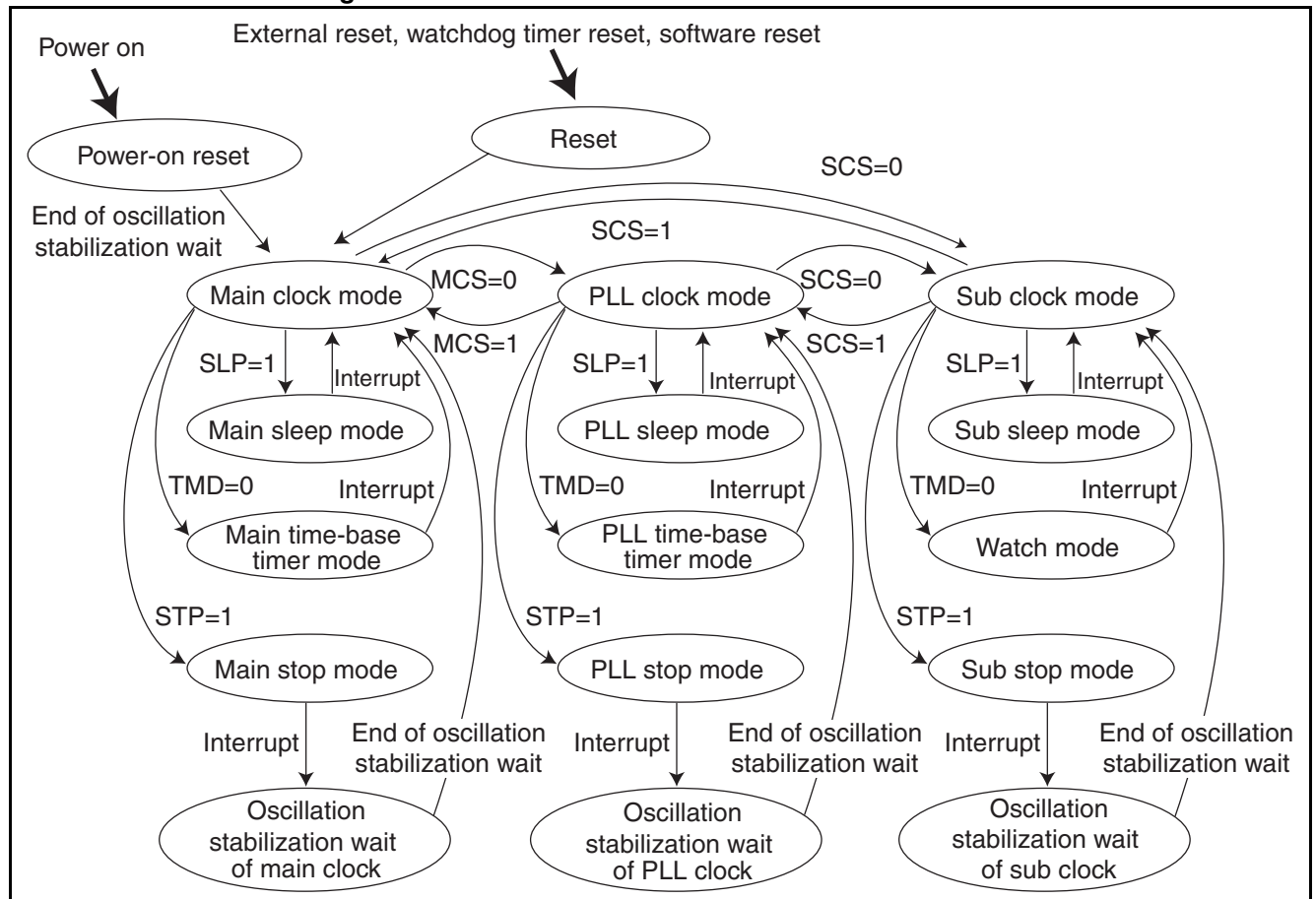
## 6.6 State Transition Diagram

This section explains the transition of operational states for the MB90480/485 series and describes the transition conditions.

### ■ State Transition Diagram

Figure 6.6-1 illustrates the transition of operational states for the MB90480/485 series and the transition conditions.

**Figure 6.6-1 State Transition and Transition Conditions**



## ■ Operational State in Low-power Consumption Mode

Table 6.6-1 lists operational states in the low-power consumption mode.

**Table 6.6-1 Operational States in Low-power Consumption Mode**

| Operational state                                | Main clock    | Sub clock     | PLL clock     | CPU           | Peripheral    | Watch         | Time-base timer | Clock source |
|--|---------------|---------------|---------------|---------------|---------------|---------------|-----------------|--------------|
| PLL clock mode                                   | Operating     | Operating     | Operating     | Operating     | Operating     | Operating     | Operating       | PLL clock    |
| PLL sleep mode                                   |               |               |               | Not operating |               |               |                 |              |
| PLL time-base timer mode                         |               |               |               |               |               |               |                 |              |
| PLL stop mode                                    | Not operating | Not operating | Not operating |               | Not operating | Not operating | Not operating   |              |
| Standby for stable oscillation of PLL clock mode | Operating     | Operating     | Operating     | Operating     |               | Operating     |                 |              |
| Main clock mode                                  | Operating     | Operating     | Not operating | Operating     | Operating     | Operating     | Operating       | Main clock   |
| Main sleep mode                                  |               |               |               | Not operating |               |               |                 |              |
| Main time-base timer mode                        |               |               |               |               |               |               |                 |              |
| Main stop mode                                   | Not operating | Not operating |               |               | Not operating | Not operating | Not operating   |              |
| Standby for stable oscillation of main clock     | Operating     | Operating     |               | Operating     |               |               |                 |              |
| Sub clock mode                                   | Not operating | Operating     | Not operating | Operating     | Operating     | Operating     | Not operating   | Sub clock    |
| Sub sleep mode                                   |               |               |               | Not operating |               |               |                 |              |
| Watch mode                                       |               | Not operating |               |               |               |               |                 |              |
| Sub clock stop mode                              |               |               |               |               | Not operating |               |                 |              |
| Standby for stable oscillation of sub clock      |               |               |               | Operating     |               |               |                 |              |
| Power-on reset                                   | Operating     | Operating     | Not operating | Not operating | Not operating | Operating     | Operating       | Main clock   |
| Reset  |               |               | Operating     |               |               |               |                 |              |

## 6.7 Pin State in Standby Mode, Hold, and Reset

The states of the pins in the standby mode and in the hold and reset states are described for each memory access mode.

### ■ Pin State in Single Chip Mode

Table 6.7-1 lists the pin states in the single-chip mode.

**Table 6.7-1 Pin States in Single Chip Mode**

| Pin name   | In sleep state                     | When stopped  |                                 | In hold state             | When reset                     |
|------------|------------------------------------|---|---------------------------------|---------------------------|--------------------------------|
|            |                                    | SPL = 0   | SPL = 1                         |                           |                                |
| P07 to P00 | Maintains the previous state<br>*1 | Input cutoff/<br>Maintains the previous state<br>*1, *2 | Input cutoff/<br>Output Hi-Z *2 | This state does not exist | Input disabled/<br>Output Hi-Z |
| P17 to P10 |                                    |   |                                 |                           |                                |
| P27 to P20 |                                    |   |                                 |                           |                                |
| P37 to P30 |                                    |   |                                 |                           |                                |
| P47 to P40 |                                    |   |                                 |                           |                                |
| P57 to P50 |                                    |   |                                 |                           |                                |
| P67 to P60 |                                    |   |                                 |                           |                                |
| P77 to P70 |                                    |   |                                 |                           |                                |
| P97 to P90 |                                    |   |                                 |                           |                                |
| PA3 to PA0 |                                    |   |                                 |                           |                                |
| P87 to P80 | Input enabled *3                   | Input enabled *3  |                                 |                           | Input disabled                 |

\*1: The state output immediately before this mode was set is output as it is. If it is input, input is disabled.

"Input disabled" means that operations of input gates located very close to the pins are enabled, but pin states cannot be accepted in internal operations because internal circuits are not operating.

\*2: In the state of "Input cutoff", input A is masked and "L" level is transmitted internally. "Output Hi-Z" means that the pin-drive transistors are disabled and the pins are set to the high-impedance state.

\*3: Same as in other ports when used in the output state. "Input enabled" means that input functions are ready and require pull up/pull down or external input.

## ■ Pin States in External Bus 16-bit Data Bus Mode and Multiplex 16-bit External Bus Mode

Table 6.7-2 summarizes pin states in the external bus 16-bit data bus mode and multiplex 16-bit external bus mode.

**Table 6.7-2 Pin States in External Bus 16-bit Data Bus Mode and Multiplex 16-bit External Bus Mode**

| Pin name                  | In sleep state                           | When stopped   |                                 | In hold state                            | When reset                           | Internal ROM access immediately after reset cancellation | Internal ROM access after external ROM access |
|---------------------------|--|--|---------------------------------|--|--------------------------------------|--|---|
|                           |  | SPL = 0  | SPL = 1                         |  |                                      |  |   |
| P07 to P00 (AD07 to AD00) | Input disabled/<br>Output Hi-Z           | Input cutoff/<br>Output Hi-Z                         | Input cutoff/<br>Output Hi-Z *5 | Input disabled/<br>Output Hi-Z           | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |
| P17 to P10 (AD15 to AD08) |  |  |                                 |  |                                      |  |   |
| P27 to P20 (A23 to A16)   | Output state *1, *2                      | Output state *1, *2                                  |                                 | Input disabled/<br>Output Hi-Z *2        | Output state *1                      | Output state *1  | Maintains the previous address                |
| P57 (CLK)                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output state *1, *2               |                                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output enabled *3 | CLK output   | CLK output                                    |
| P56 (RDY)                 | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4, *5 |                                 | Input disabled *2                        | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |
| P55 (HAK)                 |  |  |                                 | "L" output                               |                                      |  |   |
| P54 (HRQ)                 |  |  |                                 | "H" input                                |                                      |  |   |
| P53 (WRH)                 | "H" output *2                            | "H" output *2  |                                 | Input disabled/<br>Output Hi-Z *2        | "H" output                           | "H" output   | "H" output                                    |
| P52 (WRL)                 |  |  |                                 |  |                                      |  |   |
| P51 (RD)                  | "H" output                               | "H" output   |                                 | Input disabled/<br>Output Hi-Z           | Output enabled *2                    | "L" output   | "L" output                                    |
| P50 (ALE)                 | "L" output                               | "L" output   |                                 |  |                                      |  |   |
| P37 to P30                | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4     |                                 | Maintains the previous state *4          | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |
| P47 to P40                |  |  |                                 |  |                                      |  |   |
| P67 to P60                |  |  |                                 |  |                                      |  |   |
| P77 to P70                |  |  |                                 |  |                                      |  |   |
| P97 to P91                |  |  |                                 |  |                                      |  |   |
| PA3 to PA0                |  |  |                                 |  |                                      |  |   |
| P90 (CS0)                 |  |  |                                 |  | "H" output                           |  |   |
| P87 to P80                | Input enabled *6                         | Input enabled *6                                     | Input enabled *6                |  | Input disabled                       |  |   |

\*1: "Output state" means that the pin-drive transistors are set in a drive-enabled state, but internal circuits are stopped so that a fixed value, "H" or "L", is output. When internal peripheral circuits are in operation and output functions are used, output varies except when reset. Output does not vary during a reset.

\*2: The previous values are retained if used as an output port.

\*3: "Output enabled" means that the pin-drive transistors are set in a drive state and that internal circuit operations are enabled. Operational states are therefore conveyed on the pins.

\*4: The state output immediately before this mode was set is output as it is. If it is input, input is disabled.

"Input disabled" means that operations of input gates located very close to the pins are enabled, but pin states cannot be accepted in internal operations because internal circuits are not operating.

\*5: In the state of "Input cutoff", input A is masked and "L" level is transmitted internally. "Output Hi-Z" means that the pin-drive transistors are disabled and the pins are set to the high-impedance state.

\*6: Same as in other ports when used in the output state. "Input enabled" means that input functions are ready and require pull up/pull down or external input.

## ■ Pin States in External Bus 8-bit Data Bus Mode and Multiplex 8-bit External Bus Mode

Table 6.7-3 lists pin states in the external bus 8-bit data bus mode and multiplex 8-bit external bus mode.

**Table 6.7-3 Pin States in External Bus 8-bit Data Bus Mode and Multiplex 8-bit External Bus Mode**

| Pin name                  | In sleep state                           | When stopped                                     |                                 | In hold state                            | When reset                           | Internal ROM access immediately after reset cancellation | Internal ROM access after external ROM access |
|---------------------------|--|--|---------------------------------|--|--------------------------------------|--|---|
|                           |  | SPL = 0  | SPL = 1                         |  |                                      |  |   |
| P07 to P00 (AD07 to AD00) | Input disabled/<br>Output Hi-Z           | Input cutoff/<br>Output Hi-Z                     | Input cutoff/<br>Output Hi-Z *5 | Input disabled/<br>Output Hi-Z           | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |
| P17 to P10 (AD15 to AD08) | Output state *1                          | Output state *1                                  |                                 |  | Output state *1                      | Output state *1  | Maintains the previous address                |
| P27 to P20 (A23 to A16)   | Output state *1, *2                      | Output state *1, *2                              |                                 | Input disabled/<br>Output Hi-Z *2        |                                      |  |   |
| P57 (CLK)                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output state *1, *2           |                                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output enabled *3 | CLK output   | CLK output                                    |
| P56 (RDY)                 | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4 |                                 | Input disabled *2                        | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |
| P55 (HAK)                 |  |  |                                 | "L" output                               |                                      |  |   |
| P54 (HRQ)                 |  |  |                                 | "H" input                                |                                      |  |   |
| P53 (WRH)                 |  |  |                                 | Maintains the previous state *4          |                                      |  |   |
| P52 (WRL)                 | "H" output *4                            | "H" output *4                                    |                                 | Input disabled/<br>Output Hi-Z *2        | "H" output                           | "H" output   | "H" output                                    |
| P51 (RD)                  | "H" output                               | "H" output                                       |                                 | Input disabled/<br>Output Hi-Z           |                                      |  |   |
| P50 (ALE)                 | "L" output                               | "L" output                                       |                                 | "L" output                               | "L" output                           | "L" output   | "L" output                                    |
| P37 to P30                | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4 |                                 | Maintains the previous state *4          | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |
| P47 to P40                |  |  |                                 |  |                                      |  |   |
| P67 to P60                |  |  |                                 |  |                                      |  |   |
| P77 to P70                |  |  |                                 |  | "H" output                           |  |   |
| P97 to P91                |  |  |                                 |  |                                      |  |   |
| PA3 to PA0                |  |  |                                 |  | Input disabled                       |  |   |
| P90 (CS0)                 |  |  |                                 |  |                                      |  |   |
| P87 to P80                | Input enabled *6                         | Input enabled *6                                 | Input enabled *6                | Input disabled                           |                                      |  |   |

\*1: "Output state" means that the pin-drive transistors are set in a drive-enabled state, but internal circuits are stopped so that a fixed value, "H" or "L", is output. When internal peripheral circuits are in operation and output functions are used, output varies except when reset. Output does not vary during a reset.

\*2: The previous values are retained if used as an output port.

\*3: "Output enabled" means that the pin-drive transistors are set in a drive state and that internal circuit operations are enabled. Operational states are therefore conveyed on the pins.

\*4: The state output immediately before this mode was set is output as it is. If it is input, input is disabled.

"Input disabled" means that operations of input gates located very close to the pins are enabled, but pin states cannot be accepted in internal operations because internal circuits are not operating.

\*5: In the state of "Input cutoff", input A is masked and "L" level is transmitted internally. "Output Hi-Z" means that the pin-drive transistors are disabled and the pins are set to the high-impedance state.

\*6: Same as in other ports when used in the output state. "Input enabled" means that input functions are ready and require pull up/pull down or external input.



## ■ Pin States in External Bus 16-bit Data Bus Mode and Non-multiplex 16-bit External Bus Mode

Table 6.7-4 lists pin states in the external bus 16-bit data bus mode and non-multiplex 16-bit external bus mode.

**Table 6.7-4 Pin States in External Bus 16-bit Data Bus Mode and Non-multiplex 16-bit External Bus Mode**

| Pin name                  | In sleep state                           | When stopped                                     |                                 | In hold state                            | When reset                           | Internal ROM access immediately after reset cancellation | Internal ROM access after external ROM access |                                |
|---------------------------|--|--|---------------------------------|--|--------------------------------------|--|---|--------------------------------|
|                           |  | SPL = 0  | SPL = 1                         |  |                                      |  |   |                                |
| P07 to P00 (AD07 to AD00) | Input disabled/<br>Output Hi-Z           | Input cutoff/<br>Output Hi-Z                     | Input cutoff/<br>Output Hi-Z *5 | Input disabled/<br>Output Hi-Z           | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |                                |
| P17 to P10 (AD15 to AD08) |  |  |                                 |  |                                      |  |   |                                |
| P37 to P30 (A07 to A00)   | Output state *1                          | Output state *1                                  |                                 |  |                                      | Output state *1  | Output state *1                               | Maintains the previous address |
| P47 to P40 (A15 to A08)   |  |  |                                 |  |                                      |  |   |                                |
| P27 to P20 (A23 to A16)   | Output state *1, *2                      | Output state *1, *2                              |                                 | Input disabled/<br>Output Hi-Z *2        |                                      |  |   |                                |
| P57 (CLK)                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output state *1, *2           |                                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output enabled *3 | CLK output   | CLK output                                    |                                |
| P56 (RDY)                 | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4 |                                 | Input disabled *2                        | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |                                |
| P55 (HAK)                 |  |  |                                 | "L" output                               |                                      |  |   |                                |
| P54 (HRQ)                 |  |  |                                 | "H" input                                |                                      |  |   |                                |
| P53 (WRH)                 | "H" output *4                            | "H" output *4                                    |                                 | Input disabled/<br>Output Hi-Z *2        | "H" output                           | "H" output   | "H" output                                    |                                |
| P52 (WRL)                 |  |  |                                 |  |                                      |  |   |                                |
| P51 (RD)                  | "H" output                               | "H" output                                       |                                 | Input disabled/<br>Output Hi-Z           | Output enabled *3                    | "L" output   | "L" output                                    |                                |
| P50 (ALE)                 | "L" output                               | "L" output                                       |                                 |  |                                      |  |   |                                |
| P67 to P60                | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4 |                                 | Maintains the previous state *4          | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |                                |
| P77 to P70                |  |  |                                 |  |                                      |  |   |                                |
| P97 to P91                |  |  |                                 |  |                                      |  |   |                                |
| PA3 to PA0                |  |  |                                 |  |                                      |  |   |                                |
| P90 (CS0)                 |  |  |                                 |  | "H" output                           |  |   |                                |
| P87 to P80                | Input enabled *6                         | Input enabled *6                                 | Input enabled *6                | Input disabled                           |                                      |  |   |                                |

\*1: "Output state" means that the pin-drive transistors are set in a drive-enabled state, but internal circuits are stopped so that a fixed value, "H" or "L", is output. When internal peripheral circuits are in operation and output functions are used, output varies except when reset. Output does not vary during a reset.

\*2: The previous values are retained if used as an output port.

\*3: "Output enabled" means that the pin-drive transistors are set in a drive state and that internal circuit operations are enabled. Operational states are therefore conveyed on the pins.

\*4: The state output immediately before this mode was set is output as it is. If it is input, input is disabled.

"Input disabled" means that operations of input gates located very close to the pins are enabled, but pin states cannot be accepted in internal operations because internal circuits are not operating.

\*5: In the state of "Input cutoff", input A is masked and "L" level is transmitted internally. "Output Hi-Z" means that the pin-drive transistors are disabled and the pins are set to the high-impedance state.

\*6: Same as in other ports when used in the output state. "Input enabled" means that input functions are ready and require pull up/pull down or external input.

## ■ Pin States in External Bus 8-bit Data Bus Mode and Non-multiplex 8-bit External Bus Mode

Table 6.7-5 summarizes pin states in the external bus 8-bit data bus mode and non-multiplex 8-bit external bus mode.

**Table 6.7-5 Pin States in External Bus 8-bit Data Bus Mode and Non-multiplex 8-bit External Bus Mode**

| Pin name                  | In sleep state                           | When stopped                                     |                                 | In hold state                          | When reset                           | Internal ROM access immediately after reset cancellation | Internal ROM access after external ROM access |                                |
|---------------------------|--|--|---------------------------------|--|--------------------------------------|--|---|--------------------------------|
|                           |  | SPL = 0  | SPL = 1                         |  |                                      |  |   |                                |
| P07 to P00 (AD07 to AD00) | Input disabled/<br>Output Hi-Z           | Input cutoff/<br>Output Hi-Z                     | Input cutoff/<br>Output Hi-Z *5 | Input disabled/<br>Output Hi-Z         | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |                                |
| P37 to P30 (A07 to A00)   | Output state *1                          | Output state *1                                  |                                 |  | Output state *1                      | Output state *1  | Output state *1                               | Maintains the previous address |
| P47 to P40 (A15 to A08)   |  |  |                                 |  |                                      |  |   |                                |
| P27 to P20 (A23 to A16)   | Output state *1, *2                      | Output state *1, *2                              |                                 | Input disabled/<br>Output Hi-Z *2      | Input disabled/<br>Output enabled *3 | CLK output   | CLK output                                    |                                |
| P57 (CLK)                 | Input disabled/<br>Output enabled *2, *3 | Input disabled/<br>Output state *1, *2           |                                 | Input disabled/<br>Output state *2, *3 |                                      |  |   |                                |
| P56 (RDY)                 | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4 |                                 | Input disabled *2                      | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |                                |
| P55 (HAK)                 |  |  |                                 | "L" output                             |                                      |  |   |                                |
| P54 (HRQ)                 |  |  |                                 | "H" input                              |                                      |  |   |                                |
| P53                       |  |  |                                 | Maintains the previous state *4        |                                      |  |   |                                |
| P52 (WRL)                 | "H" output *2                            | "H" output *2                                    |                                 | Input disabled/<br>Output Hi-Z *2      | "H" output                           | "H" output   | "H" output                                    |                                |
| P51 (RD)                  | "H" output                               | "H" output                                       |                                 | Input disabled/<br>Output Hi-Z         |                                      |  |   |                                |
| P50 (ALE)                 | "L" output                               | "L" output                                       |                                 |  | Output enabled *2                    | "L" output   | "L" output                                    |                                |
| P17 to P10                | Maintains the previous state *4          | Input cutoff/<br>Maintains the previous state *4 |                                 | Maintains the previous state *4        | Input disabled/<br>Output Hi-Z       | Output Hi-Z/<br>Input enabled                            | Output Hi-Z/<br>Input enabled                 |                                |
| P67 to P60                |  |  |                                 |  |                                      |  |   |                                |
| P77 to P70                |  |  |                                 |  |                                      |  |   |                                |
| P97 to P91                |  |  |                                 |  | "H" output                           |  |   |                                |
| PA3 to PA0                |  |  |                                 |  |                                      |  |   |                                |
| P90 (CS0)                 |  |  |                                 |  | Input disabled                       |  |   |                                |
| P87 to P80                | Input enabled *6                         | Input enabled *6                                 | Input enabled *6                |  |                                      |  |   |                                |

\*1: "Output state" means that the pin-drive transistors are set in a drive-enabled state, but internal circuits are stopped so that a fixed value, "H" or "L", is output. When internal peripheral circuits are in operation and output functions are used, output varies except when reset. Output does not vary during a reset.

\*2: The previous values are retained if used as an output port.

\*3: "Output enabled" means that the pin-drive transistors are set in a drive state and that internal circuit operations are enabled. Operational states are therefore conveyed on the pins.

\*4: The state output immediately before this mode was set is output as it is. If it is input, input is disabled.

"Input disabled" means that operations of input gates located very close to the pins are enabled, but pin states cannot be accepted in internal operations because internal circuits are not operating.

\*5: In the state of "Input cutoff", input A is masked and "L" level is transmitted internally. "Output Hi-Z" means that the pin-drive transistors are disabled and the pins are set to the high-impedance state.

\*6: Same as in other ports when used in the output state. "Input enabled" means that input functions are ready and require pull up/pull down or external input.

## 6.8 Caution on Using Low-Power Consumption Mode

---

When operating in the low-power consumption mode, exercise reasonable care concerning the following:

- Change to the standby mode and interrupts
  - Cancellation of standby mode by interrupt
  - Cancellation of stop mode
  - Oscillation stabilization wait time
  - Notes on accessing the low-power consumption mode control register (LPMCR) to enter the standby mode
- 

### ■ Change to Standby Mode and Interrupts

When a peripheral function issues an interrupt request to the CPU, the setting of "1" to the stop mode bit (STP) and sleep mode bit (SLP) in the low-power consumption mode control register (LPMCR) and "0" to the watch/time-base timer mode bit (TMD) is ignored. A change to the appropriate standby mode is not executed. (Neither is a change to the standby mode executed after interrupt processing.) In this event, the acceptance of interrupt requests by the CPU if the interrupt level is higher than 7 is irrelevant.

Even when the CPU is processing an interrupt, the interrupt request flag bit is cleared, and a change to the standby mode is possible unless there is another interrupt request.

### ■ Cancellation of Standby Mode by Interrupt

The standby mode is canceled if a peripheral function or other device issues an interrupt request whose interrupt level is higher than 7 in the sleep, time-base timer, or stop mode. This is irrelevant to whether or not the CPU accepts an interrupt.

After the standby mode is canceled by an interrupt, branching to an interrupt processing routine is performed as in a normal interrupt operation if the priority of interrupt level setting bit (bits IL2, IL1, and IL0 of the ICR register) corresponding to an interrupt request is higher than the interrupt level mask register (ILM) and if interrupts are enabled ( $I = 1$ ) by the I-flag of the condition code register (CCR). If an interrupt is not accepted, operation is resumed beginning from an instruction next to the instruction specifying the standby mode.

In the execution of interrupt processing, interrupt processing is normally started after execution of an instruction next to the instruction specifying the standby mode.

However, depending on the conditions under which the mode is changed to the standby mode, interrupt processing may be started before the next instruction is executed.

---

#### Note:

Disabling of interrupts or other actions are required before setting the standby mode if branching to an interrupt processing routine is not performed immediately after a return.

---

## ■ Oscillation Stabilization Wait Time

### ○ Oscillation stabilization wait time of oscillation clock

The oscillator for source oscillation is stopped in the stop mode, and an oscillation stabilization wait time must be provided. Specify the oscillation stabilization wait time selected with the selection bits (WS1 and WS0) for the oscillation stabilization wait time of the clock selection register (CKSCR).

---

#### Note:

Set "00<sub>B</sub>" in the selection bits (WS1 and WS0) for the oscillation stabilization wait time of the clock selection register (CKSCR) only in the main clock mode.

---

### ○ Oscillation stabilization wait time of PLL clock

In main clock mode, the PLL multiplication circuit stops. When changing to PLL clock mode, it is necessary to reserve the PLL clock oscillation stabilization wait time. The CPU runs in main clock mode till the PLL clock oscillation stabilization wait time has elapsed.

When the main clock mode is switched to PLL clock mode, the PLL clock oscillation stabilization wait time is fixed at  $2^{14}/\text{HCLK}$  (HCLK: oscillation clock).

In sub clock mode, the main clock and PLL multiplication circuit stop. When changing to PLL clock mode, it is necessary to reserve the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of the main clock and PLL clock oscillation stabilization wait times. The PLL clock oscillation stabilization wait time, however, requires  $2^{14}/\text{HCLK}$  or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10<sub>B</sub>" or "11<sub>B</sub>".

In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires  $2^{14}/\text{HCLK}$  or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10<sub>B</sub>" or "11<sub>B</sub>".

## ■ Switching 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.

## ■ Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode

- To access the low-power consumption mode control register (LPMCR) with assembler language
- To set the low-power consumption mode control register (LPMCR) to enter the standby mode, use the instruction listed in Table 6.3-2.
- The low-power consumption mode transition instruction in Table 6.3-2 must always be followed by an array of instructions highlighted by a line below.

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

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

MOV A,#'10 ; any instruction

The devices does not guarantee its operation after releasing from the standby mode if you place an array of instructions other than the one enclosed in the line.

- To access the low-power consumption mode control register (LPMCR) with C language
- To enter the standby mode using the low-power consumption mode control register (LPMCR), use one of the following methods (1) to (3) to access the register:
- (1) Specify the standby mode transition instruction as a function and insert two `_wait_nop()` built-in functions after that instruction. If any interrupt other than the interrupt to return from the standby mode can occur within the function, optimize the function during compilation to suppress the LINK and UNLINK instructions from occurring.

Example: Watch mode or time-base timer mode transition function

```
void enter_watch(){
    IO_LPMCR.byte = 0x10; /* Set LPMCR TMD bit to "0" */
    _wait_nop();
    _wait_nop();
}
```

- (2) Define the standby mode transition instruction using `_asm` statements and insert two NOP and JMP instructions after that instruction.

Example: Transition to sleep mode

```
_asm(" MOVI: _IO_LPMCR,#H'58); /* Set LPMCR SLP bit to "1" */
_asm(" NOP");
_asm(" NOP");
_asm(" JMP $+3"); /* Jump to next instruction */
```

- (3) Define the standby mode transition instruction between #pragma asm and #pragma endasm and insert two NOP and JMP instructions after that instruction.

Example: Transition to stop mode

```
#pragma asm
    MOV  I: _IO_LPMCR,#H'98      /* Set LPMCR STP bit to "1" */
    NOP
    NOP
    JMP  $+3                      /* Jump to next instruction */
#pragma endasm
```



# CHAPTER 7    MODE SETTING

---

**This chapter explains mode setting, mode pins, mode data, external memory access and its operation.**

---

7.1 Mode Setting

7.2 Mode Pins (MD2 to MD0)

7.3 Mode Data

7.4 External Memory Access

7.5 Operation of Each Mode for Mode Setting



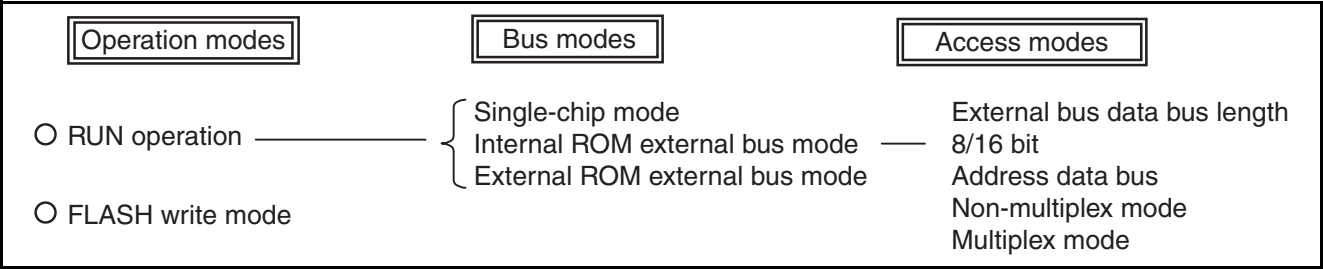
# 7.1 Mode Setting

The F<sup>2</sup>MC-16LX has different modes in each access system and access area. Each mode is set according to a mode pin at the reset state and according to mode data obtained by mode-fetch.

## ■ Mode Setting

The F<sup>2</sup>MC-16LX has different modes in each access system and access area. In this module, they are categorized as shown in Figure 7.1-1.

Figure 7.1-1 Categories of Modes



## ■ Operation Modes

Operation modes are used to control the operating conditions of devices, and they are set by mode setting pins (MDx) and with the contents of the Mx bits in mode data. By selecting an operation mode, normal operation activation or flash serial programming can be performed.

## ■ Bus Modes

Bus modes are used to control the operation of internal ROMs and of external access functions, and they are specified by mode setting pins (MDx) and with the contents of the Mx bits in mode data. Mode setting pins (MDx) specify the reset vector as well as set the bus mode for reading mode data. The Mx bits in mode data specify the bus mode during normal operation.

## ■ Access Modes

Access modes are used to control the external data bus width, and they are set by mode setting pins (MDx) and with the contents of the Sx bits in mode data. Selection of an access mode specifies either an 8-bit length or 16-bit length for the external data bus. It also specifies either the non-multiplex mode or multiplex mode for the address data bus.

## 7.2 Mode Pins (MD2 to MD0)

Mode pins are three external pins (MD2 to MD0) that specify the reset vector and mode data fetching method.

### ■ Settings of Mode Pins (MD2 to MD0)

Mode pins (MD2 to MD0) are used to select the source, either the external or internal data bus when reset vectors are read and stored, and to select the bus width when the external data bus is used. For a device with built-in FLASH ROM, the mode pins are also used to set the FLASH ROM write mode for writing built-in ROM program.

Table 7.2-1 lists the contents of mode pin settings.

**Table 7.2-1 Contents of Mode Pin Settings**

| P81 | P80 | MD2 | MD1 | MD0 | Mode name               | Reset vector access area | External data bus width | Remarks   |
|-----|-----|-----|-----|-----|-------------------------|--------------------------|-------------------------|---|
| -   | -   | 0   | 0   | 0   | External vector mode 0  | External                 | Multiplex mode          | Reset vector 8-bit bus width access                         |
| -   | -   | 0   | 0   | 1   | External vector mode 1  | External                 | Multiplex mode          | Reset vector 16-bit bus width access                        |
| -   | -   | 0   | 1   | 0   | External vector mode 2  | External                 | Non-multiplex mode      | Reset vector 8-bit bus width access                         |
| -   | -   | 0   | 1   | 1   | Internal vector mode    | Internal                 | Mode data               | Operation after reset sequence is controlled with mode data |
| -   | -   | 1   | 0   | 0   | Setting is prohibited   |                          |                         |   |
| -   | -   | 1   | 0   | 1   |                         |                          |                         |   |
| 1   | 0   | 1   | 1   | 0   | FLASH serial writing    |                          |                         |   |
| -   | -   | 1   | 1   | 1   | FLASH writer write mode | -                        | -                       | -   |

Note:

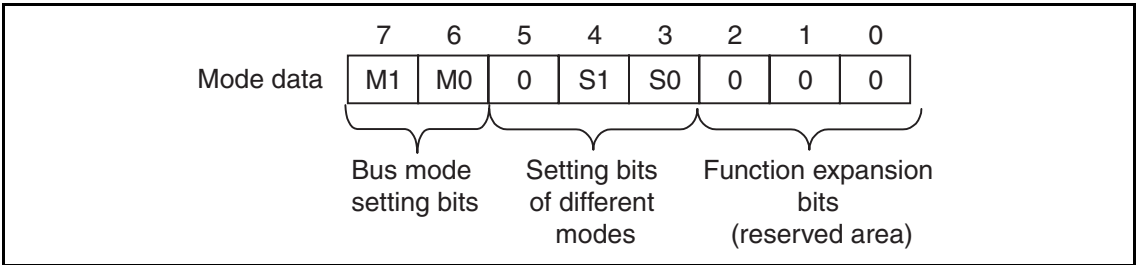
MD2 to MD0: Specify 0= $V_{SS}$  or 1= $V_{CC}$ . For external vector mode 2, the data bus width also has a default value of 8 bits. To specify 16 bits as the data bus width, specify mode data for the non-multiplex external data bus 16-bit mode, and then the IOBS and LMBS areas are set up for 16-bit size access. To set up the HMBS area for 16-bit size access, change the HMBS setting.

### 7.3 Mode Data

Mode data stored at address FFFDF<sub>H</sub> in memory specifies the operation immediately after the reset sequence. Mode data is read and stored in the CPU automatically by mode fetching.

■ Mode Data

During the reset sequence, mode data at address FFFDF<sub>H</sub> is sent to the mode register in the CPU core. The CPU uses this mode data to set the memory access mode. The contents of the mode register can be changed only by the reset sequence. Furthermore, mode data settings become valid only after the reset sequence. The configuration of mode data is shown in the figure below.



■ Setting Bits of Different Modes (S1, S0)

Bits S1 and S0 specify the bus mode and access mode that is set after completion of the reset sequence.

Table 7.3-1 lists the contents of the settings for bits S1 and S0.

Table 7.3-1 Contents of Bit S1 and S0 Settings

| S1 | S0 | Functions                     |                                |
|----|----|-------------------------------|--------------------------------|
| 0  | 0  | External data bus 8-bit mode  | Address data bus multiplex     |
| 0  | 1  | External data bus 16-bit mode |                                |
| 1  | 0  | External data bus 8-bit mode  | Address data bus non-multiplex |
| 1  | 1  | External data bus 16-bit mode |                                |

## ■ Bus Mode Setting Bits (M1, M0)

Bits M1 and M0 specify the operation mode that is set after completion of the reset sequence.

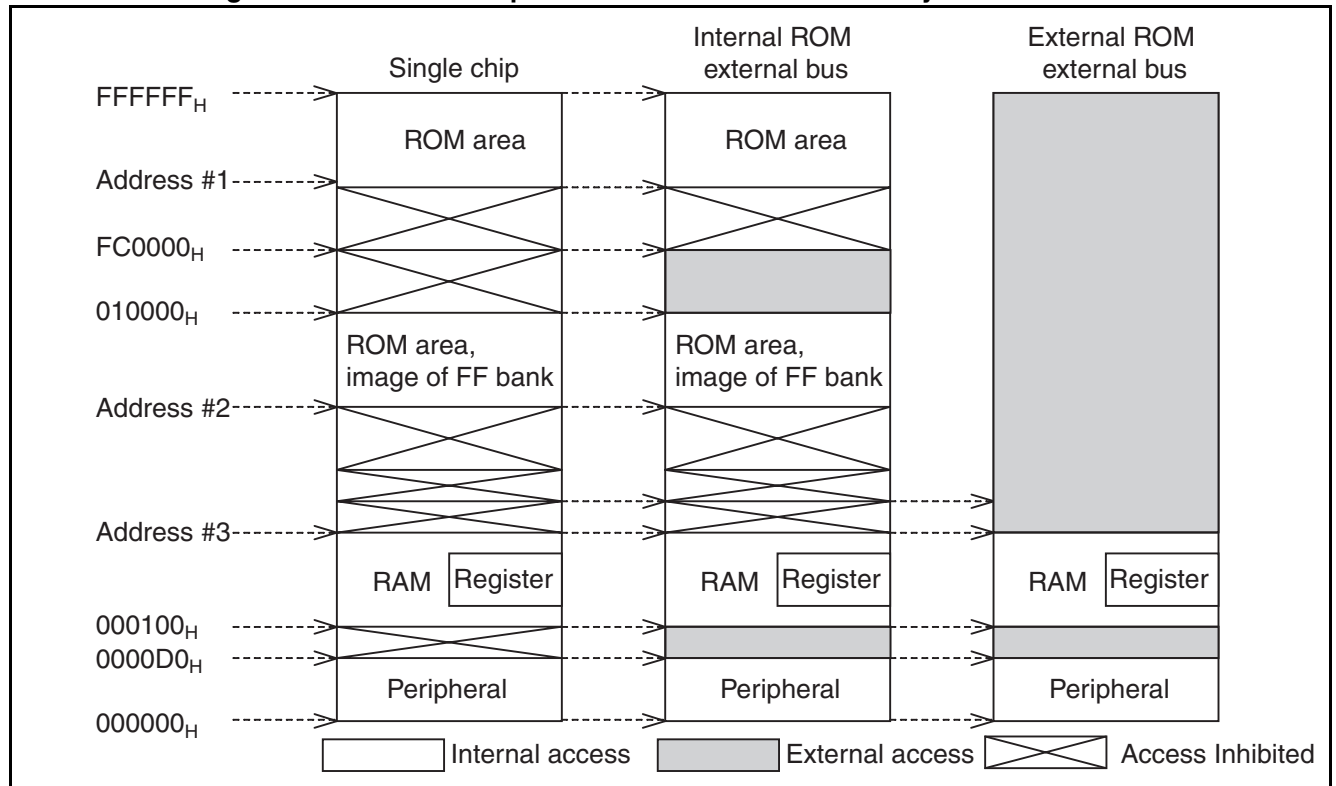
Table 7.3-2 lists the contents of the settings for bits M1 and M0.

**Table 7.3-2 Contents of Bit M1 and M0 Settings**

| M1 | M0 | Functions                          |
|----|----|------------------------------------|
| 0  | 0  | Single-chip mode                   |
| 0  | 1  | Internal ROM and external bus mode |
| 1  | 0  | External ROM and external bus mode |
| 1  | 1  | (Setting is prohibited)            |

Figure 7.3-1 shows the correspondence between access areas and physical addresses.

**Figure 7.3-1 Relationship Between Access Areas and Physical Addresses**



Note:

"Address #X" is determined based on individual models. See APPENDIX "APPENDIX A Memory Map", for details.

## ■ Relationship Between Mode Pins and Mode Data (an Example Showing Recommended Relationship)

Table 7.3-3 shows the relationship between mode pins and mode data.

**Table 7.3-3 Relationship Between Mode Pins and Mode Data**

| Mode   | MD2 | MD1 | MD0 | M1 | M0 | S1 | S0 |
|--|-----|-----|-----|----|----|----|----|
| Single chip  | 0   | 1   | 1   | 0  | 0  | X  | X  |
| Internal ROM external bus mode, 8-bit<br>(address data multiplex)                                | 0   | 1   | 1   | 0  | 1  | 0  | 0  |
| Internal ROM external bus mode, 16-bit<br>(address data multiplex)                               | 0   | 1   | 1   | 0  | 1  | 0  | 1  |
| Internal ROM external bus mode, 8-bit<br>(address data non-multiplex)                            | 0   | 1   | 1   | 0  | 1  | 1  | 0  |
| Internal ROM external bus mode, 16-bit<br>(address data non-multiplex)                           | 0   | 1   | 1   | 0  | 1  | 1  | 1  |
| External ROM external bus mode, 16-bit, bus vector<br>with 16-bit width (address data multiplex) | 0   | 0   | 1   | 1  | 0  | 0  | 1  |
| External ROM external bus mode, 8-bit<br>(address data multiplex)                                | 0   | 0   | 0   | 1  | 0  | 0  | 0  |
| External ROM external bus mode, 8-bit<br>(address data non-multiplex)                            | 0   | 1   | 0   | 1  | 0  | 1  | 0  |

Note:

If the output for high-order addresses of A23 to A16 is suppressed, the maximum capacity of accessible memory is 64 K bytes.

## ■ Operation of External Pins in Each Mode

Table 7.3-4 shows the operation of each external pin in the non-multiplex mode and multiplex mode.

**Table 7.3-4 Operation of External Pins in Each Mode**

|  | Functions                |                         |                        |                         |                          |                         |                        |                         |
|--|--------------------------|-------------------------|------------------------|-------------------------|--------------------------|-------------------------|------------------------|-------------------------|
|  | Non-multiplex mode       |                         |                        |                         | Multiplex mode           |                         |                        |                         |
|  | External address control |                         |                        |                         | External address control |                         |                        |                         |
|  | Permitted (address)      |                         | Prohibited (port)      |                         | Permitted (address)      |                         | Prohibited (port)      |                         |
|  | External bus extension   |                         | External bus extension |                         | External bus extension   |                         | External bus extension |                         |
|  | 8-bit                    | 16-bit                  | 8-bit                  | 16-bit                  | 8-bit                    | 16-bit                  | 8-bit                  | 16-bit                  |
| P07 to P00/<br>D07 to D00/<br>AD07 to AD00 | D07 to D00               |                         |                        |                         | AD07 to AD00             |                         |                        |                         |
| P17 to P10/<br>D15 to D08/<br>AD15 to AD08 | Port                     | D15 to D08              | Port                   | D15 to D08              | A15 to A08               | AD15 to AD08            | A15 to A08             | AD15 to AD08            |
| P27 to P20                                 | A23 to A16               |                         | Port                   |                         | A23 to A16               |                         | Port                   |                         |
| P37 to P30                                 | A07 to A00               |                         | A07 to A00             |                         | Port                     |                         |                        |                         |
| P47 to P40                                 | A15 to A08               |                         | A15 to A08             |                         |                          |                         |                        |                         |
| ALE  | ALE                      |                         |                        |                         | ALE                      |                         |                        |                         |
| $\overline{\text{RD}}$                     | $\overline{\text{RD}}$   |                         |                        |                         | $\overline{\text{RD}}$   |                         |                        |                         |
| P52/ $\overline{\text{WRL}}$               | $\overline{\text{WRL}}$  |                         |                        |                         | $\overline{\text{WRL}}$  |                         |                        |                         |
| P53/ $\overline{\text{WRH}}$               | Port                     | $\overline{\text{WRH}}$ | Port                   | $\overline{\text{WRH}}$ | Port                     | $\overline{\text{WRH}}$ | Port                   | $\overline{\text{WRH}}$ |
| P54/ $\overline{\text{HRQ}}$               | HRQ                      |                         |                        |                         | HRQ                      |                         |                        |                         |
| P55/ $\overline{\text{HAK}}$               | HAK                      |                         |                        |                         | HAK                      |                         |                        |                         |
| P56/ $\overline{\text{RDY}}$               | RDY                      |                         |                        |                         | RDY                      |                         |                        |                         |
| P57/ $\overline{\text{CLK}}$               | CLK                      |                         |                        |                         | CLK                      |                         |                        |                         |

- In the single-chip mode, all addresses can be used as ports.
- High-order addresses,  $\overline{\text{WRL}}$ ,  $\overline{\text{WRH}}$ , HAK, HRQ, RDY, and CLK can be used as ports depending on the selected function.
- In the non-multiplex mode, the up/down-counter, SCI2, and  $\mu\text{PG}$  cannot be used. They function as addresses.

## 7.4 External Memory Access

**This section contains block diagrams about external memory access, the configuration and functions of registers, and operation of external memory access.**

### ■ I/O Signal Pins for External Memory Access

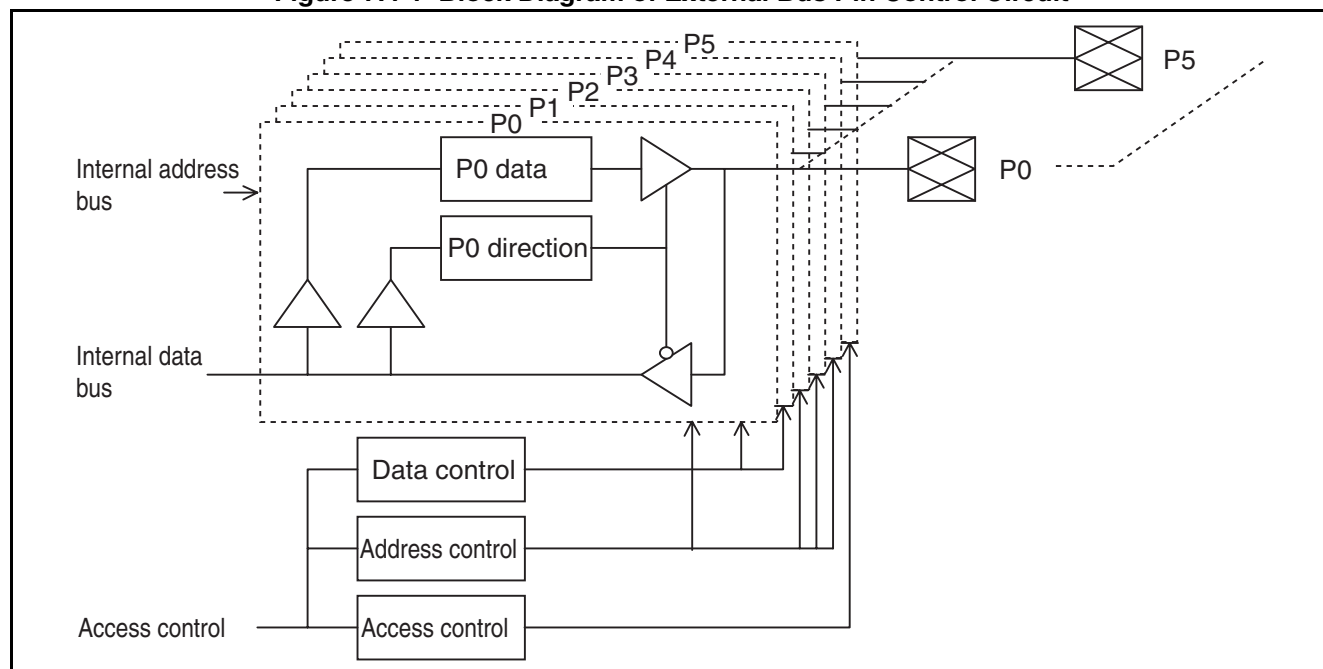
For accessing external memory and peripheral devices, the F<sup>2</sup>MC-16LX supplies the following address, data, and control signals:

- CLK (P57): Outputs the machine cycle clock (KBP)
- RDY (P56): External ready input pin
- $\overline{\text{HAK}}$  (P55): Hold acknowledge output pin
- HRQ (P54): Hold request input pin
- $\overline{\text{WRH}}$  (P53): Write signal for the high-order 8 bits on the data bus
- $\overline{\text{WRL}}$  (P52): Write signal for the low-order 8 bits on the data bus
- $\overline{\text{RD}}$  (P51): Read signal
- ALE (P50): Address latch enable signal (effective in the multiplex mode)

## ■ Block Diagram

Figure 7.4-1 is a block diagram of the external bus pin control circuit.

**Figure 7.4-1 Block Diagram of External Bus Pin Control Circuit**



## ■ List of Registers

Figure 7.4-2 shows a list of registers in the external bus pin control circuit.

**Figure 7.4-2 Registers in External Bus Pin Control Circuit**

|                     |      |      |      |      |      |     |      |      |   |
|---------------------|------|------|------|------|------|-----|------|------|---|
| bit                 | 15   | 14   | 13   | 12   | 11   | 10  | 9    | 8    |   |
| 0000A5 <sub>H</sub> | IOR1 | IOR0 | HMR1 | HMR0 | —    | —   | LMR1 | LMR0 | Automatic ready function<br>selection register (ARSR) |
|                     | (W)  | (W)  | (W)  | (W)  | (-)  | (-) | (W)  | (W)  | Read/write  |
|                     | (0)  | (0)  | (1)  | (1)  | (-)  | (-) | (0)  | (0)  | Initial value   |
| bit                 | 7    | 6    | 5    | 4    | 3    | 2   | 1    | 0    |   |
| 0000A6 <sub>H</sub> | E23  | E22  | E21  | E20  | E19  | E18 | E17  | E16  | External address<br>output control register (HACR)    |
|                     | (W)  | (W)  | (W)  | (W)  | (W)  | (W) | (W)  | (W)  | Read/write  |
|                     | (*)  | (*)  | (*)  | (*)  | (*)  | (*) | (*)  | (*)  | Initial value   |
| bit                 | 15   | 14   | 13   | 12   | 11   | 10  | 9    | 8    |   |
| 0000A7 <sub>H</sub> | CKE  | RYE  | HDE  | IOBS | HMBS | WRE | LMBS | —    | Bus control signal<br>selection register (EPCR)       |
|                     | (W)  | (W)  | (W)  | (W)  | (W)  | (W) | (W)  | (-)  | Read/write  |
|                     | (1)  | (0)  | (0)  | (0)  | (*)  | (1) | (0)  | (-)  | Initial value   |



## 7.4.1 Automatic ready function selection register (ARSR)

This section shows the configuration and explains the function of the automatic ready function selection register (ARSR)

### ■ Automatic Ready Function Selection Register (ARSR)

The bit configuration of the automatic ready function selection register (ARSR) is shown in the figure below.

|                     | 15   | 14   | 13   | 12   | 11  | 10  | 9    | 8    | Automatic ready function selection register |
|---------------------|------|------|------|------|-----|-----|------|------|---|
| 0000A5 <sub>H</sub> | IOR1 | IOR0 | HMR1 | HMR0 | —   | —   | LMR1 | LMR0 |   |
|                     | (W)  | (W)  | (W)  | (W)  | (-) | (-) | (W)  | (W)  | Read/write                                  |
|                     | (0)  | (0)  | (1)  | (1)  | (-) | (-) | (0)  | (0)  | Initial value                               |

Functions of each bit in the automatic ready function selection register (ARSR) are described below.

#### [bit15, bit14] IOR1, IOR0

These bits are used to select the automatic wait function for external access to areas in a range of 0000D0<sub>H</sub> to 0000FF<sub>H</sub>. Contents of settings are listed below.

| IOR1 | IOR0 | Setting  |
|------|------|--|
| 0    | 0    | Automatic wait prohibited [Initial value]                |
| 0    | 1    | Automatic wait in 1 machine cycle during external access |
| 1    | 0    | Automatic wait in 2 machine cycle during external access |
| 1    | 1    | Automatic wait in 3 machine cycle during external access |

#### [bit13, bit12] HMR1, HMR0

These bits are used to select the automatic wait function for external access to areas in a range of 800000<sub>H</sub> to FFFFFFF<sub>H</sub>. Contents of settings are listed below.

| HMR1 | HMR0 | Setting  |
|------|------|--|
| 0    | 0    | Automatic wait prohibited  |
| 0    | 1    | Automatic wait in 1 machine cycle during external access                 |
| 1    | 0    | Automatic wait in 2 machine cycle during external access                 |
| 1    | 1    | Automatic wait in 3 machine cycle during external access [Initial value] |

**[bit9, bit8] LMR1, LMR0**

These bits are used to select the automatic wait function for external access to areas in a range of 002000<sub>H</sub> to 7FFFFFF<sub>H</sub>. Contents of settings are listed below.

| LMR1 | LMR0 | Setting  |
|------|------|--|
| 0    | 0    | Automatic wait prohibited [Initial value]                |
| 0    | 1    | Automatic wait in 1 machine cycle during external access |
| 1    | 0    | Automatic wait in 2 machine cycle during external access |
| 1    | 1    | Automatic wait in 3 machine cycle during external access |

### 7.4.2 External address output control register (HACR)

This section shows the configuration and explains the function of the external address output control register.

■ External Address Output Control Register (HACR)

The bit configuration of the external address output control register is shown in the figure below.

|                     |     |     |     |     |     |     |     |     |   |
|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|---|
|                     | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | External address<br>output control register |
| 0000A6 <sub>H</sub> | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |   |
|                     | (W) | (W) | (W) | (W) | (W) | (W) | (W) | (W) | Read/write                                  |
|                     | (*) | (*) | (*) | (*) | (*) | (*) | (*) | (*) | Initial value                               |

The external address output control register controls external output of addresses (A23 to A16). One bit corresponds to each of addresses A23 to A16 and controls each address output pin as follows.

|   |  |
|---|--|
| 0 | The corresponding pin is for address output (AXX). |
| 1 | The corresponding pin is as an I/O port (PXX).     |

This register cannot be accessed while the device is set to the single-chip mode. In this event, all ports function as I/O ports regardless of the values in this register. All bits of this register are dedicated for writing, and the readout value is "1". Furthermore, if addresses are expected to be output with address output selected, specify the value of DDR to "0".

The initial value is 1 if the device is activated in internal vector mode. Otherwise, the initial value is 0.

Note:

When using PPG, set it to "1" (setting for an I/O port).

### 7.4.3 Bus control signal selection register (EPCR)

This section shows the configuration and explains the function of the bus control signal selection register.

#### ■ Bus Control Signal Selection Register (EPCR)

The bus control signal selection register (EPCR) sets the bus operation control function in external bus mode.

The bit configuration of the bus control signal selection register is shown in the figure below.

|                     |     |     |     |      |      |     |      |     |  |
|---------------------|-----|-----|-----|------|------|-----|------|-----|--|
| 0000A7 <sub>H</sub> | 15  | 14  | 13  | 12   | 11   | 10  | 9    | 8   | Bus control signal<br>selection register |
|                     | CKE | RYE | HDE | IOBS | HMBS | WRE | LMBS | —   |  |
|                     | (W) | (W) | (W) | (W)  | (W)  | (W) | (W)  | (-) |  |
|                     | (1) | (0) | (0) | (0)  | (*)  | (1) | (0)  | (-) |  |

This register cannot be accessed while the device is set to the single-chip mode. In the single-chip mode, all pins function as I/O ports regardless of the values in this register. All bits of this register are dedicated for writing, and the readout value is "1".

Functions of each bit in the bus control signal selection register are described below.

##### [bit15] CKE

This bit controls the external clock (CLK) output.

|   |   |
|---|---|
| 0 | I/O port (P57) operation (clock prohibited)         |
| 1 | Clock signal (CLK) output permitted [Initial value] |

##### [bit14] RYE

This bit controls the external ready (RDY) input.

|   |  |
|---|--|
| 0 | I/O port (P56) operation (external RDY input prohibited) [Initial value] |
| 1 | External ready (RDY) input permitted                                     |

##### [bit13] HDE

This bit specifies I/O enable for hold-related pins. Hold request input (HRQ) and hold acknowledge output ( $\overline{HAK}$ ) are controlled with this bit setting.

|   |   |
|---|---|
| 0 | I/O port (P55, 54) operation (hold function I/O prohibited) [Initial value]     |
| 1 | Hold request (HRQ) input/hold acknowledge ( $\overline{HAK}$ ) output permitted |

### [bit12] IOBS

This bit specifies the bus width for accessing external buses corresponding to areas in a range of 0000D0<sub>H</sub> to 0000FF<sub>H</sub> in the external data bus 16-bit mode.

|   |   |
|---|---|
| 0 | 16-bit bus width access [Initial value] |
| 1 | 8-bit bus width access                  |

### [bit11] HMBS

This bit specifies the bus width for accessing external buses corresponding to areas in a range of 800000<sub>H</sub> to FFFFFFF<sub>H</sub> in the external data bus 16-bit mode.

|   |  |
|---|--|
| 0 | 16-bit bus width access<br>[Initial value in external vector mode 1]       |
| 1 | 8-bit bus width access<br>[Initial value in external vector modes 0 and 2] |

### [bit10] WRE

This bit controls the output of external write signals (both the  $\overline{WRH}$  and  $\overline{WRL}$  pins in the external data bus 16-bit mode or the  $\overline{WRL}$  pin in the external data bus 8-bit mode).

|   |   |
|---|---|
| 0 | I/O port (P53, P52) operation (write signal output prohibited)  |
| 1 | Write strobe signal ( $\overline{WRH}$ and $\overline{WRL}$ , or only $\overline{WRL}$ ) output permitted [Initial value] |

### [bit9] LMBS

This bit specifies the bus width for accessing external buses corresponding to areas in a range of 002000<sub>H</sub> to 7FFFFFF<sub>H</sub> in the external data bus 16-bit mode.

|   |   |
|---|---|
| 0 | 16-bit bus width access [Initial value] |
| 1 | 8-bit bus width access                  |

---

#### Note:

Even when RDY and HRQ input is permitted by RYE and HDE bits, the I/O port function of a port is enabled. Therefore, be sure to set "0" (input mode) to the bit in DDR5 corresponding to that port.

---

## 7.5 Operation of Each Mode for Mode Setting

---

This section has a timing chart showing the operation of each mode for mode setting.

---

### ■ Types of Mode

Operation with the following items is categorized by function as follows:

- External memory access control signals
  - External data bus 8-bit mode (non-multiplex mode)
  - External data bus 8-bit mode (multiplex mode)
  - External data bus 16-bit mode (non-multiplex mode)
  - External data bus 16-bit mode (multiplex mode)
- Ready function
  - Non-multiplex mode
  - Multiplex mode
- Hold function
  - Non-multiplex mode
  - Multiplex mode

## 7.5.1 External memory access control signals

Access to external memory is performed in 3 cycles when the ready function is not used.

### ■ External Memory Access Control Signal

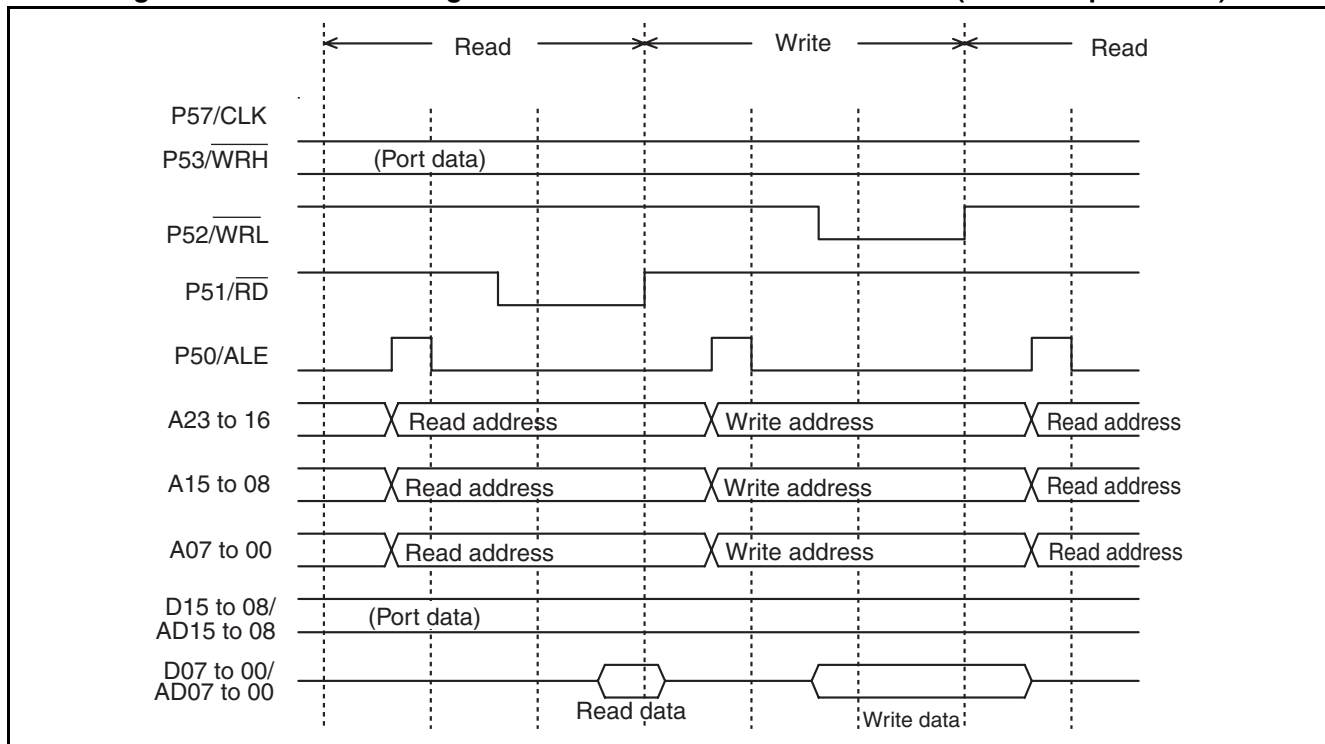
Timing charts for external access in each mode are shown in Figure 7.5-1 to Figure 7.5-4. Access with an 8-bit bus width in the 16-bit external data bus mode is a function for reading from and writing to peripheral chips of an 8-bit width when a mixture of peripheral chips of an 8-bit width and 16-bit width are connected to the external bus. Because access with an 8-bit bus width is performed using the low-order 8 bits of the data bus, connect the peripheral chips of an 8-bit bus width to the low-order 8 bits of data. Access with either a 16-bit bus width or an 8-bit bus width in the external data bus 16-bit mode is determined by the specification of the HMBS, LMBS, and/or IOBS bit of EPCR. Incidentally, there is a case where bus operation is not actually done by only outputting addresses and ALE assert results in the multiplex mode without assert for RD, WRL, and WRH.

Note:

Be sure not to perform access to peripheral chips with only ALE signals.

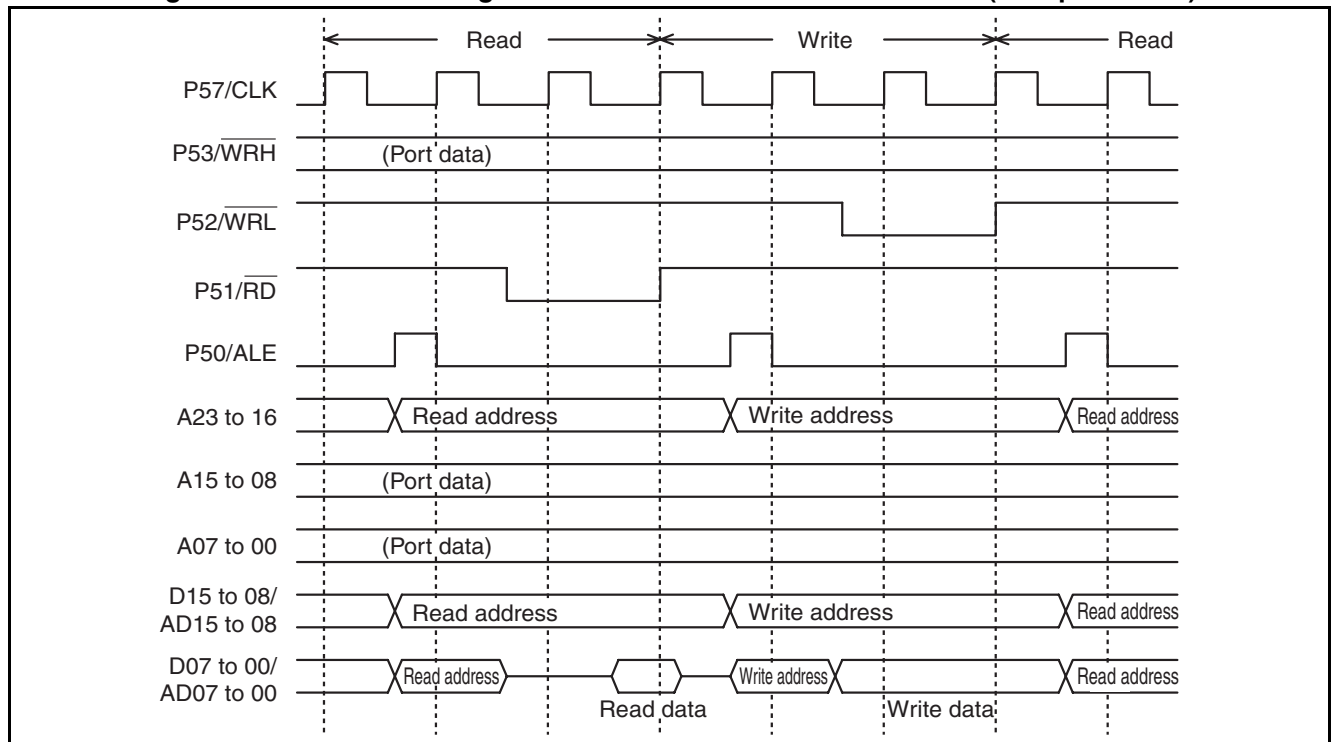
#### ○ External data bus 8-bit mode (non-multiplex mode)

Figure 7.5-1 Access Timing Chart of External Data Bus 8-bit Mode (Non-multiplex Mode)



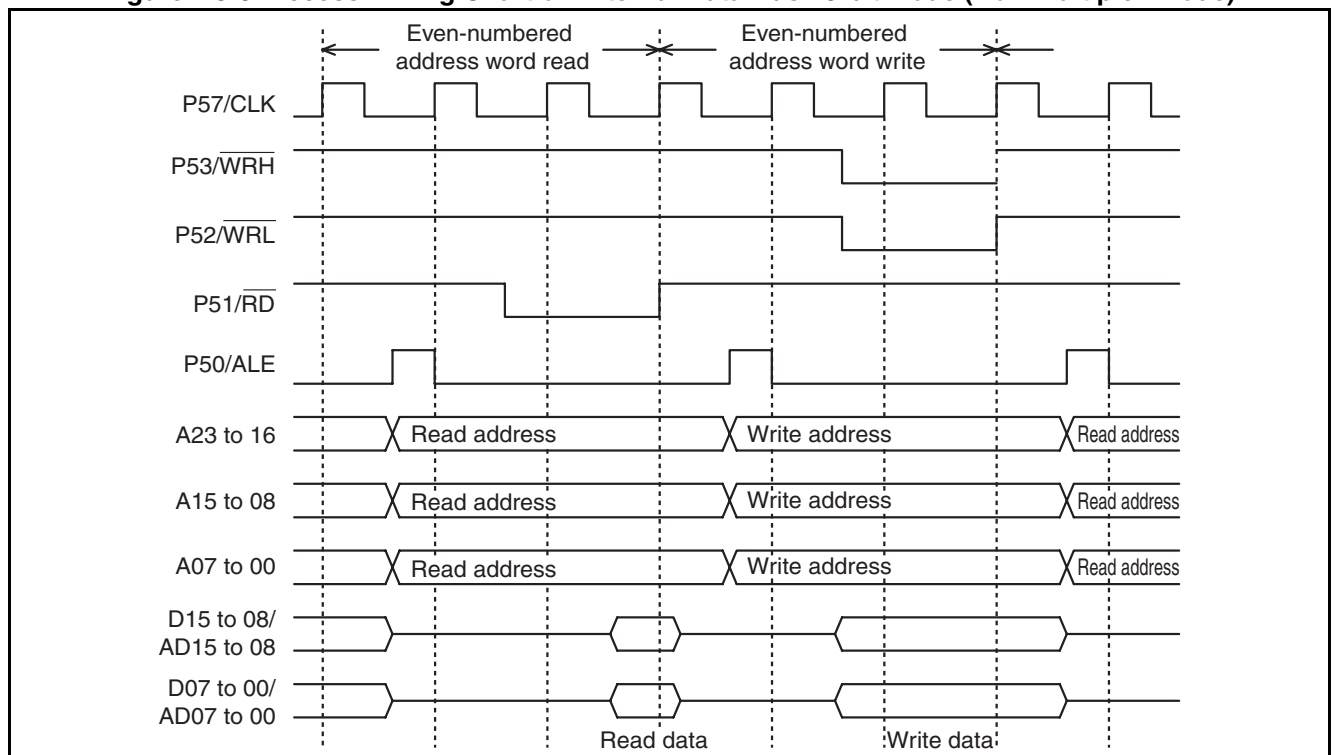
○ External data bus 8-bit mode (multiplex mode)

**Figure 7.5-2 Access Timing Chart of External Data Bus 8-bit Mode (Multiplex Mode)**



○ External data bus 16-bit mode (non-multiplex mode)

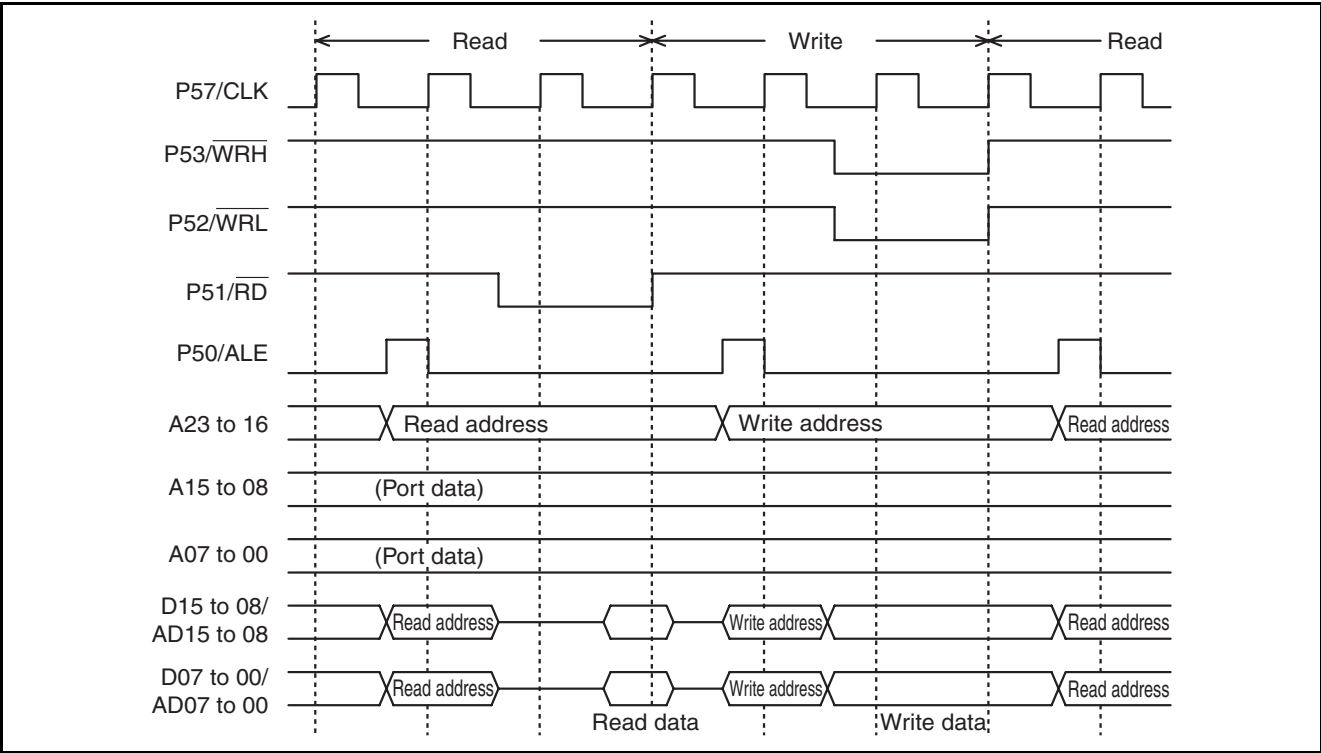
**Figure 7.5-3 Access Timing Chart of External Data Bus 16-bit Mode (Non-multiplex Mode)**





○ External data bus 16-bit mode (multiplex mode)

Figure 7.5-4 Access Timing Chart of External Data Bus 16-bit Mode (Multiplex Mode)



## 7.5.2 Ready function

---

By setting the P56/RDY pin or defining the automatic ready function selection register (ARSR), access to low-speed memory and peripheral circuits is enabled. If the RYE bit in the bus control signal selection register (EPCR) is set to "1", wait cycles are generated during the period where the "L" level is input to the P56/RDY pin while access to the external area is in progress. Thus, the access cycle can be extended.

---

### ■ Ready Function

The F<sup>2</sup>MC-16LX has two types of built-in auto-ready functions for external memory. The auto-ready functions enable the access cycle to be extended by inserting 1 to 3 wait cycles automatically without an external circuit when access occurs to the external area within the following address ranges: a low-order address allocated between 002000<sub>H</sub> and 7FFFFFF<sub>H</sub>, and a high-order address allocated between 800000<sub>H</sub> and FFFFFFF<sub>H</sub>. These functions are evoked by setting the LMR1 and LMR0 bits of ARSR (external area of a low-order address) and the HMR1 and HMR0 bits of ARSR (external area of a high-order address).

Furthermore, the F<sup>2</sup>MC-16LX has built-in auto-ready function for external I/O that is independent of those for external memory. This function enables the access cycle to be extended by inserting 1 to 3 wait cycles automatically without an external circuit when access occurs to the external area between addresses 0000D0<sub>H</sub> and 0000FF<sub>H</sub>. This function is evoked by setting the IOR1 and IOR0 bits in ARSR.

With the auto-ready functions for either external memory or external I/O, if the RYE bit of EPCR is set to "1" when the "L" level is input to the P56/RDY pin upon completion of the wait cycle generated by auto-ready, the wait cycle continues as it is.

The timing charts of the ready function in the non-multiplex mode and multiplex mode are shown below. In both modes, the top figure shows the case where the ready function is not set and the bottom figure shows the case where the ready function is set.

---

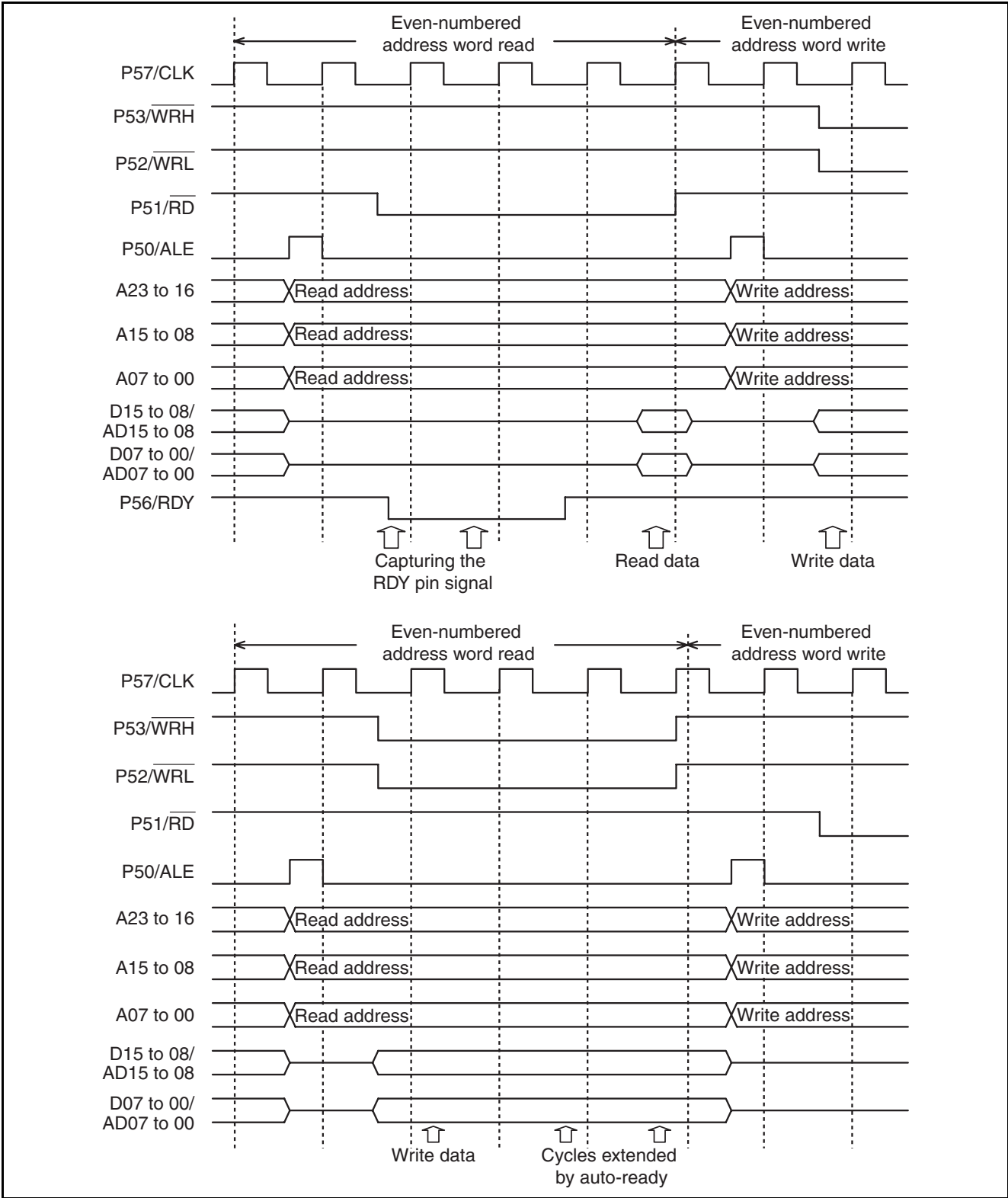
#### Note:

If the AC rating is not satisfied for input from the RDY pin, be careful because this device may enter the runaway state.

---

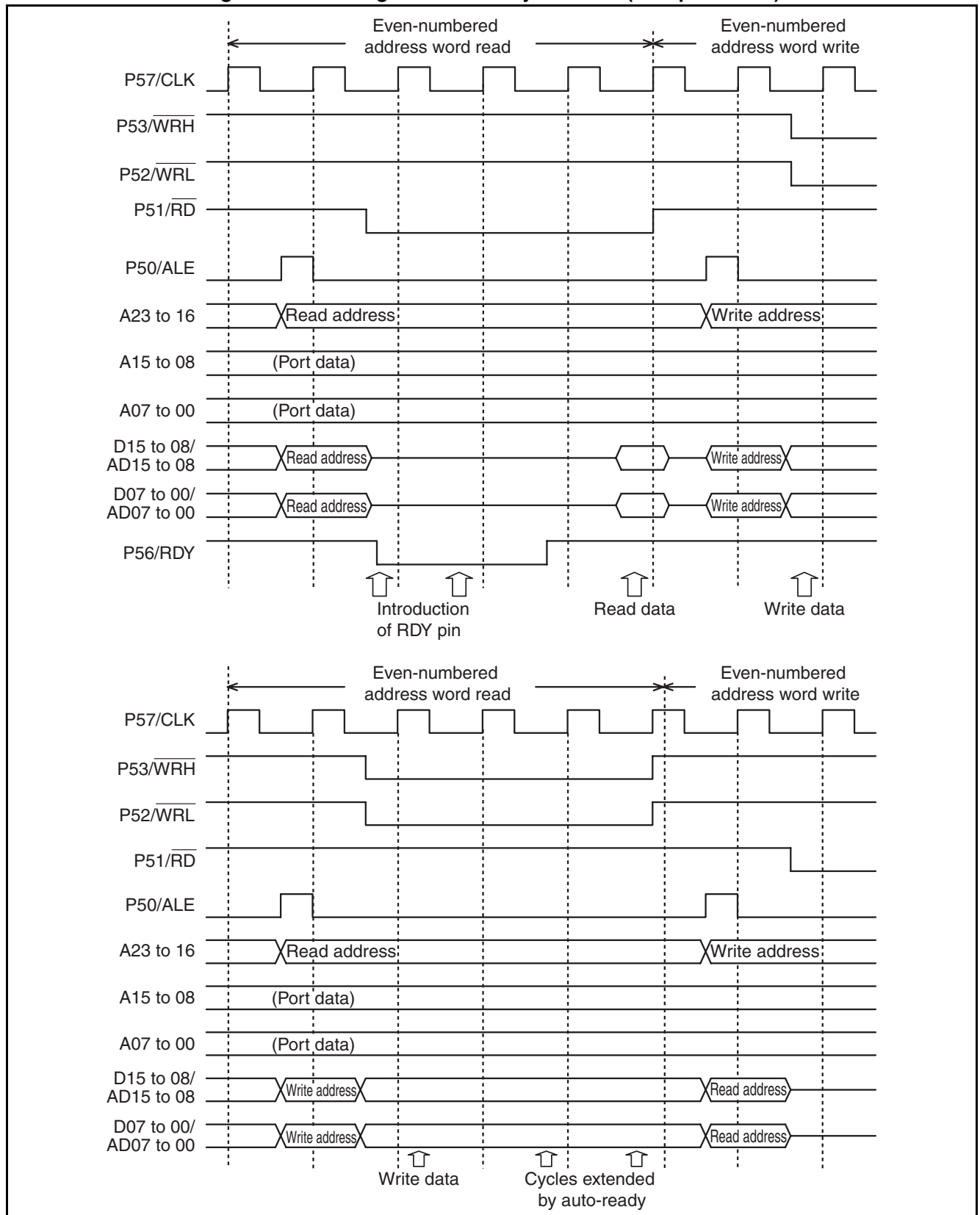
○ Non-multiplex mode

Figure 7.5-5 Timing Chart of Ready Function (Non-multiplex Mode)



## ○ Multiplex mode

Figure 7.5-6 Timing Chart of Ready Function (Multiplex Mode)



## 7.5.3 Hold function

---

This section uses timing charts to describe the operation of the hold function.

---

### ■ Operation of Hold Function

When the HDE bit of EPCR is set to "1", the external bus hold function specified by both the P54/HRQ and P55/ $\overline{\text{HAK}}$  pins becomes effective. When the "H" level is input to the P54/HRQ pin, the hold state is set upon completion of a command by the CPU (after data of 1 element is processed in the case of the string command), and the "L" level is output from P55/ $\overline{\text{HAK}}$  to set the following pins to a high-impedance state:

#### ○ Non-multiplex mode

- Address output: A23 to A00
- Data input/output: D15/AD15 to D00/AD00
- Bus control signal: P51/ $\overline{\text{RD}}$ , P52/ $\overline{\text{WRL}}$ , P53/ $\overline{\text{WRH}}$

#### ○ Multiplex mode

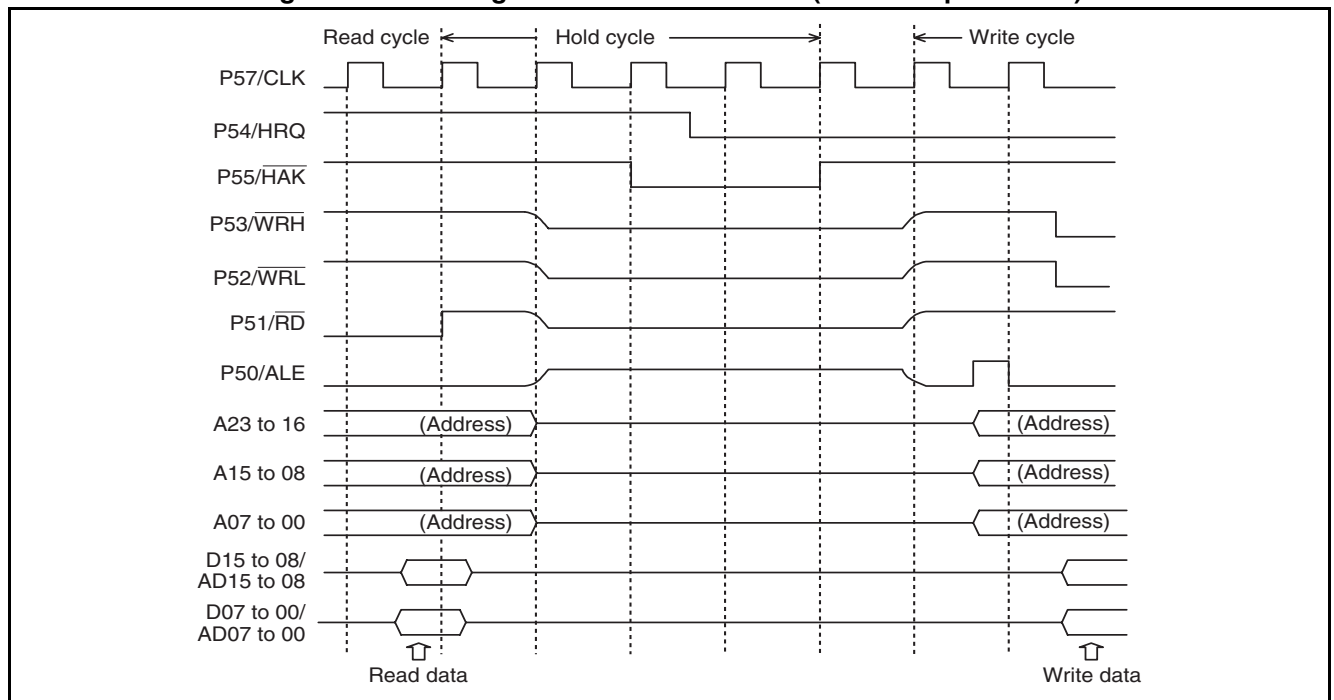
- Address output: A23 to A16
- Address output, Data input/output: D15/AD15 to D00/AD00
- Bus control signal: P51/ $\overline{\text{RD}}$ , P52/ $\overline{\text{WRL}}$ , P53/ $\overline{\text{WRH}}$

This operation enables use of the external bus via the device external circuit. When the "L" level is input to the P54/HRQ pin, the P55/ $\overline{\text{HAK}}$  pin outputs the "H" level to restore the external pin state, and the CPU restarts operation. In the STOP state, requests for hold are rejected.

## ■ Non-multiplex Mode

Figure 7.5-7 shows a timing chart of the non-multiplex-mode hold function in the external data bus 16-bit mode.

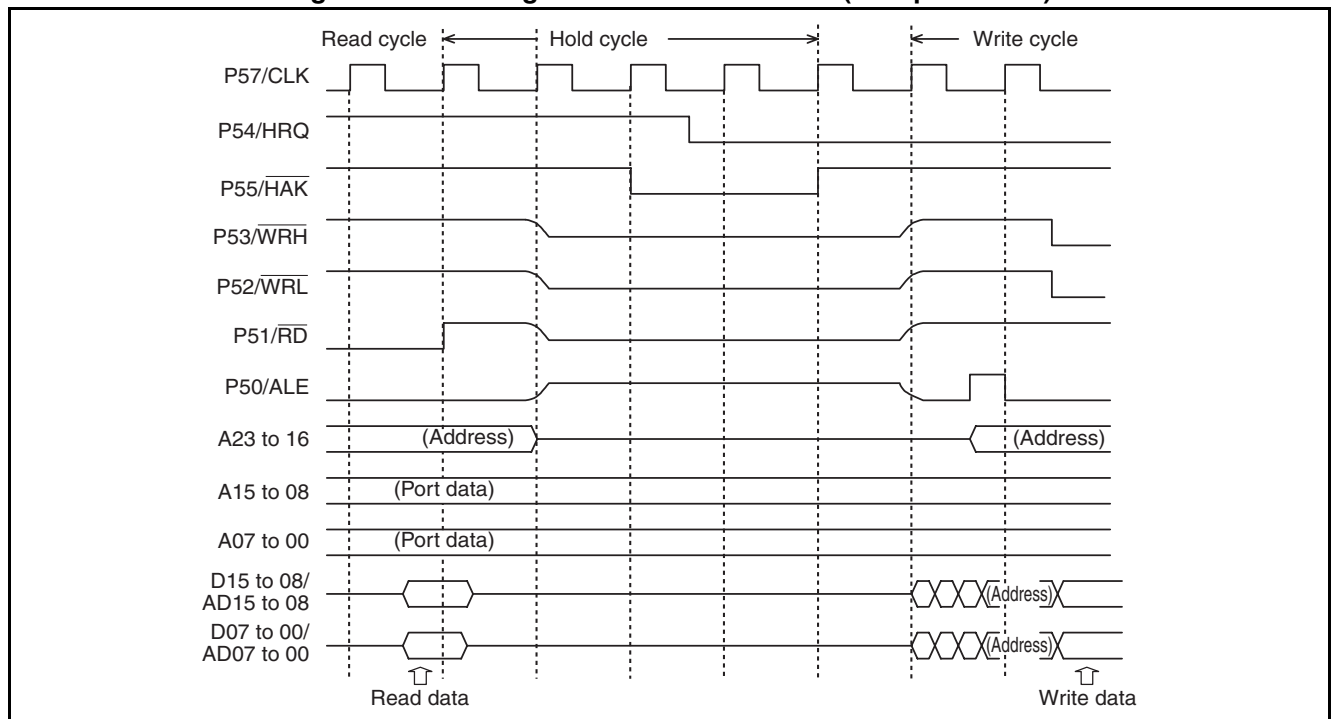
**Figure 7.5-7 Timing Chart of Hold Function (Non-multiplex Mode)**



## ■ Multiplex Mode

Figure 7.5-8 shows a timing chart of the multiplex-mode hold function in the external data bus 16-bit mode.

**Figure 7.5-8 Timing Chart of Hold Function (Multiplex Mode)**





## CHAPTER 8 I/O PORT

---

**This chapter explains the configuration and the functions of the registers used for the I/O port.**

---

8.1 Functions of I/O Port

8.2 Registers for I/O Port



## 8.1 Functions of I/O Port

---

This section outlines the functions of the I/O port.

---

### ■ Functions of I/O Port

The I/O port has functions to output data from the CPU to I/O pins and introduce the signals input to I/O pins to the CPU by using the port register (PDR). Furthermore, the I/O port enables input to and output from I/O pins to be set in any direction in units of bits by using the port direction register (DDR).

The MB90480/485 series has 84 input/output pins.

## 8.2 Registers for I/O Port

---

This section shows the configuration and explains the functions of the registers used for the I/O port.

---

### ■ Registers for I/O Port

The registers for the I/O port are listed below:

- Port registers (PDR0 to PDRA)
- Port direction registers (DDR0 to DDRA)
- Port input resistor registers (RDR0, RDR1)
- Port output pin registers (ODR7, ODR4)
- Analog input enable register (ADER)
- Up/down timer input enable register (UDER)

## 8.2.1 Port registers (PDR0 to PDRA)

This section shows the configuration and explains the functions of port registers (PDR0 to PDRA)

### ■ Port Registers (PDR0 to PDRA)

Figure 8.2-1 shows a list of port registers (PDR0 to PDRA).

**Figure 8.2-1 List of Port Registers (PDR0 to PDRA)**

| PDR0                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Initial value | Access |
|-----------------------------|-----|-----|-----|-----|-----|-----|-----|-----|---------------|--------|
| Address:000000 <sub>H</sub> | P07 | P06 | P05 | P04 | P03 | P02 | P01 | P00 | Undefined     | R/W *1 |
| PDR1                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000001 <sub>H</sub> | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 | Undefined     | R/W *1 |
| PDR2                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000002 <sub>H</sub> | P27 | P26 | P25 | P24 | P23 | P22 | P21 | P20 | Undefined     | R/W *1 |
| PDR3                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000003 <sub>H</sub> | P37 | P36 | P35 | P34 | P33 | P32 | P31 | P30 | Undefined     | R/W *1 |
| PDR4                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000004 <sub>H</sub> | P47 | P46 | P45 | P44 | P43 | P42 | P41 | P40 | Undefined     | R/W *1 |
| PDR5                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000005 <sub>H</sub> | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 | Undefined     | R/W *1 |
| PDR6                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000006 <sub>H</sub> | P67 | P66 | P65 | P64 | P63 | P62 | P61 | P60 | Undefined     | R/W *1 |
| PDR7                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000007 <sub>H</sub> | P77 | P76 | P75 | P74 | P73 | P72 | P71 | P70 | Undefined *2  | R/W *1 |
| PDR8                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000008 <sub>H</sub> | P87 | P86 | P85 | P84 | P83 | P82 | P81 | P80 | Undefined     | R/W *1 |
| PDR9                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:000009 <sub>H</sub> | P97 | P96 | P95 | P94 | P93 | P92 | P91 | P90 | Undefined     | R/W *1 |
| PDRA                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |               |        |
| Address:00000A <sub>H</sub> | -   | -   | -   | -   | PA3 | PA2 | PA1 | PA0 | Undefined     | R/W *1 |

\*1: R/W access to I/O ports slightly differs in operation from R/W access to memory. Be careful about such R/W access because it operates as follows:

- 0: Input mode

-- During reading: The level of the relevant pins is read and output.

-- During writing: Writing is performed on the latch for output.

- 1: Output mode

- During reading: The value of the data register latch is read and output.

- During writing: Output is to the relevant pins.

\*2: The initial value of MB90485 series is "11XXXXXXB"

## 8.2.2 Port direction registers (DDR0 to DDRA)

This section shows the configuration and explains the functions of port direction registers (DDR0 to DDRA.)

### ■ Port Direction Registers (DDR0 to DDRA)

Figure 8.2-2 shows a list of port direction registers (DDR0 to DDRA).

**Figure 8.2-2 List of Port Direction Registers (DDR0 to DDRA)**

|                             |                   |                   |     |     |     |     |     |     |                                     |        |
|-----------------------------|-------------------|-------------------|-----|-----|-----|-----|-----|-----|-------------------------------------|--------|
| DDR0                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   | Initial value                       | Access |
| Address:000010 <sub>H</sub> | D07               | D06               | D05 | D04 | D03 | D02 | D01 | D00 | 00000000 <sub>B</sub>               | R/W    |
| DDR1                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000011 <sub>H</sub> | D17               | D16               | D15 | D14 | D13 | D12 | D11 | D10 | 00000000 <sub>B</sub>               | R/W    |
| DDR2                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000012 <sub>H</sub> | D27               | D26               | D25 | D24 | D23 | D22 | D21 | D20 | 00000000 <sub>B</sub>               | R/W    |
| DDR3                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000013 <sub>H</sub> | D37               | D36               | D35 | D34 | D33 | D32 | D31 | D30 | 00000000 <sub>B</sub>               | R/W    |
| DDR4                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000014 <sub>H</sub> | D47               | D46               | D45 | D44 | D43 | D42 | D41 | D40 | 00000000 <sub>B</sub>               | R/W    |
| DDR5                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000015 <sub>H</sub> | D57               | D56               | D55 | D54 | D53 | D52 | D51 | D50 | 00000000 <sub>B</sub>               | R/W    |
| DDR6                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000016 <sub>H</sub> | D67               | D66               | D65 | D64 | D63 | D62 | D61 | D60 | 00000000 <sub>B</sub>               | R/W    |
| DDR7                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000017 <sub>H</sub> | D77 <sup>*1</sup> | D76 <sup>*2</sup> | D75 | D74 | D73 | D72 | D71 | D70 | 00000000 <sub>B</sub> <sup>*3</sup> | R/W    |
| DDR8                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000018 <sub>H</sub> | D87               | D86               | D85 | D84 | D83 | D82 | D81 | D80 | 00000000 <sub>B</sub>               | R/W    |
| DDR9                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:000019 <sub>H</sub> | D97               | D96               | D95 | D94 | D93 | D92 | D91 | D90 | 00000000 <sub>B</sub>               | R/W    |
| DDRA                        | 7                 | 6                 | 5   | 4   | 3   | 2   | 1   | 0   |                                     |        |
| Address:00001A <sub>H</sub> | -                 | -                 | -   | -   | DA3 | DA2 | DA1 | DA0 | - - - -0000 <sub>B</sub>            | R/W    |

<sup>\*1</sup>: The value of bit7 for MB90485 series is "-".  
<sup>\*2</sup>: The value of bit6 for MB90485 series is "-".  
<sup>\*3</sup>: The initial value of MB90485 series is "XX00000<sub>B</sub>".

R/W : Readable/Writable

#### ○ When each pin functions as a port

When each pin functions as a port, it controls the corresponding pin as follows:

- 0: Input mode
- 1: Output mode, which can be set to "0" by a reset.

### Notes:

- If this register is accessed with a command of the read-modify-write type (such as the bit-set command), the contents of the output registers corresponding to the other bits specified for input are replaced with the input value of the pin at the time of access even if the bit specified by the command is set to the required value. Therefore, when a pin for input is switched to a pin for output, be sure to define DDR after writing the desired value to PDR, and then switch it to a pin for output.
  - P77 and P76 of MB90485 series do not have DDR function. Data is always valid as a port. So set the PDR value to "1" when P77 and P76 are used as I<sup>2</sup>C pins (When using as P77 and P76, stop I<sup>2</sup>C).  
Also this port has the open drain output format (without P-ch), therefore it is necessary to set the PDR value to "1" in order to turn off the output transistor and to add a pull-up resistor to external output when this port is used as an input port.
-

## 8.2.3 Other registers

This section shows the configuration and explains the functions of registers other than port registers (PDR0 to PDRA) or port direction registers (DDR0 to DDRA).

### ■ Port Input Resistor Registers (RDR0, RDR1)

The bit configuration of port input resistor registers (RDR0, RDR1) is shown in the figure below.

|                             |      |      |      |      |      |      |      |      |                       |        |
|-----------------------------|------|------|------|------|------|------|------|------|-----------------------|--------|
| RDR0                        | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value         | Access |
| Address:00001C <sub>H</sub> | RD07 | RD06 | RD05 | RD04 | RD03 | RD02 | RD01 | RD00 | 00000000 <sub>B</sub> | R/W    |
| RDR1                        | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |                       |        |
| Address:00001D <sub>H</sub> | RD17 | RD16 | RD15 | RD14 | RD13 | RD12 | RD11 | RD10 | 00000000 <sub>B</sub> | R/W    |

R/W : Readable/Writable

Port input resistor registers (RDR0, RDR1) specify whether or not there is pull-up resistor in the input mode.

- 0: Pull-up resistor in the input mode
- 1: No pull-up resistor in the input mode

These registers have no function in the output mode (no pull-up resistor).

The input or output mode is determined by the setting on the port direction register (DDR).

During a stoppage (SPL = 1), the lack of pull-up resistor is specified (high impedance).

This function is prohibited if an external bus is used. Do not write to this register.

### ■ Port Output Pin Registers (ODR7, ODR4)

The bit configuration of port output pin registers (ODR7, ODR4) is shown in the figure below.

|                             |                    |                    |      |      |      |      |      |      |                                     |        |
|-----------------------------|--------------------|--------------------|------|------|------|------|------|------|-------------------------------------|--------|
| ODR7                        | 7                  | 6                  | 5    | 4    | 3    | 2    | 1    | 0    | Initial value                       | Access |
| Address:00001E <sub>H</sub> | OD77 <sup>*1</sup> | OD76 <sup>*2</sup> | OD75 | OD74 | OD73 | OD72 | OD71 | OD70 | 00000000 <sub>B</sub> <sup>*3</sup> | R/W    |
| ODR4                        | 7                  | 6                  | 5    | 4    | 3    | 2    | 1    | 0    |                                     |        |
| Address:00001B <sub>H</sub> | OD47               | OD46               | OD45 | OD44 | OD43 | OD42 | OD41 | OD40 | 00000000 <sub>B</sub>               | R/W    |

\*1: The value of bit7 for only MB90485 series is "-".  
 \*2: The value of bit6 for only MB90485 series is "-".  
 \*3: The initial value of only MB90485 series is "XX000000<sub>B</sub>".

R/W : Readable/Writable

Port output pin registers (ODR7, ODR4) perform open drain control in the output mode.

- 0: Sets a standard output port in the output mode
- 1: Sets an open-drain output port in the output mode

Port output pin registers (ODR7, ODR4) have no function in the input mode (Output Hi-Z).

The input or output mode is determined by the setting of the port direction register (DDR).

This function is prohibited when an external bus is used. Do not write to this register.

### ■ Analog Input Enable Register (ADER)

The bit configuration of the analog input enable register (ADER) is shown in the figure below.

| ADER                        | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value         | Access |
|-----------------------------|------|------|------|------|------|------|------|------|-----------------------|--------|
| Address:00001F <sub>H</sub> | ADE7 | ADE6 | ADE5 | ADE4 | ADE3 | ADE2 | ADE1 | ADE0 | 11111111 <sub>B</sub> | R/W    |
| R/W : Readable/Writable     |      |      |      |      |      |      |      |      |                       |        |

The analog input enable register (ADER) controls the pins of port 6 as follows:

- 0: Sets the port I/O mode
- 1: Set the analog I/O mode. "1" is restored by a reset.

In the MB90480/485 series, each bit is set as follows:

- ADE0: P60/AN0
- ADE1: P61/AN1
- ADE2: P62/AN2
- ADE3: P63/AN3
- ADE4: P64/AN4
- ADE5: P65/AN5
- ADE6: P66/AN6
- ADE7: P67/AN7

### ■ Up/down Timer Input Enable Register (UDER)

The bit configuration of the up/down timer input enable register (UDER) is shown in the figure below.

| UDER                        | 7 | 6 | 5    | 4    | 3    | 2    | 1    | 0    | Initial value         | Access |
|-----------------------------|---|---|------|------|------|------|------|------|-----------------------|--------|
| Address:00000B <sub>H</sub> | - | - | UDE5 | UDE4 | UDE3 | UDE2 | UDE1 | UDE0 | XX000000 <sub>B</sub> | R/W    |
| R/W : Readable/Writable     |   |   |      |      |      |      |      |      |                       |        |

The up/down timer input enable register (UDER) controls the pins of port 3 as follows:

- 0: Sets the port input mode
- 1: Sets the up/down timer input mode. "0" is restored by a reset.

In the MB90480/485 series, each bit is set as follows:

- UDE0: P30/AIN0
- UDE1: P31/BIN0
- UDE2: P32/ZIN0
- UDE3: P33/AIN1
- UDE4: P34/BIN1
- UDE5: P35/ZIN1

## CHAPTER 9    TIME-BASE TIMER

---

**This chapter explains the functions and operations of the time-base timer.**

---

- 9.1 Overview of Time-Base Timer
- 9.2 Time-Base Timer Configuration
- 9.3 Time-Base Timer Control Register (TBTC)
- 9.4 Time-Base Timer Interrupt
- 9.5 Time-Base Timer Operation
- 9.6 Notes on Using Time-Base Timer
- 9.7 Sample Programs of Time-Base Timer



## 9.1 Overview of Time-Base Timer

The time-base timer, which is an 18-bit free-run counter (time-base timer counter) that counts up in synchronization with the internal count clock (the source clock frequency divided by 2), has the interval timer function to enable selection for four types of interval times. Furthermore, it also has functions to supply operation clocks including timer output for the oscillation stabilization wait time as well as the watchdog timer.

### ■ Interval Timer Function

The interval timer function generates repetitive interval interrupt requests.

- An interrupt request is generated when the bit for the interval timer in the time-base counter overflows.
- The bit for the interval timer (interval time) can be selected out of four types.

Table 9.1-1 shows the interval time of the time-base timer.

**Table 9.1-1 Interval Time of Time-base Timer**

| Internal count clock cycle | Interval cycle                           |
|----------------------------|--|
| 2 / HCLK (0.5 $\mu$ s)     | $2^{12}$ / HCLK (approximately 1.0 ms)   |
|                            | $2^{14}$ / HCLK (approximately 4.1 ms)   |
|                            | $2^{16}$ / HCLK (approximately 16.4 ms)  |
|                            | $2^{19}$ / HCLK (approximately 131.1 ms) |

HCLK: Oscillation clock

The value during operation of the oscillation clock at 4 MHz is shown in ( ).

## ■ Clock Supplying Function

The clock supplying function is the function supplying the timer for the oscillation stabilization wait time and the operation clocks to some peripheral functions. Table 9.1-2 lists the cycles of the clocks supplied by the time-base timer to individual peripheral functions.

**Table 9.1-2 Clock Cycles Supplied by Time-base Timer**

| Function to which clock is supplied | Clock cycle                              | Remarks   |
|-------------------------------------|--|---|
| Oscillation stabilization wait time | $2^{13}$ / HCLK (approximately 2.0 ms)   | Oscillation stabilization wait time for ceramic resonator |
|                                     | $2^{15}$ / HCLK (approximately 8.2 ms)   | Oscillation stabilization wait time for crystal resonator |
|                                     | $2^{17}$ / HCLK (approximately 32.8 ms)  |   |
| Watchdog timer                      | $2^{12}$ / HCLK (approximately 1.0 ms)   | Up-count clock for watchdog timers                        |
|                                     | $2^{14}$ / HCLK (approximately 4.1 ms)   |   |
|                                     | $2^{16}$ / HCLK (approximately 16.4 ms)  |   |
|                                     | $2^{19}$ / HCLK (approximately 131.1 ms) |   |

HCLK: Oscillation clock

The value during operation of the oscillation clock at 4 MHz is shown in ( ).

Because the oscillation cycle immediately after the start of oscillation is unstable, the oscillation stabilization wait time is merely a guideline.

## 9.2 Time-Base Timer Configuration

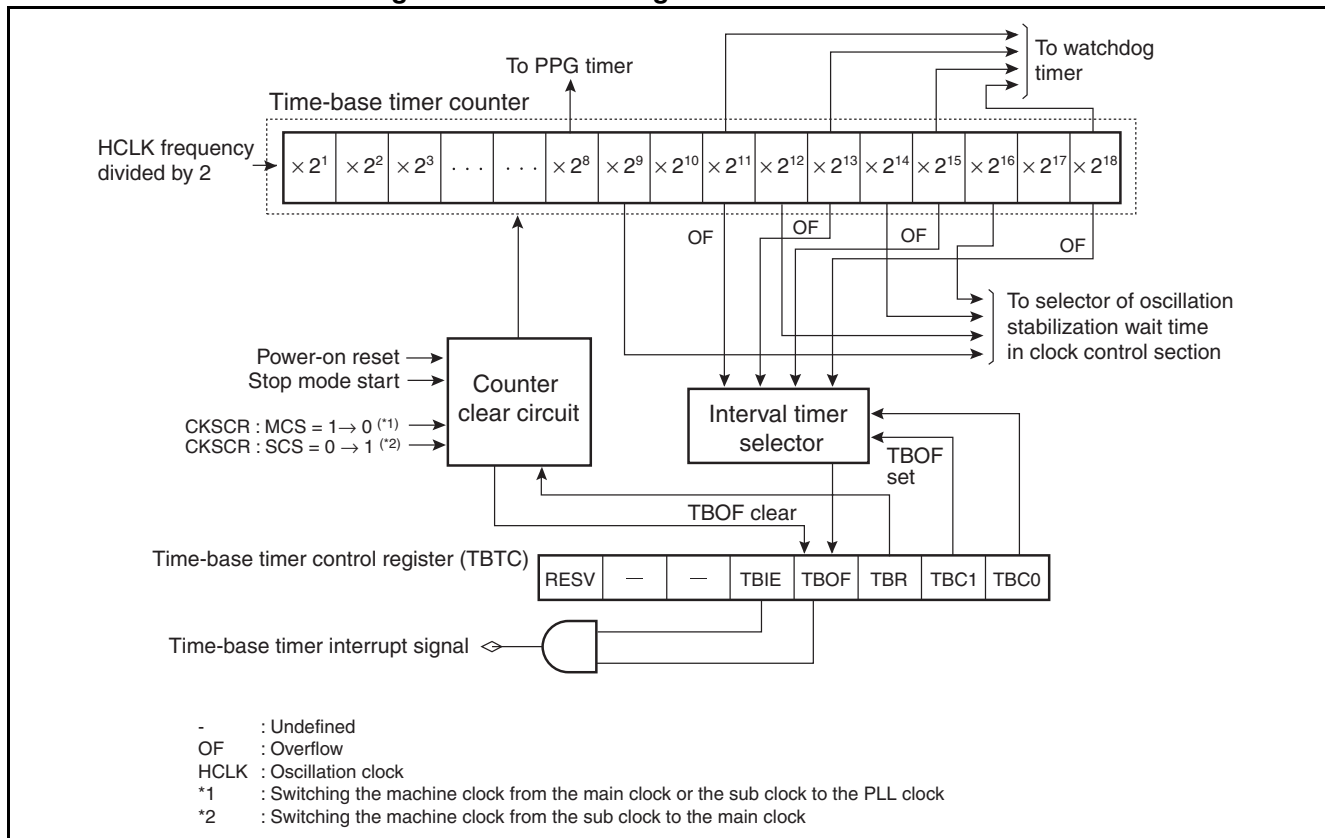
The time-base timer is composed of the following four blocks:

- Time-base timer counter
- Counter clear circuit
- Interval timer selector
- Time-base timer control register (TBTC)

### ■ Block Diagram of Time-base Timer

Figure 9.2-1 is a block diagram of the time-base timer.

**Figure 9.2-1 Block Diagram of Time-base Timer**



#### ○ Time-base timer counter

This is an 18-bit up-counter that uses the oscillation clock (HCLK) frequency divided by 2 as the count clock.

#### ○ Counter clear circuit

This circuit clears the counter at the time of writing of "0" to the time-base timer initializing bit (TBR) in the time-base timer control register (TBTC), a power-on reset, a transition to the main stop mode, a transition to the PLL stop mode, switching from the main clock mode to the PLL clock mode, switching from sub clock mode to the PLL clock mode, and switching from the sub clock mode to main clock mode.

- **Interval timer selector**

Selects one of the four types for time-base timer counter output. Overflow of the selected bit causes an interrupt.

- **Time-base timer control register (TBTC)**

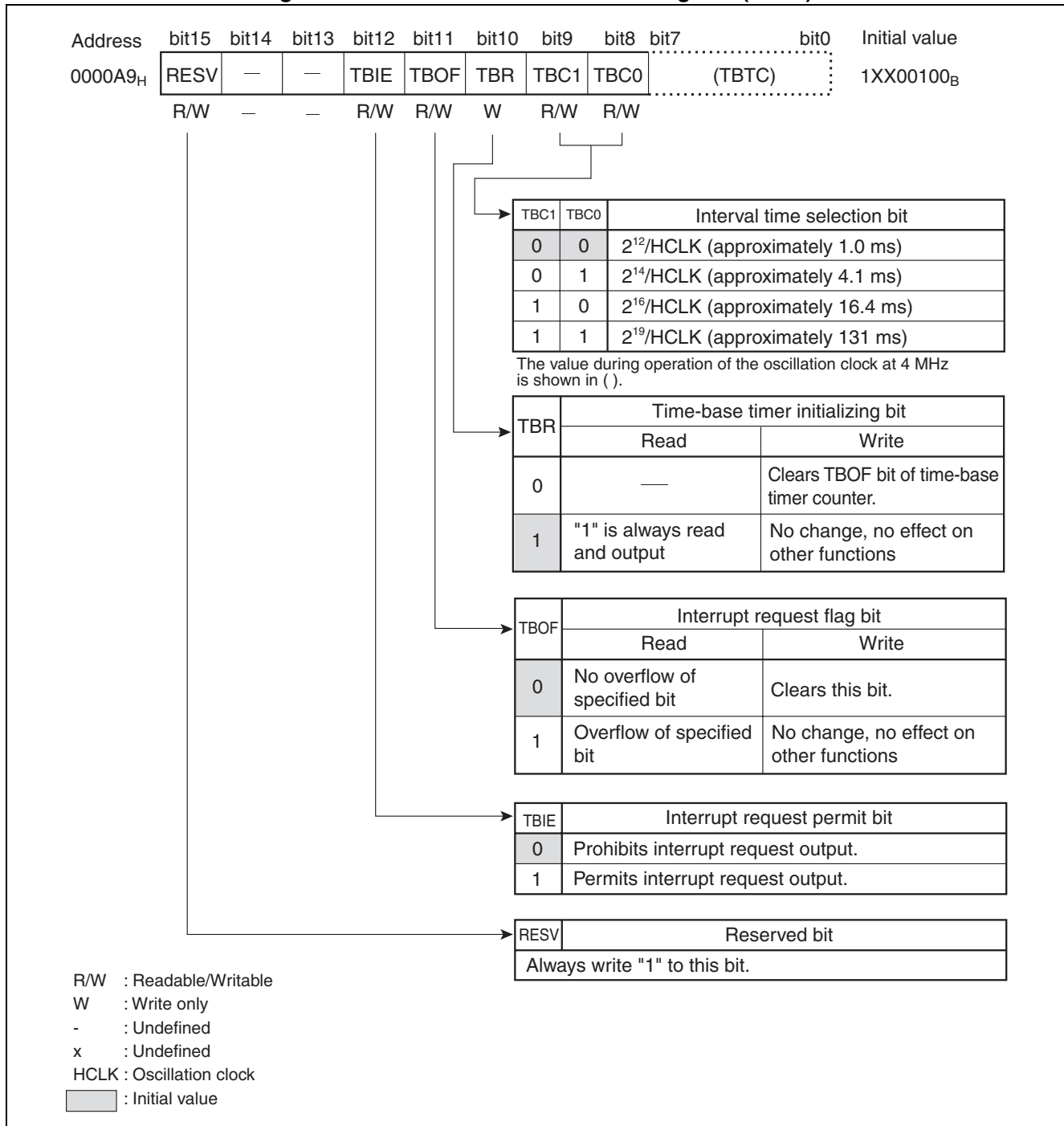
Selects the interval time, clears the counter, controls interrupt requests, and checks the current state.

## 9.3 Time-Base Timer Control Register (TBTC)

This register selects the interval time, clears the counter, controls interrupt requests, and checks the state.

### ■ Time-base Timer Control Register (TBTC)

Figure 9.3-1 Time-base Timer Control Register (TBTC)



**Table 9.3-1 Functions of Bits in Time-base Timer Control Register (TBTC)**

| Bit name        |   | Function  |
|-----------------|---|---|
| bit15           | RESV:<br>Reserved bit                         | Always write "1" to this bit.   |
| bit14,<br>bit13 | Undefined bits                                | <ul style="list-style-type: none"> <li>Undefined value in reading</li> <li>No effect on write operation</li> </ul>  |
| bit12           | TBIE:<br>Interrupt request<br>permit bit      | <p>This bit permits or prohibits output of interrupt requests to the CPU.</p> <ul style="list-style-type: none"> <li>An interrupt request is output if this bit and the interrupt request flag bit (TBOF) are set to "1".</li> </ul>  |
| bit11           | TBOF:<br>Interrupt request flag<br>bit        | <p>This bit is set to "1" when the bit specified by the time-base timer counter overflows.</p> <ul style="list-style-type: none"> <li>An interrupt request is output if this bit and the interrupt request permit bit (TBIE) are set to "1".</li> <li>This bit is cleared by writing "0" but is not changed by writing "1" so that this operation does not affect other functions.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>Clearing of the interrupt request flag bit (TBOF) must be implemented in the state where the time-base timer interrupt is prohibited by the interrupt request permit bit (TBIE) or by the interrupt level mask register (ILM) setting of the processor status (PS).</li> <li>This bit is cleared to "0" by writing "0", a transition to the main stop mode, a transition to the PLL stop mode, a transition from the sub clock mode to the main clock mode, a transition from the sub clock mode to the PLL clock mode, a transition from the main clock mode to the PLL clock mode, writing "0" to the time-base timer initializing bit (TBR) or a reset.</li> </ul> |
| bit10           | TBR:<br>Time-base timer<br>initializing bit   | <p>This bit clears the time-base timer counter.</p> <ul style="list-style-type: none"> <li>When "0" is written to this bit, the counter is cleared and the TBOF bit is cleared. This bit is not changed by writing "1" so that this operation does not affect other functions.</li> </ul> <p>[Reference]<br/>The readout value is always "1".</p>   |
| bit9,<br>bit8   | TBC1, TBC0:<br>Interval time selection<br>bit | <p>This bit specifies the cycle of the interval timer.</p> <ul style="list-style-type: none"> <li>The bit for the interval timer in the time-base timer counter is specified.</li> <li>The interval time can be selected out of four types.</li> </ul>  |

## 9.4 Time-Base Timer Interrupt

---

The time-base timer can generate the interrupt request caused by an overflow of the specified bit in the time-base timer counter (interval timer function).

---

### ■ Time-base Timer Interrupt

When the time-base timer counter counts up using the internal count clock and the bit for the selected interval timer overflows, the interrupt request flag bit (TBOF) of the time-base timer control register (TBTC) is set to "1". In this event, if an interrupt request is permitted because the interrupt request permit bit (TBIE) is set to "1", an interrupt request is generated in the CPU. Clear this interrupt request by writing "0" to the TBOF bit using the interrupt processing routine. Incidentally, TBOF bit is set when the specified bit overflows regardless for the value for the interrupt request permit bit (TBIE).

Note:

Clearing of the interrupt request flag bit (TBOF) in the time-base timer control register (TBTC) must be implemented in the state where the time-base timer interrupt is prohibited by the interrupt request permit bit (TBIE) or by the interrupt level mask register (ILM) setting of the processor status (PS).

#### References:

- If the TBOF bit is set to "1", an interrupt request is generated immediately when the TBIE bit is switched from prohibit (0) to permit (1).
- $\mu$ DMAC cannot be used in the time-base timer.

### ■ Time-base Timer Interrupt and $\mu$ DMAC

Table 9.4-1 lists time-base timer interrupts and  $\mu$ DMAC.

**Table 9.4-1 Time-base Timer Interrupt and  $\mu$ DMAC**

| Interrupt No. | Interrupt level setting register |                     | Address of the vector table |                     |                     | $\mu$ DMAC |
|---------------|----------------------------------|---------------------|-----------------------------|---------------------|---------------------|------------|
|               | Register name                    | Address             | Low-order                   | High-order          | Bank                |            |
| #41           | ICR15                            | 0000BF <sub>H</sub> | FFFF58 <sub>H</sub>         | FFFF59 <sub>H</sub> | FFFF5A <sub>H</sub> | x          |

x: Not used

Note:

ICR15 is commonly used by the time-base timer interrupt, the watch timer interrupt, and flash write. Although interrupt can be used for these three purposes, the interrupt level is the same.

## 9.5 Time-Base Timer Operation

The time-base timer has the interval timer function as well as the clock supplying function for some peripheral functions.

### ■ Operation of Interval Timer Function (Time-base Timer)

The interval timer function generates interrupt requests at any defined interval times. For its operation as an interval timer, the settings shown in Figure 9.5-1 are required.

**Figure 9.5-1 Time-base Timer Settings**

| Address                  | bit15 | bit14 | bit13 | bit12 | bit11 | bit10 | bit9 | bit8 | bit7   | bit0 |
|--------------------------|-------|-------|-------|-------|-------|-------|------|------|--------|------|
| 0000A9 <sub>H</sub> TBTC | RESV  | —     | —     | TBIE  | TBOF  | TBR   | TBC1 | TBC0 | (WDTC) |      |
|                          | 1     | —     | —     | ⊙     | 0     | 0     | ⊙    | ⊙    |        |      |

⊙ : Bits in use  
 — : Unused bits  
 0 : "0" is set  
 1 : "1" is set

- The time-base timer counter continues counting up in synchronization with the internal count clock (the source clock frequency divided by 2) as long as the clock is oscillating.
- When the counter has been cleared, the starts from "0", and an overflow of the bit used for the interval timer sets the interrupt request flag bit (TBOF) to "1". In this event, if the interrupt request output is permitted (TBIE = 1), interrupts are generated at the selected interval times with the clearing time used as a reference time point.
- The interval time may become longer than the specified time,  $e$ , when the time-base timer was cleared.

### ■ Timer Function for Oscillation Stabilization Wait Time

The time-base timer can also be used as the oscillation clock as well as the timer for the oscillation stabilization wait time of the PLL clock. The oscillation stabilization wait time is the time in which counting starts when the counter is set to "0" (clearing of counter) and continues until an overflow occurs in the bit for the oscillation stabilization wait time. However, during return from the time-base timer mode to the PLL clock mode or main clock mode, the waiting time differs because the time-base timer counter is not cleared and the time count does not start from zero. Table 9.5-1 explains the time-base timer counter clear operation and oscillation stabilization wait time.



**Table 9.5-1 Time-base Timer Counter Clear Operation and Oscillation Stabilization Wait Time.**

| Operation   | Counter clear | TBOF clear | Oscillation stabilization wait time               |
|---|---------------|------------|---|
| Writing "0" to time-base timer initializing bit (TBR) for time-base timer control register (TBTC) | ○             | ○          |   |
| Power-on reset  | ○             | ○          | Oscillation stabilization wait time of main clock |
| Watchdog reset  | ×             | ○          |   |
| Release of the main stop mode   | ○             | ○          |   |
| Release of the PLL stop mode  | ○             | ○          |   |
| Release of the sub stop mode  | ×             | ×          | Oscillation stabilization wait time of sub clock  |
| Switching from main clock mode to PLL clock mode (SCM: transition from 1 to 0)                    | ○             | ○          | Oscillation stabilization wait time of PLL clock  |
| Transition from sub clock mode to main clock mode (SCM: transition from 0 to 1)                   | ○             | ○          | Oscillation stabilization wait time of main clock |
| Release of time-base timer mode   | ×             | ×          |   |
| Release of sleep mode   | ×             | ×          |   |

○: Cleared

×: Not cleared

## ■ Clock Supplying Function

The time-base timer supplies a clock to the watchdog timer. Clearing of the time-base timer counter affects the operation of the watchdog timer.

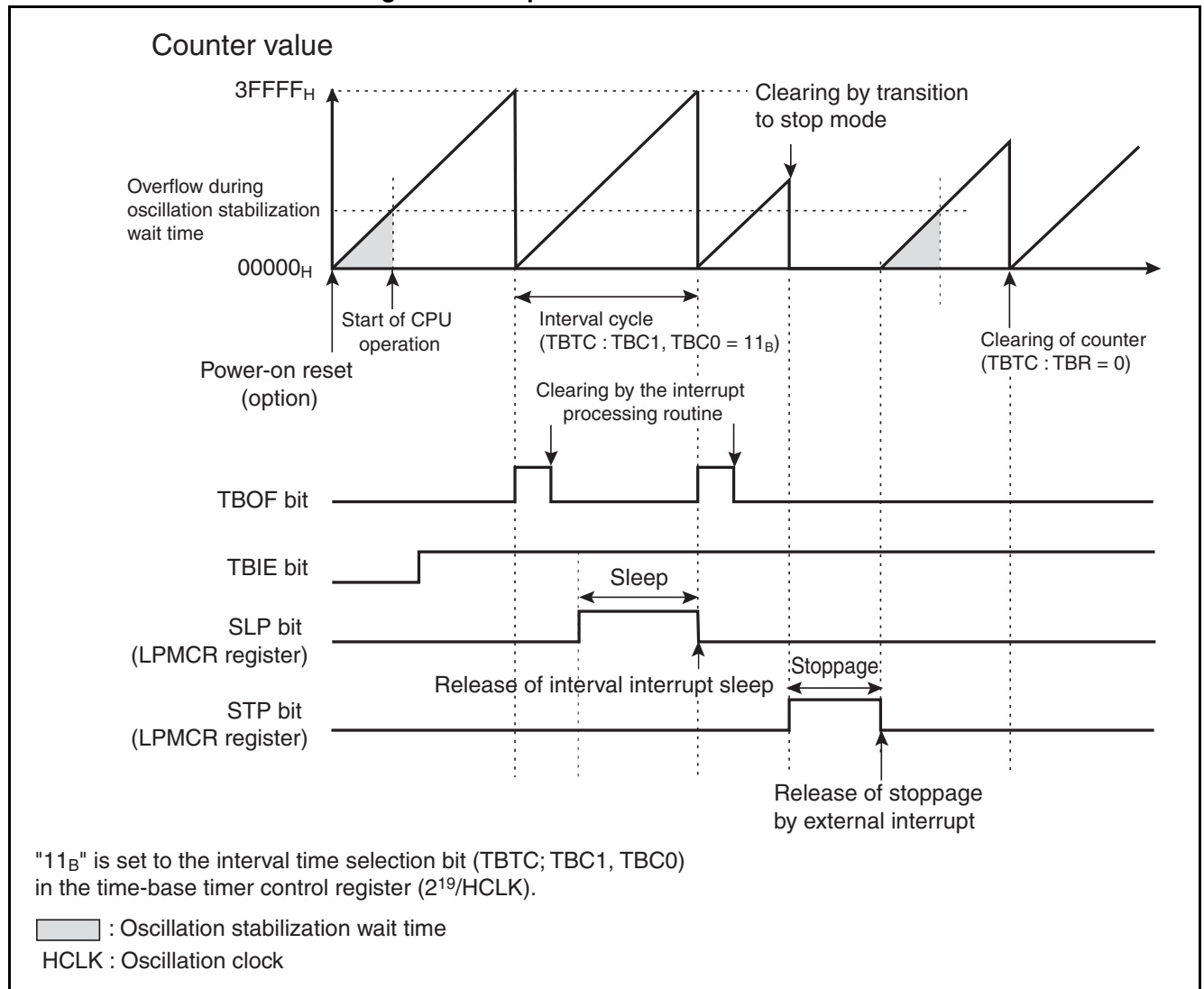
## ■ Operation of Time-base Timer

Operations in the following states are shown in Figure 9.5-2:

- Where the power-on reset has occurred
- Where transition to the sleep mode has occurred during processing for the interval timer function
- Where transition to the stop mode has occurred
- Where clearing of the counter is requested

Transition to the stop mode clears the time-base timer to stop operation. After restoration from the stop mode, the time-base timer starts an up-count of the oscillation stabilization wait time.

**Figure 9.5-2 Operation of Time-base Timer**



## 9.6 Notes on Using Time-Base Timer

---

This section explains notes on using the time-base timer, including the effects of clearing an interrupt request or clearing the time-base timer on peripheral functions.

---

### ■ Notes on Using Time-base Timer

#### ○ Clearing an interrupt request

Clearing the interrupt request flag bit (TBOF) of the time-base timer control register (TBTC) must be implemented in the state where the time-base timer interrupt is masked by the interrupt request permit bit (TBIE) or by the interrupt level mask register (ILM) setting of the processor status (PS).

#### ○ Effect of clearing the time-base timer

Clearing the time-base timer counter affects the following:

- Operations where the interval timer function (interval interrupt) is used by the time-base timer
- Operations using the watchdog timer

#### ○ Use of the timer for the oscillation stabilization wait time

When power is turned on, the oscillation clock is stopped in the main stop mode. In such a case, after the oscillator starts operating, the oscillation stabilization wait time of the oscillation clock must be provided by using as a timing reference the operation clock supplied by the time-base timer. An appropriate oscillation stabilization wait time must be selected depending on the type of oscillator (resonator) connected to the high-speed oscillation pin. See Section "5.5 Oscillation Stabilization Wait Time", for details.

#### ○ Caution on using peripheral functions whose operation clock is supplied from the time-base timer

In a mode where the main clock stops, the counter is cleared and the time-base timer stops operating. Furthermore, because the clock supplied by the time-base timer is reset to the initial state and is supplied again when the time-base timer counter is cleared, the period of "H" level may become shorter or the period of "L" level may become longer by a maximum of a 1/2 cycle. Although the clock for the watchdog timer is also supplied from the initial state, the watchdog timer operates at normal cycles because the watchdog time counter is cleared at the same time.

## 9.7 Sample Programs of Time-Base Timer

Sample programs for the time-base timer are shown below.

### ■ Sample Programs of Time-base Timer

#### ○ Specifications for processing

Repetitively generate an interval interrupt of  $2^{12}/\text{HCLK}$  (oscillation clock). The interval time in this case is approximately 1.0 ms (when operating at 4 MHz).

#### ○ Sample coding

```

ICR12 EQU    0000BCH                ; Interrupt control register for time-base timer
TBTC   EQU    0000A9H                ; Time-base timer control register
TBOF   EQU    TBTC:3                ; Interrupt request flag bit
;----- Main program -----
CODE   CSEG
START:
;      :                               ; Assumption that stack pointer
;                               ; (SP), etc., have been initialized
      AND     CCR, #0BFH              ; Disabling interrupts
      MOV     I:ICR12, #00H           ; Interrupt level 0 (highest)
      MOV     I:TBTC, #10010000B     ; Three high-order bits must be fixed
;                               ; Permitting interrupts, clearing of TBOF
;                               ; Clearing the counter
;                               ; Selection for interval time of  $2^{12}/\text{HCLK}$ 
      MOV     ILM, #07H              ; Setting ILM in PS to level 7
      OR      CCR, #40H              ; Enabling interrupts
LOOP:  MOV     A, #00H                ; Infinite loop
      MOV     A, #01H
      BRA     LOOP
;----- Interrupt program -----
WARI:
      CLR     bit BOF                ; Clearing the interrupt request flag
;      :
;      User processing
;      :
      RETI                          ; Restoration from interrupt
CODE   ENDS
;----- Specifying vectors -----
VECT   CSEG    ABS=0FFH
      ORG     0FF6CH                ; Specifying the interrupt vector
      DSL     WARI
      ORG     0FFDCH                ; Specifying the reset vector
      DSL     START
      DB      00H                    ; Setting to the single-chip mode
VECT   ENDS
      END     START

```



# CHAPTER 10 WATCHDOG TIMER

---

**This chapter provides an overview of watchdog timer, explains control register, configuration, operations and shows the precautions on use, and sample program.**

---

- 10.1 Overview of Watchdog Timer
- 10.2 Watchdog Timer Control Register (WDTC)
- 10.3 Watchdog Timer Configuration
- 10.4 Watchdog Timer Operation
- 10.5 Notes on Using Watchdog Timer
- 10.6 Sample Programs of Watchdog Timer

## 10.1 Overview of Watchdog Timer

The watchdog timer is a 2-bit counter that uses the output of the time-base timer or the watch timer as the count clock, and if it is not cleared within a certain period of time after startup, this timer resets the CPU.

### ■ Functions of Watchdog Timer

The watchdog timer is a counter used to prevent runaway programs. Once it is started, this timer must be cleared periodically within a certain period of time. If it is not cleared within a certain period of time because a program is running in an infinite loop, it generates a watchdog reset to the CPU. The interval time of the watchdog timer can be specified on the WT1 and WT0 bits in the watchdog timer control register (WDTC), as shown in Table 10.1-1. If the watchdog timer is not cleared, a watchdog reset is generated between the minimum time and the maximum time. Be sure to clear the timer within the minimum time.

The output destination of the clock source is set by the watchdog clock selection bit (WTC:WDCS) of the clock timer control register.

**Table 10.1-1 Interval Time for Watchdog Timer**

| WT1 | WT0 | WDCS                             | SCM | Interval time           |                         | Number of clock cycles                   |
|-----|-----|----------------------------------|-----|-------------------------|-------------------------|--|
|     |     |                                  |     | Minimum *               | Maximum *               |  |
| 0   | 0   | 1                                | 1   | Approximately 3.58 ms   | Approximately 4.61 ms   | $(2^{14} \pm 2^{11})/\text{HCLK cycles}$ |
| 0   | 1   | 1                                | 1   | Approximately 14.33 ms  | Approximately 18.43 ms  | $(2^{16} \pm 2^{13})/\text{HCLK cycles}$ |
| 1   | 0   | 1                                | 1   | Approximately 57.23 ms  | Approximately 73.73 ms  | $(2^{18} \pm 2^{15})/\text{HCLK cycles}$ |
| 1   | 1   | 1                                | 1   | Approximately 458.75 ms | Approximately 589.82 ms | $(2^{21} \pm 2^{18})/\text{HCLK cycles}$ |
| 0   | 0   | Combination other than the above |     | Approximately 0.457 s   | Approximately 0.576 s   | $(2^{12} \pm 2^9)/\text{SCLK cycles}$    |
| 0   | 1   |                                  |     | Approximately 3.584 s   | Approximately 4.608 s   | $(2^{15} \pm 2^{12})/\text{SCLK cycles}$ |
| 1   | 0   |                                  |     | Approximately 7.168 s   | Approximately 9.216 s   | $(2^{16} \pm 2^{13})/\text{SCLK cycles}$ |
| 1   | 1   |                                  |     | Approximately 14.336 s  | Approximately 18.432 s  | $(2^{17} \pm 2^{14})/\text{SCLK cycles}$ |

\*: Values during operation of the oscillation clock (HCLK) at 4 MHz and the sub clock (SCLK) at 32 kHz frequency divided by 4 (= 8 kHz)

The maximum and minimum watchdog timer interval times and the number of the oscillation clock cycles depend on the timing of the clear operation. The interval time is 3.5 to 4.5 times as large as the cycle of the count clock (clock supplied by the time-base timer).

For the watchdog timer interval times, see Section "10.4 Watchdog Timer Operation".

#### Note:

The watchdog counter is composed of a 2-bit counter to count the carry signals of the time-base timer. Therefore, if the time-base timer counter has been cleared, the time until the occurrence of watchdog reset may become longer than the specified time.

To use the sub clock as a machine clock, please select the output of the clock timer by setting the watchdog timer clock source selection bit (WDCS) of the clock timer control register (WTC) to "0".

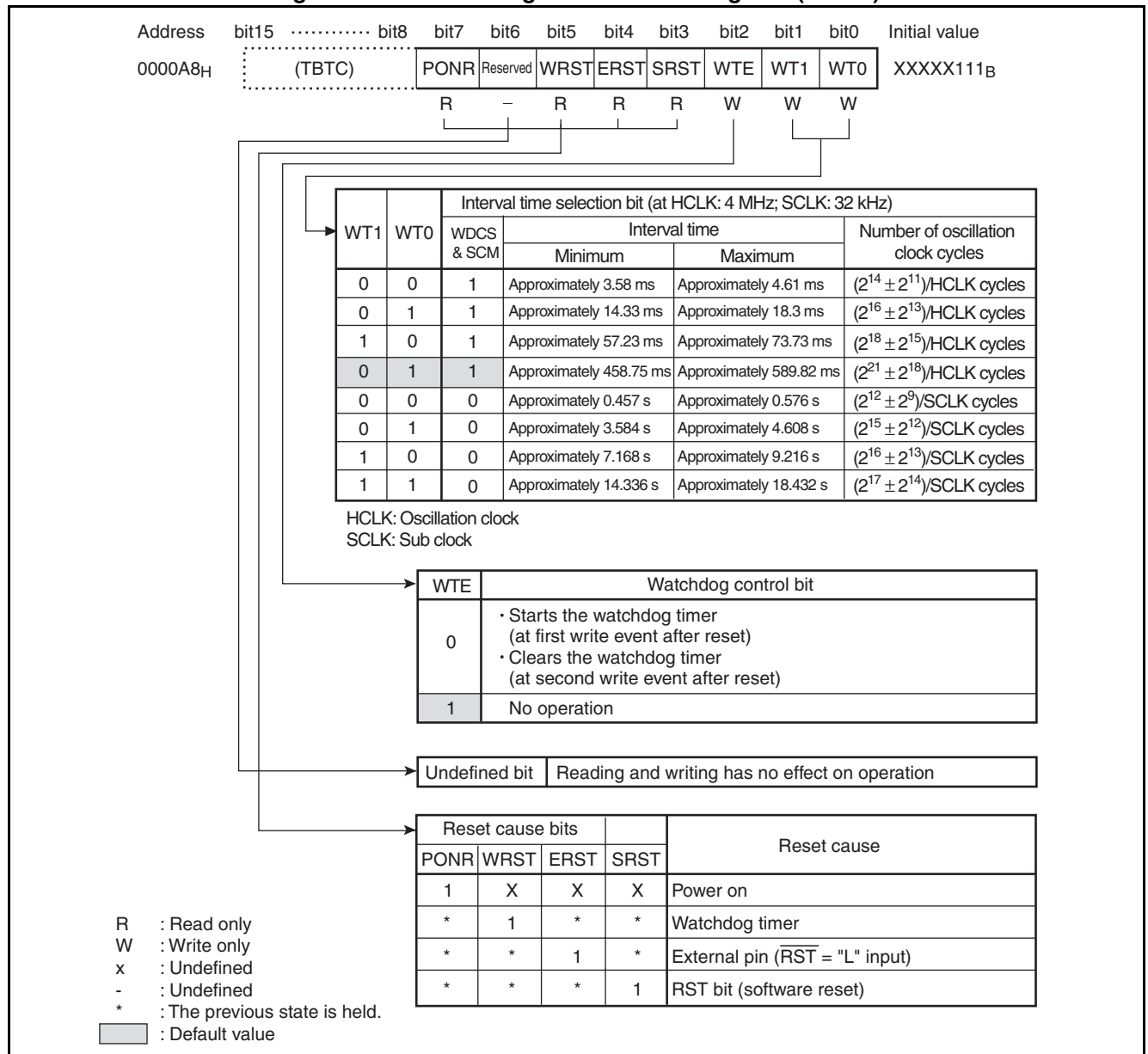
## 10.2 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) is used for the start and clearing of the watchdog timer and the display of reset causes.

### ■ Watchdog Timer Control Register (WDTC)

Figure 10.2-1 shows the configuration of the watchdog timer control register (WDTC), and Table 10.2-1 explains the function of each bit in the WDTC register.

**Figure 10.2-1 Watchdog Timer Control Register (WDTC)**



The interval time is 3.5 to 4.5 times as large as the count clock (output value of the time-base timer) cycle. See Section "10.4 Watchdog Timer Operation", for details.



Table 10.2-1 Function of Bits in Watchdog Timer Control Register (WDTC)

| Bit name                                 |  |                              | Function  |
|--|--|------------------------------|---|
| bit7,<br>bit6,<br>bit5,<br>bit4,<br>bit3 | PONR<br>Reserved<br>WRST<br>ERST<br>SRST | Reset cause bits             | <p>These are read-only bits that indicate reset causes. Each of these bits is set to "1" when the corresponding reset cause has occurred.</p> <ul style="list-style-type: none"> <li>• All of these bits are cleared to "0" after reading of the WDTC register.</li> <li>• When power is turned on, the contents of bits other than the PONR bit are not assured. Therefore, if the PONR bit is set to "1", ignore the contents of other bits.</li> </ul>   |
| bit2                                     | WTE                                      | Watchdog control bit         | <p>This is a bit for starting or clearing the watchdog timer.</p> <ul style="list-style-type: none"> <li>• When "0" is written to this bit, the watchdog timer is started (at the first write event after a reset) or the 2-bit counter is cleared (at the second and succeeding write events after a reset).</li> <li>• Writing "1" does not affect operation.</li> </ul>  |
| bit1,<br>bit0                            | WT1<br>WT0                               | Interval time selection bits | <ul style="list-style-type: none"> <li>• These are the bits for selecting the interval time of the watchdog timer.</li> <li>• The interval time varies, as shown in Figure 10.2-1, between the case where the sub clock mode is selected as the clock mode (the sub clock display bit (SCM) of the clock selection register (CKSCR) is "0") or the clock source of the watchdog timer is set to the watch timer by the watch timer control register (WTC) (the watchdog timer clock source selection bit (WDCS) is set to "0") and the case where the main clock mode or PLL clock mode is selected as the clock mode while the WDCS bit of WTC is set to "1".</li> <li>• Only data that has been defined before the start of the watchdog timer are effective.</li> <li>• Any data that is written after the start of the watchdog timer is ignored</li> <li>• These bits are write-only.</li> </ul> |

## 10.3 Watchdog Timer Configuration

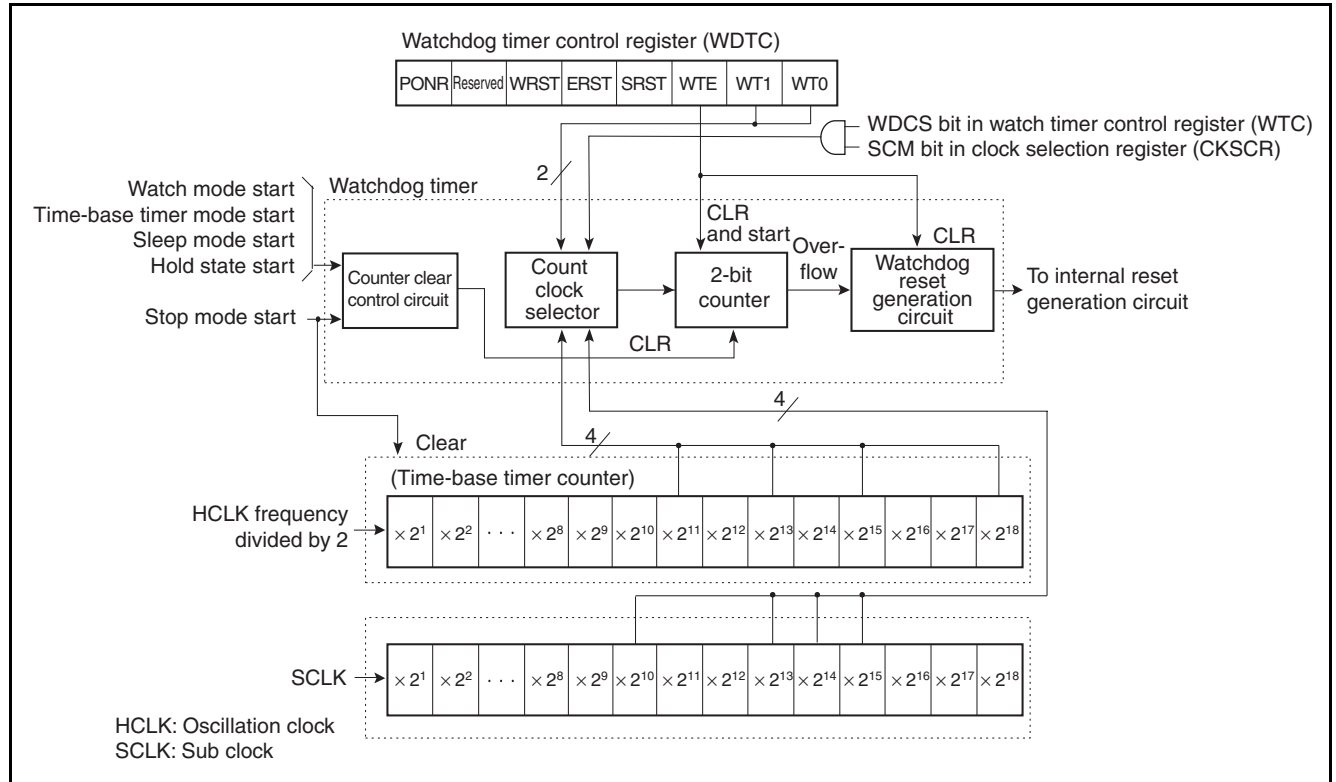
The watchdog timer is composed of the following five blocks:

- Count clock selector
- Watchdog counter (2-bit counter)
- Watchdog reset generation circuit
- Counter clear control circuit
- Watchdog timer control register (WDTC)

### ■ Block Diagram of Watchdog Timer

Figure 10.3-1 is a block diagram of the watchdog timer.

Figure 10.3-1 Block Diagram of Watchdog Timer



#### ○ Count clock selector

This circuit selects the count clock of the watchdog timer from among four types of time-base timer output and four types of watch timer output. This selection determines the time for generating a watchdog reset.

#### ○ Watchdog counter (2-bit counter)

This is a 2-bit up-counter that uses time-base timer output as the count clock.

## CHAPTER 10 WATCHDOG TIMER

- **Watchdog reset generation circuit**

This circuit generates a reset signal for an overflow of the watchdog counter.

- **Counter clear control circuit**

This circuit controls the clearing of the watchdog counter and the start and stop of the counter.

- **Watchdog timer control register (WDTC)**

This register is used for the start and clearing of the watchdog timer, and holding of the reset cause.

## 10.4 Watchdog Timer Operation

The watchdog timer generates a watchdog reset for an overflow of the watchdog counter.

### ■ Operation of Watchdog Timer

Figure 10.4-1 shows the settings required for operation of the watchdog timer.

**Figure 10.4-1 Watchdog Timer Settings**

| Address                  | bit15  | ..... | bit8 | bit7 | bit6      | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|--------------------------|--------|-------|------|------|-----------|------|------|------|------|------|------|
| 0000A8 <sub>H</sub> WDTC | (TBTC) |       |      | PONR | Re-served | WRST | ERST | SRST | WTE  | WT1  | WT0  |
|                          |        |       |      |      |           |      |      |      | 0    | ⊙    | ⊙    |

⊙ : Bits being used  
0 : Set to "0"

#### ○ Starting the watchdog timer

After a reset, the watchdog timer starts operation when the first "0" is written to the watchdog control bit (WTE) in the watchdog timer control register (WDTC). The interval time is then specified at the same time on the interval time selection bits (WT1, WT0) in the WDTC register.

#### ○ Clearing of the watchdog timer

The 2-bit counter of the watchdog timer is cleared when the second or subsequent "0" is written to the WTE bit. If the counter is not cleared within the interval time, the counter overflows to generate a watchdog reset.

The watchdog counter is cleared when a reset occurs and by a transition to the sleep mode, stop mode, time-base timer mode, or clock mode.

Notes:

- When a transition to the time-base timer mode or watch mode occurs, the watchdog counter is cleared once, but be careful because the watchdog counter does not stop after being cleared.
- When the clock timer is set as watchdog's clock with single clock products, the watchdog timer cannot be used.

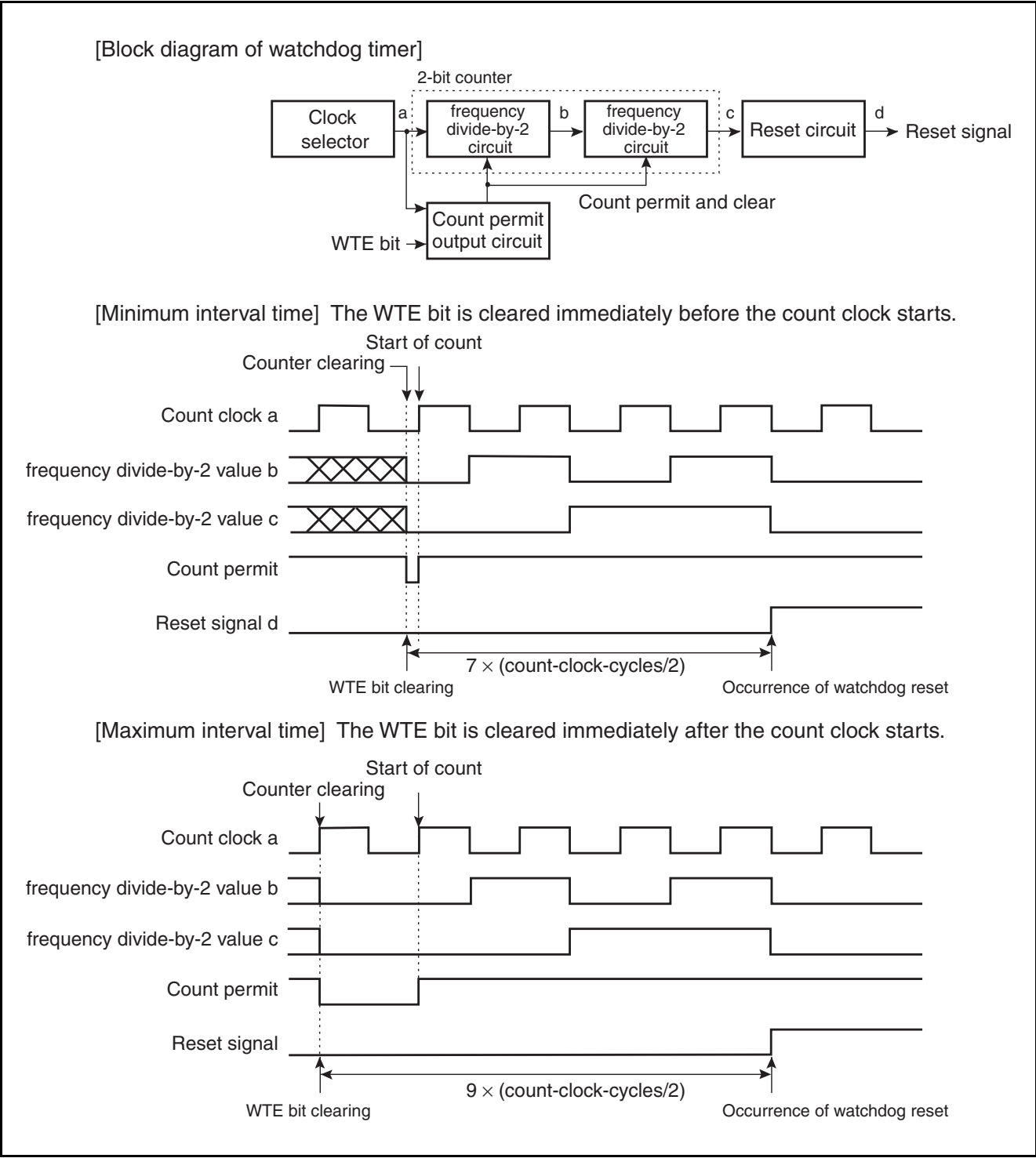
#### ○ Interval time of the watchdog timer

Figure 10.4-2 shows the relationship between the clear timing and interval time of the watchdog timer. The interval time varies depending on the timing when the watchdog timer is cleared, which takes 3.5 to 4.5 times as much time as the count clock cycle.

#### ○ Check of reset causes

After a reset, the reset cause can be found by checking the reset cause bits (PONR, WRST, ERST, and SRST) in the WDTC register.

Figure 10.4-2 Clearing Timing and Interval Time of Watchdog Timer



## 10.5 Notes on Using Watchdog Timer

---

This section explains notes on using the watchdog timer.

---

### ■ Notes on Using Watchdog Timer

- **Stopping the watchdog timer**

The watchdog timer stops by all reset causes.

- **Interval time**

Because the interval time uses the carry signals of the time-base timer as the count clock, the interval time of the watchdog timer may become longer than the specified time when the time-base timer is cleared.

The time-base timer is also cleared by writing "0" to the TBR bit in the time-base timer control register (TBTC), transition from main clock mode to PLL clock mode, transition from sub clock mode to main clock mode, and transition from sub clock mode to PLL clock mode.

- **Selection of the interval time**

The interval time can be specified when the watchdog timer is started. Any data written after the start of the watchdog timer is ignored.

- **Caution on creating programs**

When creating a program that clears the watchdog timer repetitively in the main loop, the main loop processing time, including interrupt processing, must be less than the minimum interval time of the watchdog timer.

- **Caution at sub clock mode**

Please select the output of the clock timer by setting the watchdog timer clock source selection bit (WDSCS) of the clock timer control register (WTC) to "0" on sub clock mode.

## 10.6 Sample Programs of Watchdog Timer

---

Sample programs for the watchdog timer are shown below.

---

### ■ Sample Programs for Watchdog Timer

#### ○ Specifications for processing

- Clears the watchdog timer once per loop of the main program.
- The main loop must complete a circuit within the minimum interval time of the watchdog timer.

#### ○ Sample coding

```

WDTC    EQU    0000A8H          ; Watchdog timer control register
WTE     EQU    WDTC:2          ; Watchdog control bit
;----- Main program -----
CODE    CSEG
START:
;      :                      ; Assumption that the stack pointer (SP),
;                      ; etc., have been initialized.
WDG_START:
        MOV     WDTC, #00000011B ; Start of watchdog timer Selection of
;                      ; interval time of  $2^{21} \pm 2^{18}$  cycles
;----- Main loop -----
MAIN:   CLRB    I:WTE           ; Clearing of watchdog timer
;      :                      ; Periodic 2-bit clearing
;      User processing
;      :
        JMP     MAIN           ; Loop time shorter than interval
;                      ; time of watchdog timer
CODE    ENDS
;----- Specifying vectors -----
VECT    CSEG    ABS=0FFH
        ORG     0FFDCH          ; Specifying the reset vector
        DSL     START
        DB      00H            ; Setting to the single-chip mode
VECT    ENDS
        END      START

```

# CHAPTER 11 WATCH TIMER

---

**This chapter provides an overview of the watch timer, explains the configuration and functions of its register, and the operation of the watch timer.**

---

- 11.1 Overview of Watch Timer
- 11.2 Watch Timer Configuration
- 11.3 Watch Timer Control Register (WTC)
- 11.4 Watch Timer Operation



## 11.1 Overview of Watch Timer

The watch timer is a 15-bit timer using the sub clock. This timer can generate interval interrupts. Furthermore, depending on the setting, this timer can be used as the clock source for the watchdog timer.

### ■ Functions of Watch Timer

The watch timer is composed of a 15-bit timer and a circuit to control interval interrupts.

The watch timer uses the sub clock regardless of the PLL clock selection bit (MCS) or the sub clock selection bit (SCS) in the clock selection register (CKSCR).

Table 11.1-1 lists the interval times of the watch timer.

**Table 11.1-1 Interval Times of Watch Timer**

| WTC2 | WTC1 | WTC0 | Interval time *       |
|------|------|------|-----------------------|
| 0    | 0    | 0    | 31.25 ms              |
| 0    | 0    | 1    | 62.5 ms               |
| 0    | 1    | 0    | 125 ms                |
| 0    | 1    | 1    | 250 ms                |
| 1    | 0    | 0    | 500 ms                |
| 1    | 0    | 1    | 1.000 s               |
| 1    | 1    | 0    | 2.000 s               |
| 1    | 1    | 1    | Setting is prohibited |

\*: Sub clock: 32 kHz frequency divided by 4 (= 8 kHz)

## 11.2 Watch Timer Configuration

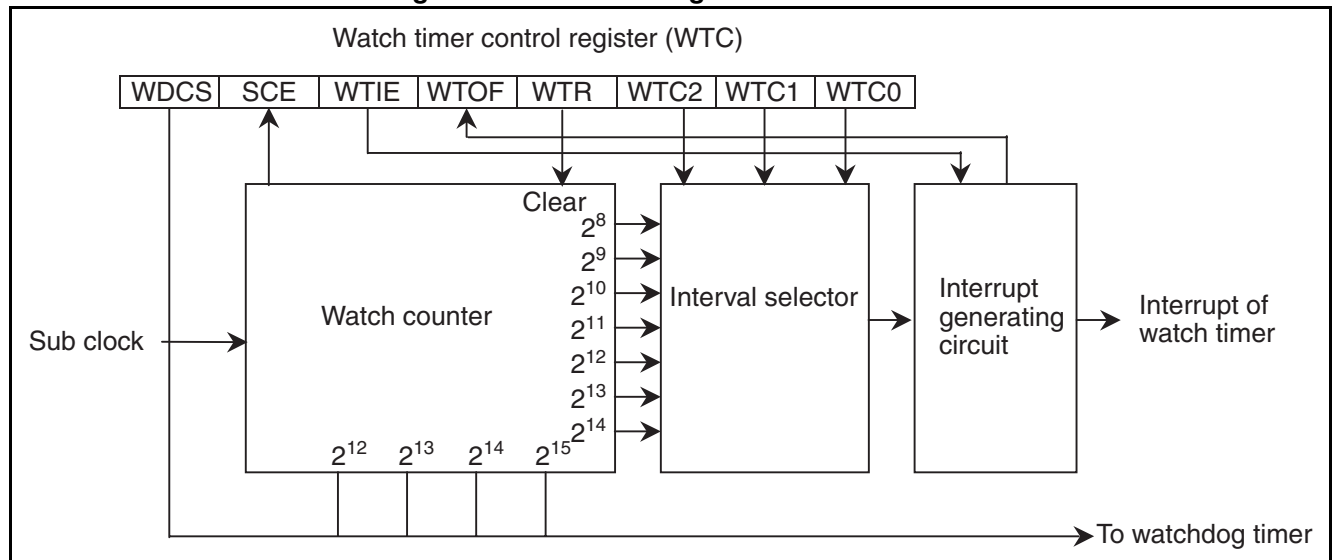
The watch timer is composed of four blocks that include the following:

- Interval selector
- Watch counter
- Watch timer interrupt generating circuit
- Watch timer control register (WTC)

### ■ Block Diagram of Watch Timer

Figure 11.2-1 is a block diagram of the watch timer.

**Figure 11.2-1 Block Diagram of Watch Timer**



#### ○ Watch counter

This is a 15-bit up-counter that uses the sub clock as the clock source.

#### ○ Interval selector

This selector selects one of seven types for watch timer interrupt intervals.

#### ○ Interrupt generating circuit

This circuit generates the interval interrupts of the watch timer.

#### ○ Watch timer control register (WTC)

This register controls operation of the watch timer and watch timer interrupt, and it specifies the clock source for the watchdog timer.

### 11.3 Watch Timer Control Register (WTC)

The watch timer control register (WTC) controls operation of the watch timer. This register also controls the time of interval interrupts.

■ Configuration of Watch Timer Control Register (WTC)

Figure 11.3-1 shows the configuration of the watch timer control register (WTC), and Table 11.3-1 lists the functions of bits in the watch timer control register (WTC).

Figure 11.3-1 Configuration of Watch Timer Control Register (WTC)

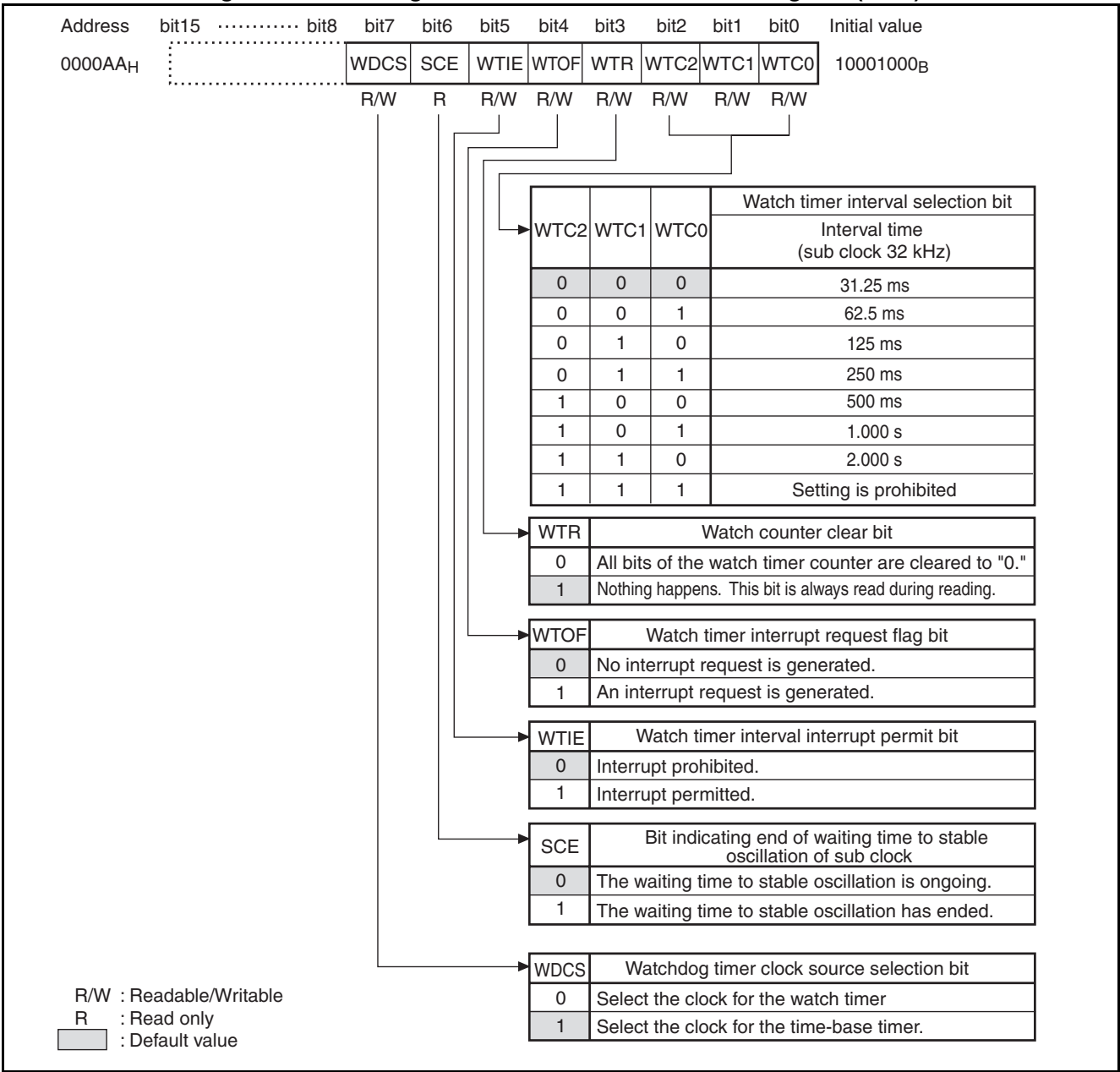


Table 11.3-1 Functions of Bits in Watch Timer Control Register (WTC)

| Bit name     |   | Function  |
|--------------|---|---|
| bit7         | WDCS:<br>Watchdog timer clock source selection bit                              | <p>This bit is for selecting the clock source for the watchdog timer.</p> <ul style="list-style-type: none"> <li>If set to "0", this bit specifies clock for the watch timer; if set to "1", it specifies clock for the time-base timer. If the mode transits to the sub clock mode with setting to "1", the watchdog timer stops.</li> <li>This bit is initialized to "1" by a reset.</li> </ul>   |
| bit6         | SCE:<br>Bit indicating end of oscillation stabilization wait time for sub clock | <p>This bit indicates that the oscillation stabilization wait time of the sub clock has ended.</p> <ul style="list-style-type: none"> <li>If set to "0", this bit indicates that the oscillation stabilization wait time of the sub clock is ongoing.</li> <li>The oscillation stabilization wait time of the sub clock is fixed at <math>2^{14}</math> cycles of the sub clock.</li> <li>This bit is initialized to "0" by a power-on reset or stoppage.</li> </ul>                |
| bit5         | WTIE:<br>Watch timer interval interrupt permit bit                              | <p>This bit is for permitting interval interrupts by the watch timer.</p> <ul style="list-style-type: none"> <li>If set to "1", this bit permits interrupts; if set to "0", it prohibits interrupts.</li> <li>This bit is initialized to "0" by a reset</li> </ul>  |
| bit4         | WTOF:<br>Watch timer interrupt request flag bit                                 | <p>This bit indicates that an interrupt request by the watch timer has been issued.</p> <ul style="list-style-type: none"> <li>If this bit is set to "1" and the WTIE bit is set to "1", an interrupt request has been issued.</li> <li>This bit is set to "1" at each interval time specified by the WTC2 to WTC0 bits.</li> <li>This bit is cleared to "0" by writing "0", a transition to the stop mode, and a reset.</li> <li>Writing "1" to this bit has no effect.</li> </ul> |
| bit3         | WTR:<br>Watch counter clear bit   | <p>This bit is for clearing all bits to 0 in the watch timer counter.</p> <ul style="list-style-type: none"> <li>Writing "0" to this bit clears the watch timer counter to "0".</li> <li>Writing "1" to this bit has no effect.</li> <li>During reading, "1" is always read and output.</li> </ul>  |
| bit2 to bit0 | WTC2, WTC1, WTC0:<br>Watch timer interval selection bits                        | <p>These bits specify the interval of the watch timer.</p> <ul style="list-style-type: none"> <li>These bits are initialized to "000<sub>B</sub>" by a reset.</li> <li>When changing the bit settings, clear the WTOF bit at the same time.</li> </ul>  |

## 11.4 Watch Timer Operation

---

The watch timer functions as a clock source for the watchdog timer, timer for the oscillation stabilization wait time of the sub clock, and interval timer to generate interrupts at fixed intervals.

---

### ■ Watch Counter

The watch counter is composed of a 15-bit counter to count the sub clock, and it always continues counting as long as the sub clock is input.

#### ○ Clearing the watch counter

The operation of clearing the watch counter is affected by a power-on reset, a transition to the stop mode, and writing "0" to the watch counter clear bit (WTR) in the watch timer control register (WTC).

---

#### Note:

Clearing the watch counter affects the watchdog timer and interval interrupts that use watch timer output.

To clear the watch timer by writing "0" to the WTR bit in the watch timer control register (WTC), set the WTIE bit to "0" and set the watch timer to interrupt inhibited state. Before permitting an interrupt, clear the interrupt request issued by writing "0" to the WTOF flag.

---

### ■ Interval Interrupt Function of Watch Timer

This function generates interrupts at fixed intervals by using the carry signals of the watch counter.

#### ○ Specification of the interval time

The interval time can be specified with the WTC2, WTC1, and WTC0 bits in the WTC register.

#### ○ Generation of watch timer interrupts

The watch timer interrupt request flag bit (WTOF) is set at each interval time specified by the WTC2 to WTC0 bits. Consequently, when interrupts are permitted by setting the watch timer interval interrupt permit bit (WTIE) to "1", a watch timer interrupt occurs.

The timing, when the WTOF bit is set, depends on the timing as a time reference when the watch timer was cleared for the last time.

Because the watch timer is used as the timer for the oscillation stabilization wait time of the sub clock after a transition to the stop mode, the WTOF bit is simultaneously cleared at the transition to this mode.

### ■ Clock Source for Watchdog Timer Specifying Function

The clock source for the watchdog timer can be specified with the watchdog timer clock source selection bit (WDCS) of the WTC register. If the sub clock is used as the machine clock, set the WDCS bit to "0" and select the output of the watch timer. If the mode transits to the sub clock mode with the WDCS bit setting to "1", the watchdog timer stops.

### ■ Sub Clock Oscillation Stabilization Wait Time Function

For a power-on reset, restoration from the stop mode or the watch timer functions as the timer for the oscillation stabilization wait time of the sub clock. The oscillation stabilization wait time of the sub clock is fixed at  $2^{14}$  cycles of the sub clock.



# CHAPTER 12 16-BIT INPUT/OUTPUT TIMER

---

**This chapter provides an overview of the 16-bit input/output timer, explains the configuration and functions of its registers, interrupt and its operation.**

---

- 12.1 Overview of 16-bit Input/Output Timer
- 12.2 Configuration of 16-bit Input/Output Timer
- 12.3 Configuration and Function of 16-bit Input/Output Timer Register
- 12.4 Interrupt of 16-bit Input/Output Timer
- 12.5 16-bit Input/Output Timer Operation
- 12.6 Program Example of 16-bit Input/Output Timer



## 12.1 Overview of 16-bit Input/Output Timer

---

The 16-bit input/output timer consists of one free-run timer, six output compares, and two input captures. An output of six independent waveforms based on the free-run timer can be obtained, and measurement of input pulse widths and external clock intervals is enabled.

---

### ■ Functions of 16-bit Input/Output Timer

The 16-bit input/output timer consists of a free-run timer, output compare, and input capture, whose functions are explained below.

#### ○ Free-run timer (× 1)

Free-run timer consists of 16-bit up-counter, control register, and prescaler.

The output value of the free-run timer uses as a base time (base timer) of input capture and output compare.

- Clock for the count operation can be selected from 8 types of clocks.
- Counter overflow interrupt can be generated.
- The counter initialization is allowed by the match between the value of compare clear register in output compare and value of free-run timer.

#### ○ Output compare (× 6)

Output compare consists of six 16-bit compare registers, a compare output latch, and a control register. If a free-run timer and compare register have matching values, the output level is reversed with a generation of an interrupt.

- Six compare registers can operate independently. Each compare register has a corresponding output pin and interrupt flag.
- Two compare registers are used as a pair to control the output pin.
- The initial value of output pin can be set.

#### ○ Input capture (× 2)

Input capture consists of two independent external input pins and corresponding capture registers, control registers, and edge detection circuit. If any edge of a signal input from the external input pin is detected, the free-run timer value is retained in the capture register and, at the same time, an interrupt is generated.

- The edge of an external input signal can be selected from its rising edge, falling edge or both edges.
- The two input capture operate independently.

Interrupts occur at the valid edge of external input signals. Input capture can start DMA or EI<sup>2</sup>OS by the interrupt.

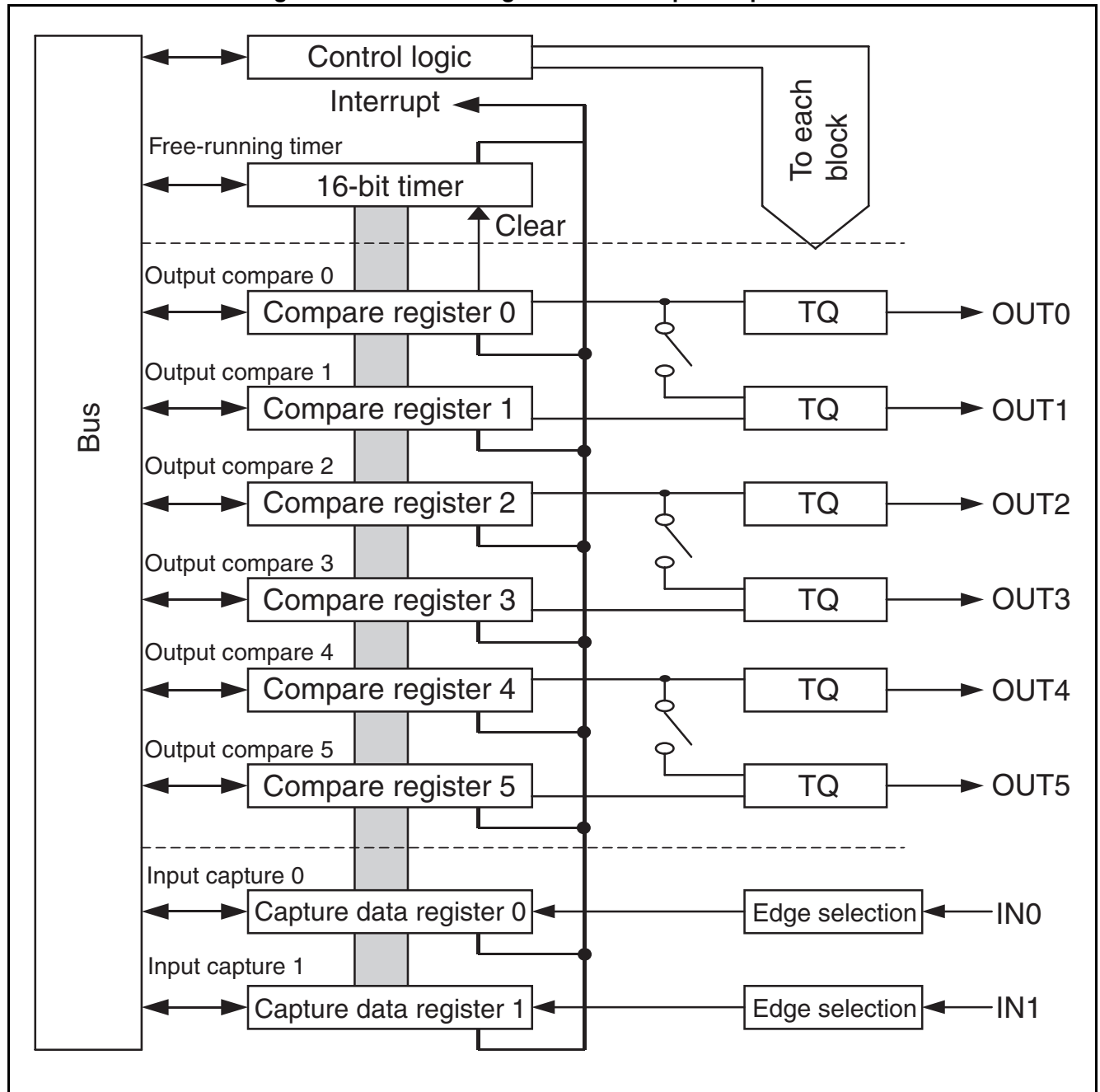
## 12.2 Configuration of 16-bit Input/Output Timer

The 16-bit input/output timer consists of three modules for the free-run timer, output compare, and input capture.

### ■ Block Diagram

Figure 12.2-1 is a block diagram of the 16-bit input/output timer.

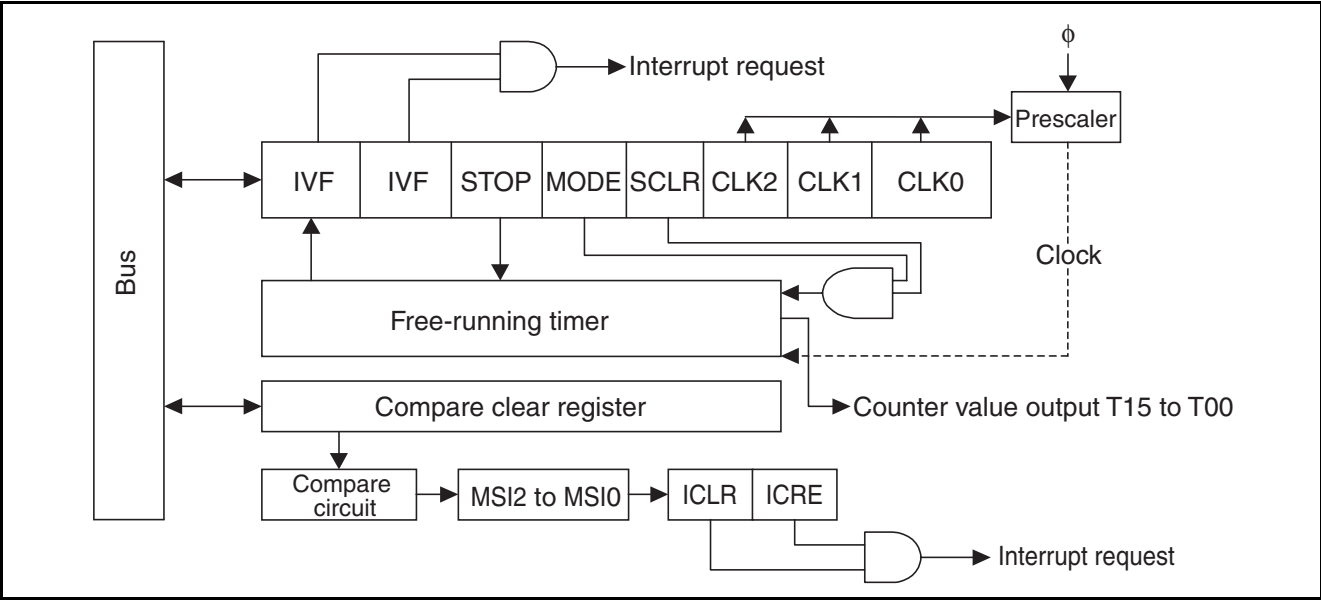
**Figure 12.2-1 Block Diagram of 16-bit Input/Output Timer**



■ Block Diagram of Free-run Timer

Figure 12.2-2 is a block diagram of the free-run timer.

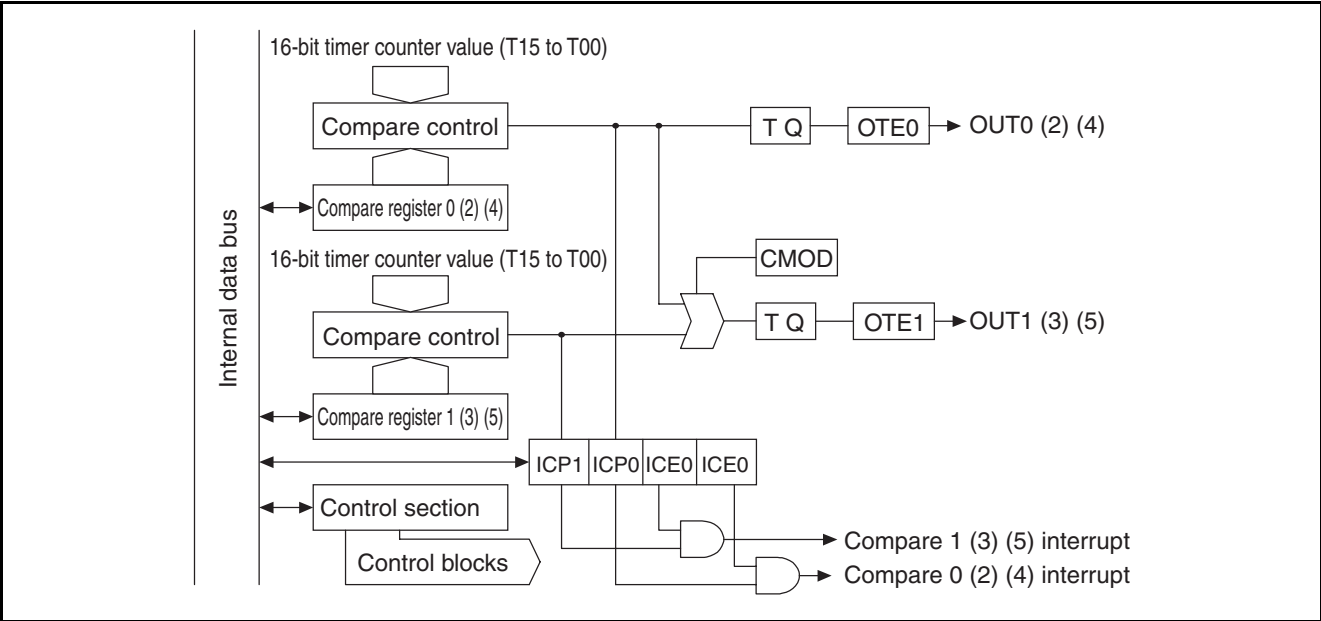
Figure 12.2-2 Block Diagram of Free-run Timer



■ Block Diagram of Output Compare

Figure 12.2-3 is a block diagram of output compare.

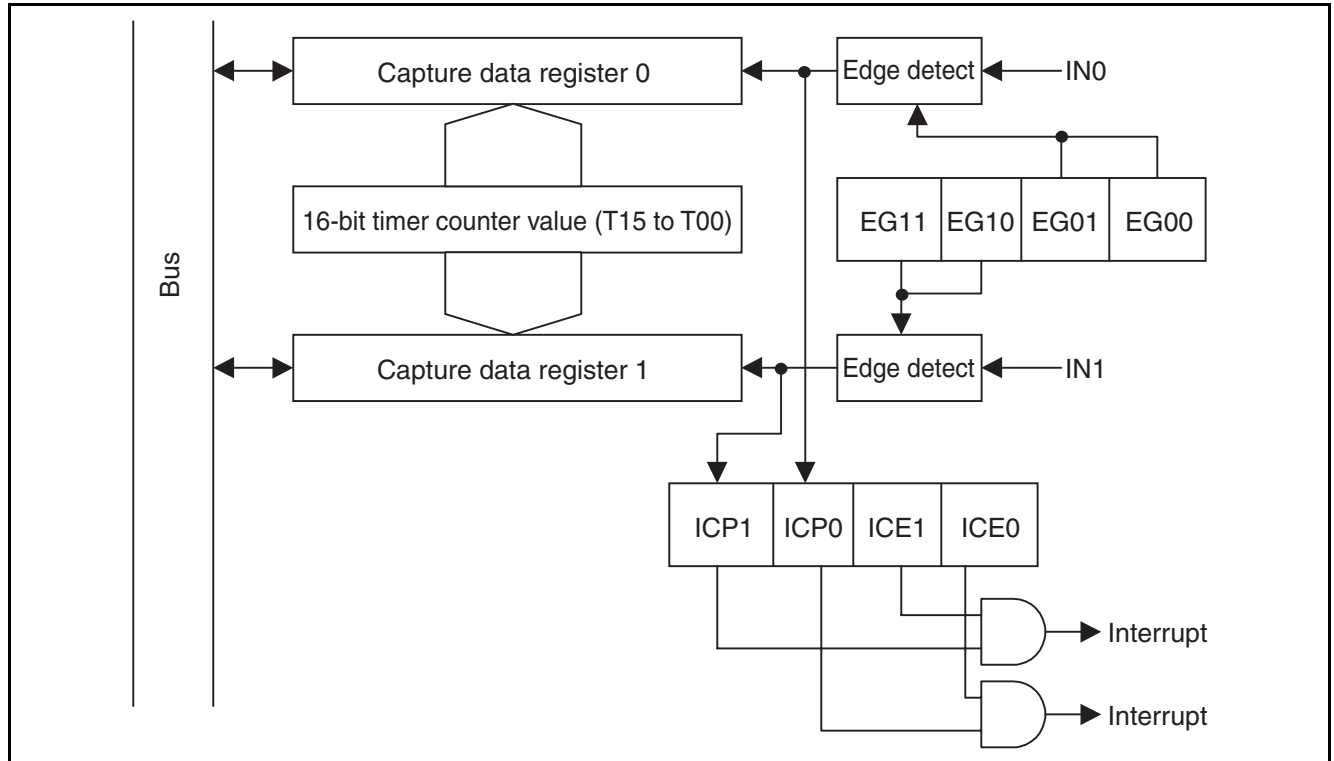
Figure 12.2-3 Block Diagram of Output Compare



## ■ Block Diagram of Input Capture

Figure 12.2-4 is a block diagram of input capture.

**Figure 12.2-4 Block Diagram of Input Capture**



## ■ Pin Related to 16-bit Input/Output Timer

The pin related to the 16-bit input/output timer has the IN0/IN1 and OUT0/OUT1/OUT2/OUT3/OUT4/OUT5 pins. The IN0/IN1 pins function as the general-purpose I/O port (P96/IN0, P97/IN1) and input capture input pin. The OUT0/OUT1/OUT2/OUT3/OUT4/OUT5 pins function as the general-purpose I/O port (PA0/OUT0, PA1/OUT1, PA2/OUT2, PA3/OUT3, P46/OUT4, P47/OUT5) and output compare output pin.

### ● Setting when using as IN0/IN1 pins

When using as the IN0/IN1 pins, the P96/IN0 and P97/IN1 pins should be set the port direction register to input port (DDR9 bit15, 14→ "0").

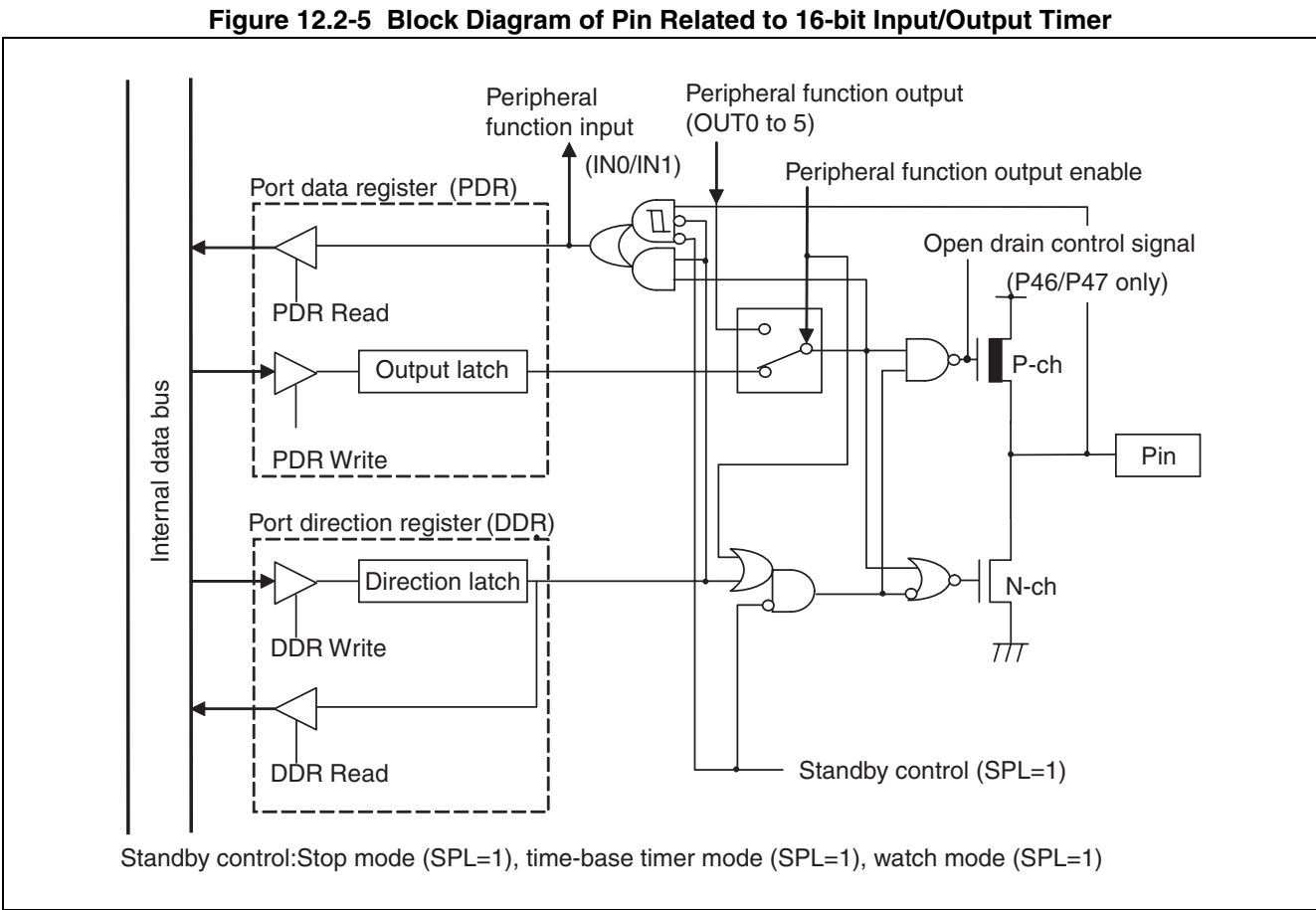
### ● Setting when using as OUT0/OUT1/OUT2/OUT3/OUT4/OUT5 pins

When using the OUT0/OUT1/OUT2/OUT3/OUT4/OUT5 pins as output, be sure to set the control register (OCS01/23/45) to the output compare pin output (OCS01/23/45 bit10, 11→ "1").

### ● Setting when using as FRCK pin

When using as the FRCK pin, the P93/FRCK pin should be set the port direction register to input port (DDR9 bit11→ "0").

■ Block Diagram of Pin Related to 16-bit Input/Output Timer

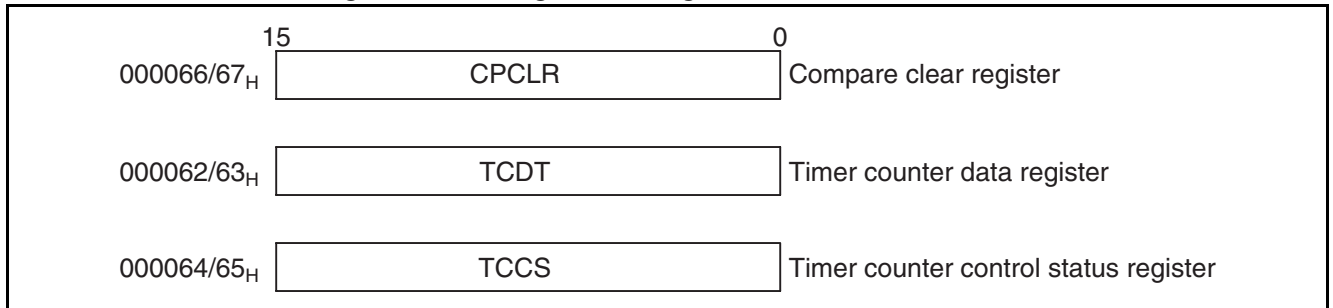


## 12.3 Configuration and Function of 16-bit Input/Output Timer Register

This section shows the configuration and functions of 16-bit input/output timer registers.

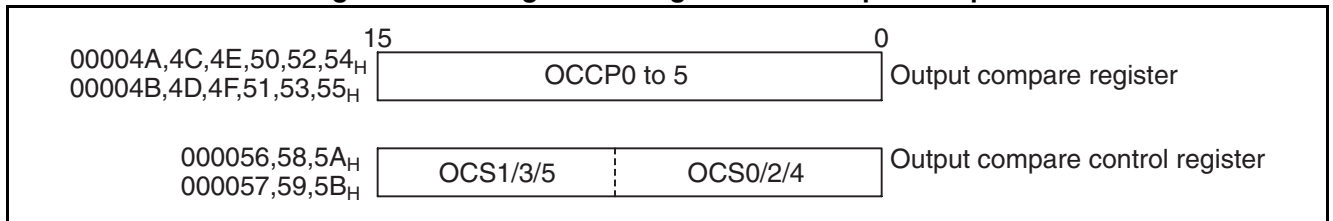
### ○ Free-run timer

**Figure 12.3-1 Register Configuration of Free-run Timer**



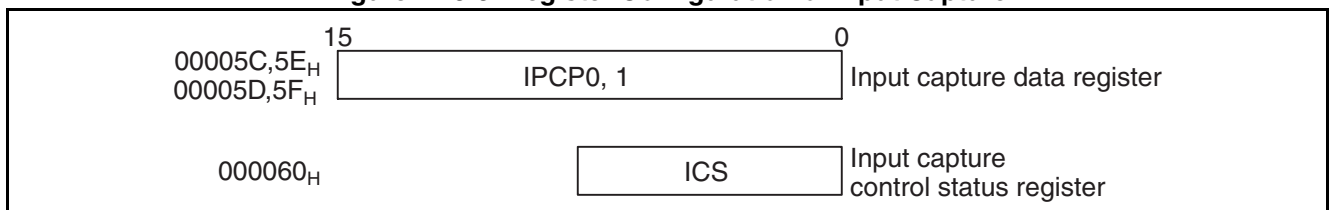
### ○ Output compare

**Figure 12.3-2 Register Configuration of Output Compare**



### ○ Input capture

**Figure 12.3-3 Register Configuration of Input Capture**



## 12.3.1 Free-Run Timer

This section shows the configuration and explains the functions of free-run timer registers.

### ■ List of Free-run Timer Registers

Figure 12.3-4 shows a list of the free-run timer registers.

**Figure 12.3-4 List of Free-run Timer Registers**

|                     |       |       |       |       |       |       |       |       |  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|--|
|                     | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | CPCLR                                  |
| 000067 <sub>H</sub> | CL15  | CL14  | CL13  | CL12  | CL11  | CL10  | CL09  | CL08  | Compare clear register                 |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value XXXXXXXX <sub>B</sub>    |
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | CPCLR                                  |
| 000066 <sub>H</sub> | CL07  | CL06  | CL05  | CL04  | CL03  | CL02  | CL01  | CL00  | Compare clear register                 |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value XXXXXXXX <sub>B</sub>    |
|                     | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TCDT                                   |
| 000063 <sub>H</sub> | T15   | T14   | T13   | T12   | T11   | T10   | T09   | T08   | Timer counter data register            |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>    |
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TCDT                                   |
| 000062 <sub>H</sub> | T07   | T06   | T05   | T04   | T03   | T02   | T01   | T00   | Timer counter data register            |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>    |
|                     | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TCCS                                   |
| 000065 <sub>H</sub> | ECKE  | -     | -     | MSI2  | MSI1  | MSI0  | ICLR  | ICRE  | Timer counter control/status register  |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 0 - - 00000 <sub>B</sub> |
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TCCS                                   |
| 000064 <sub>H</sub> | IVF   | IVFE  | STOP  | MODE  | SCLR  | CLK2  | CLK1  | CLK0  | Timer counter control/status register  |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>    |

R/W : Readable/Writable

### ■ Compare Clear Register (CPCLR)

Figure 12.3-5 shows the bit configuration of the compare clear register (CPCLR).

**Figure 12.3-5 Bit Configuration of the Compare Clear Register (CPCLR)**

|                     |       |       |       |       |       |       |       |       |                                     |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------------|
|                     | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | CPCLR                               |
| 000067 <sub>H</sub> | CL15  | CL14  | CL13  | CL12  | CL11  | CL10  | CL09  | CL08  | Compare clear register              |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value XXXXXXXX <sub>B</sub> |
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | CPCLR                               |
| 000066 <sub>H</sub> | CL07  | CL06  | CL05  | CL04  | CL03  | CL02  | CL01  | CL00  | Compare clear register              |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value XXXXXXXX <sub>B</sub> |

R/W : Readable/Writable

The compare clear register (CPCLR) is a 16-bit length compare register used to make a comparison with the free-run timer. An initial value of a register is undefined. Therefore, set the initial value, then allow the interrupt operation. This register requires word access.

When the MODE bit of the timer counter control status register (TCCS) is set to "1", the free-run timer value is initialized to "0000<sub>H</sub>" at matching this register value and the value of the free-run timer. When this register value matches the value of the free-run timer, a compare clear interrupt flag is set. When the compare interrupt flag is set to "1", an interrupt request issues to the CPU at allowing the interrupt operation.

## ■ Timer Counter Data Register (TCDT)

Bit configuration of the timer counter data register (TCDT) is shown below.

**Figure 12.3-6 Bit Configuration of Timer Counter Data Register (TCDT)**

|                         |       |       |       |       |       |       |       |       |                                     |
|-------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------------|
|                         | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TCDT                                |
| 000063 <sub>H</sub>     | T15   | T14   | T13   | T12   | T11   | T10   | T09   | T08   | Timer counter data register         |
|                         | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub> |
|                         | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TCDT                                |
| 000062 <sub>H</sub>     | T07   | T06   | T05   | T04   | T03   | T02   | T01   | T00   | Timer counter data register         |
|                         | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub> |
| R/W : Readable/Writable |       |       |       |       |       |       |       |       |                                     |

The timer counter data register (TCDT) is a 16-bit up counter and is used to read the value of the free-run timer counter. The counter value is cleared to "0000" by a reset. Writing to this register must be performed in the stopped (STOP = 1) state, and the written value defines the timer value.

This register requires word access. The free-run timer is initialized with the following causes:

- Reset
- Clear bit (SCLR) of timer counter control status register (TCCS)
- Matching between the compare clear register (CPCLR) of output compare and timer counter value (TCCS: MODE = 1) (requiring mode setting).

## ■ Timer Counter Control Status Register (TCCS)

Bit configuration of the timer counter control status register (TCCS) is shown below.

**Figure 12.3-7 Bit Configuration of Timer Counter Control Status Register (TCCS)**

|                         |       |       |       |       |       |       |       |       |                                       |
|-------------------------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------------------------|
|                         | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TCCS                                  |
| 000065 <sub>H</sub>     | ECKE  | -     | -     | MSI2  | MSI1  | MSI0  | ICLR  | ICRE  | Timer counter control status register |
|                         | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 0--00000 <sub>B</sub>   |
|                         | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TCCS                                  |
| 000064 <sub>H</sub>     | IVF   | IVFE  | STOP  | MODE  | SCLR  | CLK2  | CLK1  | CLK0  | Timer counter control status register |
|                         | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>   |
| R/W : Readable/Writable |       |       |       |       |       |       |       |       |                                       |

Timer counter control status register (TCCS) consists of bits that have the functions explained below.

### [bit15] ECKE

This bit is used to select whether the count clock source of the free-run timer is internal or external. Since the clock is changed soon after being written to this bit, change the bit setting when output compare and input capture are in the stopped state.

|   |  |
|---|--|
| 0 | Internal clock source is selected (initial value)  |
| 1 | Clock inputted by external pin (FRCK) is selected. |

### [bit14, bit13] Unused bits

These bits are not used.



### [bit12, bit11, bit10] MSI2, MSI1, MSI0

These bits specify the count with which a compare clear interrupt is masked. It consists of 3-bit reload counter that reloads the count value every time the counter value reaches "000". During writing to this register, the count value is also loaded, where mask count = specified count (e.g., set to "010" when masked twice and the third time is interrupted). However, if "000" is set, no interrupt cause is masked.

### [bit9] ICLR

This bit is the interrupt request flag for compare clear. If the compare clear register and free-run timer are found to have matching values by compare result, this bit is set to "1". If the interrupt request permit bit (bit8: ICRE) is set, an interrupt occurs. This bit is cleared by writing "0". Writing "1" has no effect. An instruction of the read-modify-write type always reads "1".

|   |   |
|---|---|
| 0 | No interrupt request issued (initial value) |
| 1 | Interrupt request issued                    |

### [bit8] ICRE

This bit is the interrupt permit bit for compare clear. If this bit is set to "1" and the interrupt flag (bit9: ICLR) is set to "1", then an interrupt occurs.

|   |                                      |
|---|--------------------------------------|
| 0 | Interrupt prohibited (initial value) |
| 1 | Interrupt permitted                  |

### [bit7] IVF

This bit is the interrupt request flag of the free-run timer. If the free-run timer causes an overflow or mode setting results in a match between the compare clear register and compare results of free-run timer so that the counter is cleared, then the IVF bit is set to "1". If the interrupt request permit bit (bit5: IVFE) is set, an interrupt occurs. This bit is cleared by setting "0". Setting "1" has no effect. An instruction of the read-modify-write type reads "1".

|   |   |
|---|---|
| 0 | No interrupt request used (initial value) |
| 1 | Interrupt request used                    |

### [bit6] IVFE

This bit is the interrupt permit bit for the free-run timer. If this bit is set to "1" when the write flag (bit5: IVF) is set to "1", an interrupt occurs.

|   |                                    |
|---|------------------------------------|
| 0 | Interrupt prohibit (initial value) |
| 1 | Interrupt permit                   |

**[bit5] STOP**

This bit sets whether to enable or disable the counting by the free-run timer. If this bit is set to "1", the timer stops counting, and if it is set to "0", the timer starts counting.

|   |  |
|---|--|
| 0 | Count permit (operation) (initial value) |
| 1 | Count prohibit (stop)                    |

If the free-run timer stops counting, the output compare operation also stops.

**[bit4] MODE**

This bit specifies the initialization conditions of the free-run timer.

If set to "0", the reset and clear bit (bit3: SCLR) initialize the counter value.

If set to "1", in addition to the reset and clear bit (bit3: SCLR), matching the compare clear register (CPCLR) value with the free-run timer, initializes the counter value.

|   |   |
|---|---|
| 0 | Initialized by reset and clear bit (initial value)          |
| 1 | Initialized by reset, clear bit, and compare clear register |

The counter value initialization occurs at the point where the counter value changes.

**[bit3] SCLR**

This bit initializes the value of the free-run timer to "0000".

Writing "1" initializes the counter value to "0000". Writing "0" has no effect. The read value is always "0". The counter value initialization occurs synchronizing with the counter value change point.

|   |   |
|---|---|
| 0 | No effect (initial value)               |
| 1 | Initializes the counter value to "0000" |

If it is initialized at the time of stopping the timer, write "0000" to the data register.

---

**Note:**

After "1" is written, the counter value of this bit is not initialized to the following count clock when "0" is written.

---

## [bit2, bit1, bit0] CLK2, CLK1, CLK0

These bits select the count clock of the free-run timer. Since the clock changes after this bit is written, change the bit setting when output compare and input capture are in the stopped state.

| CLK2 | CLK1 | CLK0 | Count clock | $\phi = 20\text{MHz}$ | $\phi = 16\text{MHz}$ | $\phi = 8\text{MHz}$ | $\phi = 4\text{MHz}$ | $\phi = 1\text{MHz}$ |
|------|------|------|-------------|-----------------------|-----------------------|----------------------|----------------------|----------------------|
| 0    | 0    | 0    | $\phi$      | 50 ns                 | 62.5 ns               | 0.125 $\mu\text{s}$  | 0.25 $\mu\text{s}$   | 1.0 $\mu\text{s}$    |
| 0    | 0    | 1    | $\phi/2$    | 100 ns                | 0.125 $\mu\text{s}$   | 0.25 $\mu\text{s}$   | 0.5 $\mu\text{s}$    | 2.0 $\mu\text{s}$    |
| 0    | 1    | 0    | $\phi/4$    | 0.2 $\mu\text{s}$     | 0.25 $\mu\text{s}$    | 0.5 $\mu\text{s}$    | 1.0 $\mu\text{s}$    | 4.0 $\mu\text{s}$    |
| 0    | 1    | 1    | $\phi/8$    | 0.4 $\mu\text{s}$     | 0.5 $\mu\text{s}$     | 1.0 $\mu\text{s}$    | 2.0 $\mu\text{s}$    | 8.0 $\mu\text{s}$    |
| 1    | 0    | 0    | $\phi/16$   | 0.8 $\mu\text{s}$     | 1.0 $\mu\text{s}$     | 2.0 $\mu\text{s}$    | 4.0 $\mu\text{s}$    | 16.0 $\mu\text{s}$   |
| 1    | 0    | 1    | $\phi/32$   | 1.6 $\mu\text{s}$     | 2.0 $\mu\text{s}$     | 4.0 $\mu\text{s}$    | 8.0 $\mu\text{s}$    | 32.0 $\mu\text{s}$   |
| 1    | 1    | 0    | $\phi/64$   | 3.2 $\mu\text{s}$     | 4.0 $\mu\text{s}$     | 8.0 $\mu\text{s}$    | 16.0 $\mu\text{s}$   | 64.0 $\mu\text{s}$   |
| 1    | 1    | 1    | $\phi/128$  | 6.4 $\mu\text{s}$     | 8.0 $\mu\text{s}$     | 16.0 $\mu\text{s}$   | 32.0 $\mu\text{s}$   | 128.0 $\mu\text{s}$  |

## 12.3.2 Output compare

This section shows the configuration and explains the functions of registers for output compare.

### ■ List of Output Compare Registers

Figure 12.3-8 shows a list of the registers for output compare.

**Figure 12.3-8 Registers of Output Compare**

|           |                     |       |       |       |       |       |       |       |       |                                      |
|-----------|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------------------------------|
| ch.0      | 00004B <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | OCCP0 to 5                           |
|           | 00004D <sub>H</sub> |       |       |       |       |       |       |       |       |                                      |
| to        | 00004F <sub>H</sub> | C15   | C14   | C13   | C12   | C11   | C10   | C09   | C08   | Output compare register              |
|           | 000051 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>  |
|           | 000053 <sub>H</sub> |       |       |       |       |       |       |       |       |                                      |
| ch.5      | 000055 <sub>H</sub> |       |       |       |       |       |       |       |       |                                      |
| ch.1      | 00004A <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | OCCP0 to 5                           |
|           | 00004C <sub>H</sub> |       |       |       |       |       |       |       |       |                                      |
| to        | 00004E <sub>H</sub> | C07   | C06   | C05   | C04   | C03   | C02   | C01   | C00   | Output compare register              |
|           | 000050 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>  |
|           | 000052 <sub>H</sub> |       |       |       |       |       |       |       |       |                                      |
| ch.5      | 000054 <sub>H</sub> |       |       |       |       |       |       |       |       |                                      |
| ch.0,ch.1 | 000057 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | OCS1/ 23/45                          |
| ch.2,ch.3 | 000059 <sub>H</sub> | -     | -     | -     | CMOD  | OTE1  | OTE0  | OTD1  | OTD0  | Output compare control register      |
| ch.4,ch.5 | 00005B <sub>H</sub> | (-)   | (-)   | (-)   | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value -- -00000 <sub>B</sub> |
| ch.0,ch.1 | 000056 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | OCS01/23/45                          |
| ch.2,ch.3 | 000058 <sub>H</sub> | ICP1  | ICP0  | ICE1  | ICE0  | -     | -     | CST1  | CST0  | Output compare control register      |
| ch.4,ch.5 | 00005A <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (-)   | (-)   | (R/W) | (R/W) | Initial value 0000- -00 <sub>B</sub> |

R/W : Readable/Writable  
- : Unused bit

#### Note:

To rewrite the output compare register, perform within the compare interrupt routine or compare operation disabled state. Be sure not to occur a compare result match and writing the compare register simultaneously.

## ■ Output Compare Register (OCCP0 to OCCP5)

Output compare register (OCCP0 to OCCP5) has the bit configuration shown below.

**Figure 12.3-9 Bit Configuration of Output Compare Register (OCCP0 to OCCP5)**

|                          |       |       |       |       |       |       |       |       |  |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|--|
| ch.0 00004B <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | OCCP0 to OCCP5                         |
| 00004D <sub>H</sub>      |       |       |       |       |       |       |       |       |  |
| to 00004F <sub>H</sub>   | C15   | C14   | C13   | C12   | C11   | C10   | C09   | C08   | Output compare register                |
| 000051 <sub>H</sub>      | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value "00000000 <sub>B</sub> " |
| ch.5 000053 <sub>H</sub> |       |       |       |       |       |       |       |       |  |
| 000055 <sub>H</sub>      |       |       |       |       |       |       |       |       |  |
| ch.0 00004A <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | OCCP0 to OCCP5                         |
| 00004C <sub>H</sub>      |       |       |       |       |       |       |       |       |  |
| to 00004E <sub>H</sub>   | C07   | C06   | C05   | C04   | C03   | C02   | C01   | C00   | Output compare register                |
| 000050 <sub>H</sub>      | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value "00000000 <sub>B</sub> " |
| 000052 <sub>H</sub>      |       |       |       |       |       |       |       |       |  |
| ch.5 000054 <sub>H</sub> |       |       |       |       |       |       |       |       |  |

R/W : Readable/Writable

The output compare register (OCCP0 to OCCP5) is a 16-bit length compare register used to make a comparison with the free-run timer. This register is initialized at reset. This register requires word access. If this register and the free-run timer have matching values, a compare signal is generated to set the output compare interrupt flag. If output enable is given, the output level corresponding to the compare register value is reversed and outputted.

## ■ Output Compare Control Register (OCS01/OCS23/OCS45)

The output compare control register (OCS01/OCS23/OCS45) has the bit configuration shown below.

**Figure 12.3-10 Bit Configuration of Output Compare Control Register (OCS01/OCS23/OCS45)**

|                               |       |       |       |       |       |       |       |       |  |
|-------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|--|
| ch.0,ch.1 000057 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | OCS01/OCS23/OCS45                        |
| ch.2,ch.3 000059 <sub>H</sub> | -     | -     | -     | CMOD  | OTE1  | OTE0  | OTD1  | OTD0  | Output compare control register          |
| ch.4,ch.5 00005B <sub>H</sub> | (-)   | (-)   | (-)   | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value "- - -00000 <sub>B</sub> " |
| ch.0,ch.1 000056 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | OCS01/OCS23/OCS45                        |
| ch.2,ch.3 000058 <sub>H</sub> | ICP1  | ICP0  | ICE1  | ICE0  | -     | -     | CST1  | CST0  | Output compare control register          |
| ch.4,ch.5 00005A <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (-)   | (-)   | (R/W) | (R/W) | Initial value "0000- -00 <sub>B</sub> "  |

R/W : Readable/Writable  
- : Unused bit

The functions of the output compare control register bits (OCS01/OCS23/OCS45) are shown below.

### [bit15, bit14, bit13] Unused bits

These bits are not used. They are always set to "0".

### [bit12] CMOD

This bit switches the pin output level reverse operation mode in compare result matching if pin output is permitted (OTE1 = 1 or OTE0 = 1).

- If CMOD = 0 (initial value), the level corresponding to the compare register value is reversed.
  - OUT0/2/4: Level is reversed if a match for compare register 0/2/4 is found
  - OUT1/3/5: Level is reversed if a match for compare register 1/3/5 is found

- If CMOD = 1, compare register 0 (2/4) inverts the output level in the same manner as in cases where CMOD = 0; however, the pin OUT1 (OUT3/OUT5) output level corresponding to compare register 1 (3/5) inverts the output level only when both compare register 0 (2/4) and compare register 1 (3/5) have a match. Also, if compare registers 0 (2/4) and 1 (3/5) have the same value, the operation is the same as that for one compare register.
  - OUT0/2/4: Level is reversed if a match for compare register 0/2/4 is found
  - OUT1/3/5: Level is reversed if a match for compare register 0/2/4 and 1/3/5 is found

**[bit11, bit10] OTE1, OTE0**

These bits permit pin output of output compare. These bits take an initial value of "0".

|   |   |
|---|---|
| 0 | General-purpose support (initial value) |
| 1 | Operate as pin output of output compare |

- OTE1: Corresponds to output compare 1/3/5
- OTE0: Corresponds to output compare 0/2/4

**[bit9, bit8] OTD1, OTD0**

These bits change the pin output level if pin output of output compare is permitted. The initial value of compare pin output is set to "0". A write operation must be performed after the compare operation stops. In a read operation, the output value of the output compare pin is read.

|   |   |
|---|---|
| 0 | Compare pin output set to "0" (initial value) |
| 1 | Compare pin output set to "1"                 |

- OTD1: Corresponds to output compare 1/3/5
- OTD0: Corresponds to output compare 0/2/4

**[bit7, bit6] ICP1, ICP0**

These bits are the interrupt flags for output compare. Set them to "1" if the compare register and free-run timer have matching values. If the interrupt request bit (ICE1, ICE0) is set to "enabled", and this bit is set, an output compare interrupt occurs. This bit is cleared if "0" is written. Writing "1" has no effect. An instruction of the read-modify type only reads "1".

|   |                                  |
|---|----------------------------------|
| 0 | No compare match (initial value) |
| 1 | Compare result match             |

- ICP1: Corresponds to output compare 1/3/5
- ICP0: Corresponds to output compare 0/2/4

### [bit5, bit4] ICE1, ICE0

These bits are the interrupt permit bits of output compare. If these bits are set to "1" and an interrupt flag (ICP1, ICP0) is set, an output compare interrupt occurs.

|   |   |
|---|---|
| 0 | Output compare interrupt prohibit (initial value) |
| 1 | Output compare interrupt permit                   |

- ICE1: Corresponds to output compare 1/3/5
- ICE0: Corresponds to output compare 0/2/4

### [bit3, bit2] Unused bits

These bits are not used.

### [bit1, bit0] CST1, CST0

These bits permit a matching operation with the compare register and free-run timer.

|   |  |
|---|--|
| 0 | Compare operation prohibit (initial value) |
| 1 | Compare operation permit                   |

- CST1: Corresponds to output compare 1/3/5
- CST0: Corresponds to output compare 0/2/4

Before a compare operation is permitted, specify the compare register value.

---

#### Note:

Output compare operates in sync with the free-run timer clock. Thus, if the free-run timer stops, the compare operation also stops.

---

### 12.3.3 Input capture

This section describes the configuration and functions of the registers for the input capture.

#### ■ List of Input Capture Registers

Figure 12.3-11 shows a list of the input capture registers.

**Figure 12.3-11 Input Capture Registers**

|  |       |       |       |       |       |       |       |       |                                       |
|--|-------|-------|-------|-------|-------|-------|-------|-------|---------------------------------------|
| ch.0 00005D <sub>H</sub><br>ch.1 00005F <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | IPCP0, 1                              |
|  | CP15  | CP14  | CP13  | CP12  | CP11  | CP10  | CP09  | CP08  | Input capture data register           |
|  | (R)   | (R)   | (R)   | (R)   | (R)   | (R)   | (R)   | (R)   | Initial value XXXXXXXX <sub>B</sub>   |
| ch.0 00005C <sub>H</sub><br>ch.1 00005E <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | IPCP0, 1                              |
|  | CP07  | CP06  | CP05  | CP04  | CP03  | CP02  | CP01  | CP00  | Input capture data register           |
|  | (R)   | (R)   | (R)   | (R)   | (R)   | (R)   | (R)   | (R)   | Initial value XXXXXXXX <sub>B</sub>   |
| 000060 <sub>H</sub>                                  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | ICS01                                 |
|  | ICP1  | ICP0  | ICE1  | ICE0  | EG11  | EG10  | EG01  | EG00  | Input capture control status register |
|  | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>   |
| R/W : Readable/Writable                              |       |       |       |       |       |       |       |       |                                       |
| R : Read only  |       |       |       |       |       |       |       |       |                                       |

#### ■ Input Capture Data Register (IPCP0, IPCP1)

The input capture data register (IPCP0, IPCP1) has the bit configuration shown below.

**Figure 12.3-12 Bit Configuration of Input Capture Data Register (IPCP0, IPCP1)**

|  |      |      |      |      |      |      |      |      |                                     |
|--|------|------|------|------|------|------|------|------|-------------------------------------|
| ch.0 00005D <sub>H</sub><br>ch.1 00005F <sub>H</sub> | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | IPCP0, 1                            |
|  | CP15 | CP14 | CP13 | CP12 | CP11 | CP10 | CP09 | CP08 | Input capture data register         |
|  | (R)  | (R)  | (R)  | (R)  | (R)  | (R)  | (R)  | (R)  | Initial value XXXXXXXX <sub>B</sub> |
| ch.0 00005C <sub>H</sub><br>ch.1 00005E <sub>H</sub> | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | IPCP0, 1                            |
|  | CP07 | CP06 | CP05 | CP04 | CP03 | CP02 | CP01 | CP00 | Input capture data register         |
|  | (R)  | (R)  | (R)  | (R)  | (R)  | (R)  | (R)  | (R)  | Initial value XXXXXXXX <sub>B</sub> |
| R : Read only  |      |      |      |      |      |      |      |      |                                     |

The input capture data register (IPCP0, IPCP1) is a register containing the free-run timer value when a valid edge of the external pin input waveform is detected.

This register requires word access. Writing to this register is not permitted.



## ■ Input Capture Control Status Register (ICS01)

The input capture control status register (ICS01) has the bit configuration shown below.

**Figure 12.3-13 Bit Configuration of Input Capture Control Status Register (ICS01)**

|                     |       |       |       |       |       |       |       |       |                                       |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------------------------|
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | ICS01                                 |
| 000060 <sub>H</sub> | ICP1  | ICP0  | ICE1  | ICE0  | EG11  | EG10  | EG01  | EG00  | Input capture control status register |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub>   |

The input capture control status register (ICS01) consists of bits that have the functions explained below.

### [bit7, bit6] ICP1, ICP0

These bits are the interrupt flags of input capture. When a valid edge of the external input pin is detected, this bit is set to "1". If the interrupt permit bit (ICE1, ICE0) is set, an interrupt may occur when a valid edge is detected.

Writing "0" clears this bit. Writing "1" has no effect. An instruction of the read-modify-write type always reads "1".

|   |  |
|---|--|
| 0 | No valid edge detected (initial value) |
| 1 | Valid edge detected                    |

- ICP1: Corresponds to input capture 1
- ICP0: Corresponds to input capture 0

### [bit5, bit4] ICE1, ICE0

These bits are used as the interrupt permit bits of input capture. If these bits are set to "1" and the interrupt flag (ICP1, ICP0) is set, then an input capture interrupt occurs.

|   |                                    |
|---|------------------------------------|
| 0 | Interrupt prohibit (initial value) |
| 1 | Interrupt permit                   |

- ICE1: Corresponds to input capture 1
- ICE0: Corresponds to input capture 0

### [bit3, bit2, bit1, bit0] EG11, EG10, EG01, EG00

These bits specify the valid edge polarity of external input. Also, they are used to specify the enable of input capture operations.

| EG11/EG01 | EG10/EG00 | Edge detection polarity                       |
|-----------|-----------|---|
| 0         | 0         | No edge detected (stop state) (initial value) |
| 0         | 1         | Rising edge detected                          |
| 1         | 0         | Falling edge detected                         |
| 1         | 1         | Both edges detected                           |

- EG11/EG10: Corresponds to input capture 1
- EG01/EG00: Corresponds to input capture 0

## 12.4 Interrupt of 16-bit Input/Output Timer

The interrupt request of the 16-bit input/output timer occurs for three following.

- The counter value of the free-run timer overflows.
- The trigger edge input to the input capture input pin is performed.
- A match with the output compare is detected.

The DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS) can be activated for the interrupt of the input capture and output compare.

### ■ Interrupt of 16-bit Input/Output Timer

Table 12.4-1 shows the interrupt control bit and interrupt source of the 16-bit input/output timer.

**Table 12.4-1 Interrupt of 16-bit Input/Output Timer**

|                                     | Timer counter overflow interrupt          | Input capture interrupt                            | Output compare interrupt   |
|-------------------------------------|---|--|--|
| Interrupt request flag              | TCCS: IVF (bit7)                          | ICS01: ICP1 (bit7) ch.1<br>ICS01: ICP0 (bit6) ch.0 | OCS01/23/45: ICP1 (bit7) ch.1,3,5<br>OCS01/23/45: ICP0 (bit6) ch.0,2,4 |
| Interrupt request output enable bit | TCCS: IVFE (bit6)                         | ICS01: ICE1 (bit5) ch.1<br>ICS01: ICE0 (bit4) ch.0 | OCS01/23/45: ICE1 (bit5) ch.1,3,5<br>OCS01/23/45: ICE0 (bit4) ch.0,2,4 |
| Interrupt generation source         | Counter overflow of 16-bit free-run timer | Valid edge input to input capture input pin        | Match between output compare register value and counter value          |

ICS01: ICP0/ICE0 correspond to input capture pin (IN0).

ICS01: ICP1/ICE1 correspond to input capture pin (IN0).

OCS01/23/45: ICP0/ICE0 correspond to output compare pins (OUT0/OUT2/OUT4).

OCS01/23/45: ICP1/ICE1 correspond to output compare pins (OUT1/OUT3/OUT5).

#### ● Timer counter overflow interrupt

##### When the timer counter overflow interrupt request flag is set

The timer counter overflow generation flag in the timer counter control status register is set when the followings occur (TCCS: IVF=1)

- When an overflow ("FFFF<sub>H</sub>" → "0000<sub>H</sub>") occurs in counting up of the free-run timer
- When the initialization by compare clear register is set to enable (TCCS: MODE=1) and a match between the setting value of the free-run timer and the value of the compare clear register occurs.

##### When the timer counter overflow interrupt request occurs

If the timer counter overflow interrupt request is set to enable (TCCS: IVFE=1), the interrupt request is generated when the timer counter overflow generation flag is set to 1 (TCCS: IVF=1).

#### ● Input capture interrupt

The interrupt operation when the valid edge (ICS: EG) set by the input capture pin is detected is shown as follows:

- The count value of the free-run timer upon detection is stored in the input capture register.
- The valid edge detection flag in the control status register is set to 1 (ICS: ICP=1).
- When the input capture interrupt request output is set to enable (ICS: ICE=1), the interrupt request occurs.

### ● Output compare interrupt

The interrupt operation when a match between the count value of the free-run timer and the setting value of the compare register is detected is shown as follows:

- The output compare match flag in the control register is set to 1 (OCS:IOP=1).
- When the output compare interrupt request is set to enable (OCS: IOE=1), the interrupt request occurs.

### ■ Interrupt of 16-bit Input/Output Timer, DMA Transfer, and EI<sup>2</sup>OS

Table 12.4-2 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 12.4-2 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source  | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|---|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|   |                          |                      | Number           | Address             | Number                     | Address             |
| Input capture (ch.0) fetch*                                       | ○                        | 5                    | #26              | FFFF94 <sub>H</sub> | ICR07                      | 0000B7 <sub>H</sub> |
| Input capture (ch.1) fetch  | ○                        | 6                    | #27              | FFFF90 <sub>H</sub> | ICR08                      | 0000B8 <sub>H</sub> |
| Output compare (ch.0) match                                       | ○                        | 8                    | #28              | FFFF8C <sub>H</sub> |                            |                     |
| Output compare (ch.1) match                                       | ○                        | 9                    | #29              | FFFF88 <sub>H</sub> | ICR09                      | 0000B9 <sub>H</sub> |
| Output compare (ch.2) match                                       | ○                        | 10                   | #30              | FFFF84 <sub>H</sub> |                            |                     |
| Output compare (ch.3) match                                       | ○                        | ×                    | #31              | FFFF80 <sub>H</sub> | ICR10                      | 0000BA <sub>H</sub> |
| Output compare (ch.4) match                                       | ○                        | ×                    | #32              | FFFF7C <sub>H</sub> |                            |                     |
| Output compare (ch.5) match*                                      | ○                        | ×                    | #33              | FFFF78 <sub>H</sub> | ICR11                      | 0000BB <sub>H</sub> |
| 16-bit free-run timer overflow,*<br>16-bit reload timer underflow | ○                        | 12                   | #35              | FFFF70 <sub>H</sub> | ICR12                      | 0000BC <sub>H</sub> |

× : Interrupt request flag is not cleared.

○ : Interrupt request flag is cleared.

\* : This interrupt source shares the interrupt source and interrupt number of other peripheral function.  
For details, see Table 3.2-2.

#### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, other interrupt function cannot be used. The interrupt request enable bit of the relevant resource is set to "0" to execute the software polling processing.

### ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The input capture and free-run timer correspond to the DMA transfer and EI<sup>2</sup>OS functions. The output compare corresponds to the DMA transfer function only for EI<sup>2</sup>OS function and ch.0 to ch.2. When the DMA or EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 12.5 16-bit Input/Output Timer Operation

---

This section explains the operation and timing of the 16-bit input/output timer.

---

### ■ Operation and Timing of 16-bit Input/Output Timer

The 16-bit input/output timer handles the operation and timing for the following items:

- Free-run timer operation
- Output compare operation
- Input capture operation
- Free-run timer timing
  - Count timing
  - Clear timing
- Output compare timing
  - Interrupt timing
  - Change timing of the output pin
- Timing of input capture
  - Capture timing to the input signal

## 12.5.1 Operation of Free-Run Timer

This section explains the operation and timing of the free-run timer.

### ■ Operation of Free-run Timer

The free-run timer starts counting at a counter value of "0000" after clearing reset operation. This counter value is used as a reference time for output compare and input capture.

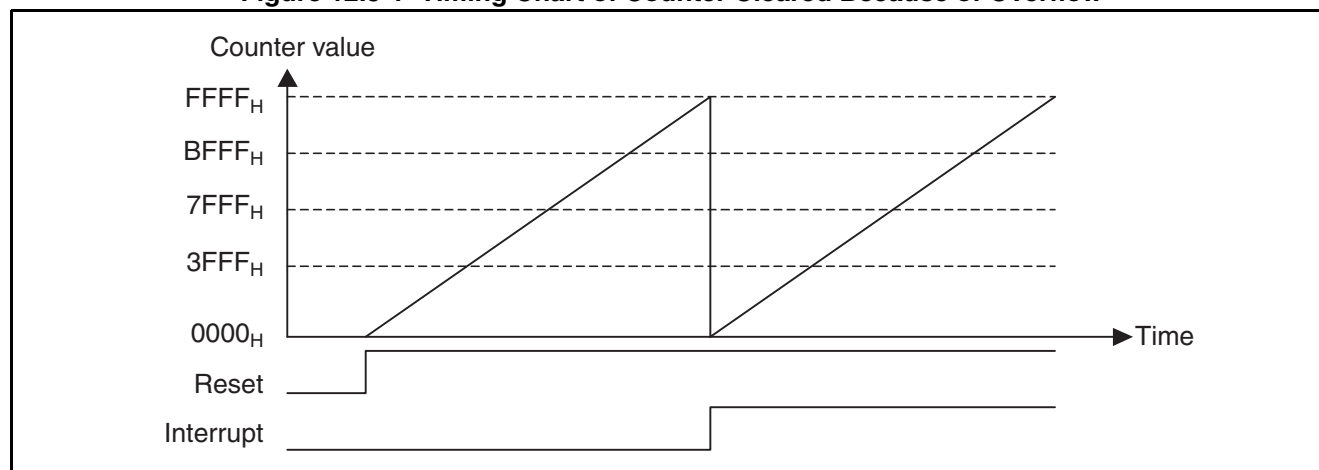
The count value is cleared under the following conditions:

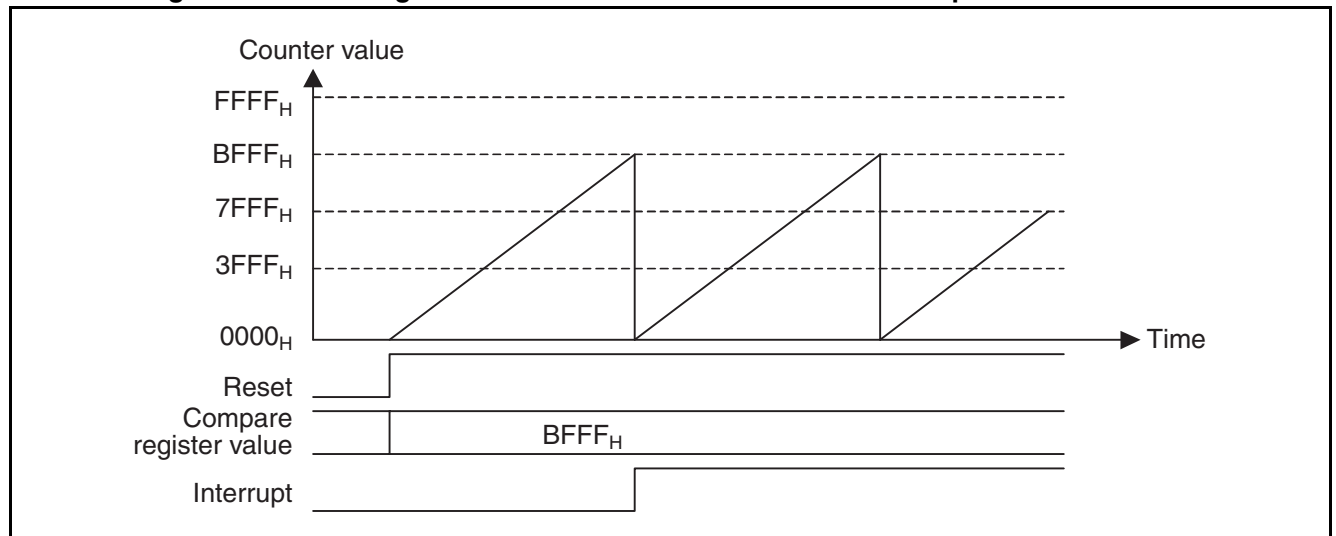
- Overflow occurs
- Compare match is found with the output compare value 0 (mode setting is required)
- SCLR bit of TCCS register is set to "1"
- TCDT register is set to "0000"
- Reset occurs

An interrupt occurs if an overflow is generated or the counter value of free-run timer the value of compare register 0 matches compare results (a compare results match interrupt requires mode setting).

Figure 12.5-1 shows the timing chart of the counter cleared because of an overflow. Figure 12.5-2 shows the timing chart of the counter cleared because of a compare results match.

**Figure 12.5-1 Timing Chart of Counter Cleared Because of Overflow**



**Figure 12.5-2 Timing Chart of Counter Cleared Because of Compare Results Match**

## 12.5.2 Operation of output compare

Output compare compares the specified compare register value with the free-run timer value, and if they match, it issues an interrupt request and reverses the output level.

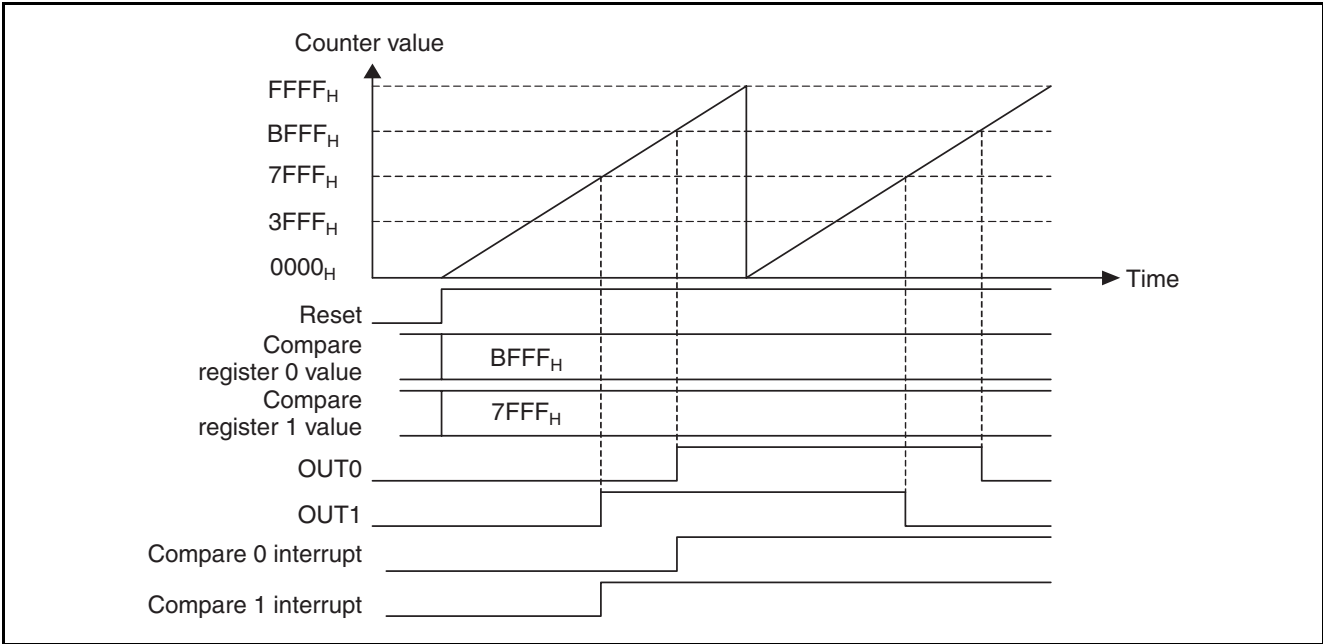
### ■ Examples of Output Waveform

Examples of output waveform are shown below.

○ **Example of output waveform where compare registers 0 and 1 are used**

Figure 12.5-3 shows an example of output waveform where the initial value of output is specified as "0".

**Figure 12.5-3 Example of Output Waveform Where Compare Registers 0 and 1 are Used  
(Initial Value of Output = 0)**

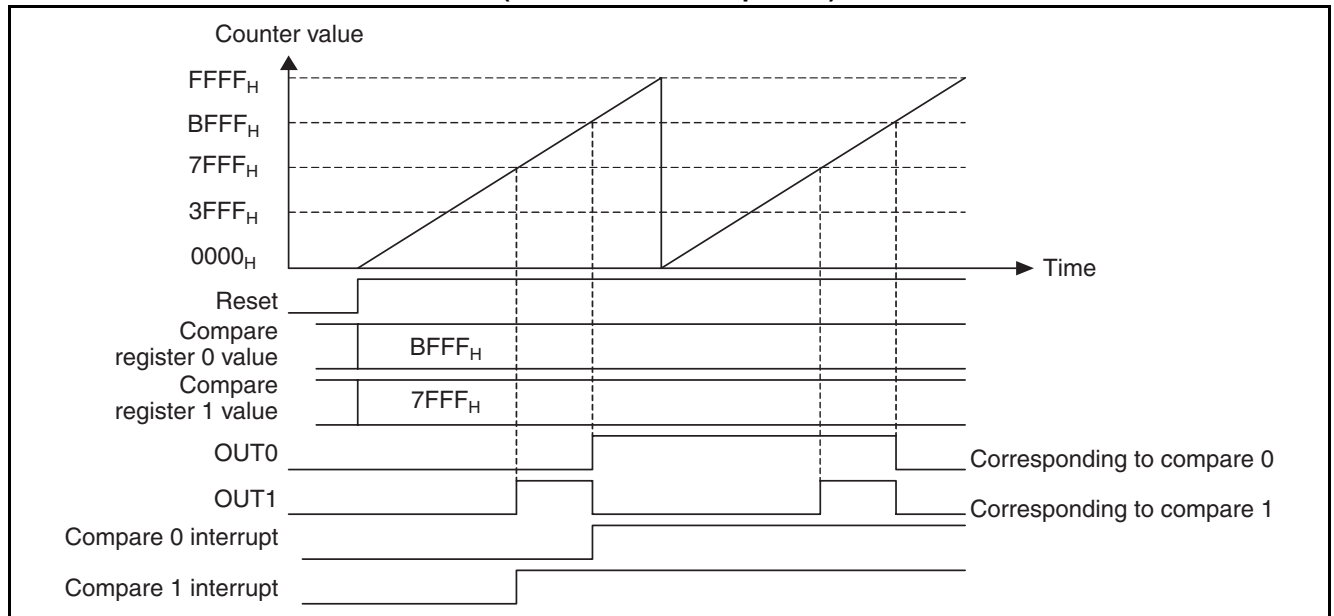


If CMOD = 1, two pairs of compare registers may be used to change the output level.

○ **Example of output waveform from two pairs of compare registers**

Figure 12.5-4 shows an example of output waveform where the initial value of output is specified as "0".

**Figure 12.5-4 Example of Output Waveform from Two Pairs of Compare Registers  
(Initial Value of Output = 0)**



**Note:**

To rewrite the compare register, perform within the compare interrupt routine or compare operation disabled state. Be sure not to occur a compare result match and writing the compare register simultaneously.



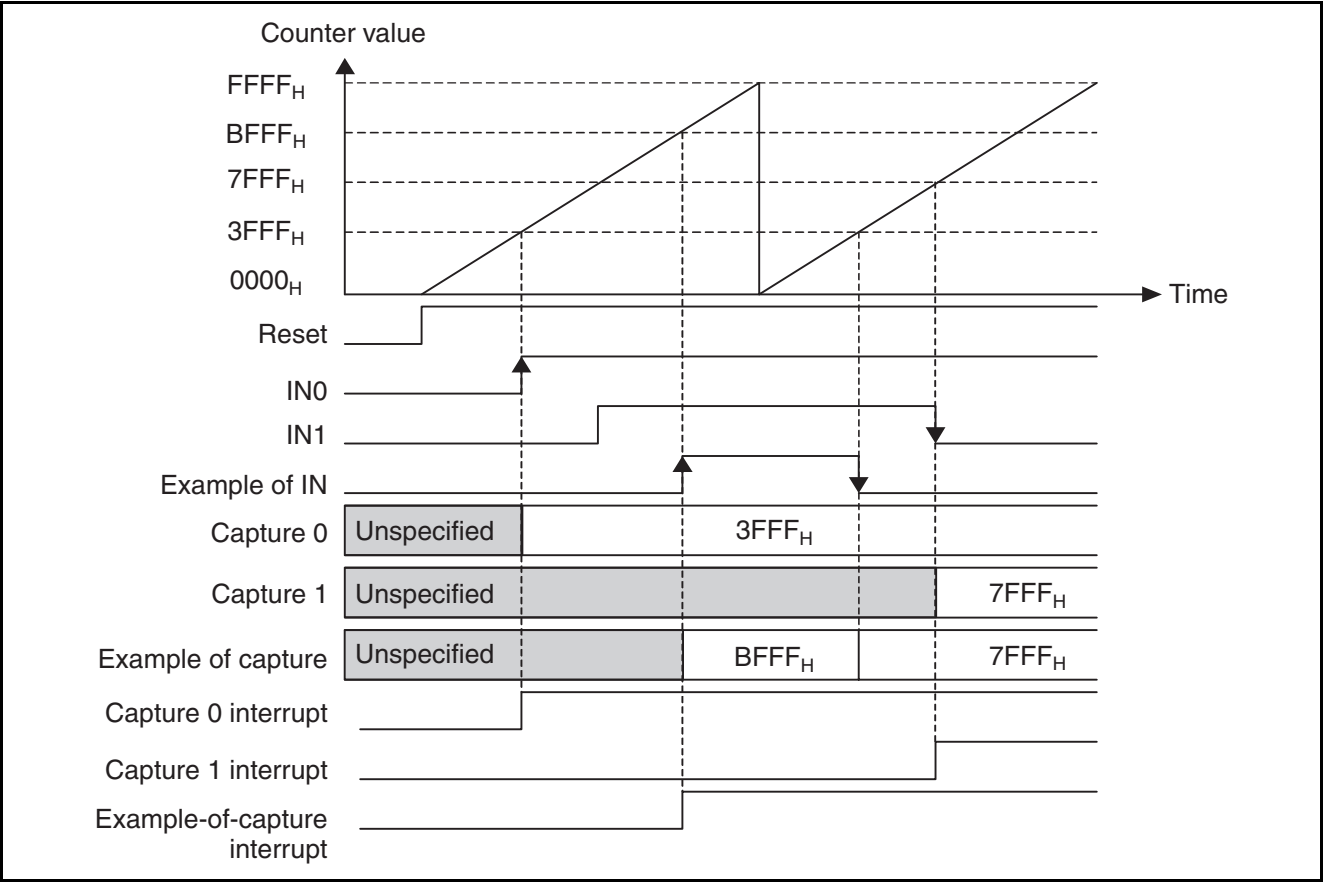
### 12.5.3 Operation of input capture

Input capture generates interrupt request by reading the free-run timer value into the capture register when the specified valid edge is detected.

■ Example of Input Capture Timing

Figure 12.5-5" shows an example of input capture timing.

Figure 12.5-5 Example of Input Capture Timing



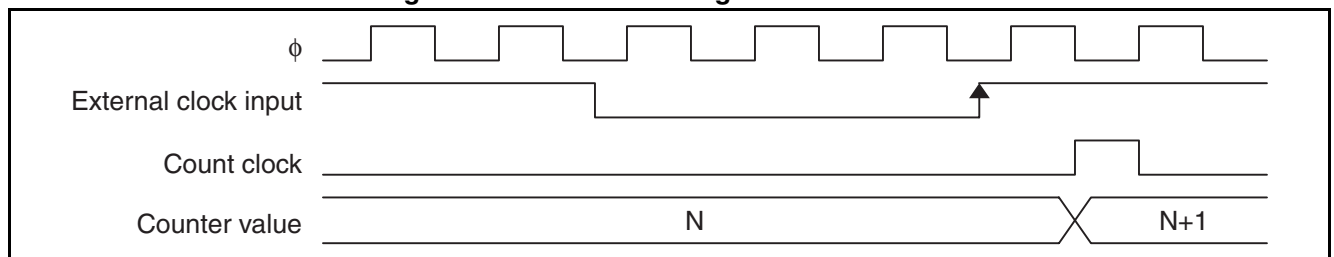
## 12.5.4 Free-Run Timer Timing

The free-run timer is incremented according to the timing of input clock (internal or external clock). If an external clock is selected, counting is performed at the rising edge.

### ■ Count Timing of Free-run Timer

Figure 12.5-6 shows the count timing of the free-run timer.

**Figure 12.5-6 Count Timing of Free-run Timer**

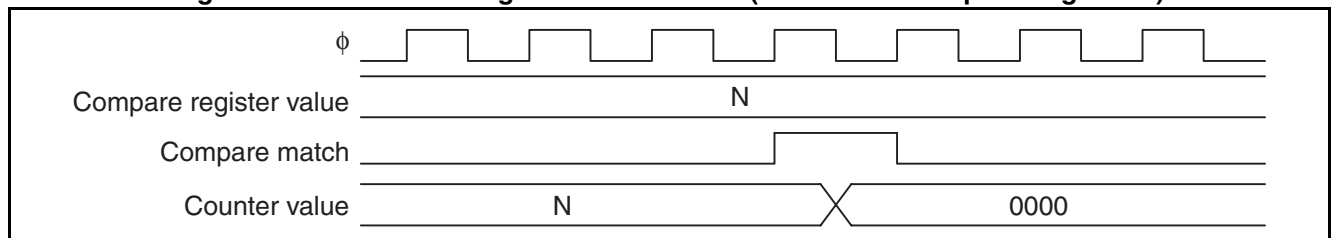


The counter is cleared by a reset, clearing by software, or a match with compare register 0 and free-run timer. Counter clear by a reset or software occurs when a clear operation is performed. Counter clear because of a match with compare register 0 occurs in sync with count timing.

### ■ Clear Timing of Free-run Timer (Match with Compare Register 0)

Figure 12.5-7 shows the clear timing of the free-run timer caused by a match with compare register 0.

**Figure 12.5-7 Clear Timing of Free-run Timer (match with Compare Register 0)**



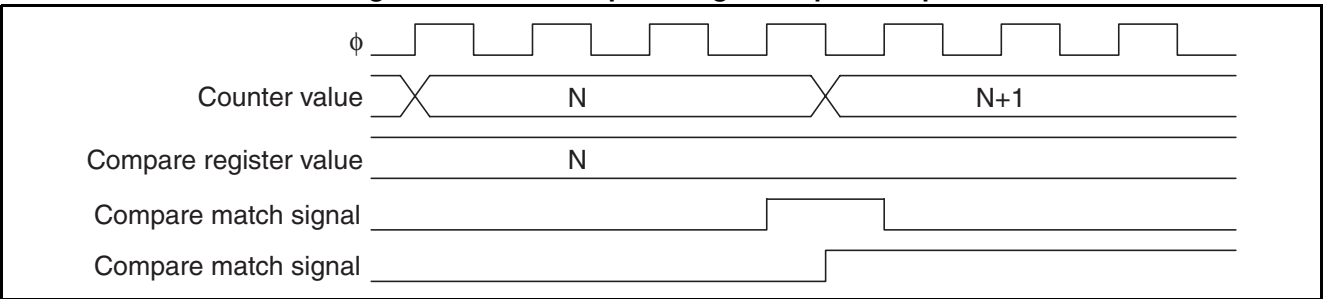
### 12.5.5 Output compare timing

The output compare is used to issue compare match signals when the free-run timer and the compare register have matching values, to reverse the output value, and to generate interrupts. Output reverse timing at the compare match is in sync with the counter timing.

■ Interrupt Timing

Figure 12.5-8 shows the interrupt timing of output compare.

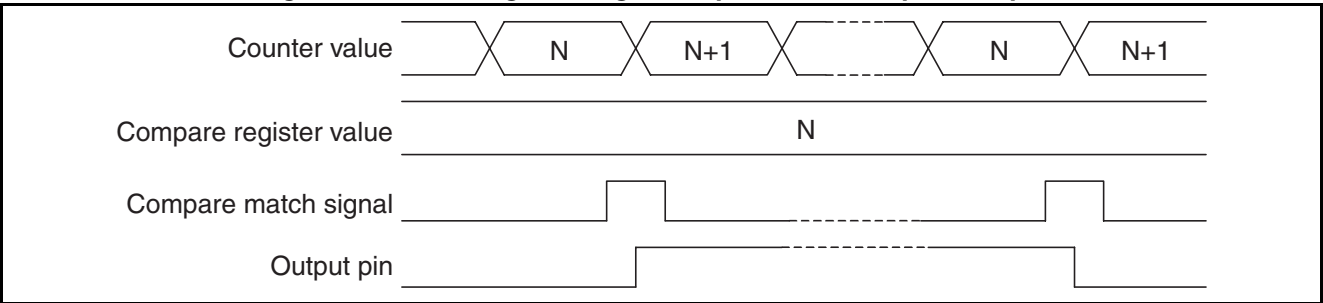
Figure 12.5-8 Interrupt Timing of Output Compare



■ Change Timing of Output Pin

Figure 12.5-9 shows the change timing of the output pin for output compare.

Figure 12.5-9 Change Timing of Output Pin for Output Compare



Note:

To rewrite the compare register, perform within the compare interrupt routine or compare operation disabled state. Be sure not to occur a compare result match and writing the compare register simultaneously.

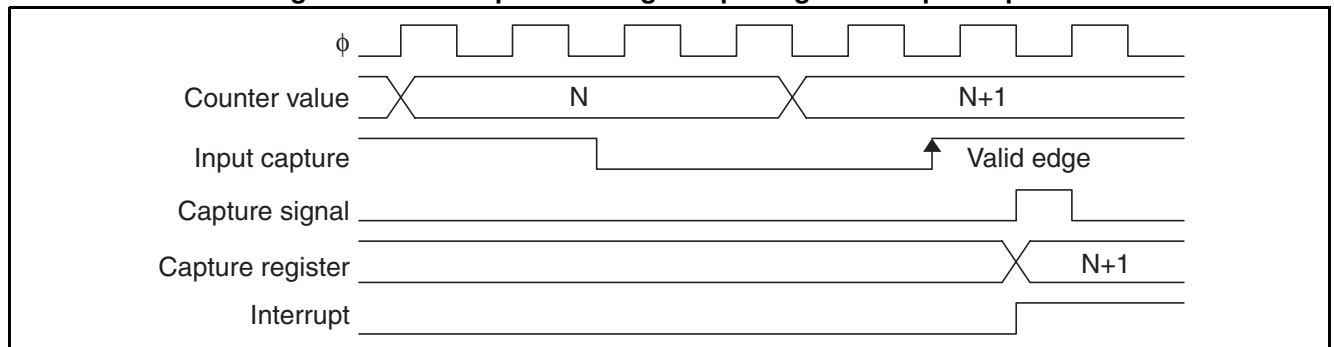
## 12.5.6 Timing of input capture

This section describes a capture timing of the input signal for input capture.

### ■ Capture Timing to Input Signal

Figure 12.5-10 shows the capture timing of input signal for input capture.

**Figure 12.5-10 Capture Timing of Input Signal for Input Capture**



## 12.6 Program Example of 16-bit Input/Output Timer

This section describes the program example of the 16-bit input/output timer.

### ■ Program Example of Free-run Timer

|   |                         |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
|---|-------------------------|------|-------------------|-------|--------------------------|------|----------------------------|-------|-------------------|-------|-------------------------------------|-------|--------------|-------|---------------|---------|----------------------|------|---------------------|-------|------------|-------|--|-------------------------|-----------------------|------------|--|-------------------------|------------------------------|-----------|------------------------|--|-------------------|--|--|
| <p>Example of setting procedure</p> <p>Count number of times overflow by free-run timer, clock = <math>\phi/2^4</math>, and interrupt processing</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Control free-run timer</li> </ul> <table border="1"> <tr> <td>Set control register</td><td>TCCS</td></tr> <tr> <td>Clock selection&gt;&gt;</td><td>.ECKE</td></tr> <tr> <td>Interrupt request flag&gt;&gt;</td><td>.IVF</td></tr> <tr> <td>Interrupt request enable&gt;&gt;</td><td>.IVFE</td></tr> <tr> <td>Count operation&gt;&gt;</td><td>.STOP</td></tr> <tr> <td>Initialization condition of timer&gt;&gt;</td><td>.MODE</td></tr> <tr> <td>TCDT clear&gt;&gt;</td><td>.SCLR</td></tr> <tr> <td>Count clock&gt;&gt;</td><td>.CLK2-0</td></tr> <tr> <td>Set timer data value</td><td>TCDT</td></tr> </table> <ul style="list-style-type: none"> <li>Interrupt related</li> </ul> <table border="1"> <tr> <td>Set interrupt level</td><td>ICR12</td></tr> <tr> <td>Set I flag</td><td>(CCR)</td></tr> </table> <ul style="list-style-type: none"> <li>Set variable</li> </ul> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start free-run timer ch.0</li> </ul> <table border="1"> <tr> <td></td><td>Register name, bit name</td></tr> <tr> <td>Start count operation</td><td>TCCS .STOP</td></tr> </table> <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Interrupt processing</li> </ul> <table border="1"> <tr> <td></td><td>Register name, bit name</td></tr> <tr> <td>Clear interrupt request flag</td><td>TCCS .IVF</td></tr> <tr> <td>(Arbitrary processing)</td><td></td></tr> <tr> <td>Count of variable</td><td></td></tr> </table> <p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>Note:<br/>Setting related to clock and setting of <code>_set_il</code> (numeric value) are required in advance. See the chapter of clock and interrupt.</p> | Set control register    | TCCS | Clock selection>> | .ECKE | Interrupt request flag>> | .IVF | Interrupt request enable>> | .IVFE | Count operation>> | .STOP | Initialization condition of timer>> | .MODE | TCDT clear>> | .SCLR | Count clock>> | .CLK2-0 | Set timer data value | TCDT | Set interrupt level | ICR12 | Set I flag | (CCR) |  | Register name, bit name | Start count operation | TCCS .STOP |  | Register name, bit name | Clear interrupt request flag | TCCS .IVF | (Arbitrary processing) |  | Count of variable |  | <p>Program example</p> <pre> void FREE_RUN_TIMER_sample(void) {     FREERUN_initial();     FREERUN_start(); }  void FREERUN_initial(void) {     IO_TCCS.word = 0x006C; /* Setting value=0000_0000_0110_1100 */                           /* bit15 = 0  ECKE internal clock source */                           /* bit7 = 0  IVF interrupt request flag */                           /* bit6 = 1  Enable IVFE interrupt */                           /* bit5 = 1  Disable STOP count */                           /* bit4 = 0  Initialize by MODE reset, clear bit */                           /* bit3 = 1  Initialize SCLR free-run timer value */                           /* bit2-0 = 100 CLK2-0 count clock <math>\phi/16</math> */     IO_TCDT = 0x0000; /* Initialize timer data value */      IO_ICR12.byte = 0x00; /* Set interrupt level of free-run timer (arbitrary value) */     __EI();               /* Enable interrupt */     count = 0; }  void FREERUN_start(void) {     IO_TCCS.bit.STOP = 0; /* bit5 = 0  Enable STOP count */ }  __interrupt void FREE_RUN_TIMER_int(void) {     IO_TCCS.bit.IVF = 0; /* bit7 = 0  Clear IVF overflow flag */      count++; }  #pragma intvect FREE_RUN_TIMER_int 35 </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
| Set control register  | TCCS                    |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Clock selection>>   | .ECKE                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Interrupt request flag>>  | .IVF                    |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Interrupt request enable>>  | .IVFE                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Count operation>>   | .STOP                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Initialization condition of timer>>   | .MODE                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| TCDT clear>>  | .SCLR                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Count clock>>   | .CLK2-0                 |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Set timer data value  | TCDT                    |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Set interrupt level   | ICR12                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Set I flag  | (CCR)                   |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
|   | Register name, bit name |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Start count operation   | TCCS .STOP              |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
|   | Register name, bit name |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Clear interrupt request flag  | TCCS .IVF               |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| (Arbitrary processing)  |                         |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |
| Count of variable   |                         |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                     |       |            |       |  |                         |                       |            |  |                         |                              |           |                        |  |                   |  |  |

## ■ Setting Method Other than Program Example

### ● Type of internal clock and selection method

There are eight internal clocks and they are set by the clock selection bit (TCCS.ECKE) and count clock bit (TCCS.CLK[2:0]).

| Internal clock       | Setting                    |                            | Count cycle           |                       |
|----------------------|----------------------------|----------------------------|-----------------------|-----------------------|
|                      | Clock selection bit (ECKE) | Count clock bit (CLK[2:0]) | $\phi = 20\text{MHz}$ | $\phi = 16\text{MHz}$ |
| To select $\phi$     | Set to "0"                 | Set to "000 <sub>B</sub> " | 50ns                  | 62.5ns                |
| To select $\phi/2$   | Set to "0"                 | Set to "001 <sub>B</sub> " | 100ns                 | 0.125 $\mu\text{s}$   |
| To select $\phi/4$   | Set to "0"                 | Set to "010 <sub>B</sub> " | 0.2 $\mu\text{s}$     | 0.25 $\mu\text{s}$    |
| To select $\phi/8$   | Set to "0"                 | Set to "011 <sub>B</sub> " | 0.4 $\mu\text{s}$     | 0.5 $\mu\text{s}$     |
| To select $\phi/16$  | Set to "0"                 | Set to "100 <sub>B</sub> " | 0.8 $\mu\text{s}$     | 1.0 $\mu\text{s}$     |
| To select $\phi/32$  | Set to "0"                 | Set to "101 <sub>B</sub> " | 1.6 $\mu\text{s}$     | 2.0 $\mu\text{s}$     |
| To select $\phi/64$  | Set to "0"                 | Set to "110 <sub>B</sub> " | 3.2 $\mu\text{s}$     | 4.0 $\mu\text{s}$     |
| To select $\phi/128$ | Set to "0"                 | Set to "111 <sub>B</sub> " | 6.4 $\mu\text{s}$     | 8.0 $\mu\text{s}$     |

### ● Selection method of external clock

Set by the clock selection bit (TCCS.ECKE), data direction bit, and port function bit.

| Setting                             |  | Pin  | Count cycle      |
|-------------------------------------|--|------|------------------|
| Set clock selection bit (ECKE) to 1 | Set data direction bit (DDR9.P93) to "0" | FRCK | $\phi/2$ or more |

### ● Method to enable/disable the count operation of free-run timer

Set by count operation bit (TCCS.STOP)

| Operation  | Count operation bit (STOP) |
|--|----------------------------|
| To enable the count operation of free-run timer  | Set to "0"                 |
| To disable the count operation of free-run timer | Set to "1"                 |

### ● Method to clear the free-run timer

The following method is used to clear the free-run timer.

- Set by clear bit (TCCS.SCLR)

| Operation                   | Clear bit (SCLR) |
|-----------------------------|------------------|
| To clear the free-run timer | Write "1"        |

- Match between free-run timer value and compare clear register value (set by timer initialization condition bit (TCCS.MODE))

| Operation   | Timer initialization condition bit (MODE) |
|---|---|
| To clear the free-run timer by match of comparison result | Set to "1"                                |

- Reset

The free-run timer is cleared by reset (external reset, watchdog reset, software reset).

- Set "0000<sub>H</sub>" to timer counter data register (TCDT)

When "0000<sub>H</sub>" is written to the timer counter data register (TCDT) during operation of the free-run timer is stopped, the count value is cleared to "0000<sub>H</sub>".

- Interrupt related register

The relationship between the interrupt level and interrupt vector is shown in the following table.

For details on the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

| Interrupt vector                    | Interrupt level setting register                                 |
|-------------------------------------|--|
| #35<br>Address: FFFF70 <sub>H</sub> | Interrupt level register (ICR12)<br>Address: 0000BC <sub>H</sub> |

Clear the interrupt request flag (TCCS.IVF) with software before returning from the interrupt processing because the flag is not cleared automatically (write "0" to IVF bit).

- Type of interrupt

One interrupt is provided only. Caused by overflow of the free-run timer.

- Method to enable interrupt

Enabling/disabling interrupt sets using the interrupt request enable bit (TCCS.IVFE).

| Control           | Interrupt request enable bit (IVFE) |
|-------------------|-------------------------------------|
| Disable interrupt | Set to "0"                          |
| Enable interrupt  | Set to "1"                          |

Clearing the interrupt request sets using the interrupt request bit (TCCS.IVF).

| Control                 | Interrupt request bit (IVF) |
|-------------------------|-----------------------------|
| Clear interrupt request | Write "0"                   |

- Method to stop operation of free-run timer

Set by count operation bit (TCCS.STOP).

| Operation                           | Count operation bit (STOP) |
|-------------------------------------|----------------------------|
| To stop count operation of free-run | Set to "1"                 |

## ■ Program Example of Output Compare

|  |                      |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
|--|----------------------|------|-------------------|-------|--------------------------|------|----------------------------|-------|-------------------|-------|-------------------------------------|-------|--------------|-------|---------------|---------|----------------------|------|----------------------|-------|---|-------|---------------------|------------|----------------------------|------------|--------------------------|------------|----------------------------|------------|------------------------|------------|------------------------|-------|------------------------|-------|---------------------|-------|---------------------|-------|------------|-------|-------------------|------------------|-------------------------|------------------|-----------------------|------------|---|
| <p>Example of setting procedure</p> <p>2-channel independent output compare operation<br/>(7FFFF, BFFFF), interrupt generation, no compare clear</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Control free-run timer</li> </ul> <table border="1"> <tr> <td>Set control register</td><td>TCCS</td></tr> <tr> <td>Clock selection&gt;&gt;</td><td>.ECKE</td></tr> <tr> <td>Interrupt request flag&gt;&gt;</td><td>.IVF</td></tr> <tr> <td>Interrupt request enable&gt;&gt;</td><td>.IVFE</td></tr> <tr> <td>Count operation&gt;&gt;</td><td>.STOP</td></tr> <tr> <td>Initialization condition of timer&gt;&gt;</td><td>.MODE</td></tr> <tr> <td>TCDT clear&gt;&gt;</td><td>.SCLR</td></tr> <tr> <td>Count clock&gt;&gt;</td><td>.CLK2-0</td></tr> <tr> <td>Set timer data value</td><td>TCDT</td></tr> </table> <ul style="list-style-type: none"> <li>Control output compare</li> </ul> <table border="1"> <tr> <td>Set control register</td><td>OCS01</td></tr> <tr> <td>Reverse operation of pin output level&gt;&gt;</td><td>.CMOD</td></tr> <tr> <td>Pin output enable&gt;&gt;</td><td>.OTE1,OTE0</td></tr> <tr> <td>Specify pin output level&gt;&gt;</td><td>.OTD1,OTD0</td></tr> <tr> <td>Interrupt request flag&gt;&gt;</td><td>.ICP1,ICP0</td></tr> <tr> <td>Interrupt request enable&gt;&gt;</td><td>.ICE1,ICE0</td></tr> <tr> <td>Set operation enable&gt;&gt;</td><td>.CST1.CST0</td></tr> <tr> <td>Set compare value ch.0</td><td>OCCP0</td></tr> <tr> <td>Set compare value ch.1</td><td>OCCP1</td></tr> </table> <ul style="list-style-type: none"> <li>Interrupt related</li> </ul> <table border="1"> <tr> <td>Set interrupt level</td><td>ICR08</td></tr> <tr> <td>Set interrupt level</td><td>ICR09</td></tr> <tr> <td>Set I flag</td><td>(CCR)</td></tr> </table> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start output compare</li> </ul> <table border="1"> <tr> <td>Interrupt control</td><td>OCS01 .ICE1.ICE0</td></tr> <tr> <td>Start compare operation</td><td>OCS01 .CST1.CST0</td></tr> </table> <ul style="list-style-type: none"> <li>Start free-run timer</li> </ul> <table border="1"> <tr> <td>Start count operation</td><td>TCCS .STOP</td></tr> </table> | Set control register | TCCS | Clock selection>> | .ECKE | Interrupt request flag>> | .IVF | Interrupt request enable>> | .IVFE | Count operation>> | .STOP | Initialization condition of timer>> | .MODE | TCDT clear>> | .SCLR | Count clock>> | .CLK2-0 | Set timer data value | TCDT | Set control register | OCS01 | Reverse operation of pin output level>> | .CMOD | Pin output enable>> | .OTE1,OTE0 | Specify pin output level>> | .OTD1,OTD0 | Interrupt request flag>> | .ICP1,ICP0 | Interrupt request enable>> | .ICE1,ICE0 | Set operation enable>> | .CST1.CST0 | Set compare value ch.0 | OCCP0 | Set compare value ch.1 | OCCP1 | Set interrupt level | ICR08 | Set interrupt level | ICR09 | Set I flag | (CCR) | Interrupt control | OCS01 .ICE1.ICE0 | Start compare operation | OCS01 .CST1.CST0 | Start count operation | TCCS .STOP | <p>Program example</p> <pre> void OUTPUT01_sample(void) {     freerun_initial();     OUTPUT01_initial();     OUTPUT01_start();     freerun_start(); }  void freerun_initial(void) {     IO_TCCS.word = 0x0020; /* Setting value=0000_0000_0010_0000 */                           /* bit15 = 0   ECKE internal clock source */                           /* bit7 = 0   IVF interrupt request flag */                           /* bit6 = 0   Enable IVFE interrupt */                           /* bit5 = 1   Disable STOP count */                           /* bit4 = 0   Initialize by MODE reset, clear bit */                           /* bit3 = 0   Initialize SCLR free-run timer value */                           /* bit2-0 = 000 CLK2-0 count clock <math>\phi/4=32\text{MHz}/4</math> */     IO_TCDT = 0x0000; /* Initialize timer data value */ }  void OUTPUT01_initial(void) {     IO_OCS01.word = 0x0C00; /* Setting value=0000_1100_0000_0000 */                           /* bit15-13 = 000 Undefined bit */                           /* bit12 = 0   Reverse CMOD ch.0,ch.1 level */                           /* bit11-10 = 11 Enable OTE1,OTE0 pin output */                           /* bit9-8 = 00 OTD1,OTD0 compare pin output 0 */                           /* bit7-6 = 00 Clear ICP1,ICP0 output compare flag */                           /* bit5-4 = 00 Disable ICE1,ICE0 output compare interrupt */                           /* bit3-2 = 00 Undefined bit */                           /* bit1-0 = 00 Disable CST1,CST0 compare operation */     IO_OCCP0 = BFFF; /* Set Compare register ch.0 */     IO_OCCP1 = 7FFF; /* Set Compare register ch.1 */      IO_ICR08.byte = 0x00; /* Set output compare ch.0 interrupt level (arbitrary value) */     IO_ICR09.byte = 0x00; /* Set output compare ch.1 interrupt level (arbitrary value) */     __EI(); /* Enable interrupt */ }  void OUTPUT01_start(void) {     IO_OCS01.word = 0x0C30; /* bit5-4 = 11 Enable ICE1,ICE0 output compare interrupt */     IO_OCS01.word = 0x0C33; /* bit1-0 = 11 Enable CST1,CST0 compare operation */ }  void freerun_start(void) {     IO_TCCS.bit.STOP = 0; /* bit4 = 0 Enable STOP count */ } </pre> |
| Set control register   | TCCS                 |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Clock selection>>  | .ECKE                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Interrupt request flag>>   | .IVF                 |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Interrupt request enable>>   | .IVFE                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Count operation>>  | .STOP                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Initialization condition of timer>>  | .MODE                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| TCDT clear>>   | .SCLR                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Count clock>>  | .CLK2-0              |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set timer data value   | TCDT                 |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set control register   | OCS01                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Reverse operation of pin output level>>  | .CMOD                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Pin output enable>>  | .OTE1,OTE0           |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Specify pin output level>>   | .OTD1,OTD0           |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Interrupt request flag>>   | .ICP1,ICP0           |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Interrupt request enable>>   | .ICE1,ICE0           |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set operation enable>>   | .CST1.CST0           |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set compare value ch.0   | OCCP0                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set compare value ch.1   | OCCP1                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set interrupt level  | ICR08                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set interrupt level  | ICR09                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Set I flag   | (CCR)                |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Interrupt control  | OCS01 .ICE1.ICE0     |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Start compare operation  | OCS01 .CST1.CST0     |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |
| Start count operation  | TCCS .STOP           |      |                   |       |                          |      |                            |       |                   |       |                                     |       |              |       |               |         |                      |      |                      |       |   |       |                     |            |                            |            |                          |            |                            |            |                        |            |                        |       |                        |       |                     |       |                     |       |            |       |                   |                  |                         |                  |                       |            |   |

(Continued)



(Continued)

| <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Interrupt processing</li> </ul> <table border="1" data-bbox="194 273 617 493"> <thead> <tr> <th colspan="2">Register name. bit name</th></tr> </thead> <tbody> <tr> <td>Clear interrupt request flag</td><td>OCS01 .ICP0</td></tr> <tr> <td>(Arbitrary processing)</td><td></td></tr> <tr> <td>*****</td><td></td></tr> <tr> <td>Clear interrupt request flag</td><td>OCS01 .ICP1</td></tr> <tr> <td>(Arbitrary processing)</td><td></td></tr> <tr> <td>*****</td><td></td></tr> </tbody> </table> <p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>Note:<br/>Setting related to clock and setting of _set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</p> | Register name. bit name |  | Clear interrupt request flag | OCS01 .ICP0 | (Arbitrary processing) |  | ***** |  | Clear interrupt request flag | OCS01 .ICP1 | (Arbitrary processing) |  | ***** |  | <pre> __interrupt void OUTPUT0_int(void) {     IO_OCS01.bit.ICP0 = 0; /* bit6 = 0      Clear ICP0 interrupt flag */     . . . . . }  __interrupt void OUTPUT1_int(void) {     IO_OCS01.bit.ICP1 = 0; /* bit7 = 0      Clear ICP1 interrupt flag */     . . . . . }  #pragma intvect OUTPUT0_int 28 #pragma intvect OUTPUT1_int 29     </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
|---|-------------------------|--|------------------------------|-------------|------------------------|--|-------|--|------------------------------|-------------|------------------------|--|-------|--|---|
| Register name. bit name   |                         |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |
| Clear interrupt request flag  | OCS01 .ICP0             |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |
| (Arbitrary processing)  |                         |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |
| *****   |                         |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |
| Clear interrupt request flag  | OCS01 .ICP1             |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |
| (Arbitrary processing)  |                         |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |
| *****   |                         |  |                              |             |                        |  |       |  |                              |             |                        |  |       |  |   |

## ■ Setting Method Other than Program Example

### ● Method to set compare value

The compare value is written to the compare registers (OCCP0 to OCCP5).

### ● Method to set compare mode (valid for OUT1, OUT2, OUT3, OUT4, OUT5 output)

Set by compare mode bit (OCS01.CMOD, OCS23.CMOD, OCS45.CMOD).

| Operation  | Compare mode bit            |
|--|-----------------------------|
| To reverse OUT1 output by a match with comparison result between free-run timer and compare register 1   | Set (OCS01.CMOD) bit to "0" |
| To reverse OUT3 output by a match with comparison result between free-run timer and compare register 3   | Set (OCS23.CMOD) bit to "0" |
| To reverse OUT5 output by a match with comparison result between free-run timer and compare register 5   | Set (OCS45.CMOD) bit to "0" |
| To reverse Out1 output by match with comparison result between free-run timer and compare register 0 and between free-run timer and compare register 1 | Set (OCS01.CMOD) bit to "1" |
| To reverse Out3 output by match with comparison result between free-run timer and compare register 2 and between free-run timer and compare register 3 | Set (OCS23.CMOD) bit to "1" |
| To reverse Out5 output by match with comparison result between free-run timer and compare register 4 and between free-run timer and compare register 5 | Set (OCS45.CMOD) bit to "1" |

The following is output regardless of the CMOD bit.

- OUT0 output reverses output by a match with comparison result between free-run timer and compare register 0.
- OUT2 output reverses output by a match with comparison result between free-run timer and compare register 2.

- OUT4 output reverses output by a match with comparison result between free-run timer and compare register 4.

● Method to enable/disable compare operation

Set by compare operation bit (OCS01.CST[1:0], OCS23.CST[1:0], OCS45.CST[1:0]).

| Operation                           | Compare   | Compare operation enable bit |
|-------------------------------------|-----------|------------------------------|
| To stop (disable) compare operation | Compare 0 | Set (OCS01.CST0) to "0".     |
|                                     | Compare 1 | Set (OCS01.CST1) to "0".     |
|                                     | Compare 2 | Set (OCS23.CST0) to "0".     |
|                                     | Compare 3 | Set (OCS23.CST1) to "0".     |
|                                     | Compare 4 | Set (OCS45.CST0) to "0".     |
|                                     | Compare 5 | Set (OCS45.CST1) to "0".     |
| To enable compare operation         | Compare 0 | Set (OCS01.CST0) to "1".     |
|                                     | Compare 1 | Set (OCS01.CST1) to "1".     |
|                                     | Compare 2 | Set (OCS23.CST1) to "1".     |
|                                     | Compare 3 | Set (OCS23.CST0) to "1".     |
|                                     | Compare 4 | Set (OCS45.CST0) to "1".     |
|                                     | Compare 5 | Set (OCS45.CST1) to "1".     |

● Method to set the initial level of compare pin output

Set by compare pin output specification bit (OCS01.OTD[1:0], OCS23.OTD[1:0], OCS45.OTD[1:0]).

| Operation                | Compare pin output specification bit |
|--------------------------|--------------------------------------|
| Set compare 0 pin to "L" | Set (OCS01.OTD0) to "0"              |
| Set compare 0 pin to "H" | Set (OCS01.OTD0) to "1"              |
| Set compare 1 pin to "L" | Set (OCS01.OTD1) to "0"              |
| Set compare 1 pin to "H" | Set (OCS01.OTD1) to "1"              |
| Set compare 2 pin to "L" | Set (OCS23.OTD0) to "0"              |
| Set compare 2 pin to "H" | Set (OCS23.OTD0) to "1"              |
| Set compare 3 pin to "L" | Set (OCS23.OTD1) to "0"              |
| Set compare 3 pin to "H" | Set (OCS23.OTD1) to "1"              |
| Set compare 4 pin to "L" | Set (OCS45.OTD1) to "0"              |
| Set compare 4 pin to "H" | Set (OCS45.OTD1) to "1"              |
| Set compare 5 pin to "L" | Set (OCS45.OTD1) to "0"              |
| Set compare 5 pin to "H" | Set (OCS45.OTD1) to "1"              |

## CHAPTER 12 16-BIT INPUT/OUTPUT TIMER

- Method to set compare pin OUT0-OUT5 to output

Set by port function register (OCS01.OTE[1:0], OCS23.OTE[1:0], OCS45.OTE[1:0]).

| Operation                             | Port function bit       |
|---------------------------------------|-------------------------|
| To set compare 0 pin (OUT0) to output | Set (OCS01.OTE0) to "1" |
| To set compare 1 pin (OUT1) to output | Set (OCS01.OTE1) to "1" |
| To set compare 2 pin (OUT2) to output | Set (OCS23.OTE0) to "1" |
| To set compare 3 pin (OUT3) to output | Set (OCS23.OTE1) to "1" |
| To set compare 4 pin (OUT4) to output | Set (OCS45.OTE0) to "1" |
| To set compare 5 pin (OUT5) to output | Set (OCS45.OTE0) to "1" |

- Method to clear free-run timer

Set by clear bit (TCCS.SCLR).

| Operation               | Clear bit (SCLR) |
|-------------------------|------------------|
| To clear free-run timer | Write "1"        |

For other methods, see "12.3.1 Free-Run Timer".

- Method to enable compare operation

Set by the compare operation enable bit (OCS01.CST[1:0], OCS23.CST[1:0], OCS45.CST[1:0]).

- Method to clear the free-run timer when the free-run timer value matches with the compare register value

Set by timer initialization condition bit (TCCS.MODE).

| Operation                                       | Timer initialization condition bit (MODE) |
|---|---|
| To clear free-run timer 0 by match of compare 0 | Set (TCCS.MODE) to "1"                    |

- Interrupt related register

The relationship between channel, interrupt level, and interrupt vector is shown in the following table.

For details on the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

| Channel          | Interrupt vector                     | Interrupt level setting register                                  |
|------------------|--------------------------------------|---|
| Output compare 0 | #28<br>Address : FFFF8C <sub>H</sub> | Interrupt level register (ICR08)<br>Address : 0000B8 <sub>H</sub> |
| Output compare 1 | #29<br>Address : FFFF88 <sub>H</sub> | Interrupt level register (ICR09)<br>Address : 0000B9 <sub>H</sub> |
| Output compare 2 | #30<br>Address : FFFF84 <sub>H</sub> | Interrupt level register (ICR09)<br>Address : 0000B9 <sub>H</sub> |
| Output compare 3 | #31<br>Address : FFFF80 <sub>H</sub> | Interrupt level register (ICR10)<br>Address : 0000BA <sub>H</sub> |
| Output compare 4 | #32<br>Address : FFFF7C <sub>H</sub> | Interrupt level register (ICR10)<br>Address : 0000BA <sub>H</sub> |
| Output compare 5 | #33<br>Address : FFFF78 <sub>H</sub> | Interrupt level register (ICR11)<br>Address : 0000BB <sub>H</sub> |

Clear the interrupt request flag (OCS01.ICP[1:0], OCS23.ICP[1:0], OCS45.ICP[1:0]) by writing "0" to ICP[1:0] bit with software before returning from the interrupt processing because the flag is not cleared automatically.

- Type of interrupt

One interrupt is provided. Caused by a match of the comparison result.

- Method to enable interrupt

Enabling the interrupt is set by the interrupt request enable bit (OCS01.ICE[1:0], OCS23.ICE[1:0], OCS45.ICE[1:0]).

| Control           | Interrupt request enable bit (ICE0, ICE1) |
|-------------------|---|
| Disable interrupt | Set to "0"                                |
| Enable interrupt  | Set to "1"                                |

The interrupt request is cleared by the interrupt request bit (OCS01.ICP[1:0], OCS23.ICP[1:0], OCS45.ICP[1:0]).

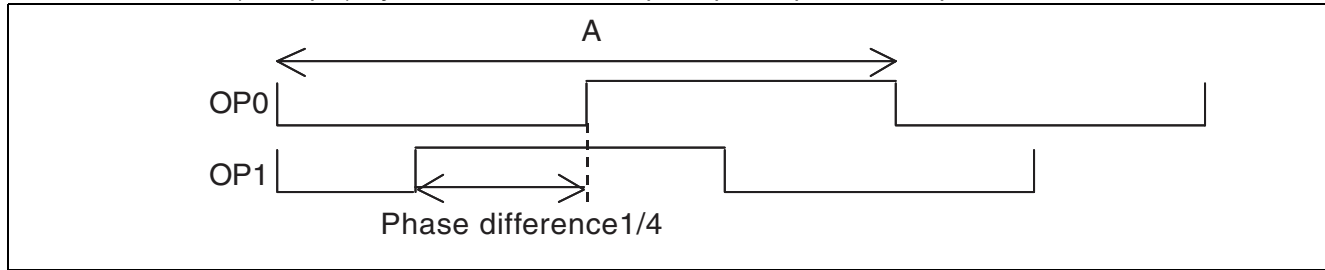
| Control                 | Interrupt request bit (ICP0, ICP1) |
|-------------------------|------------------------------------|
| Clear interrupt request | Write "0"                          |

## CHAPTER 12 16-BIT INPUT/OUTPUT TIMER

- Calculation method of compare value

- Toggle output pulse

(Example) Cycle : A, method to output 2-phase pulse of 1/4 phase difference



Formula : Compare 0 value =  $(A/2) / \text{count clock}$

Compare 1 value =  $(A/4) / \text{count clock}$

(Count clock : time set by free-run timer)

Note:

Setting to clear the free-run timer 0 by a match of compare 0 (TCCS0.MODE = 1 ) and Setting of CMOD = 0 are required.

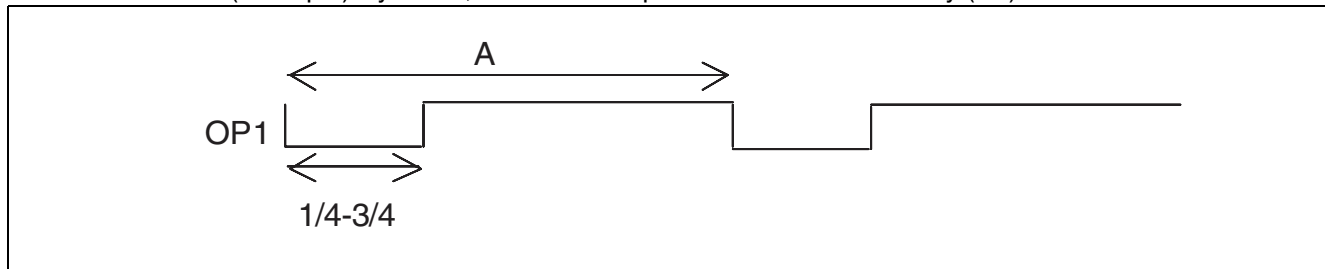
Calculation example :  $A=1024\mu\text{s}$ , Count clock= $125\text{ns}$

Compare 0 value =  $(1024000 / 2) / 125 - 1 = 4095 = \text{FFF}_H$

Compare 1 value =  $(1024000 / 4) / 125 - 1 = 1023 = 7FF_H$

- PWM output

(Example) Cycle : A, method to output PWM of 1/4 to 3/4 duty ("L")



Formula: Compare 0 value =  $A / \text{Count clock}$

Compare 1 value =  $(A/4) / \text{Count clock}$  (at 1/4 duty)

$(A \times 3/4) / \text{Count clock}$  (at 3/4 duty)

(Count clock: time set by free-run timer)

Note:

Setting to clear the free-run timer 0 by a match of compare 0 (TCCS0.MODE=1) and setting of CMOD= 1 are required.

Calculation example:  $A=1024\mu\text{s}$ , Count clock= $125\text{ns}$

Compare 0 value =  $1024000 / 125 - 1 = 8191 = 1FFF_H$

Compare 1 value =  $(1024000 / 4) / 125 - 1 = 1023 = 7FF_H$   
(at 1/4 duty)

$(1024000 \times 3 / 4) / 125 - 1 = 1023 = BFF_H$   
(at 3/4 duty)

## ■ Program Example of Input Capture

### Example of setting procedure

Rising edge of the pulse input to IN0 is detected, and the value of the free-run timer is recorded. Repeat this twice and measure the time between triggers. However reading of capture value and calculation processing is interrupt processing.

#### <Initial setting>

- Control free-run timer

|                                     |         |
|-------------------------------------|---------|
| Set control register                | TCCS    |
| Clock selection>>                   | .ECKE   |
| Interrupt request flag>>            | .IVF    |
| Interrupt request enable>>          | .IVFE   |
| Count operation>>                   | .STOP   |
| Initialization condition of timer>> | .MODE   |
| TCDT clear>>                        | .SCLR   |
| Count clock>>                       | .CLK2-0 |
| Set timer data value                | TCDT    |

- Port

Register name. bit name

|                       |           |
|-----------------------|-----------|
| Set IN0 input of port | DDR9 .P96 |
|-----------------------|-----------|

- Control input capture

|                                   |            |
|-----------------------------------|------------|
| Set control register              | ICS01      |
| Interrupt request flag>>          | .ICP1,ICP0 |
| Interrupt request enable>>        | .ICE1,ICE0 |
| Select ch.1 valid edge polarity>> | .EG11,EG10 |
| Select ch.0 valid edge polarity>> | .EG01,EG00 |

- Interrupt related

|                     |       |
|---------------------|-------|
| Set interrupt level | ICR07 |
| Set I flag          | (CCR) |

- Set variable

#### <Start>

- Start input capture ch.0

Register name. bit name

|                   |             |
|-------------------|-------------|
| Interrupt control | ICS01 .ICE0 |
|-------------------|-------------|

- Start free-run timer

Register name. bit name

|                       |            |
|-----------------------|------------|
| Start count operation | TCCS .STOP |
|-----------------------|------------|

#### <Interrupt>

- Interrupt processing

Register name. bit name

|                              |             |
|------------------------------|-------------|
| Clear interrupt request flag | ICS01 .ICP0 |
| (Arbitrary processing)       |             |
| *****                        |             |

#### <Interrupt vector>

- Set vector table

Note:

Setting related to clock and setting of `_set_il` (numeric value) are required in advance. See the chapter of clock and interrupt.

### Program example

```
void INPUT0_sample(void)
```

```
{
    freerun_initial();
    INPUT0_initial();
    INPUT0_start();
    freerun_start();
}
```

```
void freerun_initial(void)
```

```
{
    IO_TCCS.word = 0x20; /* Setting value=0000_0000_0010_0000 */
                          /* bit15 = 0  ECKE internal clock source */
                          /* bit7 = 0  IVF interrupt request flag */
                          /* bit6 = 0  IVFE disable interrupt */
                          /* bit5 = 1  Disable STOP count */
                          /* bit4 = 0  Initialize by MODE reset, clear bit */
                          /* bit3 = 0  Initialize SCLR free-run timer value */
                          /* bit2-0 = 000 CLK2-0 count clock  $\phi$  */
    IO_TCDT = 0x0000; /* Initialize timer data value */
}
```

```
void INPUT0_initial(void)
```

```
{
    IO_DDR9.byte = 0x00; /* DDR9 IN0(P96) input */
    IO_ICS01.byte = 0x01; /* Setting value=0000_0001 */
                          /* bit7-6 = 00 Clear ICP1, 0 valid edge flag */
                          /* bit5-4 = 00 Disable ICE1, 0 interrupt */
                          /* bit3-2 = 00 No EG11, EG10 ch.1 edge detection */
                          /* bit1-0 = 01 EG01, EG00 ch.0 rising edge detection */
}
```

```
IO_ICR07.byte = 0x10; /* Set input capture ch.0 interrupt level
                        (arbitrary value) */
```

```
_EI(); /* Enable interrupt */
count = 0;
```

```
void INPUT0_start(void)
```

```
{
    IO_ICS01.bit.ICE0 = 1; /* bit4 = 1 Enable ICE0 ch.0 interrupt */
}
```

```
void freerun_start(void)
```

```
{
    IO_TCCS.bit.STOP = 0; /* bit5 = 0 Enable STOP count */
}
```

```
__interrupt void INPUT0_int(void)
```

```
{
    IO_ICS01.bit.ICP0 = 0; /* bit6 = 0 Clear ICP0 valid edge detection flag */
    if(count==0)
        Data1 = IO_IPCP0; /* Record value of free-run timer (first time) */
    else if(count==1) {
        Data2 = IO_IPCP0; /* Record value of free-run timer (second time) */
        cycle = (data2-data1)*125; /* Measure time */
    }
    count++;
}
```

```
#pragma intvect INPUT0_int 26
```

Note:

For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".

## ■ Setting Methods Other than Program Example

- Type of external input valid edge polarity and selection method

The valid edge polarity has rising edge, falling edge, and both edges.

Set by the valid edge polarity bit of external input (ICS01.EG[01:00], ICS01.EG[11:10]).

| Operation              | Valid edge polarity bit of external input<br>(EG[01:00], EG[11:10]) |
|------------------------|---|
| To select rising edge  | Select "00 <sub>B</sub> "   |
| To select falling edge | Select "10 <sub>B</sub> "   |
| To select both edges   | Select "11 <sub>B</sub> "   |

- Method to set to external input pin (IN0, IN1)

Set by data direction bit (DDR9.P96, DDR9.P97).

| Operation                               | Data direction bit (P96, P97) |
|---|-------------------------------|
| To set to external input pin (IN0, IN1) | Set to "0"                    |

- Interrupt related register

The relationship between channel, interrupt level, and interrupt vector is shown in the following table.

For details on the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

| Channel         | Interrupt vector                    | Interrupt level setting register                                  |
|-----------------|-------------------------------------|---|
| Input capture 0 | #26<br>Address: FFFF94 <sub>H</sub> | Interrupt level register (ICR07)<br>Address : 0000B7 <sub>H</sub> |
| Input capture 1 | #27<br>Address: FFFF90 <sub>H</sub> | Interrupt level register (ICR08)<br>Address : 0000B8 <sub>H</sub> |

The interrupt request flag (ICS01.ICP0, ICS01.ICP1) is not cleared automatically. Clear the flag by writing "0" to the input capture interrupt request flag (ICP1, ICP0) with software before returning from interrupt processing.

- Type of interrupt

One interrupt is provided only. Occurs at edge detection of the input signal.

● Method to enable interrupt

Enabling the interrupt sets using the interrupt request enable bit (ICS01.ICE0, ICS01.ICE1).

|                   | Interrupt request enable bit (ICE0, ICE1) |
|-------------------|---|
| Disable interrupt | Set to "0"                                |
| Enable interrupt  | Set to "1"                                |

Clearing the interrupt request set using the interrupt request bit (ICS01.ICP0, ICS01.ICP1).

|                         | Interrupt request bit (ICP0, ICP1) |
|-------------------------|------------------------------------|
| Clear interrupt request | Write "0"                          |

● Method to measure pulse width of input signal

• "H" width measurement:

Both edges are set at edge detection.

Rising edge is detected and then falling edge is detected.

Pulse width = { value recorded at falling (value of input capture register) +  
 "10000<sub>H</sub>" × number of times overflow -  
 value recorded at rising (value of input capture register) } ×  
 count clock width of free-run timer

Example: Value recorded at falling = 2320<sub>H</sub>, value recorded at rising = A635<sub>H</sub>,

number of times overflow = 1, count clock = 125ns

→ pulse width = (2320<sub>H</sub> + 10000<sub>H</sub> - A635<sub>H</sub>) × 125ns = 3997.375μs

• Cycle measurement:

Rising (or falling) is set at edge detection.

Edge is detected twice.

Cycle = {value recorded at second time (value of input capture register) +  
 "10000<sub>H</sub>" × number of times overflow -  
 value recorded at first time (value of input capture register) } ×  
 count clock width of free-run timer





# CHAPTER 13 8/16-BIT UP/DOWN COUNTER/TIMER

---

**This chapter provides an overview of the 8/16-bit up/down counter/timer, explains the configuration and functions of its registers, interrupt and its operation.**

---

- 13.1 Overview of 8/16-bit Up/Down Counter Timer
- 13.2 Configuration of 8/16-bit Up/Down Counter/Timer
- 13.3 Configuration and Functions of Registers for 8/16-bit Up/Down Counter/Timer
- 13.4 Interrupt of 8/16-bit Up/Down Counter/Timer
- 13.5 8/16-bit Up/Down Counter/Timer Operation
- 13.6 Program Example of 8/16-bit Up/Down Counter/Timer

## 13.1 Overview of 8/16-bit Up/Down Counter Timer

---

The 8/16-bit up/down counter/timer consists of six event input pins, two 8-bit up/down counter registers, two 8-bit reload/compare registers, and their control circuits.

---

### ■ Major Functions of 8/16-bit Up/Down Counter/Timer

- 8-bit count register used for counting in a range of 0 to 255 (in the 16 bits × one operation mode, counting in a range of 0 to 65535 is possible).
- Four types of count modes can be selected for the count clock.
  - Timer mode
  - Up/down count mode
  - Phase difference decremented mode (two times)
  - Phase difference decremented mode (eight times)
- In the timer mode, the count clock is selected from two types of internal clocks:
  - Divided by 2 (80ns at internal machine cycle  $f = 25$  MHz)
  - Divided by 8 (320ns at internal machine cycle  $f = 25$  MHz)
- In the up/down count mode, a detection edge of the external pin input signal may be selected:
  - Falling edge detection
  - Rising edge detection
  - Both falling and rising edges detection
  - Edge detection prohibited
- The phase difference count mode is suitable for counting encoder output such as motor, where encoder output from phases A, B, and Z is used as input, thereby facilitating high-precision counting of rotation angle and rotations.
- The ZIN pin is used to select from two types of functions:
  - Counter clear function
  - Gate function
- Compare and reload functions are provided, where each function or a combination of them are available. Starting both functions allows up/down counting with any time width.
  - Compare function (an interrupt is issued during compare)
  - Compare function (an interrupt is issued and the counter is cleared during compare)
  - Reload function (an interrupt is issued and reloaded during compare)
  - Compare and reload functions (an interrupt is issued and the counter is cleared during compare, and an interrupt is issued and reloaded when an underflow occurs)
  - Compare and reload prohibited
- Interrupt generation is individually controlled during comparison, a reload (underflow), or an overflow.
- The count direction flag identifies the count direction of the last counter operation
- An interrupt occurs when the count direction is switched

## 13.2 Configuration of 8/16-bit Up/Down Counter/Timer

The 8-bit up/down counter/timer has two channels and consists of three event input pins, one 8-bit up/down count, and one 8-bit reload/compare register per channel. Also, one of two 8-bit up/down counter/timer channels can be used as the 16-bit up/down counter/timer. (When using as the 16-bit up/down counter/timer, the register of ch.0 is valid.)

### ■ Block Diagram of 8/16-bit Up/Down Counter/Timer

Figure 13.2-1 and Figure 13.2-2 are block diagrams of the 8/16-bit up/down counter/timer.

**Figure 13.2-1 Block Diagram of 8/16-bit Up/Down Counter/Timer (ch.0)**

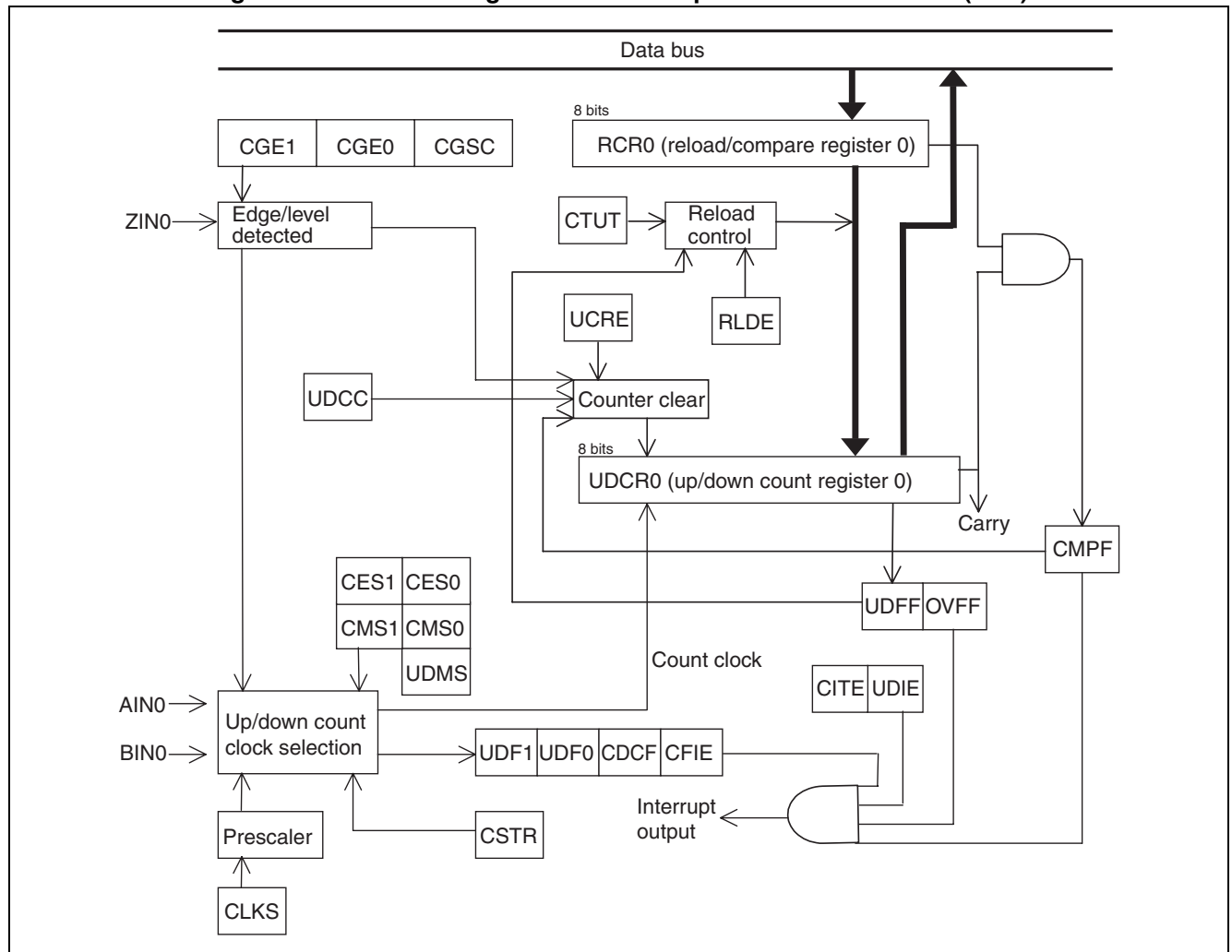
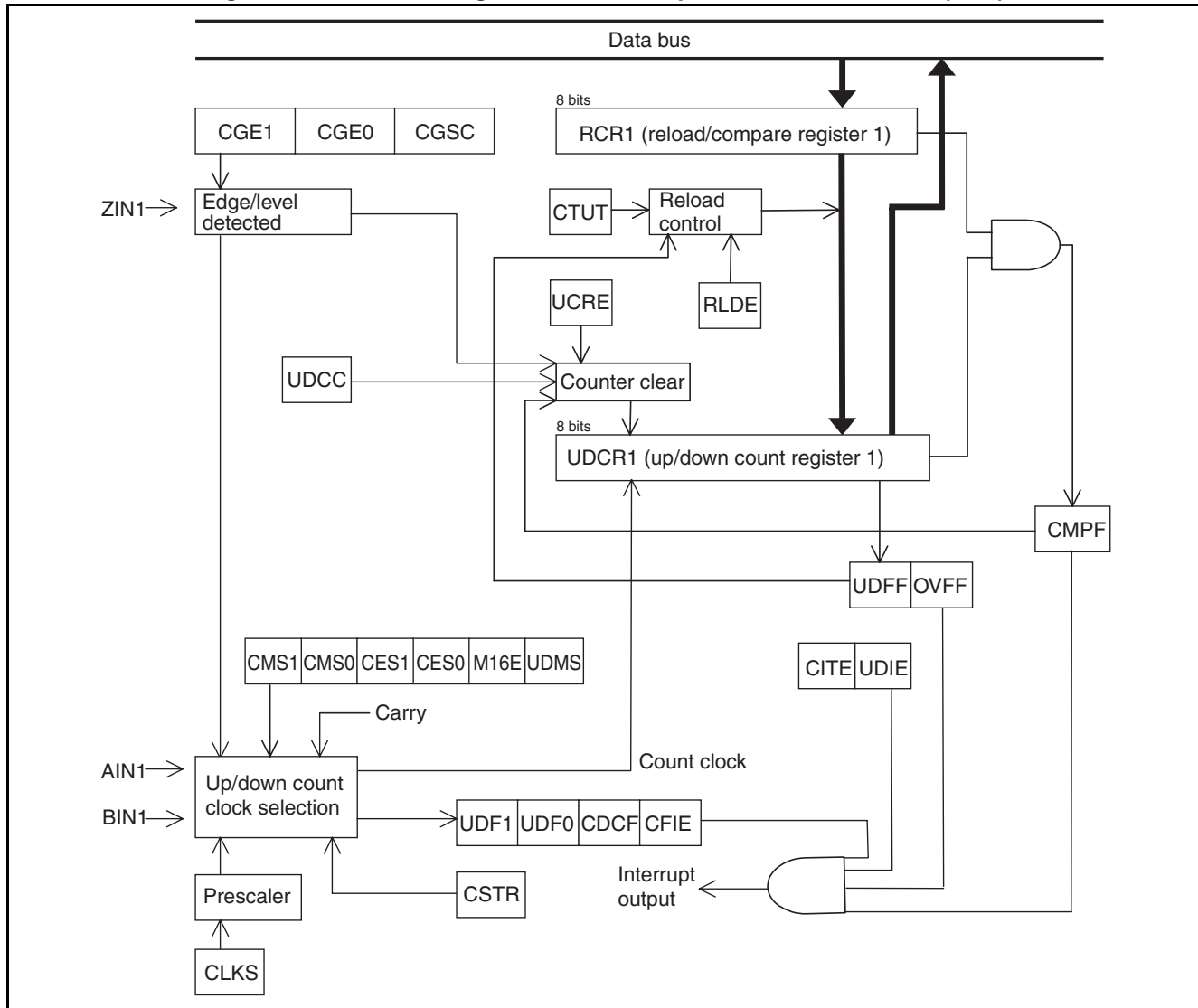


Figure 13.2-2 Block Diagram of 8/16-bit Up/Down Counter/Timer (ch.1)



### ■ Pin Related to 8/16-bit Up/Down Counter/Timer

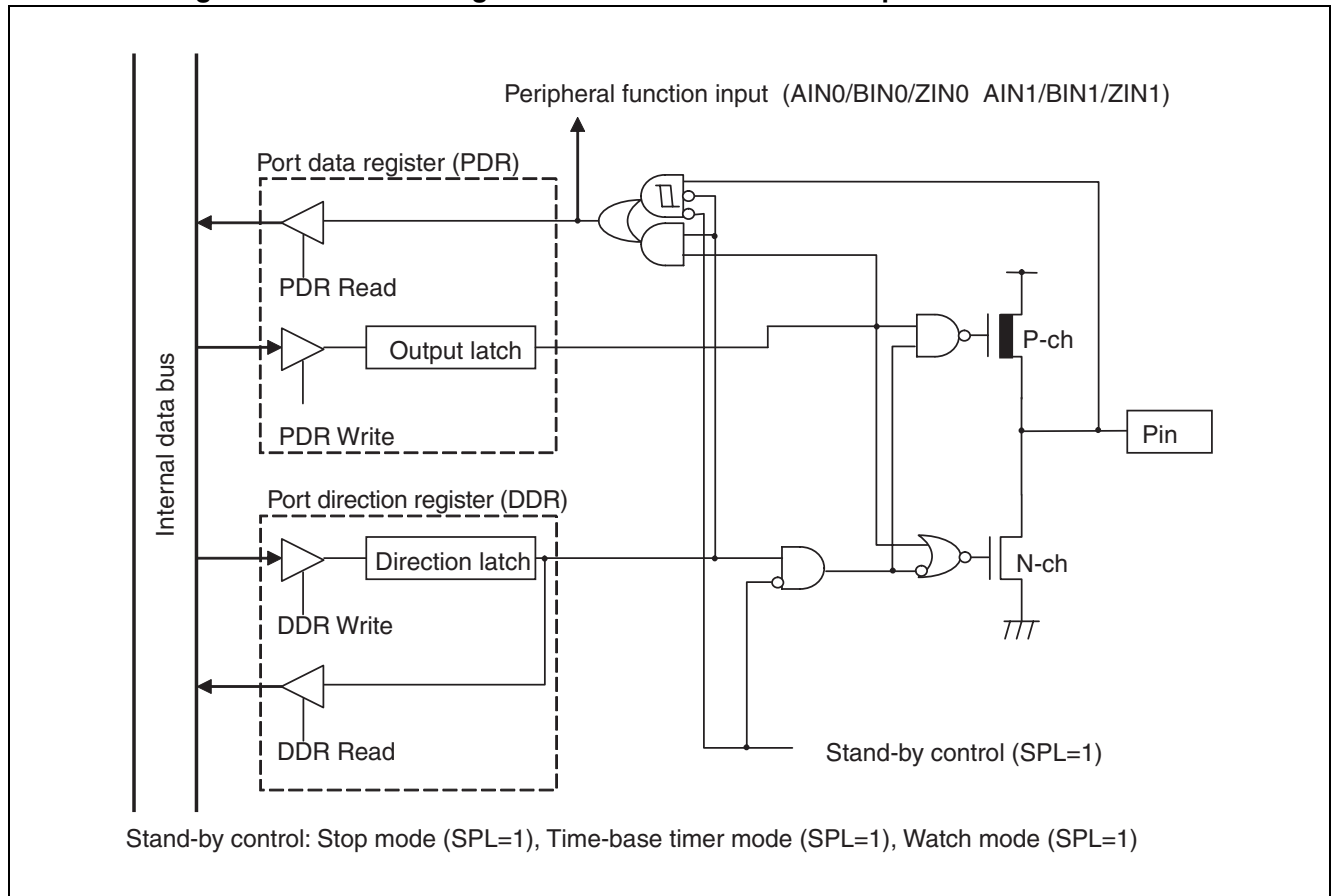
The pin related to the 8/16-bit up/down counter/timer has the AIN0/BIN0/ZIN0 and AIN1/BIN1/ZIN1 pins. The AIN0/BIN0/ZIN0 pin functions as the general-purpose I/O port (AIN0/P30, BIN0/P31, ZIN0/P32) and the input pin of the up/down counter/timer. The AIN1/BIN1/ZIN1 pin functions as the general-purpose I/O port (AIN1/P33, BIN1/P34, ZIN1/P35) and the input pin of the up/down counter/timer.

#### ○ Setting when using as AIN0/BIN0/ZIN0 and AIN1/BIN1/ZIN1 pins

When using as the AIN/BIN/ZIN input pin, the AIN0/P30, BIN/P31, ZIN0/P32 and AIN1/P33, BIN1/P31, ZIN1/P35 pins should be set to input port by the port direction register (DDR3 bit8, 9, 10, 11, 12, 13 → "0").

## ■ Block Diagram of Pin Related to 8/16-bit Up/Down Counter/Timer

Figure 13.2-3 Block Diagram of Pin Related to 8/16-bit Up/Down Counter/Timer



## 13.3 Configuration and Functions of Registers for 8/16-bit Up/Down Counter/Timer

This section shows the configuration and explains the function of the 8/16-bit up/down counter/timer registers.

### ■ List of 8/16-bit Up/Down Counter/Timer Registers

Figure 13.3-1 shows a list of registers for the 8/16-bit up/down counter/timer.

**Figure 13.3-1 List of Registers for 8/16-bit Up/Down Counter/Timer**

|              |     |  |  |  |  |          |   |  |  |
|--------------|-----|--|--|--|--|----------|---|--|--|
| 15           | 8 7 |  |  |  |  |          | 0 |  |  |
| UDCR1        |     |  |  |  |  | UDCR 0   |   |  |  |
| RCR1         |     |  |  |  |  | RCR 0    |   |  |  |
| Reserve area |     |  |  |  |  | CSR 0    |   |  |  |
| CCRH 0       |     |  |  |  |  | CCRL 0   |   |  |  |
| Reserve area |     |  |  |  |  | CSR 1    |   |  |  |
| CCRH 1       |     |  |  |  |  | CCRL 1   |   |  |  |
| ← 8 bits     |     |  |  |  |  | 8 bits → |   |  |  |

|                              |      |      |      |      |      |      |      |      |                        |
|------------------------------|------|------|------|------|------|------|------|------|------------------------|
| ch.0 UDCR0                   | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value          |
| Address: 000068 <sub>H</sub> | D07  | D06  | D05  | D04  | D03  | D02  | D01  | D00  | 00000000 <sub>B</sub>  |
| ch.1 UDCR1                   | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | Initial value          |
| Address: 000069 <sub>H</sub> | D17  | D16  | D15  | D14  | D13  | D12  | D11  | D10  | 00000000 <sub>B</sub>  |
| ch.0 RCR0                    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value          |
| Address: 00006A <sub>H</sub> | D07  | D06  | D05  | D04  | D03  | D02  | D01  | D00  | 00000000 <sub>B</sub>  |
| ch.1 RCR1                    | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | Initial value          |
| Address: 00006B <sub>H</sub> | D17  | D16  | D15  | D14  | D13  | D12  | D11  | D10  | 00000000 <sub>B</sub>  |
| ch.0 CSR0                    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value          |
| Address: 000072 <sub>H</sub> | CSTR | CITE | UDIE | CMPF | OVFF | UDFF | UDF1 | UDF0 | 00000000 <sub>B</sub>  |
| ch.1 CSR1                    |      |      |      |      |      |      |      |      |                        |
| Address: 000074 <sub>H</sub> |      |      |      |      |      |      |      |      |                        |
| ch.0 CCRL0                   | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value          |
| Address: 00006C <sub>H</sub> | UDMS | CTUT | UCRE | RLDE | UDCC | CGSC | CGE1 | CGE0 | 0X00X000 <sub>B</sub>  |
| ch.1 CCRL1                   |      |      |      |      |      |      |      |      |                        |
| Address: 000070 <sub>H</sub> |      |      |      |      |      |      |      |      |                        |
| ch.0 CCRH0                   | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | Initial value          |
| Address: 00006D <sub>H</sub> | M16E | CDCF | CFIE | CLKS | CMS1 | CMS0 | CES1 | CES0 | 00000000 <sub>B</sub>  |
| ch.1 CCRH1                   | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | Initial value          |
| Address: 000071 <sub>H</sub> | -    | CDCF | CFIE | CLKS | CMS1 | CMS0 | CES1 | CES0 | -00000000 <sub>B</sub> |

### 13.3.1 Counter control register (ch.0) upper (CCRH0)

This section shows the configuration and explains the functions of counter control register (ch.0) upper (CCRH0).

#### ■ Counter Control Register (ch.0) Upper (CCRH0)

The bit configuration of counter control register (ch.0) upper (CCRH0) is shown below.

**Figure 13.3-2 Bit Configuration of Counter Control Register (ch.0) Upper (CCRH0)**

|                                   |     |      |      |      |      |      |      |      |      |                       |
|-----------------------------------|-----|------|------|------|------|------|------|------|------|-----------------------|
| CCRH0                             | bit | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | Initial value         |
| ch.0 Address: 00006D <sub>H</sub> |     | M16E | CDCF | CFIE | CLKS | CMS1 | CMS0 | CES1 | CES0 | 00000000 <sub>B</sub> |
|                                   |     | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |                       |

Counter control register (ch.0) upper (CCRH0) consists of bits that have the functions explained below.

##### [bit15] M16E (16-bit mode permit)

This bit is used to select (switch) an operation mode of 8 bits × 2 channels or 16 bits × 1 channel.

| M16E | Setting 16-bit mode permit                         |
|------|--|
| 0    | 8 bits × 2 channels operation mode (initial value) |
| 1    | 16 bits × 1 channel operation mode                 |

If this bit is rewritten after its start, the count value is not assured.

##### [bit14] CDCF (count direction reversal flag)

This bit is a flag that is set when the count direction is switched. It is set in the count start mode when the count direction is switched from either up to down or down to up.

The initialization (writing "0") is only permitted.

Read-modify-write type instructions read "1" irrespective of bit values.

| CDCF | Direction reversal detection             |
|------|--|
| 0    | No reversal of direction (initial value) |
| 1    | One or more reversals of direction       |

##### [bit13] CFIE (count direction reversal interrupt enable)

If CDCF is defined, this bit is used to control interrupt output to the CPU. An interrupt occurs if count direction changes even a single time in the count start mode when this bit is set to "1".



| <b>CFIE</b> | <b>Direction reversal interrupt output</b>                   |
|-------------|--|
| 0           | Direction reversal interrupt output prohibit (initial value) |
| 1           | Direction reversal interrupt output permit                   |

**[bit12] CLKS (built-in prescaler selection)**

This bit is used to select the frequency of built-in prescaler in the selection of the timer mode.

It is only valid in the timer mode, and only decrementing (down count) is permitted.

| <b>CLKS</b> | <b>Internal clock selected</b>   |
|-------------|----------------------------------|
| 0           | 2 machine cycles (initial value) |
| 1           | 8 machine cycles                 |

If this bit is rewritten after its start, the count value is not assured.

**[bit11, bit10] CMS1, CMS0 (count mode selection)**

These bits are used to select the count mode.

| <b>CMS1</b> | <b>CMS0</b> | <b>Count mode</b>                                      |
|-------------|-------------|--|
| 0           | 0           | Timer mode [decremented] (initial value)               |
| 0           | 1           | Up/down count mode                                     |
| 1           | 0           | Phase difference count mode: frequency multiplied by 2 |
| 1           | 1           | Phase difference count mode: frequency multiplied by 4 |

If this bit is rewritten after its start, the count value is not assured.

**[bit9, bit8] CES1, CES0 (count clock edge selection)**

These bits are used to select the detection edge of external pins AIN and BIN in the up/down count mode.

This setting is invalid in modes other than up/down count.

| <b>CES1</b> | <b>CES0</b> | <b>Selected edge</b>                   |
|-------------|-------------|--|
| 0           | 0           | Edge detect prohibit (initial value)   |
| 0           | 1           | Falling edge detect                    |
| 1           | 0           | Rising edge detect                     |
| 1           | 1           | Both rising and falling edges detected |

If this bit is rewritten after its start, the count value is not assured.

## 13.3.2 Counter control register (ch.1) upper (CCRH1)

This section describes the configuration and explains the function of counter control register (ch.1) upper (CCRH1).

### ■ Counter Control Register (ch.1) Upper (CCRH1)

The bit configuration of the counter control register (ch.1) upper (CCRH1) is shown below.

**Figure 13.3-3 Bit Configuration of Counter Control Register (ch.1) Upper (CCRH1)**

| CCRH1                             | bit | 15 | 14   | 13   | 12   | 11   | 10   | 9    | 8    | Initial value         |
|-----------------------------------|-----|----|------|------|------|------|------|------|------|-----------------------|
| ch.1 Address: 000071 <sub>H</sub> |     | -  | CDCF | CFIE | CLKS | CMS1 | CMS0 | CES1 | CES0 | -0000000 <sub>B</sub> |
|                                   |     |    | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |                       |

Counter control register (ch.1) upper (CCRH1) consists of bits that have the functions explained below.

#### [bit14] CDCF (count direction reversal flag)

This bit is set when the count direction changes. It is set in the count start mode when the count direction changes from up to down or from down to up.

The initialization (writing "0") is only permitted.

Read-modify-write type instructions read "1" irrespective of bit values.

| CDCF | Direction reversal detection           |
|------|--|
| 0    | No direction reversals (initial value) |
| 1    | One or more reversals of direction     |

#### [bit13] CFIE (count direction reversal interrupt enable)

This bit is used to control interrupt output to the CPU if CDCF is defined. It generates an interrupt in the count start mode when the count direction changes if this bit is set to "1".

| CFIE | Direction reversal interrupt output                          |
|------|--|
| 0    | Direction reversal interrupt output prohibit (initial value) |
| 1    | Direction reversal interrupt output permit                   |

**[bit12] CLKS (built-in prescaler selection)**

This bit is used to select the frequency of built-in prescaler when the timer mode is selected.

This is only valid in the timer mode, where only decrementing is permitted.

| <b>CLKS</b> | <b>Selection internal clock</b>  |
|-------------|----------------------------------|
| 0           | 2 machine cycles (initial value) |
| 1           | 8 machine cycles                 |

If this bit is rewritten after its start, the count value is not assured.

**[bit11, bit10] CMS1, CMS0 (count mode selection)**

These bits are used to select the count mode.

| <b>CMS1</b> | <b>CMS0</b> | <b>Count mode</b>                                      |
|-------------|-------------|--|
| 0           | 0           | Timer mode [decremented] (initial value)               |
| 0           | 1           | Up/down count mode                                     |
| 1           | 0           | Phase difference count mode: frequency multiplied by 2 |
| 1           | 1           | Phase difference count mode: frequency multiplied by 4 |

If this bit is rewritten after its start, the count value is not assured.

**[bit9, bit8] CES1, CES0 (count clock edge selection)**

These bits are used in the up/down count mode to select a detection edge for external pins AIN and BIN.

This setting is invalid in modes other than up/down count.

| <b>CES1</b> | <b>CES0</b> | <b>Selected edge</b>                   |
|-------------|-------------|--|
| 0           | 0           | Edge detect prohibit (initial value)   |
| 0           | 1           | Falling edge detect                    |
| 1           | 0           | Rising edge detect                     |
| 1           | 1           | Both rising and falling edges detected |

If this bit is rewritten after its start, the count value is not assured.

### 13.3.3 Counter control register (ch.0/ch.1) lower (CCRL0/CCRL1)

This section describes the configuration and explains the function of counter control register (ch.0/ch.1) lower (CCRL0/CCRL1).

#### ■ Counter Control Register (ch.0/ch.1) Lower (CCRL0/CCRL1)

The bit configuration of counter control register (ch.0/ch.1) lower (CCRL0/CCRL1) is shown below.

**Figure 13.3-4 Bit Configuration of Counter Control Register (ch.0/ch.1) Lower (CCRL0/CCRL1)**

| CCRL0                             | bit | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value         |
|-----------------------------------|-----|------|------|------|------|------|------|------|------|-----------------------|
| ch.0 Address: 00006C <sub>H</sub> |     | UDMS | CTUT | UCRE | RLDE | UDCC | CGSC | CGE1 | CGE0 | 0X00X000 <sub>B</sub> |
| CCRL1                             |     | R/W  | W    | R/W  | R/W  | W    | R/W  | R/W  | R/W  |                       |
| ch.1 Address: 000070 <sub>H</sub> |     |      |      |      |      |      |      |      |      |                       |

Counter control register (ch.0/ch.1) lower (CCRL0/CCRL1) consists of bits that have the functions explained below.

#### [bit7] UDMS (up/down mode selection)

This bit is used to control the up/down at the falling edge of the BIN pin in the phase difference counter mode at frequency multiplied by 2.

It is initialized to "0" by a reset. Read and write operations are possible.

| UDMS | Operation  |
|------|--|
| 0    | Decrement if the AIN pin value detected at the falling edge of the BIN pin is "H" (initial value)<br>Increment if the AIN pin value detected at the falling edge of the BIN pin is "L" (initial value) |
| 1    | Decrement if the AIN pin value detected at the falling edge of the BIN pin is "L"<br>Increment if the AIN pin value detected at the falling edge of the BIN pin is "H"                                 |

If this bit is rewritten after its start, the count value is not assured.

#### [bit6] CTUT (counter write)

This bit is used to control data transfers from RCR to UDCR.

If this bit is set to "1", data is transferred from RCR to UDCR.

Writing "0" has no effect.

**[bit5] UCRE (UDCR clear enable)**

This bit is used to control UDCR clear caused by compare.

This does not affect the UDCR clear function (such as caused by the ZIN pin setting) other than clear because of compare generation.

| UCRE | Counter clear caused by compare        |
|------|--|
| 0    | Counter clear prohibit (initial value) |
| 1    | Counter clear permit                   |

**[bit4] RLDE (reload enable)**

This bit is used to control the start of the reload function. The RCR value is transferred to UDCR if an underflow occurs when the reload function starts.

| RLDE | Reload function                          |
|------|--|
| 0    | Reload function prohibit (initial value) |
| 1    | Reload function permit                   |

**[bit3] UDCC (UDCR clear)**

This bit is used to clear UDCR. Writing "0" to this bit clears UDCR to "0000<sub>H</sub>".

Writing "1" has no effect.

**[bit2] CGSC (counter clear/gate selection)**

This bit is used to select a function of external pin ZIN.

| CGSC | ZIN function                           |
|------|--|
| 0    | Counter clear function (initial value) |
| 1    | Gate function                          |

**[bit1, 0] CGE1, CGE0 (counter clear/gate edge selection)**

These bits are used to select a detection edge/level for external pin ZIN.

| CGE1 | CGE0 | For selecting the counter clear function | For selecting the gate function         |
|------|------|--|---|
| 0    | 0    | Edge detect prohibited (initial value)   | Level detect prohibited (count disable) |
| 0    | 1    | Falling edge                             | "L" level                               |
| 1    | 0    | Rising edge                              | "H" level                               |
| 1    | 1    | Setting is prohibited                    | Setting is prohibited                   |

If this bit is rewritten after its start, the count value is not assured.

### 13.3.4 Counter status register 0/1 (CSR0/CSR1)

This section describes the configuration and explains the function of counter status register 0/1 (CSR0/CSR1).

#### ■ Counter Status Register 0/1 (CSR0/CSR1)

The bit configuration of the counter status register 0/1 (CSR0/CSR1) is shown below.

**Figure 13.3-5 Bit Configuration of Counter Status Register 0/1(CSR0/CSR1)**

| CSR0                              | bit | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value         |
|-----------------------------------|-----|------|------|------|------|------|------|------|------|-----------------------|
| ch.0 Address: 000072 <sub>H</sub> |     | CSTR | CITE | UDIE | CMPF | OVFF | UDFF | UDF1 | UDF0 | 00000000 <sub>B</sub> |
| CSR 1                             |     |      |      |      |      |      |      |      |      |                       |
| ch.1 Address: 000074 <sub>H</sub> |     | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R    | R    |                       |

Counter status register 0/1 (CSR0/CSR1) consists of bits that have the functions explained below.

#### [bit7] CSTR (count start)

This bit is used to control the UDCR count start/stop operation.

| CSTR | Count start/stop operation           |
|------|--------------------------------------|
| 0    | Count operation stop (initial value) |
| 1    | Count operation start                |

#### [bit6] CITE (compare interrupt output control)

This bit is used to control permit/prohibition of interrupt output to the CPU if CMPF is defined (if a compare occurs).

| CITE | Permit/prohibit of compare interrupt output         |
|------|---|
| 0    | Compare interrupt output prohibited (initial value) |
| 1    | Compare interrupt output permitted                  |

#### [bit5] UDIE (overflow/underflow interrupt output control)

This bit is used to control the permit/prohibition of interrupt output to the CPU if OVFF/UDFF is defined (if overflow/underflow occurs).

| UDIE | Permit/prohibit of overflow/underflow interrupt output         |
|------|--|
| 0    | Overflow/underflow interrupt output prohibited (initial value) |
| 1    | Overflow/underflow interrupt output permitted                  |

**[bit4] CMPF (compare detect flag)**

This bit is a flag indicating that the UDCR and RCR values match each other after a comparison.

The initialization (writing "0") is only permitted.

Read-modify-write type instructions read "1" irrespective of bit values.

| <b>CMPF</b> | <b>Match/no match at compare detection</b>  |
|-------------|---|
| 0           | No match in compare results (initial value) |
| 1           | Match in compare results                    |

**[bit3] OVFF (overflow detect flag)**

This bit is a flag indicating an overflow occurred.

The initialization (writing "0") is only permitted.

Read-modify-write type instructions read "1" irrespective of bit values.

| <b>OVFF</b> | <b>Overflow occurrence</b>           |
|-------------|--------------------------------------|
| 0           | No overflow occurred (initial value) |
| 1           | Overflow occurred                    |

**[bit2] UDFF (underflow detect flag)**

This bit is a flag indicating that an underflow occurs.

The initialization (writing "0") is only permitted.

Read-modify-write type instructions read "1" irrespective of bit values.

| <b>UDFF</b> | <b>Underflow occurrence</b>           |
|-------------|---------------------------------------|
| 0           | No underflow occurred (initial value) |
| 1           | Underflow occurred                    |

**[bit1, bit0] UDF1, UDF0 (up/down flag)**

These bits are used to indicate the last count operation (up/down) performed.

Only reading is permitted but writing is not.

| <b>UDF1</b> | <b>UDF0</b> | <b>Detect edge</b>            |
|-------------|-------------|-------------------------------|
| 0           | 0           | No input (initial value)      |
| 0           | 1           | Decrement                     |
| 1           | 0           | Increment                     |
| 1           | 1           | Simultaneous up/down occurred |

### 13.3.5 Up/down count register (ch.0/ch.1) (UDCR0/UDCR1)

This section describes the configuration and explains the function of up/down count register (ch.0/ch.1) (UDCR0/UDCR1).

#### ■ Up/Down Count Register (ch.0/ch.1) (UDCR0/UDCR1)

The bit configuration of the up/down count register (ch.0/ch.1) (UDCR0/UDCR1) is shown below.

**Figure 13.3-6 Bit Configuration of Up/Down Count Register (ch.0/ch.1) (UDCR0/UDCR1)**

|                                   |     |     |     |     |     |     |     |     |     |                       |
|-----------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
| UDCR 1                            | bit | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | Initial value         |
| ch.1 Address: 000069 <sub>H</sub> |     | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | 00000000 <sub>B</sub> |
|                                   |     | R   | R   | R   | R   | R   | R   | R   | R   |                       |
| UDCR 0                            | bit | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Initial value         |
| ch.0 Address: 000068 <sub>H</sub> |     | D07 | D06 | D05 | D04 | D03 | D02 | D01 | D00 | 00000000 <sub>B</sub> |
|                                   |     | R   | R   | R   | R   | R   | R   | R   | R   |                       |

This register is an 8-bit count register. With an internal prescaler or AIN/BIN pin input, an up/down count operation is performed. It operates as a 16-bit count register in the 16-bit count mode. In this case, the high-order 8-bit setting of the control register is disabled.

Writing to this register directly is not allowed. To write to this register, be sure to write via PCR. The value to be written to this register must first be written to RCR, and then it is transferred from RCR value to this register by setting the CCRL: CTUT bit to "1" (reloading by software).

This register requires word access for reading.



13.3.6 Reload/compare register (ch.0/ch.1) (RCR0/RCR1)

This section describes the configuration and explains the function of reload/compare register (ch.0/ch.1) (RCR0/RCR1).

■ Reload/compare Register (ch.0/ch.1) (RCR0/RCR1)

Reload/compare register (ch.0/ch.1) (RCR0/RCR1) has the bit configuration shown below.

Figure 13.3-7 Bit Configuration of Reload/Compare Register (ch.0/ch.1) (RCR0/RCR1)

|                                   |     |     |     |     |     |     |     |     |     |                       |
|-----------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
| RCR1                              | bit | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | Initial value         |
| ch.1 Address: 00006B <sub>H</sub> |     | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | 00000000 <sub>B</sub> |
|                                   |     | W   | W   | W   | W   | W   | W   | W   | W   |                       |
| RCR0                              | bit | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Initial value         |
| ch.0 Address: 00006A <sub>H</sub> |     | D07 | D06 | D05 | D04 | D03 | D02 | D01 | D00 | 00000000 <sub>B</sub> |
|                                   |     | W   | W   | W   | W   | W   | W   | W   | W   |                       |

Reload/compare registers (ch.0/ch.1) (RCR0/RCR1) are used to specify a reload value and compare value. The reload value and compare value have the same value and, by starting the reload function and compare function, an up/down count is available between the 00<sub>H</sub> and RCR values (16-bit operation mode: 0000<sub>H</sub> to RCR value).

This register allows write-only operations but not read operations. Writing "1" to the CCR0/CCR1: CTUT bit transfers the register value to UDCR (reloading by software).

Write to this register with word access.

## 13.4 Interrupt of 8/16-bit Up/Down Counter/Timer

The interrupt of the 8/16-bit up/down counter/timer occurs when the count direction is changed only once during count start, when an match of comparison result is detected, or when the overflow/underflow occurs.

The DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS) cannot be activated for the interrupt of the 8/16-bit up/down counter/timer.

### ■ Interrupt of 8/16-bit Up/Down Counter/Timer

Table 13.4-1 shows the interrupt control bit and interrupt source of the 8/16-bit up/down counter/timer.

**Table 13.4-1 Interrupt of 8/16-bit Up/Down Counter/Timer**

|  | Count direction<br>detection interrupt             | Overflow/<br>underflow interrupt   | Counter compare<br>match interrupt   |
|--|--|--|--|
| Interrupt request flag                 | CCR0: CDCF (bit14) ch.0<br>CCR1: CDCF (bit14) ch.1 | CSR0: OVFF (bit3) ch.0<br>UDFF (bit2)<br>CSR1: OVFF (bit3) ch.1<br>UDFF (bit2) | CSR0: CMPF (bit4) ch.0<br>CSR1: CMPF (bit4) ch.1                                 |
| Interrupt request output<br>enable bit | CCR0: CFIE (bit13) ch.0<br>CCR1: CFIE (bit13) ch.1 | CSR0: UDIE (bit5) ch.0<br>CSR1: UDIE (bit5) ch.1                               | CSR0: CITE (bit6) ch.0<br>CSR1: CITE (bit6) ch.1                                 |
| Interrupt generation source            | Up/down counter<br>direction detection             | Overflow/underflow detec-<br>tion  | Match between value of<br>up/down counter and that of<br>reload/compare register |

CCR0/OCR0 correspond to up/down counter pins (AIN0/BIN0/ZIN0).

CCR1/OCR1 correspond to up/down counter pins (AIN1/BIN1/ZIN1).

#### ● Count direction change interrupt

The operation for generating the count direction change interrupt is shown below.

- Bit14: CDCF flag of the counter control register (CCR0/1) is set to "1".
- While bit13: CFIE of the interrupt request (CCR0/1) is enabled ("1"). When the count direction is changed only once during count start, the interrupt occurs.

#### ● Overflow/underflow interrupt

The operation for generating the overflow/underflow interrupt is shown below.

- Bit5: UDIE flag of the counter status register (CSR0/1) is set to "1".
- If bit3: OVFF or bit2: UDFF of the counter status register (CSR0/1) is set to "1", the interrupt request occurs.

#### ● Counter compare match interrupt

The operation for generating the compare interrupt is shown below.

- Bit6: CITE flag of the counter status register (CSR0/CSR1) is set to "1".
- When a comparison result between the UDCR value and RCR value using bit4: CMPF of the counter status register (CSR0/1) matches, the interrupt request occurs.

## ■ Interrupt of 8/16-bit Up/Down Counter/Timer, DMA Transfer, and EI<sup>2</sup>OS

Table 13.4-2 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 13.4-2 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source  | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|---|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|   |                          |                      | Number           | Address             | Number                     | Address             |
| 8/16-bit up/down counter/timer*<br>(ch.0, ch.1)<br>Compare/underflow/overflow/reverse up/down | ○                        | ×                    | #25              | FFFF98 <sub>H</sub> | ICR07                      | 0000B7 <sub>H</sub> |

×: Interrupt request flag is not cleared.

○: Interrupt request flag is cleared.

\*: This interrupt source shares the interrupt source and interrupt number of other peripheral function.

For details, see Table 3.3-2.

### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, other interrupt function cannot be used. The interrupt request enable bit of the relevant resource is set to "0" to execute the software polling processing.

## ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The 8/16-bit up/down counter/timer does not correspond to the DMA transfer function, but the EI<sup>2</sup>OS function. When the EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 13.5 8/16-bit Up/Down Counter/Timer Operation

This section explains different count modes in the 8/16-bit up/down counter/timer and the operation of the reload/compare function.

### ■ Selection of Count Mode

The 8/16-bit up/down counter/timer has four types of count modes. These count modes are selected by CCRH: CMS1 or CMS0.

**Table 13.5-1 Selection of Count Mode**

| CMS1 | CMS0 | Count mode   |
|------|------|--|
| 0    | 0    | Timer mode (decremented)                               |
| 0    | 1    | Up/down count mode                                     |
| 1    | 0    | Phase difference count mode: frequency multiplied by 2 |
| 1    | 1    | Phase difference count mode: frequency multiplied by 4 |

#### ○ Timer mode (decremented)

In the timer mode, output of the internal prescaler is decremented. The built-in prescaler enables selection of either 2 machine cycles or 8 machine cycles with CCRH: CLKS.

#### ○ Up/down count mode

In the up/down count mode, counting the input of external pins AIN and BIN enables an up/down count. AIN pin input controls increments, and BIN pin input controls decrements.

Input of the AIN pin and BIN pin indicates an edge detection for input, detection edge can be selected by CCRH: CES1 and CES0.

**Table 13.5-2 Selection of Edge Detection**

| CES1 | CES0 | Detect edge                            |
|------|------|--|
| 0    | 0    | Edge detect prohibit                   |
| 0    | 1    | Falling edge detect                    |
| 1    | 0    | Rising edge detect                     |
| 1    | 1    | Both falling and rising edges detected |

#### ○ Phase difference count mode (at frequency multiplied by 2/frequency multiplied by 4)

In the phase difference count mode, to count the encoder phase difference between output signal phases A and B, the BIN pin input level is checked for counting if the AIN pin input edge is detected, and the AIN pin input level is checked for counting if the BIN pin input edge is detected.

In the modes at frequency multiplied by 2 and frequency multiplied by 4, the phase difference is checked between AIN and BIN pin inputs. If the AIN pin is advanced, it is incremented and, if the BIN pin is advanced, it is decremented.

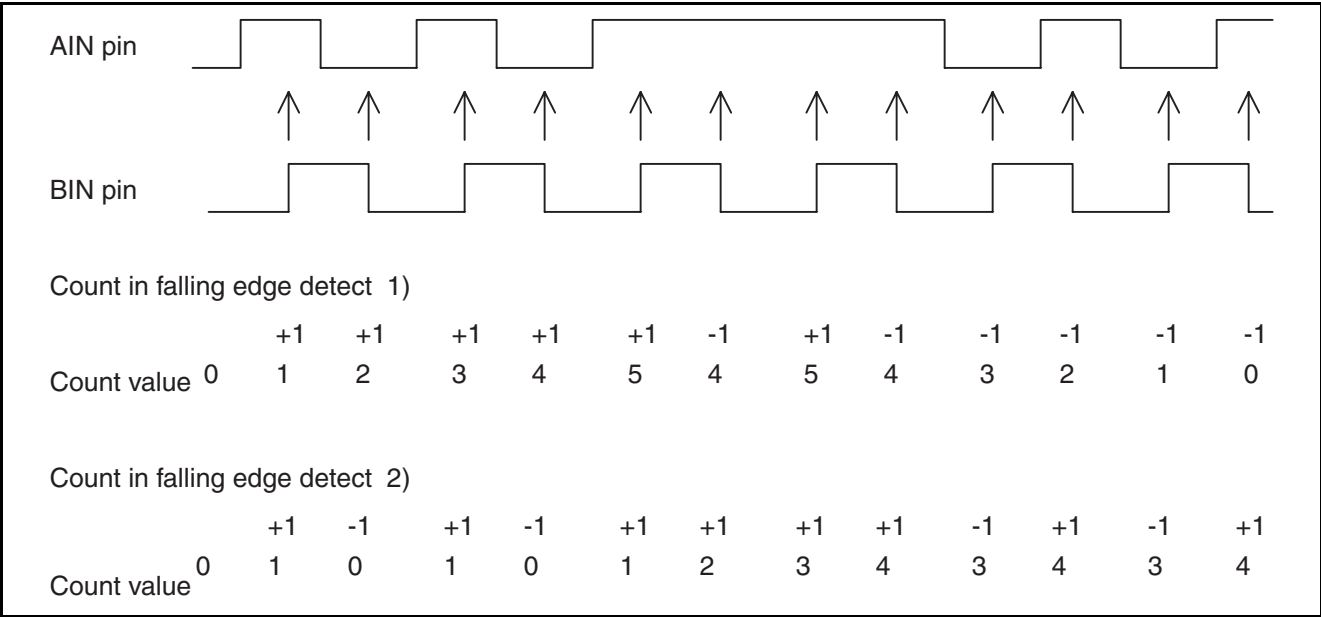
In the mode at frequency multiplied by 2, at the timing of both the rising and falling edges of the BIN pin, counting is done as required by checking for the AIN pin value. Count operations in this case are as follows:

- Incremented if the AIN pin value detected at the rising edge of the BIN pin is "H"
- Decrementated if the AIN pin value detected at the rising edge of the BIN pin is "L"

The AIN pin value detected at the falling edge of the BIN pin is selected from the following two types 1) and 2):

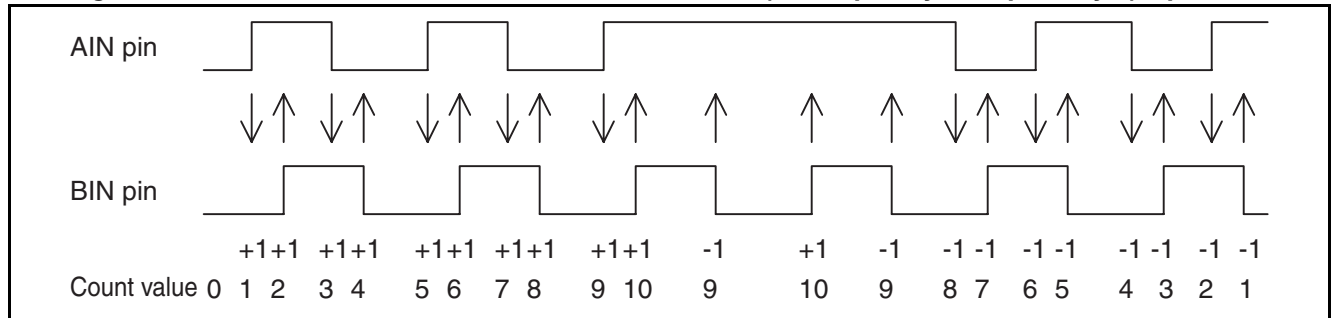
- 1) Decrementated if the AIN pin value detected at the falling edge of BIN pin is "H"  
Incremented if the AIN pin value detected at the falling edge of the BIN pin is "L"
- 2) Decrementated if the AIN pin value detected at the falling edge of the BIN pin is "L"  
Incremented if the AIN pin value detected at the falling edge of the BIN pin is "H"

Figure 13.5-1 Outline of Phase Difference Count Mode (at Frequency Multiplied by 2) Operation



In the mode at frequency multiplied by 4, the AIN pin value is checked for counting at the timing of both the BIN pin rising and falling edges. BIN pin value is checked for counting at the timing of both the AIN pin rising and falling edges. Count operations for such cases are described below.

- Incremented if the AIN pin value detected at the rising edge of the BIN pin is "H"
- Decrementated if the AIN pin value detected at the rising edge of the BIN pin is "L"
- Decrementated if the AIN pin value detected at the falling edge of the BIN pin is "H"
- Incremented if the AIN pin value detected at the falling edge of the BIN pin is "L"
- Decrementated if the BIN pin value detected at the rising edge of the AIN pin is "H"
- Incremented if the BIN pin value detected at the rising edge of the AIN pin is "L"
- Incremented if the BIN pin value detected at the falling edge of the AIN pin is "H"
- Decrementated if the BIN pin value detected at the falling edge of the AIN pin is "L"

**Figure 13.5-2 Outline of Phase Difference Count Mode (at Frequency Multiplied by 4) Operation**

In counting the encoder output, the input condition must be arranged by defining the relationship between the phases and pins shown below. Thus, high-precision detection can enable the rotation angle, rotation count, and rotation direction to be measured.

- Inputting phase A to the AIN pin
- Inputting phase B to the BIN pin
- Inputting phase Z to the ZIN pin

If this count mode is selected, the selection of detection edge via CCRH: CES1/0 and CCRL: CGE1/0 is disabled.

### 13.5.1 Reload/compare function

The 8/16-bit up/down counter/timer has reload and compare functions. These two functions may be mixed for processing.

■ Selection of Reload and Compare Functions

Table 13.5-3 shows an example of selecting reload and compare functions.

Table 13.5-3 Selection Example of Reload/Compare Function

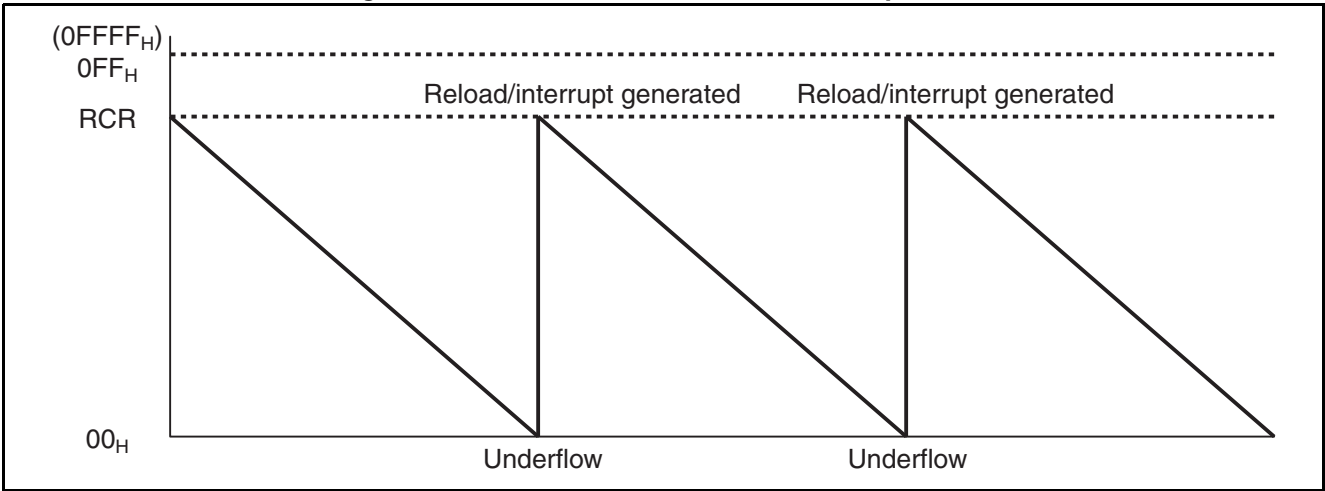
| RLDE, UCRE      | Reload and compare functions            |
|-----------------|---|
| 00 <sub>B</sub> | Reload/compare prohibit (initial value) |
| 01 <sub>B</sub> | Compare permit                          |
| 10 <sub>B</sub> | Reload permit                           |
| 11 <sub>B</sub> | Reload/compare permit                   |

■ Reload Function

At the start of the reload function, the RCR value is transferred to UDCR at the timing of the down-count clock next to the clock in which an underflow occurs. In this example, UDFF is specified, and an interrupt request occurs.

In a mode where there is no down counting (decrement), the start of this function is disabled.

Figure 13.5-3 Outline of Reload Function Operation

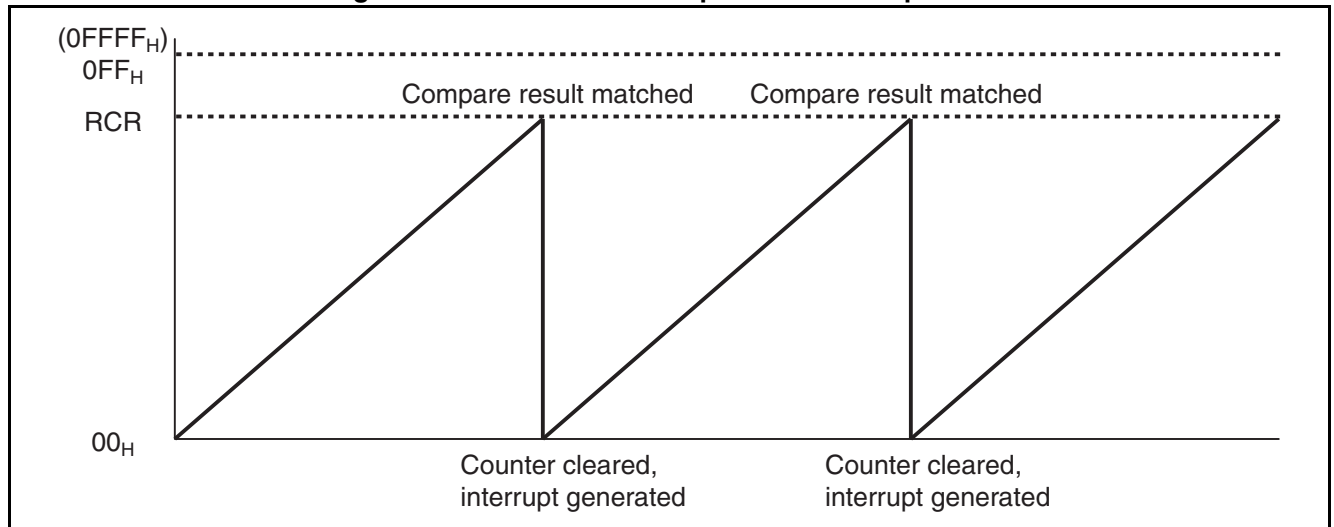


## ■ Compare Function

The compare function is available in any modes other than the timer mode. If RCR and UDCR values match at the start of the compare function, CMPF is specified and an interrupt request occurs. When the compare clear function starts, UDCR is cleared at the next timing of the incremented clock.

In a mode where there is no up counting, the start of this function is disabled.

### Figure 13.5-4 Outline of Compare Function Operation

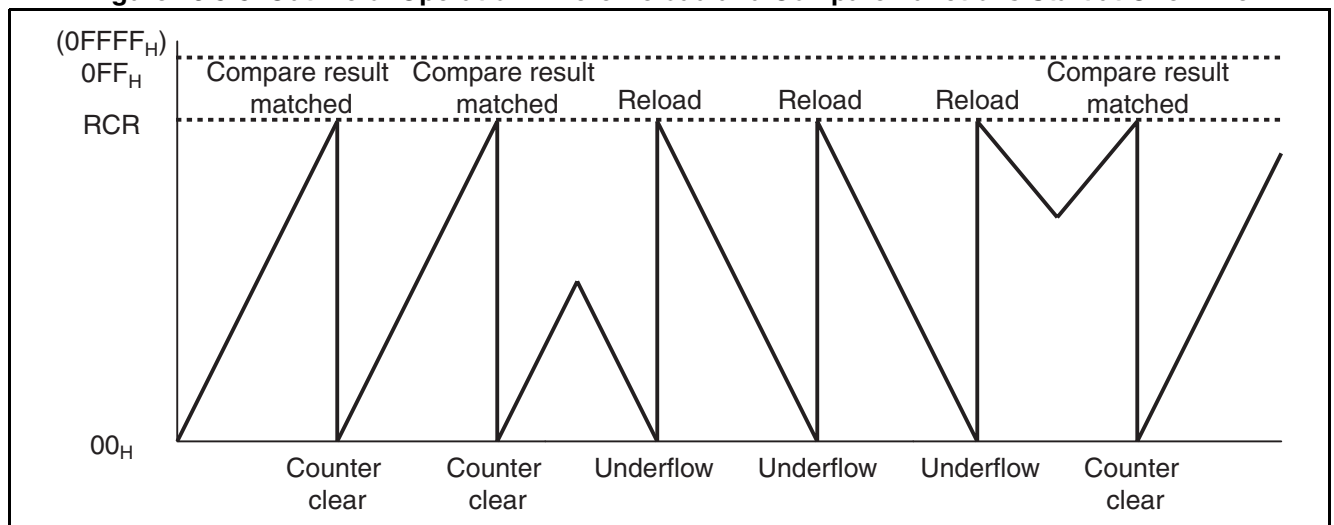


### ■ Up/Down Count at any Width in Reload/Compare Function

When a reload/compare function starts, an up/down count is available at any width.

When an underflow occurs by the reload function, the RCR value is transferred to UDCR. The compare function clears UDCR if RCR and UDCR have matching values. Using both functions, an up/down count is performed in a range of 00<sub>H</sub> to RCR.

**Figure 13.5-5 Outline of Operation Where Reload and Compare Functions Start at One Time**



If compare result finds a match or a reload (underflow) occurs, an interrupt can be generated to CPU. These interrupt outputs are controlled so that they are enabled independently.

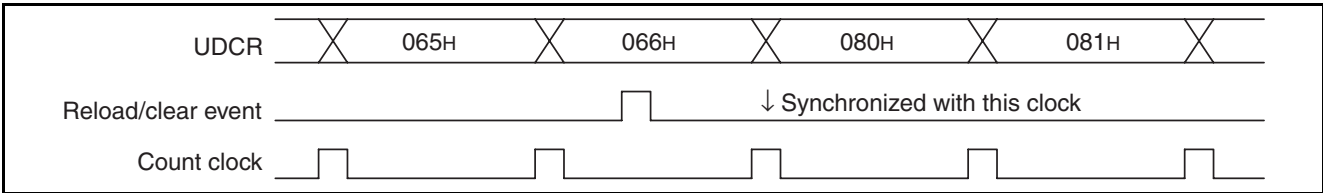
The timing of reload and clear operations for UDCR varies between the count start and stop modes.



○ If reload or clear events are generated in a count operation

All updating operations of UDCR are in sync with the count clock. Figure 13.5-6 shows an example of reloading 080<sub>H</sub>.

Figure 13.5-6 Normal Operation Counting

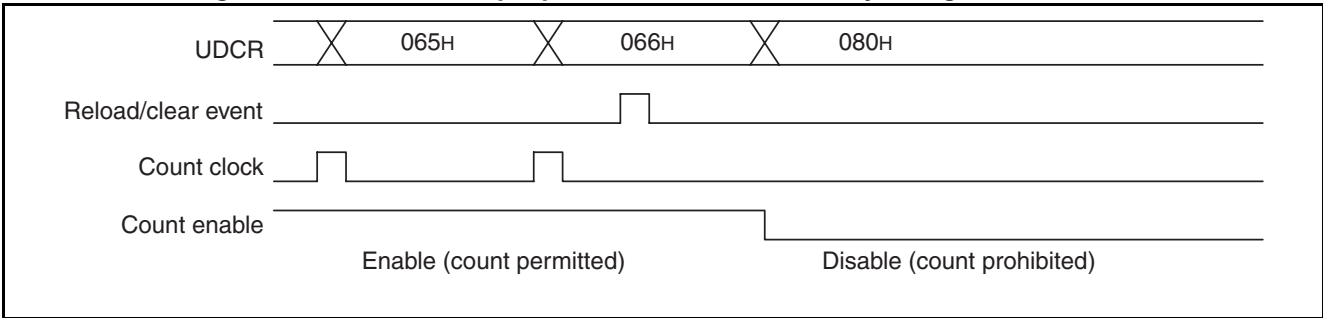


○ If reload and clear events are generated just before the count operation stops

If counting stops in the count clock sync wait mode (state where count input is held for synchronization), reload and clear operations are performed when the stop occurs.

Figure 13.5-7 shows an example of reloading 080<sub>H</sub>.

Figure 13.5-7 Count Stop Operation in Count Clock sync Signal Wait Mode

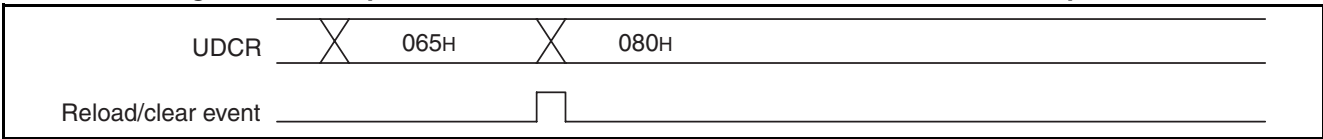


○ If reload and clear events are generated in the count stop mode

Update of UDCR is performed when an event occurs.

Figure 13.5-8 shows an example of reloading "080<sub>H</sub>".

Figure 13.5-8 Operation When Reload/Clear Event Occurs in Count Stop Mode



○ If counter is cleared by the comparison result match

A clear operation caused by compare is performed if the UDCR and RCR values match and incrementing (up count) occurs. Even if the UDCR and RCR values match, no clear operation is performed if a down-count or count stop occurs subsequently.

A clear operation is performed at the above timing for all events other than reset input. Reload is also performed at the above timing in any event.

If clear and reload events occur at the same time, the clear event has priority.

## 13.5.2 Writing data to up/down count register (UDCR)

Writing data directly to UDCR from a data bus is not permitted. This section includes procedures for writing any data to UDCR.

### ■ Writing Data to UDCR

Data can be written to UDCR with the following procedures:

1. Write the data to be written to UDCR to RCR first.
2. Writing "1" to CCRH: CTUT transfers the data from RCR to UDCR.

### ■ Clearing the Counter

In addition to write "0000<sub>H</sub>" to UDCR, the following procedures also clear the counter.

- Clearing with reset input (initialize)
- Clearing with an edge input from the ZIN pin
- Clearing by writing "0" to CCRL: UDCC
- Clearing with the compare function

Such clear operations are performed regardless of the occurrence for count start/stop.

### ■ Count Clear/Gate Function

The ZIN pin is used as either a count clear or gate function by CCRH: CGSC.

If the count clear function starts, the counter is cleared by the edge input from the ZIN pin. CCRL: CGE1/CGE0 select the edge of the ZIN pin input signal where the counter is cleared.

When the gate function starts, the count is enabled or disabled depending on the level input from the ZIN pin. The level of the ZIN pin input signal used to enable the count selects using CCRL: CGE1/CGE0.

This function is available for all count modes.

**Table 13.5-4 Selection of ZIN Pin Function**

| CGSC | ZIN pin function       | CGE1, CGE0      | Counter clear function | Gate function     |
|------|------------------------|-----------------|------------------------|-------------------|
| 0    | Counter clear function | 00 <sub>B</sub> | Detect prohibited      | Detect prohibited |
| 1    | Gate function          | 01 <sub>B</sub> | Rising edge            | "L" level         |
|      |                        | 10 <sub>B</sub> | Falling edge           | "H" level         |

## ■ Count Direction Flag, Count Direction Reversal Flag

The count direction flag (UDF1, UDF0) indicates whether the last count operation was either an up-count or down-count when an up- or down-count was performed. By evaluating the count clock generated by input of both the AIN and BIN pins, a flag is updated at every count operation. If information on the current rotation direction is required for controlling the motor, it is identified by checking this flag.

This function is available in all count modes.

**Table 13.5-5 Count Direction Flag**

| UDF1, UDF0      | Count direction   |
|-----------------|---|
| 01 <sub>B</sub> | Down count (decrement)  |
| 10 <sub>B</sub> | Up count (increment)  |
| 11 <sub>B</sub> | Up/down simultaneously generated (no count operation performed) |

The count direction reversal flag (CDCF) is set when the count direction is switched between counting up and counting down. When this flag is set, an interrupt is generated to the CPU. By referring to this interrupt and the count direction flag (UDF1, UDF0), changes of direction are identified. However, note that the direction indicated by the flag may be restored to the original one and the correct count direction reversal cannot be detected after one reversal of direction if the duration of the direction reversal is short and/or occurs repeatedly.

**Table 13.5-6 Count Direction Reversal Flag**

| CDCF | Count direction reversal detection     |
|------|--|
| 0    | No direction reversal                  |
| 1    | Direction reversal (one or more times) |

## 13.6 Program Example of 8/16-bit Up/Down Counter/Timer

This section describes the program example of the 8/16-bit up/down counter/timer.

### ■ Program Example of 8/16-bit Up/Down Counter/Timer

| <p>Example of setting procedure</p> <p>16-bit mode timer mode (down count) Count clock = 8 division<br/>Reload count value and generate interval interrupt. Interrupt source = underflow</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Control up/down counter ch.0</li> </ul> <table border="1"> <tr> <td>Set control register</td><td>CCR0</td></tr> <tr> <td>Set 16-bit mode enable&gt;&gt;</td><td>.M16E</td></tr> <tr> <td>Reverse count direction&gt;&gt;</td><td>.CDCF</td></tr> <tr> <td>Enable count direction reversal interrupt&gt;&gt;</td><td>.CFIE</td></tr> <tr> <td>Select built-in prescaler&gt;&gt;</td><td>.CLKS</td></tr> <tr> <td>Select count mode&gt;&gt;</td><td>.CMS1-0</td></tr> <tr> <td>Select count clock edge&gt;&gt;</td><td>.CES1-0</td></tr> <tr> <td>Select up/down mode&gt;&gt;</td><td>.UDMS</td></tr> <tr> <td>Write counter&gt;&gt;</td><td>.CTUT</td></tr> <tr> <td>Enable UDCR clear&gt;&gt;</td><td>.UCRE</td></tr> <tr> <td>Enable reload function&gt;&gt;</td><td>.RLDE</td></tr> <tr> <td>UDCR clear&gt;&gt;</td><td>.UDCC</td></tr> <tr> <td>Counter clear/gate selection&gt;&gt;</td><td>.CGSC</td></tr> <tr> <td>Counter clear/gate edge selection&gt;&gt;</td><td>.CGE1-0</td></tr> </table> <ul style="list-style-type: none"> <li>Set reload value/compare value</li> </ul> <table border="1"> <tr> <td>Set reload value</td><td>RCR0</td></tr> <tr> <td></td><td>RCR1</td></tr> </table> <ul style="list-style-type: none"> <li>Interrupt related</li> </ul> <table border="1"> <tr> <td>Set UD counter 0 interrupt level</td><td>ICR07</td></tr> <tr> <td>Set I flag</td><td>(CCR)</td></tr> </table> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start up/down counter ch.0</li> </ul> <table border="1"> <tr> <th colspan="2">Register name, bit name</th></tr> <tr> <td>Control underflow interrupt</td><td>CSR0 .UDIE</td></tr> <tr> <td>Transfer data from RCR to UDCR</td><td>CCR0 .CTUT</td></tr> <tr> <td>Start count operation</td><td>CSR0 .CSTR</td></tr> </table> <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Interrupt processing</li> </ul> <table border="1"> <tr> <td colspan="2">Check underflow detection flag</td></tr> <tr> <td colspan="2"></td></tr> <tr> <td>Clear interrupt request flag</td><td>CSR0 .UDFF</td></tr> <tr> <td colspan="2">(Arbitrary processing)</td></tr> </table> <p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>Note:<br/>Setting related to clock and setting of _set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</p> | Set control register | CCR0 | Set 16-bit mode enable>> | .M16E | Reverse count direction>> | .CDCF | Enable count direction reversal interrupt>> | .CFIE | Select built-in prescaler>> | .CLKS | Select count mode>> | .CMS1-0 | Select count clock edge>> | .CES1-0 | Select up/down mode>> | .UDMS | Write counter>> | .CTUT | Enable UDCR clear>> | .UCRE | Enable reload function>> | .RLDE | UDCR clear>> | .UDCC | Counter clear/gate selection>> | .CGSC | Counter clear/gate edge selection>> | .CGE1-0 | Set reload value | RCR0 |  | RCR1 | Set UD counter 0 interrupt level | ICR07 | Set I flag | (CCR) | Register name, bit name |  | Control underflow interrupt | CSR0 .UDIE | Transfer data from RCR to UDCR | CCR0 .CTUT | Start count operation | CSR0 .CSTR | Check underflow detection flag |  |  |  | Clear interrupt request flag | CSR0 .UDFF | (Arbitrary processing) |  | <p>Program example</p> <pre> void UD0_sample_1(void) {     UD0_initial();     UD0_start(); }  void UD0_initial(void) {     IO_CCR0.word = 0x9018; /* Setting value=1001_0000_0001_1000 */     /* bit15 = 1    M16E 16bit×1ch operation mode */     /* bit14 = 0    Clear CDCF count direction reversal flag */     /* bit13 = 0    Disable CFIE direction reversal interrupt */     /* bit12 = 1    CLKS 8 machine cycles */     /* bit11-10 = 00 CMS1, 0 timer mode */     /* bit9-8 = 00  Disable CES1, 0 edge detection */     /* bit7 = 0     Select UMDS up/down mode */     /* bit6 = 0     Write CTUT counter (invalid) */     /* bit5 = 0     Disable UCRE counter clear */     /* bit4 = 1     Enable RLDE reload function */     /* bit3 = 1     Clear UDCC UDCR (invalid) */     /* bit2 = 0     CGSC ZIN counter clear function */     /* bit1-0 = 00  Disable CGE1, 0 ZIN edge detection */      IO_RCR0 = 0xff; /* Set 16-bit mode reload value (arbitrary value) */     IO_RCR1 = 0xff;      IO_ICR07.byte = 0x10; /* Set interrupt level (arbitrary value) */     __EI(); /* Enable interrupt */ }  void UD0_start(void) {     IO_CSR0.bit.UDIE = 1; /* bit5 = 1  Enable UDIE underflow interrupt */     IO_CCR0.bit.CTUT = 1; /* bit6 = 1  Write CTUT counter */     IO_CSR0.bit.CSTR = 1; /* bit7 = 1  Activate CSTR count operation */ }  __interrupt void UD0_int(void) {     if(IO_CSR0.bit.UDFF)     {         IO_CSR0.bit.UDFF = 0; /* bit2 = 0  Clear UDFF underflow detection flag */         *****     } }  #pragma intvect UD0_int 25 </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
|---|----------------------|------|--------------------------|-------|---------------------------|-------|---|-------|-----------------------------|-------|---------------------|---------|---------------------------|---------|-----------------------|-------|-----------------|-------|---------------------|-------|--------------------------|-------|--------------|-------|--------------------------------|-------|-------------------------------------|---------|------------------|------|--|------|----------------------------------|-------|------------|-------|-------------------------|--|-----------------------------|------------|--------------------------------|------------|-----------------------|------------|--------------------------------|--|--|--|------------------------------|------------|------------------------|--|--|
| Set control register  | CCR0                 |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Set 16-bit mode enable>>  | .M16E                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Reverse count direction>>   | .CDCF                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Enable count direction reversal interrupt>>   | .CFIE                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Select built-in prescaler>>   | .CLKS                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Select count mode>>   | .CMS1-0              |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Select count clock edge>>   | .CES1-0              |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Select up/down mode>>   | .UDMS                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Write counter>>   | .CTUT                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Enable UDCR clear>>   | .UCRE                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Enable reload function>>  | .RLDE                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| UDCR clear>>  | .UDCC                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Counter clear/gate selection>>  | .CGSC                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Counter clear/gate edge selection>>   | .CGE1-0              |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Set reload value  | RCR0                 |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
|   | RCR1                 |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Set UD counter 0 interrupt level  | ICR07                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Set I flag  | (CCR)                |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Register name, bit name   |                      |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Control underflow interrupt   | CSR0 .UDIE           |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Transfer data from RCR to UDCR  | CCR0 .CTUT           |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Start count operation   | CSR0 .CSTR           |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Check underflow detection flag  |                      |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
|   |                      |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| Clear interrupt request flag  | CSR0 .UDFF           |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |
| (Arbitrary processing)  |                      |      |                          |       |                           |       |   |       |                             |       |                     |         |                           |         |                       |       |                 |       |                     |       |                          |       |              |       |                                |       |                                     |         |                  |      |  |      |                                  |       |            |       |                         |  |                             |            |                                |            |                       |            |                                |  |  |  |                              |            |                        |  |  |

## ■ Setting Method Other than Program Example

### ● Method to select 8-bit or 16-bit operation

Set by the 16-bit mode enable setting bit (CCR0.M16E).

| Bit length of up/down counter | 16-bit mode enable setting bit (M16E) |
|-------------------------------|---------------------------------------|
| To set to 8-bit               | Set to "0"                            |
| To set to 16-bit              | Set to "1"                            |

### ● Type of count mode and setting method

There are four count modes.

The count mode is set by the count mode selection bits (CCR0.CMS[1:0], CCR1.CMS[1:0]).

| Count mode   | Count mode selection bit (CMS[1:0]) |
|--|-------------------------------------|
| To set to timer mode                                     | Set to "00 <sub>B</sub> "           |
| To set to up/down count mode                             | Set to "01 <sub>B</sub> "           |
| To set to phase difference count mode (2 multiplication) | Set to "10 <sub>B</sub> "           |
| To set to phase difference count mode (4 multiplication) | Set to "11 <sub>B</sub> "           |

### ● Method to select count source at timer mode operation

Set by built-in prescaler selection bit (CCR0.CLKS, CCR1.CLKS).

| Count source at timer mode operation                  | Built-in prescaler selection bit (CLKS) |
|---|---|
| To set clock that is frequency of $\phi$ divided by 2 | Set to "0"                              |
| To set clock that is frequency of $\phi$ divided by 8 | Set to "1"                              |

### ● Method to select edge when input signals (AIN, BIN) are detected at up/down counter operation

Set by the count clock edge selection bits (CCR0.CES[1:0], CCR1.CES[1:0]).

| Count detection edge         | Count clock edge selection bit (CES[1:0]) |
|------------------------------|---|
| To set detection prohibition | Set to "00 <sub>B</sub> "                 |
| To set falling detection     | Set to "01 <sub>B</sub> "                 |
| To set falling detection     | Set to "10 <sub>B</sub> "                 |
| To set both-edge detection   | Set to "11 <sub>B</sub> "                 |

● Method to set the value to up/down counter

The value can be set to the up/down counter when "1" is written to the counter write bit (CCR0.CTUT, CCR1.CTUT) after the value is written to the reload/compare register (RCR).

● Method to enable up/down counter clear when up count value of up/down counter and compare value (RCR[0:1]) match and up count is performed

Set by the up/down counter clear enable bit (CCR0.UCRE, CCR1.UCRE).

| Operation                        | Up/down counter clear enable bit (UCRE) |
|----------------------------------|---|
| To disable up/down counter clear | Set to "0"                              |
| To enable up/down counter clear  | Set to "1"                              |

● Method to enable the reload value (RCR[1:0]) to reload to the up/down counter when underflow of the up/down counter is generated

Set by the reload enable bit (CCR0.RLDE, CCR1.RLDE).

| Operation  | Reload enable bit (RLDE) |
|--|--------------------------|
| To disable the reload value (RCR) reloading from up/down counter | Set to "0"               |
| To enable the reload value (RCR) to reload to up/down counter    | Set to "1"               |

● Method to clear up/down counter

The up/down counter can be cleared using the following five methods:

- Write "0" to the up/down counter clear bit (CCR0.UDCC, CCR1.UDCC)
- Edge input to ZIN pin
- Match between compare value and up count value of up/down counter
- Count up operation from full count
- Reset input (external reset, watchdog reset, software reset)

## ● Method to clear up/down counter by ZIN pin

Set by the counter clear gate bit (CCR0.CGSC, CCR1.CGSC) and counter clear gate edge selection bits (CCR0.CGE[1:0], CCR1.CGE[1:0]). (valid for all count modes).

| ZIN pin input                            | Counter clear gate bit (CGSC) | Counter clear gate edge selection bit (CGE[1:0]) |
|--|-------------------------------|--|
| To disable edge detection (no clear)     | Set to "0"                    | Set to "00 <sub>B</sub> "                        |
| To clear up/down counter at falling edge | Set to "0"                    | Set to "01 <sub>B</sub> "                        |
| To clear up/down counter at rising edge  | Set to "0"                    | Set to "10 <sub>B</sub> "                        |

Setting GCE[1:0]=11 is prohibited

## ● Method to control up/down count operation by ZIN pin

Set by the counter clear gate bit (CCR0.CGSC, CCR1.CGSC) and counter clear gate edge selection bits (CCR0.CGE[1:0], CCR1.CGE[1:0]). (valid for all count modes).

| ZIN pin input   | Counter clear gate bit (CGSC) | Counter clear gate edge selection bit (CGE[1:0]) |
|---|-------------------------------|--|
| To disable level detection (count disabled state)                         | Set to "1"                    | Set to "00 <sub>B</sub> "                        |
| To operate up/down count at "L" level and stop up/down count at "H" level | Set to "1"                    | Set to "01 <sub>B</sub> "                        |
| To stop up/down count at "L" level and operate up/down count at "H" level | Set to "1"                    | Set to "10 <sub>B</sub> "                        |

Setting GCE[1:0]=11 is prohibited

## ● Method to enable/disable count operation of up/down counter

Set by the count start bit (CSR0.CSTR, CSR1.CSTR) .

| Operation  | Count start bit (CSR0.CSTR, CSR1.CSTR) |
|--|--|
| To disable count operation of up/down counter            | Set to "0"                             |
| To enable count operation of up/down counter (for start) | Set to "1"                             |

Starting counter is depending on the count mode.

Timer mode → Start count by internal clock

Up/down count mode → Start count upon detection edge of AIN pin, BIN pin

Phase difference count mode → Start count upon detection of phase difference of AIN pin, BIN pin

However, it is necessary to detect the count operation enable level when the gate function of the ZIN pin is selected.

- Method to know the previous count direction (method to know the current rotation direction)

Set by the up/down flag (CSR0.UDF[1:0], CSR1.UDF[1:0]) .

| Setting            | Up/down flag (UDF[1:0])                                      |
|--------------------|--|
| "00 <sub>B</sub> " | No count after a reset                                       |
| "01 <sub>B</sub> " | Down count   |
| "10 <sub>B</sub> " | Up count   |
| "11 <sub>B</sub> " | Up and down occur simultaneously (neither up nor down count) |

Because this flag has no connection the interrupt, use the count direction reversal flag (CCR0.CDCF, CCR1.CDCF) when processing the interrupt.

- Method to know direction reversal

Set by the count direction reversal flag (CCR0.CDCF, CCR1.CDCF) .

| Setting | Count direction reversal flag (CDCF)              |
|---------|---|
| "0"     | No direction reversal after flag clear            |
| "1"     | Direction reversal after flag clear ("1" or more) |

- Method to know a match of comparison result

Set by the compare detection flag (CSR0.CMPF, CSR1.CMPF).

| Setting | Compare detection flag (CMPF)                                     |
|---------|---|
| "0"     | No match between count value of up/down counter and compare value |
| "1"     | Match between count value of up/down counter and compare value    |

- This flag is set to "1" when the comparison result of operation is matched regardless of the up/down operation, setting value, and reload value.

- Method to know generation of overflow/underflow

Set by the overflow detect flag (CSR0.OVFF, CSR1.OVFF) and underflow detect flag (CSR0.UOFF, CSR1. UOFF).

|            |                                     |
|------------|-------------------------------------|
| (OVFF = 1) | Up/down counter generates overflow  |
| (UOFF = 1) | Up/down counter generates underflow |



- Method to set reload value and compare value

Set the value to the reload/compare registers (RCR0, RCR1) (same value is set for compare value and reload value).

- Interrupt related register

The relationship between the up/down counter number, interrupt level, and interrupt vector is shown in the following table.

For details of the interrupt level and interrupt vector, see CHAPTER 3 INTERRUPT.

| Source          | Interrupt vector                     | Interrupt level setting register                                  |
|-----------------|--------------------------------------|---|
| Up/down counter | #25<br>Address : FFFF98 <sub>H</sub> | Interrupt level register (ICR07)<br>Address : 0000B7 <sub>H</sub> |

## Interrupt request flag

Count direction reversal: (CCR0.CDCF), (CCR1.CDCF)

Compare detection: (CSR0.CMPF), (CSR1.CMPF)

Overflow: (CSR0.OVFF), (CSR1.OVFF)

Underflow: (CSR0.UDFF), (CSR1.UDFF)

The above interrupt request flag does not clear automatically. Write "0" to the interrupt request flag with software before returning from the interrupt processing.

- Type of interrupt and selection method

The following three interrupt sources are provided:

count direction reversal, match of comparison result, overflow/underflow

The interrupt occurs with OR of the above three interrupt sources. The interrupt source is selected by the interrupt request enable bit.

- Method to enable (select)/disable/clear interrupt

Enabling (selecting)/disabling interrupt sets using the interrupt request enable bits below:

Count direction reversal interrupt request enable bit: (CCR0.CFIE), (CCR1.CFIE)

Compare interrupt request enable bit : (CSR0.CITE), (CSR1.CITE)

Overflow/underflow interrupt request enable bit : (CSR0.UDIE), (CSR1.UDIE)

| Control                      | Interrupt request enable bit (CFIE, CITE, UDIE) |
|------------------------------|---|
| To disable interrupt request | Set to "0"                                      |
| To enable interrupt request  | Set to "1"                                      |

Clearing interrupt request sets using the interrupt request bits below:

Count direction reversal: (CCR0.CDCF), (CCR1.CDCF)

Compare detection : (CSR0.CMPF), (CSR1.CMPF)

Overflow : (CSR0.OVFF), (CSR1.OVFF)

Underflow : (CSR0.UDFF), (CSR1.UDFF)

| Control                    | Interrupt request bits (CDCF, CMPF, OVFF, UDFF) |
|----------------------------|---|
| To clear interrupt request | Write "0"                                       |

# CHAPTER 14 16-BIT RELOAD TIMER

---

**This chapter provides an overview of the 16-bit reload timer, explains the configuration and functions of its registers, interrupt and its operation.**

---

- 14.1 Overview of 16-Bit Reload Timer
- 14.2 Configuration and Functions of 16-Bit Reload Timer Registers
- 14.3 Interrupt of 16-Bit Reload Timer
- 14.4 Operations of the 16-Bit Reload Timer
- 14.5 Program Example of 16-Bit Reload Timer

## 14.1 Overview of 16-Bit Reload Timer

The 16-bit reload timer has the following functions:

- Clock mode can be selected from the internal clock mode and event count mode.
- When an underflow of 16-bit timer register (TMR) occurs, the count operation can be selected either the one-shot mode that stops the operation or the reload mode that continues its operation by reloading count setting value.
- When an underflow of 16-bit timer register (TMR) generated, the timer can be used as the interval timer by generating an interrupt.

### ■ Operation Modes of the 16-bit Reload Timer

Operation modes of the 16-bit reload timer are shown below.

| Clock mode                                | Counting      | 16-bit reload timer operation                               |
|---|---------------|---|
| Internal clock mode                       | Reload mode   | Software trigger operation                                  |
|   | One-shot mode | External trigger operation<br>External gate input operation |
| Event count mode<br>(External clock mode) | Reload mode   | Software trigger operation                                  |
|   | One-shot mode |   |

### ■ Internal Clock Mode

One type of count clock can be selected among three types of internal clocks.

#### ○ Software trigger operation

Sets the TRG bits of the timer control status register (TMCSR) to "1" to start a counting operation. Trigger input by the TRG bit can also be enabled when external trigger operation and external gate input operation are performed.

#### ○ External trigger operation

Starts counting operation when the selected edge (rising, falling edges, or both) is input to the TIN0 pin.

#### ○ External gate input operation

Continues counting operation while the selected signal ("L" or "H") is input to the TIN0 pin.

### ■ Event Count Mode (External Clock Mode)

The function that starts countdown when the selected edge (rising, falling, or both) is input to the TIN0 pin. This can also be used as an interval timer if an external clock with constant interval time is used.

## ■ Counter Operation Modes

### ○ Reload mode

If countdown causes an underflow ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the specified count value is reloaded to continue with counting. An underflow can generate an interrupt request that can then be used as an interval timer. A toggle waveform that reverses itself at every underflow is used to output from the TOT0 pin.

| Count clock    | Count clock interval        | Interval time             |
|----------------|-----------------------------|---------------------------|
| Internal clock | $\phi / 2^1$ (80 ns)        | 80 ns to 5.243 ms         |
|                | $\phi / 2^3$ (0.32 $\mu$ s) | 0.32 $\mu$ s to 20.972 ms |
|                | $\phi / 2^5$ (1.28 $\mu$ s) | 1.28 $\mu$ s to 83.886 ms |
| External clock | $\phi / 2^3$ (0.32 $\mu$ s) | 0.32 $\mu$ s or more      |

Notes:

- Machine cycle ( $\phi$ ) = 25 MHz      Example)  $25 \text{ MHz} / 2^1 = 12.5 \text{ MHz} = 80 \text{ ns}$
- Maximum value of interval time is "0000<sub>H</sub>" to "FFFF<sub>H</sub>".  
Example)  $\phi / 2^1 (80 \text{ ns}) \times 65536 \approx 5.243 \text{ ms}$

### ○ One-shot mode

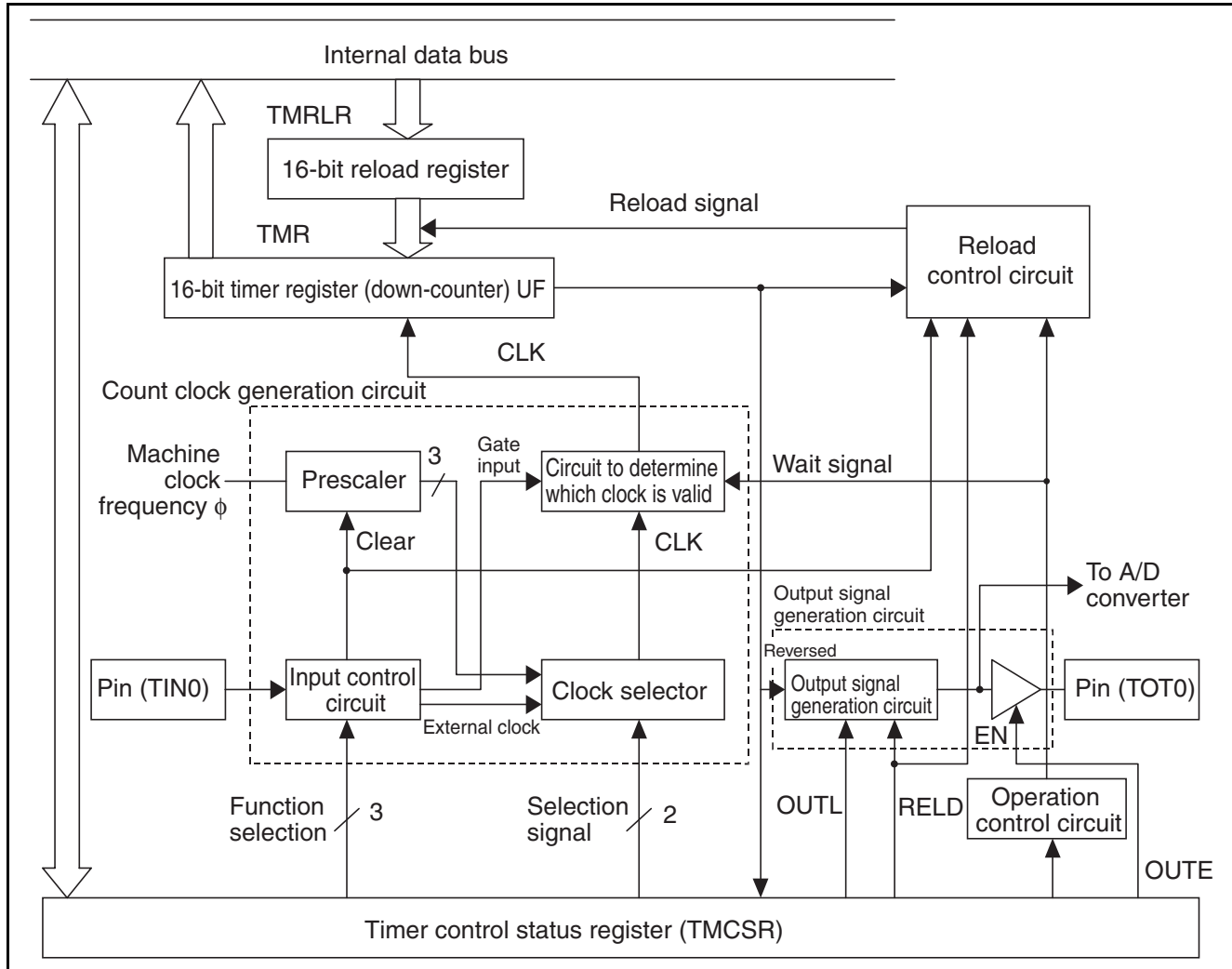
If countdown causes an underflow ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), counting stops. An underflow causes an interrupt. While counting is in progress, a square wave that indicates that counting is in progress is output from the TOT0 pin.

#### Reference:

The 16-bit reload timer can be used for A/D converter start trigger.

## ■ Block Diagram of the 16-bit Reload Timer

Figure 14.1-1 Block Diagram of the 16-bit Reload Timer



## ■ Pin Related to 16-bit Reload Timer

The pin related to the 16-bit reload timer has the TIN0 and TOT0 pins. The TIN0 pin functions as the general-purpose I/O port (P73/TIN0) and the input pin of the 16-bit reload timer. The TOT0 pin functions as the general-purpose I/O port (P74/TOT0) and the output pin of the 16-bit reload timer.

### ● Setting when using as TIN0 pin

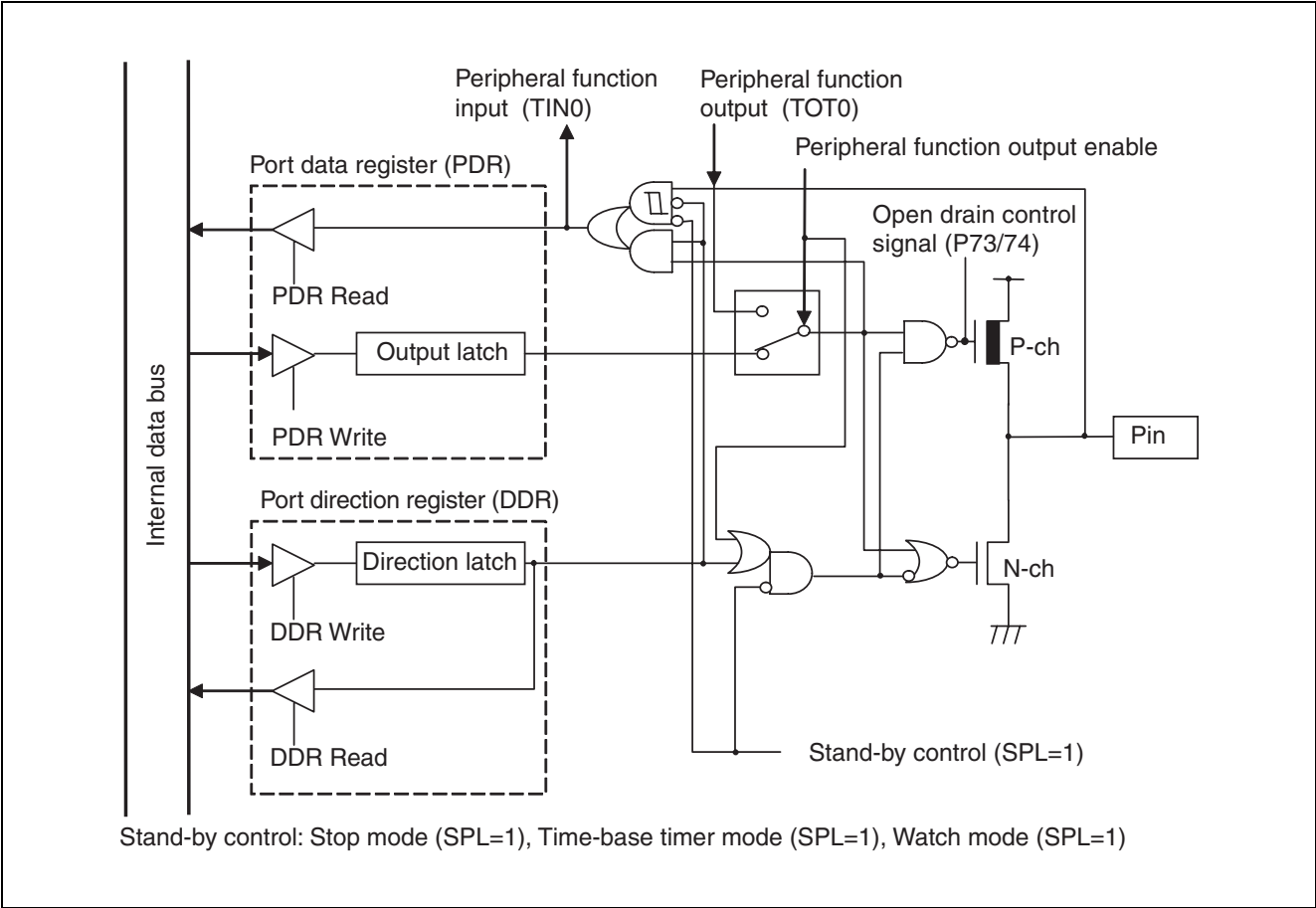
When the TIN0 pin is used as input, P73/TIN0 pin should be set to the input port by the port direction register (DDR7 bit12→ "0").

### ● Setting when using as TOT0 pin

When the using TOT0 pin is used as output, be sure to set the timer control status register (TMCSR) to output enable (OUTE bit6→ "1").

■ Block Diagram of Pin Related to 16-bit Reload Timer

Figure 14.1-2 Block Diagram of Pin Related to 16-bit Reload Timer



## 14.2 Configuration and Functions of 16-Bit Reload Timer Registers

This section describes the configuration and functions of the registers used in the 16-bit reload timer.

### ■ List of Registers

Figure 14.2-1 shows the list of the registers of the 16-bit reload timer.

**Figure 14.2-1 16-bit Reload Timer Registers**

|                     |       |       |       |       |       |       |       |       |   |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|---|
| 0000CB <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TMCSR   |
|                     | -     | -     | -     | -     | CSL1  | CSL0  | MOD2  | MOD1  | Timer control status register (upper bits)                    |
|                     | (-)   | (-)   | (-)   | (-)   | (R/W) | (R/W) | (R/W) | (R/W) | Read/write  |
|                     | (-)   | (-)   | (-)   | (-)   | (0)   | (0)   | (0)   | (0)   | Initial value   |
| 0000CA <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TMCSR   |
|                     | MOD0  | OUTE  | OUTL  | RELD  | INTE  | UF    | CNTE  | TRG   | Timer control status register(lower bits)                     |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write  |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | Initial value   |
| 0000CD <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TMR/TMRLR   |
|                     | D15   | D14   | D13   | D12   | D11   | D10   | D09   | D08   | 16-bit timer register/<br>16-bit reload register (upper bits) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write  |
|                     | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value   |
| 0000CC <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TMR/TMRLR   |
|                     | D07   | D06   | D05   | D04   | D03   | D02   | D01   | D00   | 16-bit timer register/<br>16-bit reload register (lower bits) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write  |
|                     | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value   |

## 14.2.1 Timer Control Status Register (TMCSR)

This section describes the configuration and functions of the timer control status register (TMCSR).

### ■ Timer Control Status Register (TMCSR)

The timer control status register (TMCSR) is used to control the operation mode and interrupts of the 16-bit reload timer. If CNTE = 0, bits other than UF/CNTE/TRG are modified.

The figure below shows the bit configuration of the timer control status register (TMCSR).

|                     |       |       |       |       |       |       |       |       |  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|--|
| 0000CB <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | TMCSR                                      |
|                     | -     | -     | -     | -     | CSL1  | CSL0  | MOD2  | MOD1  | Timer control status register (upper bits) |
|                     | (-)   | (-)   | (-)   | (-)   | (R/W) | (R/W) | (R/W) | (R/W) | Read/write                                 |
|                     | (-)   | (-)   | (-)   | (-)   | (0)   | (0)   | (0)   | (0)   | Initial value                              |
| 0000CA <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | TMCSR                                      |
|                     | MOD0  | OUTE  | OUTL  | RELD  | INTE  | UF    | CNTE  | TRG   | Timer control status register (lower bits) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write                                 |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | Initial value                              |

The functions of the bits in the timer control status register (TMCSR) are described below.

#### [bit11, bit10] CSL1, CSL0 (Clock selection)

These bits are used to select the clock source using the count clock selection bit.

| CSL1 | CSL0 | Clock source (machine clock $\phi = 16 \text{ MHz}$ ) |
|------|------|---|
| 0    | 0    | $\phi/2^1$ (0.125 $\mu\text{s}$ ) (initial value)     |
| 0    | 1    | $\phi/2^3$ (0.5 $\mu\text{s}$ )                       |
| 1    | 0    | $\phi/2^5$ (2.0 $\mu\text{s}$ )                       |
| 1    | 1    | Event count mode                                      |

#### [bit9, bit8, bit7] MOD2, MOD1, MOD0

These bits set the operation mode and the input/output pin functions. With MOD2 = 0, the input pin operates as a trigger. If an active edge is input to the input pin and count operation is in progress, the data from the reload register is loaded into the counter. With MOD2 = 1, the timer operates in gate counter mode and the input pin operates as a gate input. In this mode, the counter only counts when the active level is applied to the input pin.

By combination of the MOD2 to MOD0 bits, the internal clock mode and event counter mode can be selected from the modes listed in Table 14.2-1 and Table 14.2-2.



**Table 14.2-1 Internal Clock Mode (CLS1/0 = 00<sub>B</sub>, 01<sub>B</sub>, or 10<sub>B</sub>)**

| MOD2 | MOD1 | MOD0 | Input pin function | Active edge or level | Initial value |
|------|------|------|--------------------|----------------------|---------------|
| 0    | 0    | 0    | Trigger invalid    | -                    |               |
| 0    | 0    | 1    | Trigger input      | Rising edge          |               |
| 0    | 1    | 0    |                    | Falling edge         |               |
| 0    | 1    | 1    |                    | Both edges           |               |
| 1    | X    | 0    | Gate input         | "L" level            |               |
| 1    | X    | 1    |                    | "H" level            |               |

**Table 14.2-2 Event Count Mode (CLS1, 0 = 11<sub>B</sub>)**

| MOD2 | MOD1 | MOD0 | Input pin function | Active edge or level | Initial value |
|------|------|------|--------------------|----------------------|---------------|
| X    | 0    | 0    | Trigger invalid    | -                    |               |
| X    | 0    | 1    | Trigger input      | Rising edge          |               |
| X    | 1    | 0    |                    | Falling edge         |               |
| X    | 1    | 1    |                    | Both edges           |               |

**[bit6] OUTE (Output enable)**

This bit is used to control output enable.

The TOT pin operates as a general-purpose port when the OUTE bit is set to "0", or as a timer output pin when the OUTE bit is set to "1". In reload mode, the output waveform becomes a toggle waveform. In one-shot mode, the TOT pin outputs a square wave that indicates the counting is in progress.

| OUTE | Function                             |
|------|--------------------------------------|
| 0    | General-purpose port (initial value) |
| 1    | Timer output                         |

**[bit5] OUTL (Output level)**

This bit is used to specify the output level of the TOT pin. Depending on whether OUTL is set to "0" or "1", the output pin level becomes reversed.

| OUTL | One-shot mode (RELD = 0)                  | Reload mode (RELD = 1) | Initial value |
|------|---|------------------------|---------------|
| 0    | Square wave of "H" level in counting mode | 0                      |               |
| 1    | Square wave of "L" level in counting mode | 1                      |               |
| X    | 1   | 0                      |               |
| X    | 1   | 1                      |               |

**[bit4] RELD (Reload operation enable)**

This bit enables reload operation. With RELD set to "1", the timer operates in reload mode. In this mode, the timer loads the reload register data into the counter and continues counting even if an underflow occurs. With RELD set to "0", the timer operates in one-shot mode. If an underflow occurs in this mode, counter operation stops.

| RELD | Function                      |
|------|-------------------------------|
| 0    | One-shot mode (initial value) |
| 1    | Reload mode                   |

**[bit3] INTE (Timer interrupt request enable)**

This bit is used to enable timer interrupt requests. With INTE = 0, no interrupt request is generated even if UF is set to "1".

| INTE | Function  |
|------|---|
| 0    | Interrupt request output prohibited (initial value) |
| 1    | Interrupt request output allowed                    |

**Note:**

Please write "0" in the INTE bit after prohibiting interrupt by setting the IL2 bit to IL0 bit of the interrupt control register (ICR12) to "111" if the reload timer underflow interrupt setting is changed from enable (INTE bit of TMCSR registers =1) to prohibit (INTE bit of TMCSR registers =0).

**[bit2] UF (Timer interrupt request flag)**

This bit is used as a timer interrupt request flag. If an underflow occurs, UF is set to "1". It is cleared by writing "0" or by  $\mu$ DMAC. Writing "1" has no effect. Read-modify-write instruction always reads "1".

| UF | At reading                           | At writing                          |
|----|--------------------------------------|-------------------------------------|
| 0  | No counter underflow (initial value) | This bit is cleared (initial value) |
| 1  | Counter underflow generated          | No change (no effect on others)     |

**Note:**

Please set ILM (interrupt level mask register) value in the interruption routine of 16-bit reload timer after clearing the UF bit (timer interrupt request flag) to "0".

**[bit1] CNTE (Timer counter enable)**

This bit enables the timer counter.

| <b>CNTE</b> | <b>Function</b>                                |
|-------------|--|
| 0           | Counter operation stopped (initial value)      |
| 1           | Counter operation allowed (start trigger wait) |

**[bit0] TRG (Software trigger)**

This bit operates as a software trigger bit. With TRG set to "1", a software trigger is applied, data from the timer reload register is loaded into the counter and counting starts. Writing "0" has no effect. Read operations always read "0". Only when CNTE = 1, this bit is valid irrespective of the operation mode.

| <b>TRG</b> | <b>Function</b>           |
|------------|---------------------------|
| 0          | No change (initial value) |
| 1          | Count operation start     |

## 14.2.2 16-Bit Timer Register (TMR)/16-Bit Reload Register (TMRLR)

This section describes the configuration and functions of the 16-bit timer register (TMR)/16-bit reload register (TMRLR).

### ■ 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)

The bit configuration of the 16-bit timer register (TMR)/16-bit reload register (TMRLR) is shown below.

|                     |       |       |       |       |       |       |       |       |                        |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------------|
|                     | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |                        |
| 0000CD <sub>H</sub> | D15   | D14   | D13   | D12   | D11   | D10   | D09   | D08   | TMR/TMRLR (upper bits) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write             |
|                     | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value          |
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |                        |
| 0000CC <sub>H</sub> | D07   | D06   | D05   | D04   | D03   | D02   | D01   | D00   | TMR/TMRLR (lower bits) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write             |
|                     | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value          |

### ■ 16-bit Timer Register (TMR)

This register reads the counter value of the 16-bit decrement counter. If counter operation is enabled (TMCSR: CNTE = 1) and counting starts, the value written to the 16-bit reload register is loaded into this register to start count-down. In count stop state (TMCSR: CNTE = 0), the value of this register is retained.

Note:

This register can be read during count operation, but always use a word transfer instruction (such as "MOVW A 003AH").

The 16-bit timer register (TMR) is functionally a read-only register; however, it is allocated at the same address as the write-only 16-bit reload register (TMRLR). Thus, write operations will not affect the TMR value, even though writing to TMRLR is performed.

### ■ 16-bit Reload Register (TMRLR)

The 16-bit reload register sets the initial counter value while count operation is disabled (TMCSR: CNTE=0). When the counter is started by enabling counter operation (TMCSR: CNTE=1), the count-down will start from the value that was written to this register. The value set in this register is reloaded to the counter in reload mode if an underflow occurs, and count-down continues. In one-shot mode, the counter stops at "FFFF<sub>H</sub>" after an underflow occurs.

---

#### Note:

Write to this register only in counter stop mode (TMCSR: CNTE = 0), and always write by using a word transfer instruction (such as "MOVW A 003AH").

The 16-bit reload register (TMRLR) is functionally a write-only register; however, it is allocated at the same address as the read-only 16-bit timer register (TMR). Thus, the valid returned value in read operations is the value of the TMR value. Read-modify-write instructions (such as the INC or DEC instructions) cannot be used.

---

## 14.3 Interrupt of 16-Bit Reload Timer

The interrupt of the 16-bit reload timer occurs when underflow of the counter is detected. The underflow interrupt of counter can activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt of 16-bit Reload Timer

Table 14.3-1 shows the interrupt control bit and interrupt source of the 16-bit reload timer.

**Table 14.3-1 Interrupt of 16-bit Reload Timer**

| Reload timer                        | Underflow interrupt              |
|-------------------------------------|----------------------------------|
| Timer interrupt request flag        | TMCSR: UF (bit2)                 |
| Interrupt request output enable bit | TMCSR: INTE (bit3)               |
| Interrupt generation source         | Underflow of 16-bit reload timer |

When the value of the TMR value is decremented from "0000" to "FFFF" during the 16-bit timer register (TMR) count operation, an underflow occurs. When an underflow occurs, the timer interrupt request flag (UF = 1) in the timer control status register (TMCSR) is set. When an underflow interrupt is enabled (INTE = 1), an interrupt request is generated.

### ■ Interrupt of 16-bit Reload Timer, DMA Transfer, and EI<sup>2</sup>OS

Table 14.3-2 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 14.3-2 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source   | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|--|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|  |                          |                      | Number           | Address             | Number                     | Address             |
| 16-bit free-run timer overflow, *<br>16-bit reload timer underflow | ○                        | 12                   | #35              | FFFF70 <sub>H</sub> | ICR12                      | 0000BC <sub>H</sub> |

○: Interrupt request flag is cleared.

\* : This interrupt source shares the interrupt source and interrupt number of other peripheral function. For details, see Table 3.2-2.

#### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing.

### ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The 16-bit reload timer corresponds to the DMA transfer function and EI<sup>2</sup>OS function.

To use DMA or EI<sup>2</sup>OS function, other interrupt that shares the interrupt control register (ICR) must be disabled.

# 14.4 Operations of the 16-Bit Reload Timer

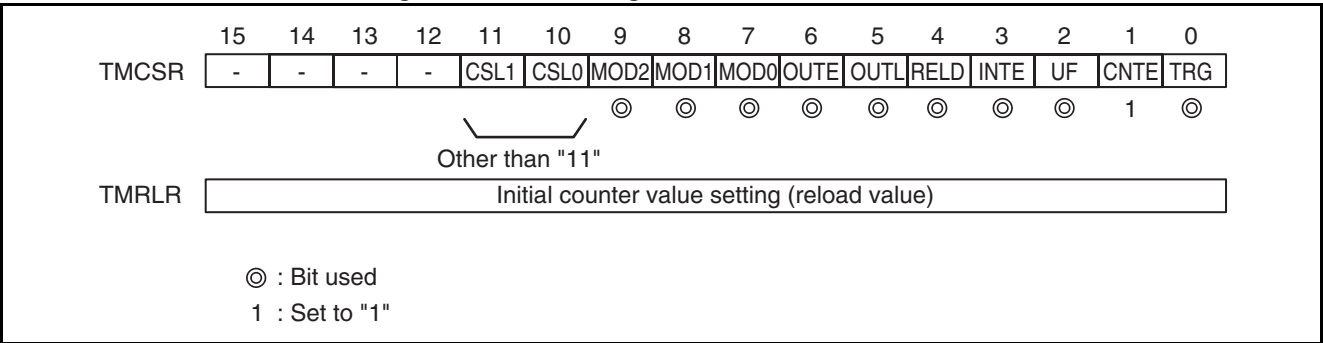
This section describes the settings of the 16-bit reload timer.

## ■ Settings of the 16-bit Reload Timer

### ○ Settings for internal clock mode

For interval timer operation, the settings shown in Figure 14.4-1 are required.

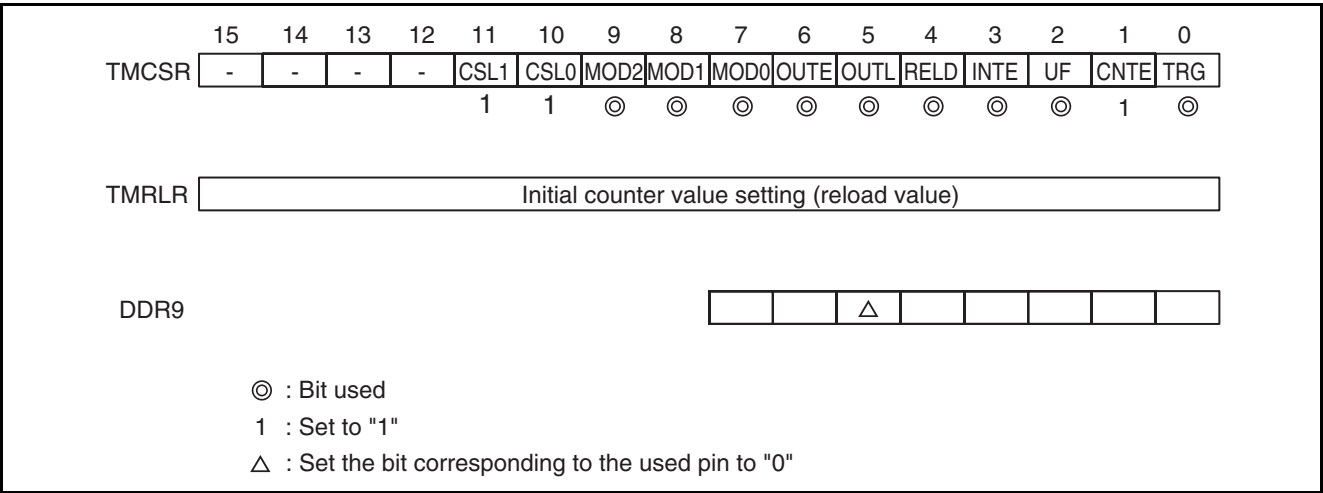
Figure 14.4-1 Settings of Internal Clock Mode



### ○ Settings for event count mode

For event count mode operation, the settings shown in Figure 14.4-2 are required.

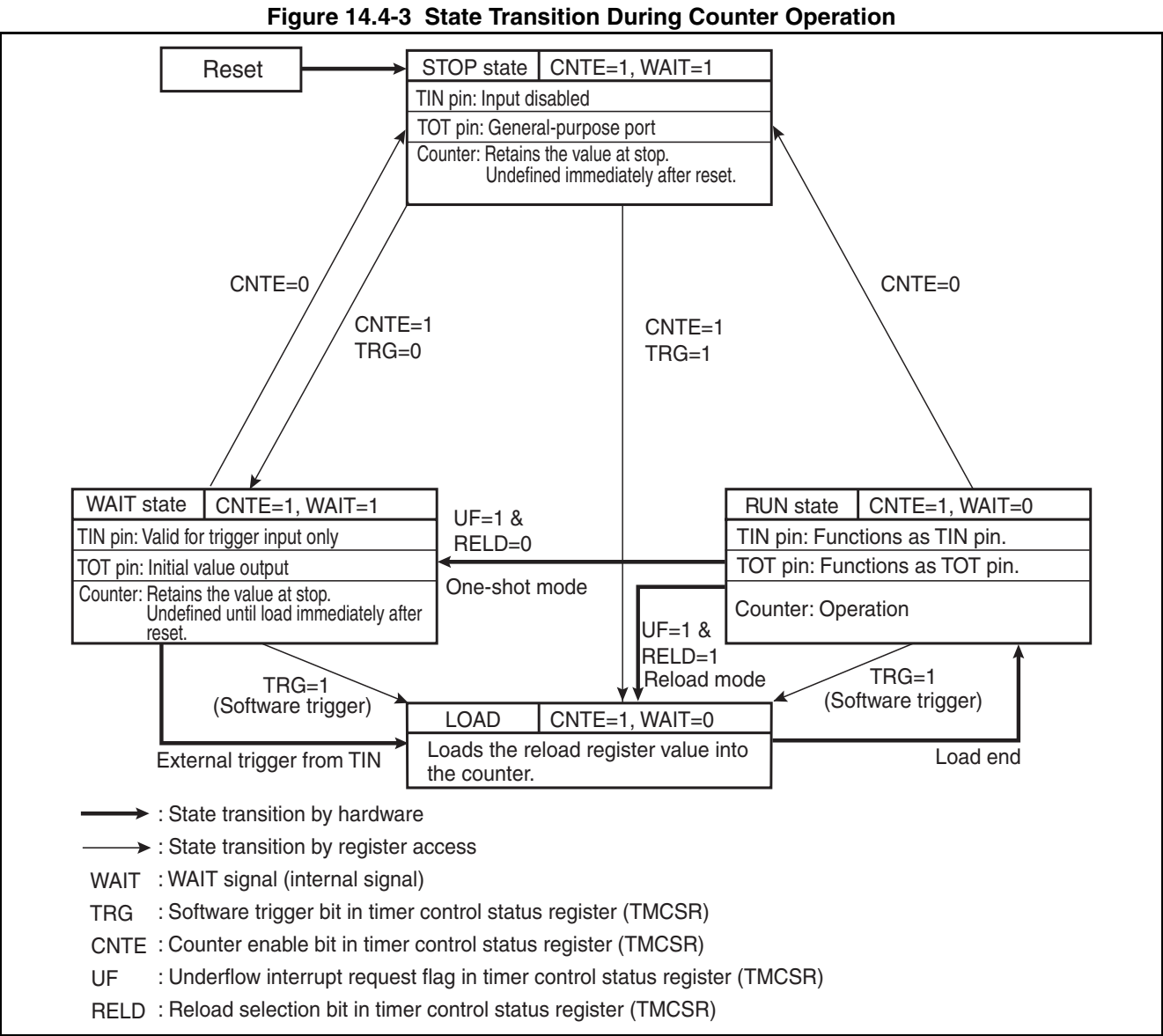
Figure 14.4-2 Settings of Event Counter Mode



### 14.4.1 State Transitions During Count Operation

This section describes the state transitions during count operation.

■ State Transitions During Count Operation





## 14.4.2 Operations of Internal Clock Mode (Reload Mode)

The counter operates in sync with the internal count clock to count down the 16-bit counter and generate an interrupt request to the CPU in case of counter underflow. The counter also outputs a toggle waveform from the timer output pin.

### ■ Operations of Internal Clock Mode (Reload Mode)

If, with count operation enabled (TMCSR: CNTE = 1), the software trigger bit (TMCSR: TRG) or external trigger is set to start the timer, the value in the reload register (TMRLR) is loaded into the counter, and counter operation starts.

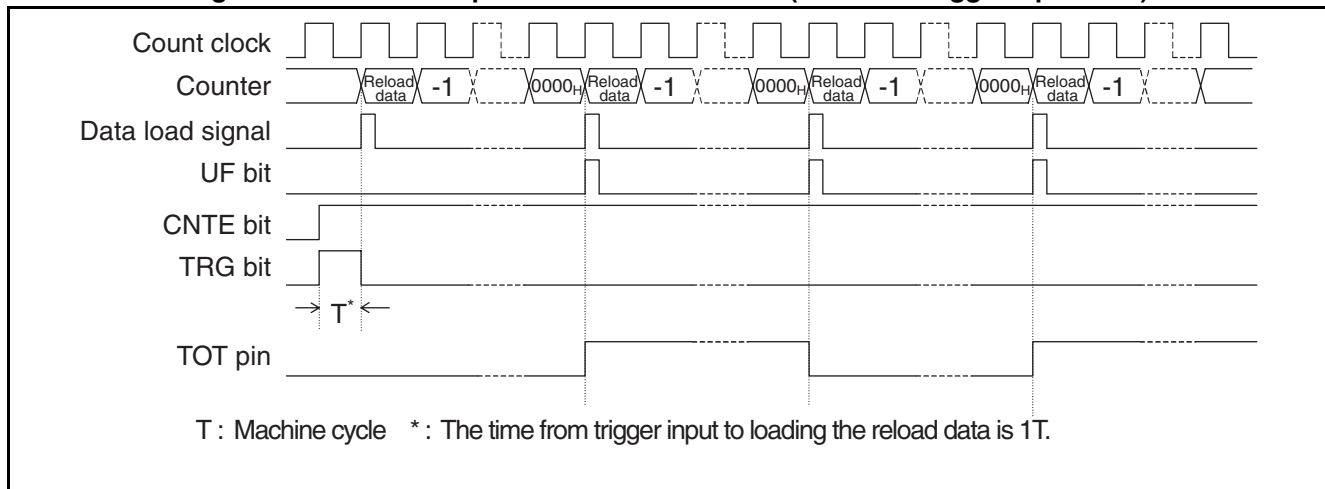
If both the counter enabled bit and the software trigger bit is set to "1" at the same time, counting will start as soon as counter operation is enabled. If the counter value causes an underflow ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the value in the 16-bit reload register (TMRLR) is loaded into the counter to continue with the counting operation. At this time, the underflow interrupt request flag bit (UF) is set to "1", and if the interrupt request enable bit (INTE) is set to "1", an interrupt request is generated. It outputs a toggle waveform that is inverted at every underflow via the TOT pin.

#### ○ Software trigger operation

With the TRG bit of the timer control status register (TMCSR) set to "1", the counter is started.

Figure 14.4-4 shows the software trigger operation for a reload.

**Figure 14.4-4 Count Operation in Reload Mode (Software Trigger Operation)**

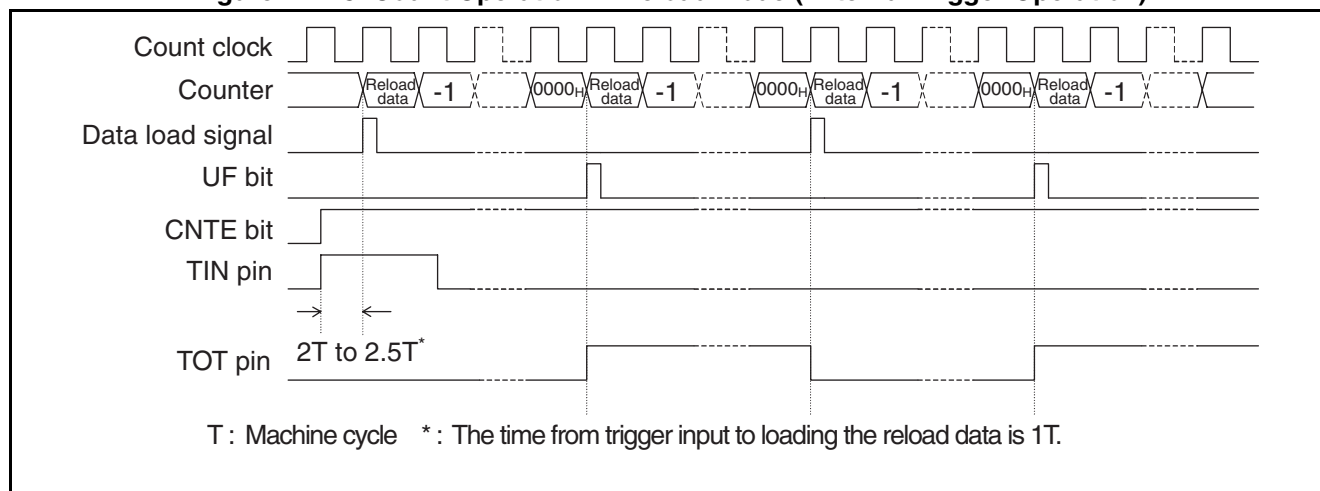


### ○ External trigger operation

The counter is started if a valid edge (rising, falling, or both edges can be selected) is input to the TIN pin.

Figure 14.4-5 shows the external trigger operation in reload mode.

**Figure 14.4-5 Count Operation in Reload Mode (External Trigger Operation)**



Note:

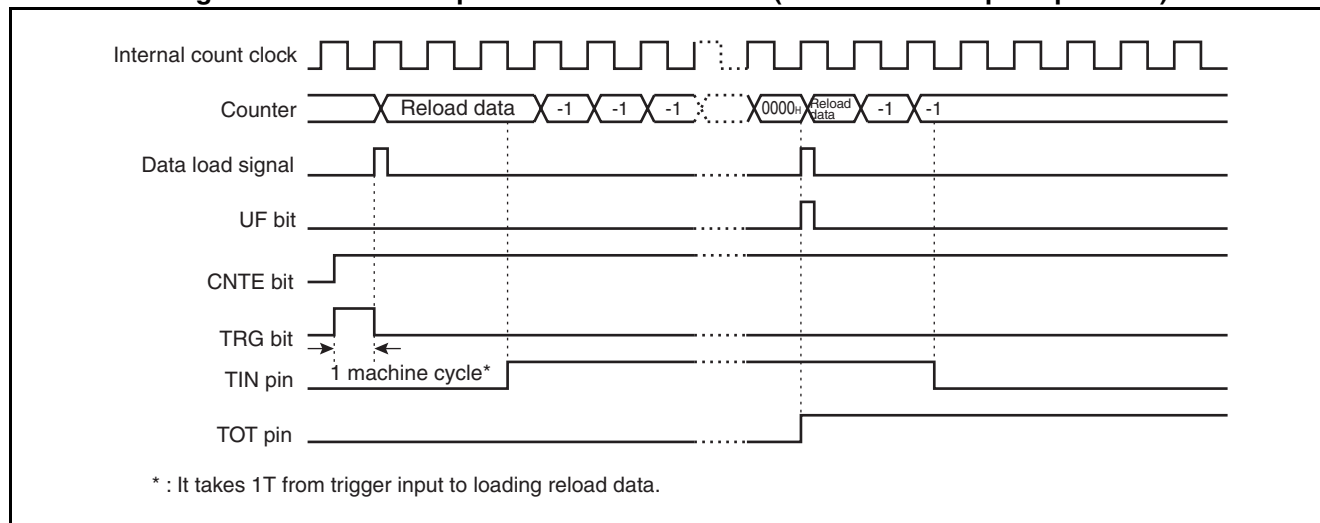
The trigger pulse width input to the TIN pin shall be 2T (T: machine cycle) or more.

### ○ External gate input operation

Counting starts when the software trigger bit (TRG) is set to "1" while the timer counter enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

Counting starts when a valid level ("L" level or "H" level can be selected) of gate input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins.

**Figure 14.4-6 Count Operation in Reload Mode (External Gate Input Operation)**



Note:

The trigger pulse width input to the TIN pin shall be 2T (T: machine cycle) or more.

### 14.4.3 Internal Clock Mode (One-Shot Mode)

The counter is in synchronization with the internal count clock in this mode to count down the 16-bit counter and generate an interrupt request to the CPU at counter underflow. It also outputs a square wave from the TOT pin to indicate that counting is in progress.

#### ■ Operation in Internal Clock Mode (One-Shot Mode)

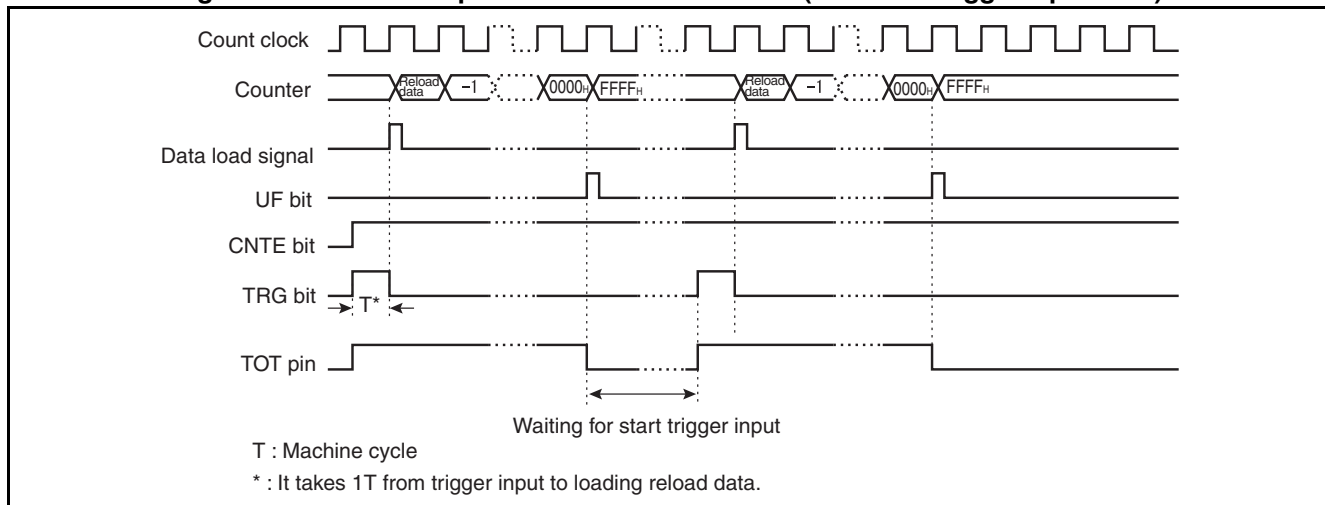
When count operation is allowed (TMCSR: CNTE=1) and the timer is started by the software trigger bit (TMCSR: TRG) or external trigger, count operation will start. When both the count enable bit and software trigger bit are set to "1" at the same time, counting will start with counter allowance. If the counter value causes an underflow ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the counter stops at "FFFF<sub>H</sub>", and the timer interrupt request flag (UF) is set to "1". If the enable bit for interrupt request (INTE) is set to "1", an interrupt request occurs.

The TOT pin outputs a square wave to indicate that counting is in progress.

#### ○ Software trigger operation

The counter is started as soon as the TRG bit of the timer control status registers (TMCSR) is set to "1". Figure 14.4-7 shows the software trigger operation in one-shot mode.

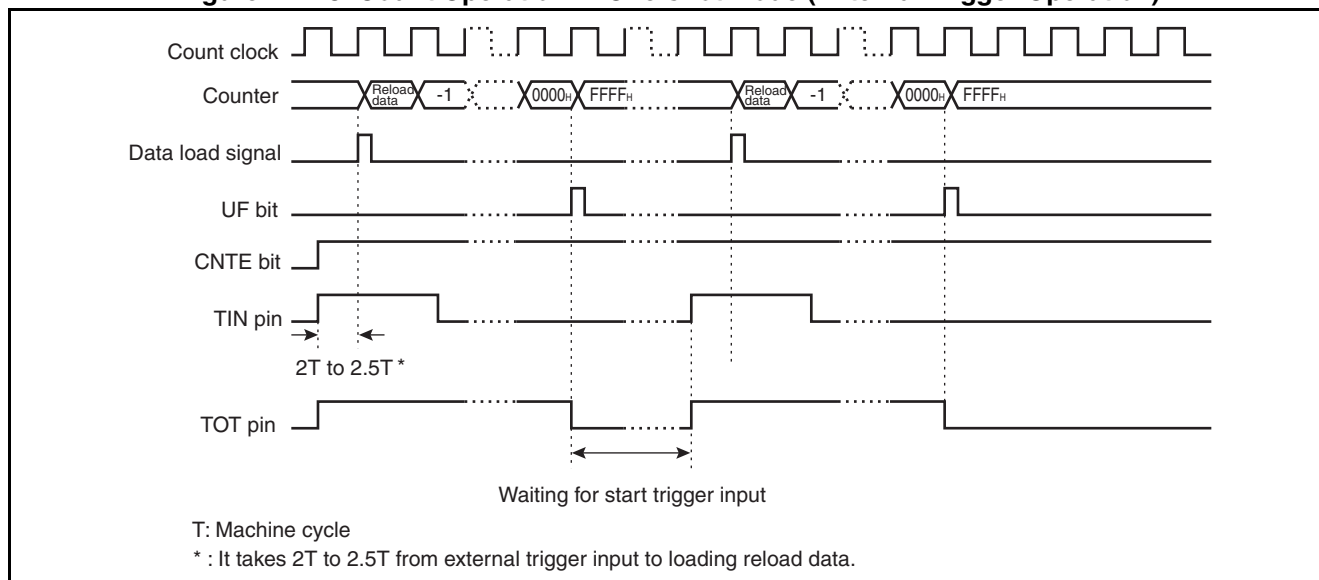
**Figure 14.4-7 Count Operation in One-shot Mode (Software Trigger Operation)**



### ○ External trigger operation

When a valid edge (leading, trailing, or both can be selected) is input to the TIN pins, the counter is started. Figure 14.4-8 shows the external trigger operation in one-shot mode.

**Figure 14.4-8 Count Operation in One-shot Mode (External Trigger Operation)**



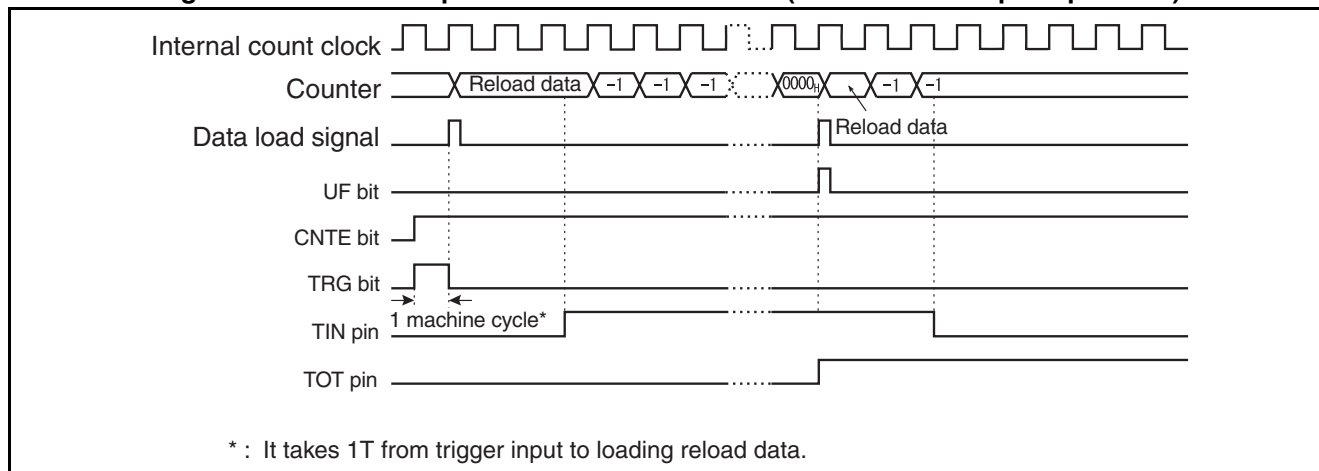
Note:

The trigger pulse width input to the TIN pin shall be 2T (T: machine cycle) or more.

### ○ External gate input operation

When a valid level ("H" and "L" level can be selected) is input to the TIN pin, counting starts. Figure 14.4-9 shows the gate input operation in one-shot mode.

**Figure 14.4-9 Count Operation in One-shot Mode (External Gate Input Operation)**



Note:

The trigger pulse width input to the TIN pin shall be 2T (T: machine cycle) or more.

## 14.4.4 Event Count Mode

The counter counts input edges from the TIN pin to count down the 16-bit counter and generate an interrupt request to the CPU when a counter underflow occurs. The TOT pin can output either a toggle waveform or a square wave.

### ■ Operation in Event Count Mode

When count operation is allowed (TMCSR: CNTE=1) to start the counter (TMCSR: TRG=1), the value of the 16-bit reload registers (TMRLR) is loaded into the counter for a countdown whenever a valid edge (leading, trailing, or both edges can be selected) of pulses (external count clock) input to the TIN pin is detected.

When both the count enable bit and software trigger bit are set to "1" at the same time, counting will start with counter allowance.

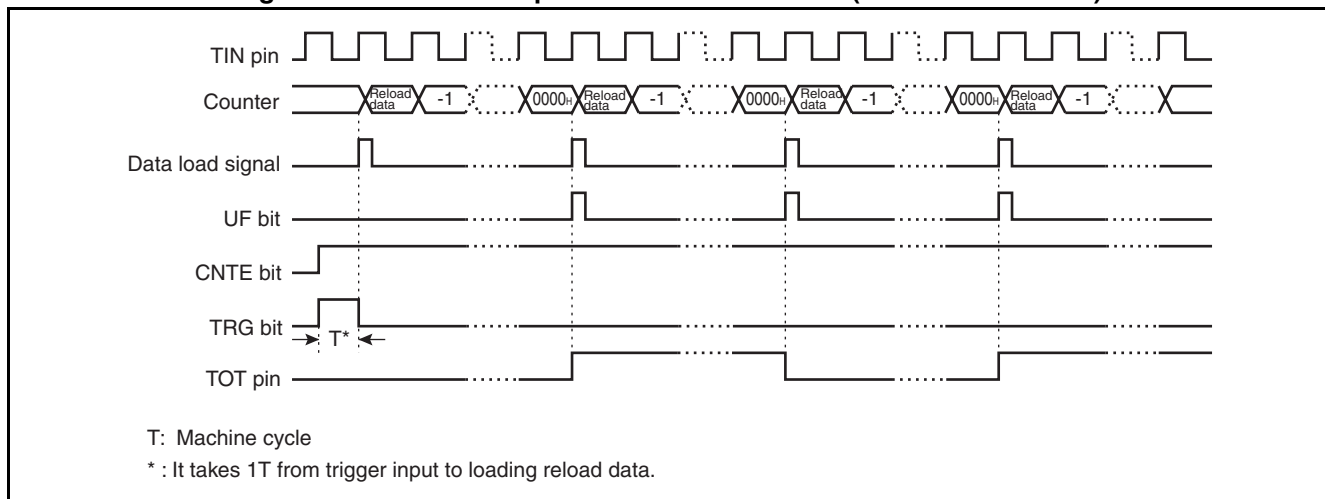
#### ○ Operation in reload mode

If the counter value causes an underflow ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the value of the 16-bit reload registers (TMRLR) is loaded into the counter to continue counting. In this case, an interrupt request is issued when the timer interrupt request flag (UF) and enable bit for interrupt requests (TMCSR: INTE) are both set to "1".

The TOT pin outputs a toggle waveform, which is reversed at every occurrence of underflow.

Figure 14.4-10 shows the counting operation in reload mode.

**Figure 14.4-10 Count Operation in Reload Mode (Event Count Mode)**



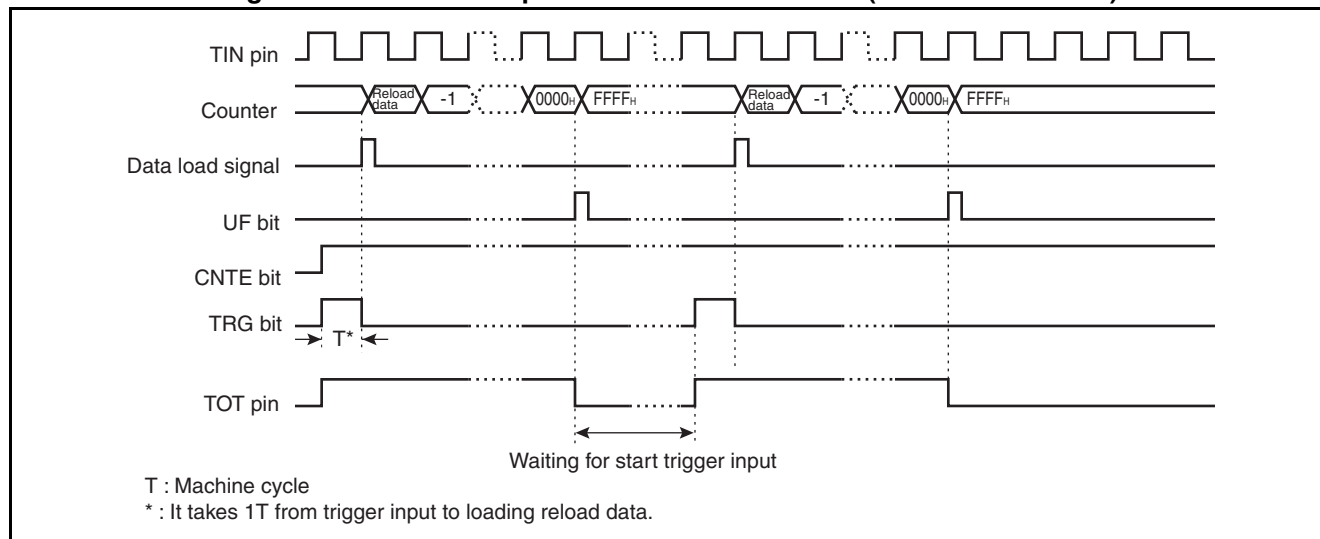
#### Note:

Both the "H" width and "L" width of clock input to the TIN pin shall be 4T (T: machine cycle) or more.

### ○ Operation in one-shot mode

If the counter value causes an underflow ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the counter stops at "FFFF<sub>H</sub>". In this case, the timer interrupt request flag (UF) is set to "1". If the interrupt request enable bit (INTE) is set to "1", an interrupt request occurs. The TOT pin outputs a square wave that indicates counting in progress. Figure 14.4-11 shows the count operation in one-shot mode.

**Figure 14.4-11 Count Operation in One-shot Mode (Event Count Mode)**



#### Note:

Both the "H" width and "L" width of clock input to the TIN pin shall be 4T (T: machine cycle) or more.

## 14.5 Program Example of 16-Bit Reload Timer

This section explains the program example of the 16-bit reload timer.

### ■ Program Example of 16-bit Reload Timer

| <p>Example of setting procedure</p> <p>Software trigger output pulse from TOT0, duty1/2, normal polarity</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Setting           <table border="1" data-bbox="203 682 630 735"> <thead> <tr> <th>Set reload value</th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td></td> <td>TMRLR</td> </tr> </tbody> </table> </li> <li>Control reload timer 0           <table border="1" data-bbox="203 787 630 1071"> <thead> <tr> <th>Set control register</th> <th>TMCSR</th> </tr> </thead> <tbody> <tr> <td>Select clock source&gt;&gt;</td> <td>.CSL1,CSL0</td> </tr> <tr> <td>Select trigger&gt;&gt;</td> <td>.MOD</td> </tr> <tr> <td>Enable output&gt;&gt;</td> <td>.OUTE</td> </tr> <tr> <td>Select output level&gt;&gt;</td> <td>.OUTL</td> </tr> <tr> <td>Select operation mode&gt;&gt;</td> <td>.RELD</td> </tr> <tr> <td>Disable interrupt&gt;&gt;</td> <td>.INTE</td> </tr> <tr> <td>Clear interrupt flag&gt;&gt;</td> <td>.UF</td> </tr> <tr> <td>Stop count&gt;&gt;</td> <td>.CNTE</td> </tr> <tr> <td>Software trigger&gt;&gt;<br/>(no processing)</td> <td>.TRG</td> </tr> </tbody> </table> </li> <li>Interrupt related           <table border="1" data-bbox="203 1123 630 1176"> <thead> <tr> <th>Interrupt level of reload timer</th> <th>ICR12</th> </tr> </thead> <tbody> <tr> <td>Set I flag</td> <td>(CCR)</td> </tr> </tbody> </table> </li> </ul> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start reload timer           <table border="1" data-bbox="203 1270 630 1428"> <thead> <tr> <th>Start PPG</th> <th>TMCSR</th> </tr> </thead> <tbody> <tr> <td>Clear interrupt flag&gt;&gt;</td> <td>.UF</td> </tr> <tr> <td>Enable interrupt&gt;&gt;</td> <td>.INTE</td> </tr> <tr> <td>Enable count&gt;&gt;</td> <td>.CNTE</td> </tr> <tr> <td>Software trigger (start)&gt;&gt;</td> <td>.TRG</td> </tr> </tbody> </table> </li> </ul> <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Interrupt processing           <table border="1" data-bbox="203 1543 630 1596"> <thead> <tr> <th>(Arbitrary processing)</th> <th></th> </tr> </thead> <tbody> <tr> <td>Clear interrupt request flag</td> <td>TMCSR.UF</td> </tr> </tbody> </table> </li> </ul> <p>Interrupt vector</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>Note:<br/>Setting related to clock and setting of _set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</p> | Set reload value        | Register name, bit name |  | TMRLR | Set control register | TMCSR | Select clock source>> | .CSL1,CSL0 | Select trigger>> | .MOD | Enable output>> | .OUTE | Select output level>> | .OUTL | Select operation mode>> | .RELD | Disable interrupt>> | .INTE | Clear interrupt flag>> | .UF | Stop count>> | .CNTE | Software trigger>><br>(no processing) | .TRG | Interrupt level of reload timer | ICR12 | Set I flag | (CCR) | Start PPG | TMCSR | Clear interrupt flag>> | .UF | Enable interrupt>> | .INTE | Enable count>> | .CNTE | Software trigger (start)>> | .TRG | (Arbitrary processing) |  | Clear interrupt request flag | TMCSR.UF | <p>Program example</p> <pre> void RT_sample(void) {     RT_initial();     RT_start(); }  void RT_initial(void) {     IO_TMRLR = 0xAA ;    /* Insert any value for reload value */      IO_TMCSR.word = 0x0050; /* Setting value =0000_0000_0101_0000 */                           /* bit15-12=0000 Undefined bit */                           /* bit11-10=00   CSL1,CSL0 internal clock φ/2 */                           /* bit9-7=000    MOD software trigger */                           /* bit6=1       Enable OUTE output */                           /* bit5=0       OUTL external output level Low */                           /* bit4=1       Enable RELD reload */                           /* bit3=0       Disable INTE interrupt */                           /* bit2=0       Clear UF interrupt request flag */                           /* bit1=0       Stop CNTE count */                           /* bit0=0       TRG software trigger */      IO_ICR12.byte = 0x10; /* Set interrupt level */     __EI();               /* Enable interrupt */ }  void RT_start(void) {     IO_TMCSR.word = 0x005B;                           /* bit3=1       Enable INTE interrupt */                           /* bit2=0       Clear UF interrupt request flag */                           /* bit1=1       Start CNTE count */                           /* bit0=1       TRG software trigger */ }  __interrupt void RT_int(void) /* Interrupt occurs at generation of underflow */ {     IO_TMCSR.bit.UF = 0; /* bit2=0   Clear UF interrupt request flag */     .....              /* Any processing operation */ }  #pragma intvect RT_int 35 </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
|--|-------------------------|-------------------------|--|-------|----------------------|-------|-----------------------|------------|------------------|------|-----------------|-------|-----------------------|-------|-------------------------|-------|---------------------|-------|------------------------|-----|--------------|-------|---------------------------------------|------|---------------------------------|-------|------------|-------|-----------|-------|------------------------|-----|--------------------|-------|----------------|-------|----------------------------|------|------------------------|--|------------------------------|----------|--|
| Set reload value   | Register name, bit name |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
|  | TMRLR                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Set control register   | TMCSR                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Select clock source>>  | .CSL1,CSL0              |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Select trigger>>   | .MOD                    |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Enable output>>  | .OUTE                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Select output level>>  | .OUTL                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Select operation mode>>  | .RELD                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Disable interrupt>>  | .INTE                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Clear interrupt flag>>   | .UF                     |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Stop count>>   | .CNTE                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Software trigger>><br>(no processing)  | .TRG                    |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Interrupt level of reload timer  | ICR12                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Set I flag   | (CCR)                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Start PPG  | TMCSR                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Clear interrupt flag>>   | .UF                     |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Enable interrupt>>   | .INTE                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Enable count>>   | .CNTE                   |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Software trigger (start)>>   | .TRG                    |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| (Arbitrary processing)   |                         |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |
| Clear interrupt request flag   | TMCSR.UF                |                         |  |       |                      |       |                       |            |                  |      |                 |       |                       |       |                         |       |                     |       |                        |     |              |       |                                       |      |                                 |       |            |       |           |       |                        |     |                    |       |                |       |                            |      |                        |  |                              |          |  |

## ■ Setting Method Other than Program Example

### ● Method to set (rewrite) reload value

The reload value is set to the 16-bit reload register (TMRLR).

The following shows the calculation formula of the set value.

<Formula>

$$\text{TMRLR register value} = \{\text{Reload interval/count clock}\} - 1$$

<Range that can be set>

$$\text{TMRLR register value} = 0 \text{ to } \text{FFFF}_{\text{H}} (65535)$$

### ● Type of count clock and selection method

The count clock can be selected by the count clock selection bits (TMCSR.CSL[1:0]) from four kinds of clocks as shown in the following table.

| Count clock    | Count clock selection bits |      | Example of count clock              |                          |                         |
|----------------|----------------------------|------|-------------------------------------|--------------------------|-------------------------|
|                | CSL1                       | CSL0 | At $\phi = 32\text{MHz}$            | At $\phi = 16\text{MHz}$ | At $\phi = 8\text{MHz}$ |
| $\phi/2$       | 0                          | 0    | 62.5ns                              | 125ns                    | 250ns                   |
| $\phi/8$       | 0                          | 1    | 250ns                               | 500ns                    | 1.0 $\mu\text{s}$       |
| $\phi/32$      | 1                          | 0    | 1.0 $\mu\text{s}$                   | 2.0 $\mu\text{s}$        | 4.0 $\mu\text{s}$       |
| External event | 1                          | 1    | Pulse width : $2/\phi_{\text{min}}$ |                          |                         |

### ● Method to enable/stop count operation of reload timer

Set by the timer counter enable bit (TMCSR: CNTE) .

| Content of control                        | Operation enable bit (CNTE) |
|---|-----------------------------|
| To stop reload timer                      | Set to "0"                  |
| To enable count operation of reload timer | Set to "1"                  |

Cannot restart in the stop state. Operation is enabled before or simultaneously starting it.

### ● Method to set mode (reload/one-shot) of reload timer

Set by the mode selection bit (TMCSR.RELD).

| Operation mode     | Mode selection bit (RELD) |
|--------------------|---------------------------|
| To set to one-shot | Set to "0"                |
| To set to reload   | Set to "1"                |

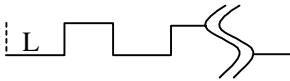
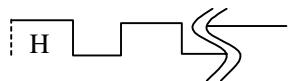
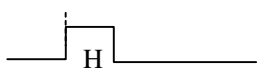



## CHAPTER 14 16-BIT RELOAD TIMER

### ● Method to reverse output level

The output level is shown in the following table.

Set by the timer output level bit (TMCSR.OUTL).

| Output level  | Timer output level bit (OUTL) |
|---|-------------------------------|
| Reload mode, "L" level output of initial value<br>           | Set to "0"                    |
| Reload mode, "H" level output of initial value (Reverse)<br> | Set to "1"                    |
| One-shot mode, "H" level output while counting<br>           | Set to "0"                    |
| One-shot mode, "L" level output while counting (Reverse)<br> | Set to "1"                    |

### ● Start method

- There are four start triggers in the internal clock mode. Set by the trigger selection bits (TMCSR.MOD[2:0]).

| Trigger                                       | Trigger setting bits (MOD[2:0]) |
|---|---------------------------------|
| Software trigger (set TRG bit)                | Set to "000 <sub>B</sub> "      |
| External trigger from TIN0 pin (rising edge)  | Set to "001 <sub>B</sub> "      |
| External trigger from TIN0 pin (falling edge) | Set to "010 <sub>B</sub> "      |
| External trigger from TIN0 pin (both edge)    | Set to "011 <sub>B</sub> "      |

- Event count mode is activated with software. The timer counter enable bit (CNTE) and software trigger bit (TRG) in the timer control status register (TMCSR) are set to "1" simultaneously.

- Types of valid edge in event count mode and selection method

Set by the trigger selection bits (TMCSR.MOD[1:0]).

Three valid edges are provided.

| Valid edge   | Trigger selection bits (MOD1, MOD0) |
|--------------|-------------------------------------|
| Rising edge  | Set to "01 <sub>B</sub> "           |
| Falling edge | Set to "10 <sub>B</sub> "           |
| Both edges   | Set to "11 <sub>B</sub> "           |

Setting MOD2 has no meaning regardless of the value of "0" or "1".

- Method to set TIN pin to external event input pin or external trigger input pin

Write "0" to the data direction specification bit (DDR7.P73).

| Pin      | Control bit                      |                          |
|----------|----------------------------------|--------------------------|
| TIN0 pin | Port 7 direction register (DDR7) | Data direction bit (P73) |

- Method to generate start trigger

- Method to generate software trigger

Set by the software trigger bit (TMCSR.TRG).

Writing "1" to the software trigger bit (TGR) generates the trigger.

To enable and start operation at the same time, set the timer counter enable bit (TMCSR: CNTE) and software trigger bit (TMCSR.TRG) simultaneously.

- Method to generate external trigger

When the edge specified by the trigger selection bits is input to the trigger pin corresponding to each reload timer, the trigger occurs.

| Timer        | Trigger pin |
|--------------|-------------|
| Reload timer | TIN0        |

### ● Interrupt related register

The relationship among the reload timer number, interrupt level, and interrupt vector, and interrupt control register is shown below.

For details of the interrupt level and interrupt vector, see CHAPTER 3 INTERRUPT.

|              | Interrupt vector                     | Interrupt level setting register                                       |
|--------------|--------------------------------------|--|
| Reload timer | #35<br>Address : FFFF70 <sub>H</sub> | Interrupt control register 12 (ICR12)<br>Address : 0000BC <sub>H</sub> |

Clear the timer interrupt request flag (TMCSR: UF) by writing "0" to the UF bit before returning from the interrupt processing because the flag is not cleared automatically.

### ● Method to enable interrupt

Interrupt request enabled, interrupt request flag

Enabling interrupt is set by the interrupt request enable bit (TMCSR.INTE).

|                              | Interrupt request enable bit (INTE) |
|------------------------------|-------------------------------------|
| To disable interrupt request | Set to "0"                          |
| To enable interrupt request  | Set to "1"                          |

The interrupt request is cleared by the timer interrupt request flag (TMCSR: UF).

|                              | Timer interrupt request flag (UF) |
|------------------------------|-----------------------------------|
| To disable interrupt request | Set to "0"                        |

#### Note:

Please write "0" in the INTE bit, after prohibiting interrupt by setting the IL2 to IL0 bit of the interrupt control register to "111", if the reload timer underflow interrupt setting is changed from enable (INTE bit of TMCSR registers =1) to prohibit (INTE bit of TMCSR registers =0).

### ● Method to stop reload timer

Set by the reload timer stop bit.

| Content of control   | Operation enable bit (CNTE) |
|----------------------|-----------------------------|
| To stop reload timer | Set to "0"                  |

# CHAPTER 15 8/16-BIT PPG TIMER

---

**This chapter provides an overview of the 8/16-bit PPG timer, explains the configuration and functions of its registers interrupt and its operation.**

---

- 15.1 Overview of 8/16-Bit PPG Timer
- 15.2 Configuration of 8/16-Bit PPG Timer
- 15.3 Configuration and Functions of 8/16-Bit PPG Timer Registers
- 15.4 Interrupt of 8/16-Bit PPG Timer
- 15.5 Operations of 8/16-Bit PPG Timer
- 15.6 Program Example of 8/16-Bit PPG Timer

## 15.1 Overview of 8/16-Bit PPG Timer

---

The 8/16-bit PPG timer is a 6-channel reload timer module that can be used any interval and output pulse of duty ratio.

On the hardware level, the timer consists of six 8-bit decrement counters, twelve 8-bit reload timers, three 16-bit control registers, six external pulse output pins and six interrupt outputs.

The MB90480/485 series has six channels that can be used for 8-bit PPG, enabling paired operation of PPG0 + PPG1, PPG2 + PPG3, and PPG4 + PPG5 for 16-bit PPG operation (3 channels).

---

### ■ Functions of 8/16-bit PPG Timer

- **8-bit PPG output 6-channel independent operation mode**

Provides independent PPG output operation with six channels.

- **16-bit PPG output operation mode**

Provides 16-bit PPG output operation with three channels.

This is achieved by combining PPG0 + PPG1, PPG2 + PPG3 and PPG4 + PPG5.

- **8 + 8-bit PPG output operation mode**

PPG0 (PPG2/PPG4) output is inputted to PPG1 (PPG3/PPG5) clock, enabling an arbitrary interval to be used for 8-bit PPG output.

- **PPG output operation**

Provides an arbitrary interval and duty ratio for pulse wave output.

Used in combination with an externally attached circuit for providing a D/A converter.

## 15.2 Configuration of 8/16-Bit PPG Timer

This section shows the configuration of ch.0/2/4 and ch.1/3/5 of the 8/16-bit PPG timer.

### ■ Block Diagram of the 8/16-bit PPG Timer

Figure 15.2-1 shows a block diagram of ch.0, ch.2, and ch.4. Figure 15.2-2 shows a block diagram of ch.1, ch.3, and ch.5.

**Figure 15.2-1 Block Diagram of the 8/16-bit PPG Timer (ch.0/2/4)**

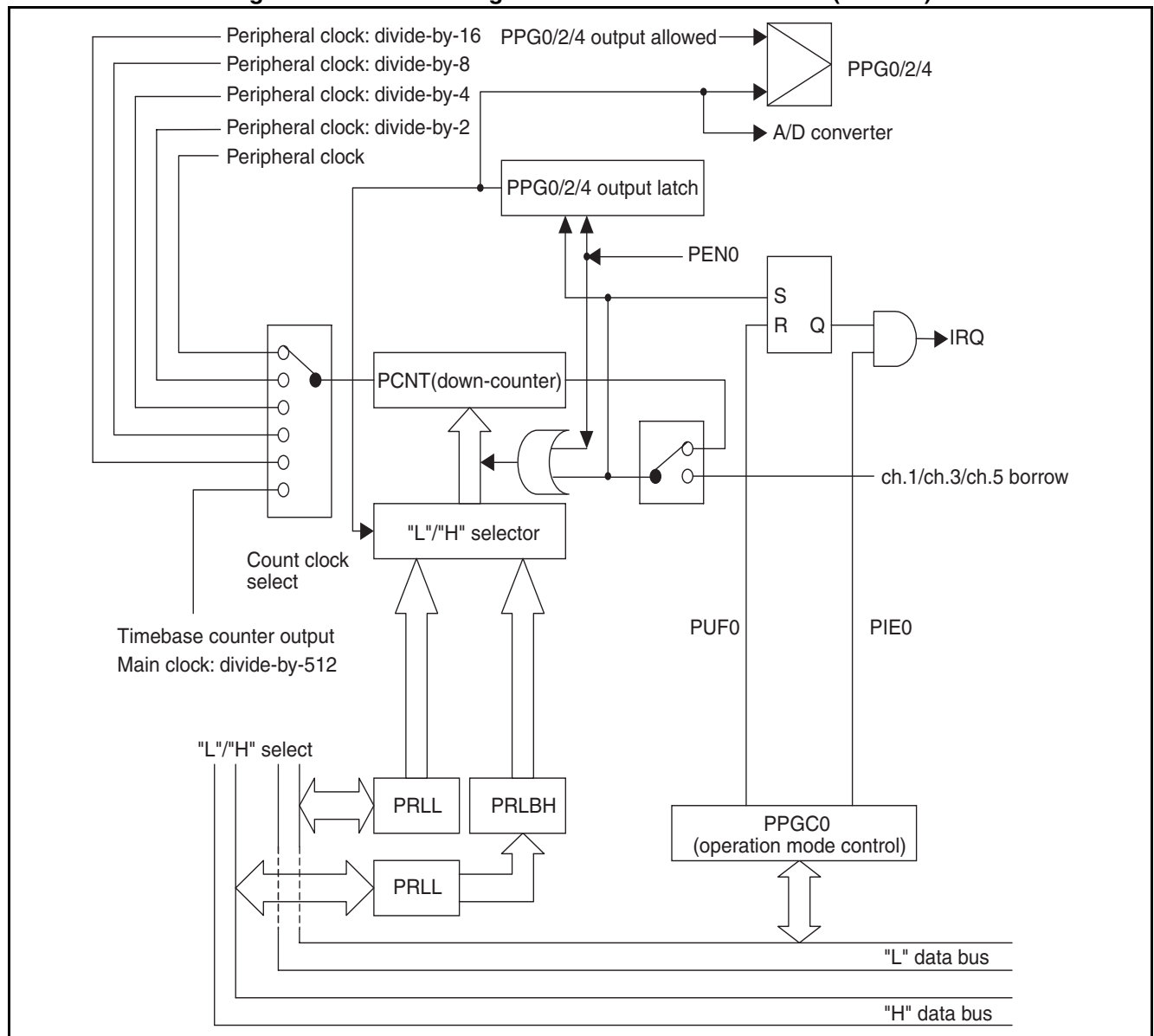
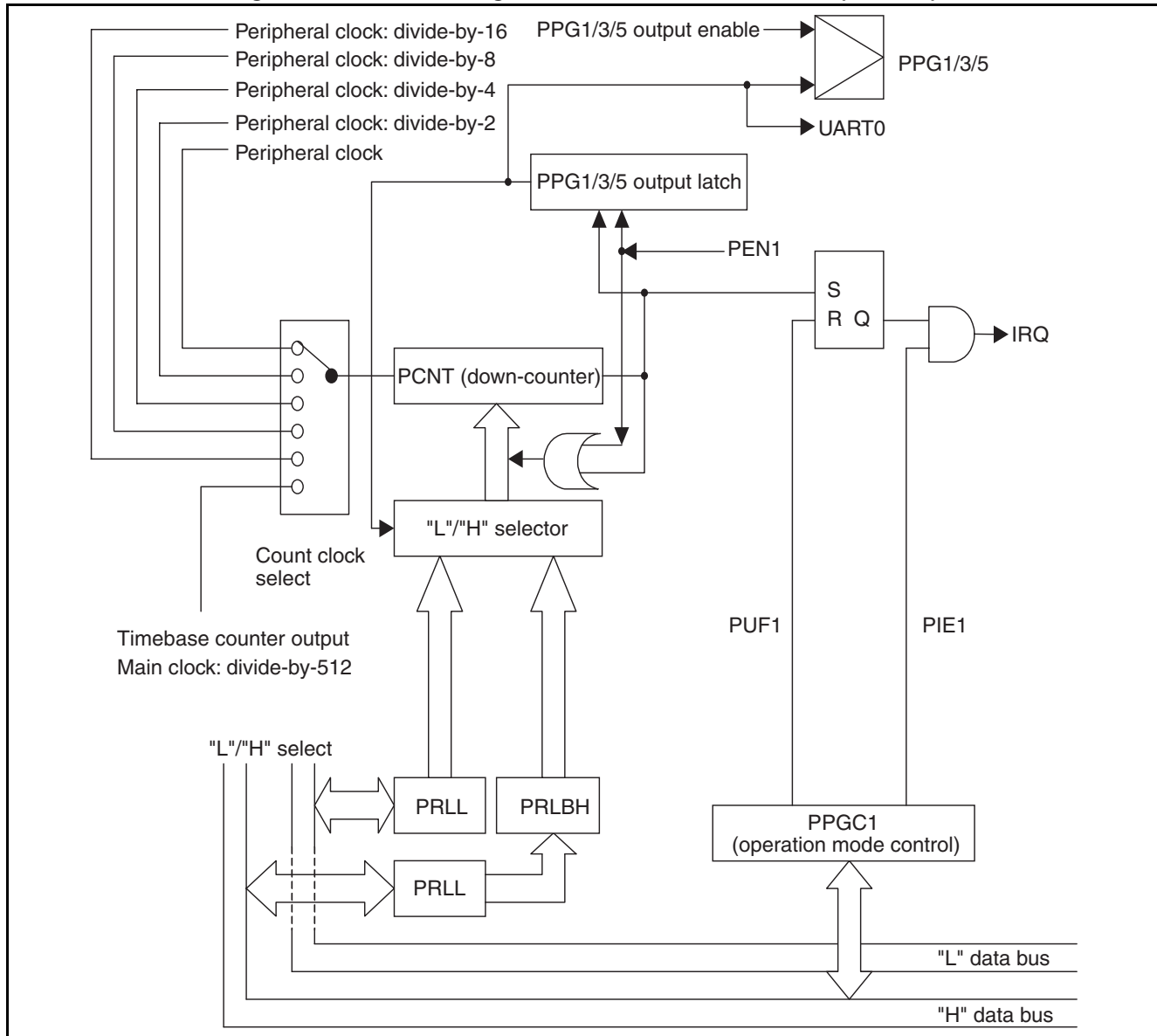


Figure 15.2-2 Block Diagram of the 8/16-bit PPG Timer (ch.1/3/5)



## ■ Pin Related to 8/16-bit PPG Timer

Pin related to the 8/16-bit PPG timer has the PPG0/PPG1/PPG2/PPG3/PPG4/PPG5 pins. These pins function as the general-purpose I/O port (P24/PPG0, P25/PPG1, P26/PPG2, P27/PPG3, P94/PPG4, P95/PPG5) and the output pin of the PPG timer.

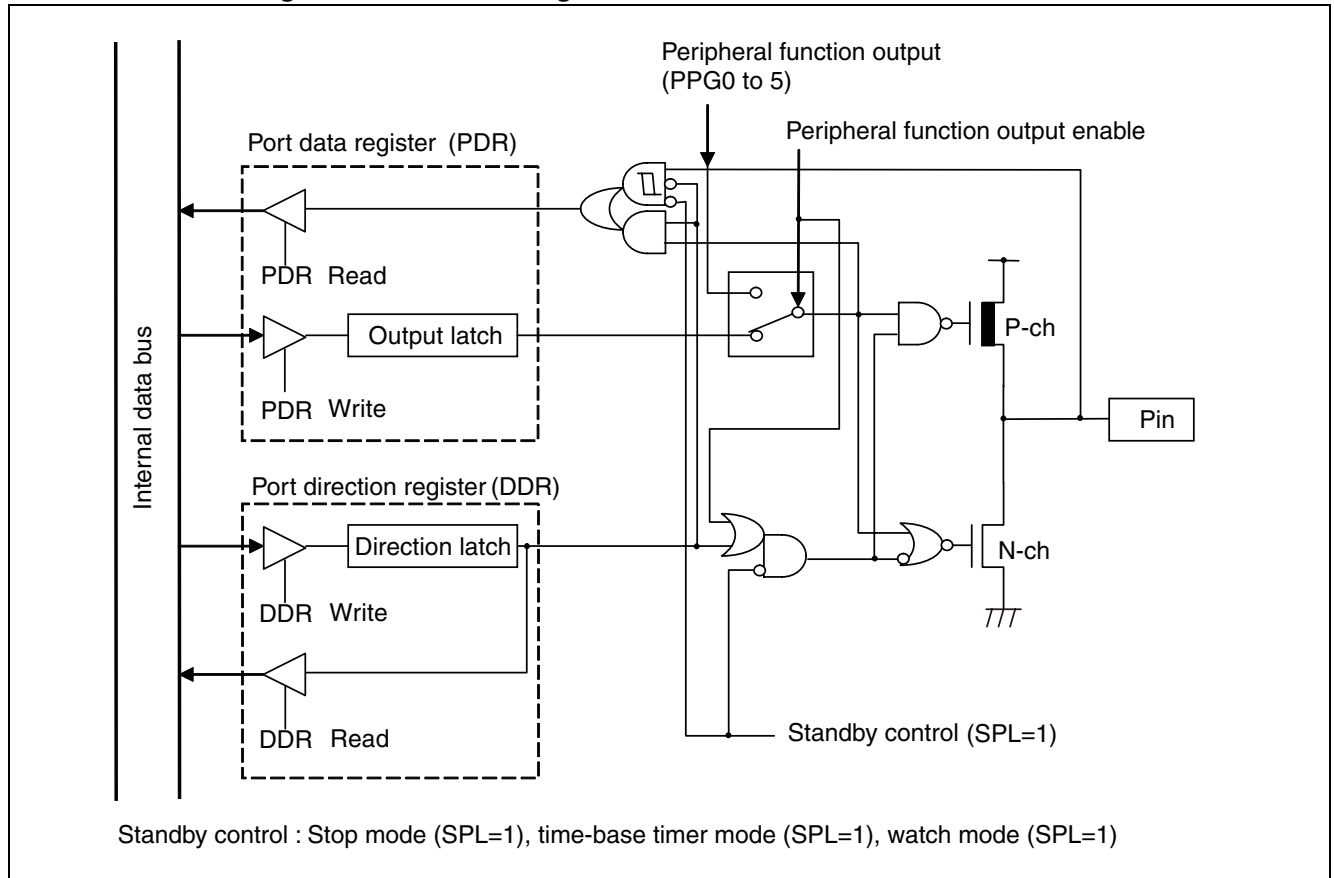
### ● Setting when using as PPG0/PPG1/PPG2/PPG3/PPG4/PPG5 pins

When the PPG0/PPG1/PPG2/PPG3/PPG4/PPG5 pins are used as output, they are set to the output pin automatically regardless of the value of the port direction register (DDR).

When using the PPG0/PPG1/PPG2/PPG3 pins, set E23 bit to "1" from the E20 of the external address output control register (HACR) (set I/O port).

## ■ Block Diagram of Pin Related to 8/16-bit PPG Timer

Figure 15.2-3 Block Diagram of Pin Related to 8/16-bit PPG Timer





## 15.3 Configuration and Functions of 8/16-Bit PPG Timer Registers

This section describes the configuration and functions of the registers used in the 8/16-bit PPG timer.

### ■ List of 8/16-bit PPG Timer Registers

Figure 15.3-1 shows a list of the registers for the 8/16-bit PPG timer.

**Figure 15.3-1 List of 8/16-bit PPG Timer Registers**

|  |       |       |       |       |       |       |          |          |                                 |
|--|-------|-------|-------|-------|-------|-------|----------|----------|---------------------------------|
| ch.0 00003A <sub>H</sub><br>ch.2 00003C <sub>H</sub><br>ch.4 00003E <sub>H</sub>   | 7     | 6     | 5     | 4     | 3     | 2     | 1        | 0        | PPGC0/PPGC2/PPGC4               |
|  | PEN0  | -     | PE00  | PIE0  | PUF0  | -     | -        | Reserved | Operation mode control register |
|  | (R/W) | (-)   | (R/W) | (R/W) | (R/W) | (-)   | (-)      | (-)      | Read/write                      |
|  | (0)   | (X)   | (0)   | (0)   | (0)   | (X)   | (X)      | (1)      | Initial value                   |
| ch.1 00003B <sub>H</sub><br>ch.3 00003D <sub>H</sub><br>ch.5 00003F <sub>H</sub>   | 15    | 14    | 13    | 12    | 11    | 10    | 9        | 8        | PPGC1/PPGC3/PPGC5               |
|  | PEN1  | -     | PE10  | PIE1  | PUF1  | MD1   | MD0      | Reserved | Operation mode control register |
|  | (R/W) | (-)   | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (-)      | Read/write                      |
|  | (0)   | (X)   | (0)   | (0)   | (0)   | (0)   | (0)      | (1)      | Initial value                   |
| ch.0,1 000040 <sub>H</sub><br>ch.2,3 000042 <sub>H</sub><br>ch.4,5 000044 <sub>H</sub>   | 7     | 6     | 5     | 4     | 3     | 2     | 1        | 0        | PPG01/PPG23/PPG45               |
|  | PCS2  | PCS1  | PCS0  | PCM2  | PCM1  | PCM0  | Reserved | Reserved | Output control register         |
|  | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (R/W)    | Read/write                      |
|  | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)      | (0)      | Initial value                   |
| ch.0 00002E <sub>H</sub><br>ch.1 000030 <sub>H</sub><br>ch.2 000032 <sub>H</sub><br>ch.3 000034 <sub>H</sub><br>ch.4 000036 <sub>H</sub><br>ch.5 000038 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1        | 0        | PPLL0 to PPLL5                  |
|  | D07   | D06   | D05   | D04   | D03   | D02   | D01      | D00      | Reload register "L"             |
|  | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (R/W)    | Read/write                      |
|  | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)      | (X)      | Initial value                   |
| ch.0 00002F <sub>H</sub><br>ch.1 000031 <sub>H</sub><br>ch.2 000033 <sub>H</sub><br>ch.3 000035 <sub>H</sub><br>ch.4 000037 <sub>H</sub><br>ch.5 000039 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9        | 8        | PPLH0 to PPLH5                  |
|  | D15   | D14   | D13   | D12   | D11   | D10   | D09      | D08      | Reload register "H"             |
|  | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (R/W)    | Read/write                      |
|  | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)      | (X)      | Initial value                   |

### 15.3.1 PPG0/2/4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4)

This section describes the configuration and functions of the PPG0/PPG2/PPG4 operation mode control register (PPGC0/PPGC2/PPGC4).

#### ■ PPG0/PPG2/PPG4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4)

The PPG0/PPG2/PPG4 operation mode control register (PPGC0/PPGC2/PPGC4) is used to select the ch.0/2/4 operation mode, control the pin output, select the count clock, and control the trigger.

The bit configuration of the PPG0/PPG2/PPG4 operation mode control register (PPGC0/PPGC2/PPGC4) is shown below.

|                     |       |     |       |       |       |     |     |          |                                 |
|---------------------|-------|-----|-------|-------|-------|-----|-----|----------|---------------------------------|
| 00003A <sub>H</sub> | 7     | 6   | 5     | 4     | 3     | 2   | 1   | 0        | PPGC0/PPGC2/PPGC4               |
| 00003C <sub>H</sub> | PEN0  | -   | PE00  | PIE0  | PUF0  | -   | -   | Reserved | Operation mode control register |
| 00003E <sub>H</sub> | (R/W) | (-) | (R/W) | (R/W) | (R/W) | (-) | (-) | (-)      | Read/write                      |
|                     | (0)   | (X) | (0)   | (0)   | (0)   | (X) | (X) | (1)      | Initial value                   |

The functions of the bits in the PPG0/PPG2/PPG4 operation mode control register (PPGC0/PPGC2/PPGC4) are described below.

#### [bit7] PEN0: ppg Enable (operation enable)

This bit is used to select the PPG operation mode.

| PEN0 | Operation state                               |
|------|---|
| 0    | Operation stop ("L" level output is retained) |
| 1    | PPG operation enable                          |

- When this bit is set to "1", the PPG starts counting.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

#### [bit5] PE00: ppg Output Enable 00 (PPG0/PPG2/PPG4 output pin enable)

This bit is used to prohibit/allow pulse output via the pulse output external pin PPG0/PPG2/PPG4.

| PE00 | Operation state                                    |
|------|--|
| 0    | General-purpose port pin (pulse output prohibited) |
| 1    | PPG0/PPG2/PPG4 pulse output (pulse output allowed) |

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit4] PIE0:ppg Interrupt Enable (PPG interrupt enable)**

This bit is used to prohibit/allow PPG interrupts.

| PIE0 | Operation state       |
|------|-----------------------|
| 0    | Interrupts prohibited |
| 1    | Interrupts allowed    |

- If PUF0 is changed to "1" while this bit is "1", an interrupt request occurs. If this bit is "0", no interrupt generates.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit3] PUF0: ppg Underflow Flag (PPG counter underflow)**

This bit indicates the result of a PPG counter underflow detection.

| PUF0 | Operation state                   |
|------|-----------------------------------|
| 0    | No PPG counter underflow detected |
| 1    | PPG counter underflow detected    |

In 8-bit PPG6 channel mode (PPG0/PPG1,PPG2/PPG3,PPG4/PPG5) and 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" if an underflow occurs because the counter value for ch.0/2/4 changes "00<sub>H</sub>" → "FF<sub>H</sub>". In 16-bit PPG3 channel mode (PPG0/PPG1, PPG2/PPG3, PPG4/PPG5), this bit is set to "1" due to underflow if the counter value of ch.1, 3, 5 or ch.0, 2, 4 changes "0000<sub>H</sub>" → "FFFF<sub>H</sub>". Writing "0" clears this bit to "0". Writing "1" has no effect. Reading by read-modify-write type instructions always read "1".

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit0] reserved bit**

This bit is reserved. When setting PPGC0/PPGC2/PPGC4, always set this bit to "1".

## 15.3.2 PPG1/3/5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5)

This section describes the configuration and functions of the PPG1/PPG3/PPG5 operation mode control register (PPGC1/PPGC3/PPGC5).

### ■ PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5)

The PPG1/PPG3/PPG5 operation mode control register (PPGC1/PPGC3/PPGC5) is used to select the ch.1/3/5 operation mode, control pin output, and select the count clock. It is also used for trigger control.

The bit configuration of the PPG1/PPG3/PPG5 operation mode control register (PPGC1/PPGC3/PPGC5) is shown below.

|                          |       |     |       |       |       |       |       |          |                                 |
|--------------------------|-------|-----|-------|-------|-------|-------|-------|----------|---------------------------------|
| ch.1 00003B <sub>H</sub> | 15    | 14  | 13    | 12    | 11    | 10    | 9     | 8        | PPGC1/3/5                       |
| ch.3 00003D <sub>H</sub> | PEN1  | -   | PE10  | PIE1  | PUF1  | MD1   | MD0   | Reserved | Operation mode control register |
| ch.5 00003F <sub>H</sub> | (R/W) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (-)      | Read/write                      |
|                          | (0)   | (X) | (0)   | (0)   | (0)   | (0)   | (0)   | (1)      | Initial value                   |

The functions of the bits in the PPG1/PPG3/PPG5 operation mode control register (PPGC1/PPGC3/PPGC5) are described below.

#### [bit15] PEN1: ppg Enable (operation enable)

This bit is used to select the PPG operation mode.

| PEN1 | Operation state                               |
|------|---|
| 0    | Operation stop ("L" level output is retained) |
| 1    | PPG operation enabled                         |

- When this bit is set to "1", PPG count starts.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

#### [bit13] PE10: ppg output Enable 10 (PPG1/PPG3/PPG5 output pin enable)

This bit is used to allow or prohibit pulse output to the pulse output external pin PPG1/PPG3/PPG5.

| PE10 | Operation state                                    |
|------|--|
| 0    | General-purpose port pin (pulse output prohibited) |
| 1    | PPG1/PPG3/PPG5 pulse output (pulse output allowed) |

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit12] PIE1: ppg Interrupt Enable (PPG interrupt enable)**

This bit is used to prohibit/allow PPG interrupts.

| PIE1 | Operation state       |
|------|-----------------------|
| 0    | Interrupts prohibited |
| 1    | Interrupts allowed    |

If PUF0 is set to "1" when this bit is "1", an interrupt request occurs. When this bit is "0", no interrupt generates.

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit11] PUF1: ppg Underflow Flag (PPG counter underflow)**

This bit is used to indicate the result of PPG counter underflow detection.

| PUF1 | Operation state                   |
|------|-----------------------------------|
| 0    | No PPG counter underflow detected |
| 1    | PPG counter underflow detected    |

In 8-bit PPG6 channel mode (PPG0/PPG1, PPG2/PPG3, PPG4/PPG5) and 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs because the counter value of ch.1, 3, 5 changes "00<sub>H</sub>" → "FF<sub>H</sub>". In 16-bit PPG3 channel mode (PPG0/PPG1, PPG2/PPG3, PPG4/PPG5), this bit is set to "1" when an underflow occurs because the counter value of ch.1, 3, 5 or ch.0, 2, 4 changes "0000<sub>H</sub>" → "FFFF<sub>H</sub>". Writing "0" clears this bit to "0". Writing "1" has no effect. Reading by read-modify-write type instructions will always read "1".

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit10, bit9] MD1, 0: ppg Count Mode (operation mode selection)**

These bits are used to select the PPG timer operation mode.

| MD1 | MD0 | Operation mode                            |
|-----|-----|---|
| 0   | 0   | 8-bit PPG2 channel independent mode (× 3) |
| 0   | 1   | 8-bit prescaler/8-bit PPG1ch              |
| 1   | 0   | Reserved (setting prohibited)             |
| 1   | 1   | 16-bit PPG1 channel mode (× 3)            |

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

## Notes:

- Do not set these bits to "10<sub>B</sub>".
  - To set these bits to "01<sub>B</sub>", do not set the PEN0 bit of PPGC0 and the PEN1 bit of PPGC1 to "01<sub>B</sub>". It is recommended that the PEN0 bit and the corresponding PEN1 bit be set to "11<sub>B</sub>" or "00<sub>B</sub>" at the same time.
  - To set these bits to "11<sub>B</sub>", rewrite the contents of PPGC0/PPGC1 by word transfer and set the PEN0/PEN1 bits to "11<sub>B</sub>" or "00<sub>B</sub>" at the same time.
- 

**[bit8] reserved bit**

This bit is reserved. When setting PPGC1/PPGC3/PPGC5, always set this bit to "1".

### 15.3.3 PPG0 to PPG5 Output Control Registers (PPG01, PPG23, PPG45)

This section describes the configuration and functions of the PPG0 to PPG5 output control registers (PPG01, PPG23, PPG45).

#### ■ PPG0 to PPG5 Output Control Registers (PPG01, PPG23, PPG45)

The bit configuration of the PPG0 to 5 output control registers (PPG01, PPG23, PPG45) is described below.

|                               |       |       |       |       |       |       |          |          |                         |
|-------------------------------|-------|-------|-------|-------|-------|-------|----------|----------|-------------------------|
| ch.0,ch.1 000040 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1        | 0        | PPG01, PPG23, PPG45     |
| ch.2,ch.3 000042 <sub>H</sub> | PCS2  | PCS1  | PCS0  | PCM2  | PCM1  | PCM0  | Reserved | Reserved | Output control register |
| ch.4,ch.5 000044 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (R/W)    | Read/write              |
|                               | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)      | (0)      | Initial value           |

The functions of the bits in the PPG0 to PPG5 output control registers (PPG01, PPG23, PPG45) are described below.

#### [bit7, bit6, bit5] PCS2 to 0:ppg Count Select (count clock selection)

These bits are used to select the operation clock for the down counter of ch.1, ch.3, and ch.5.

| PCS2 | PCS1 | PCS0 | Operation mode  |
|------|------|------|---|
| 0    | 0    | 0    | Peripheral clock (62.5 ns machine clock for 16 MHz)   |
| 0    | 0    | 1    | Peripheral clock/2 (125 ns machine clock for 16 MHz)  |
| 0    | 1    | 0    | Peripheral clock/4 (250 ns machine clock for 16 MHz)  |
| 0    | 1    | 1    | Peripheral clock/8 (500 ns machine clock for 16 MHz)  |
| 1    | 0    | 0    | Peripheral clock/16 (1 μs machine clock for 16 MHz)   |
| 1    | 1    | 1    | Input clock from the time-base counter<br>( $2^9 \times 250 \text{ ns} = 128 \text{ μs}$ oscillation for 4 MHz) |

- These bits are initialized to "000<sub>B</sub>" at reset.
- Reading and writing are allowed.

Note:

In 8-bit prescaler + 8-bit PPG mode and in 16-bit PPG mode, setting bits PCS2 to 0 is disabled since the PPG of ch.1, ch.3, and ch.5 receives the counter clock signal from ch.0, ch.2, and ch.4.

**[bit4, bit3, bit2] PCM2 to 0: ppg Count Mode (count clock selection)**

These bits are used to select the operation clock for the down counter of ch.0, ch.2, and ch.4.

| PCM2 | PCM1 | PCM0 | Operation mode   |
|------|------|------|--|
| 0    | 0    | 0    | Peripheral clock (62.5 ns machine clock for 16 MHz)  |
| 0    | 0    | 1    | Peripheral clock/2 (125 ns machine clock for 16 MHz)   |
| 0    | 1    | 0    | Peripheral clock/4 (250 ns machine clock for 16 MHz)   |
| 0    | 1    | 1    | Peripheral clock/8 (500 ns machine clock for 16 MHz)   |
| 1    | 0    | 0    | Peripheral clock/16 (1 $\mu$ s machine clock for 16 MHz)   |
| 1    | 1    | 1    | Input clock from the time-base counter<br>( $2^9 \times 250 \text{ ns} = 128 \mu\text{s}$ oscillation for 4 MHz) |

- These bits are initialized to "000<sub>B</sub>" at reset.
- Reading and writing are allowed.

**[bit1, bit0] reserved bits**

These bits are reserved. When setting PPG01, PPG23, or PPG45, set these bits to "00<sub>B</sub>".



### 15.3.4 Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5)

This section describes the configuration and functions of the reload registers (PRL0 to PRL5, PRLH0 to PRLH5).

#### ■ Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5)

The bit configuration of the reload registers (PRL0 to PRL5, PRLH0 to PRLH5) is shown below.

|                          |       |       |       |       |       |       |       |       |                     |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------|
| ch.0 00002E <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | PRL0 to PRL5        |
| ch.1 000030 <sub>H</sub> | D07   | D06   | D05   | D04   | D03   | D02   | D01   | D00   | Reload register "L" |
| ch.2 000032 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write          |
| ch.3 000034 <sub>H</sub> | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value       |
| ch.4 000036 <sub>H</sub> |       |       |       |       |       |       |       |       |                     |
| ch.5 000038 <sub>H</sub> |       |       |       |       |       |       |       |       |                     |
| ch.0 00002F <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | PRLH0 to PRLH5      |
| ch.1 000031 <sub>H</sub> | D15   | D14   | D13   | D12   | D11   | D10   | D09   | D08   | Reload register "H" |
| ch.2 000033 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write          |
| ch.3 000035 <sub>H</sub> | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value       |
| ch.4 000037 <sub>H</sub> |       |       |       |       |       |       |       |       |                     |
| ch.5 000039 <sub>H</sub> |       |       |       |       |       |       |       |       |                     |

The reload registers (PRL0 to PRL5, PRLH0 to PRLH5) are 8-bit registers which each store a reload value for the down counter (PCNT). The registers have the following functions.

| Register name | Function                         |
|---------------|----------------------------------|
| PRL           | Stores the "L" side reload value |
| PRLH          | Stores the "H" side reload value |

- Both registers can be read and written.

Note:

In 8-bit prescaler + 8-bit PPG mode, it is recommended that PRL and PRLH for ch.0/2/4 are set to the same value.

## 15.4 Interrupt of 8/16-Bit PPG Timer

The interrupt of the 8/16-bit PPG timer occurs when the PPG counter underflow is detected.

The interrupt of the PPG counter underflow cannot activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt of 8/16-bit PPG Timer

Table 15.4-1 shows the interrupt control bit and interrupt source of the 8/16-bit PPG timer.

**Table 15.4-1 Interrupt of 8/16-bit Input/Output Timer**

|                                     | PPG0/2/4 overflow interrupt             |      | PPG1/3/5 overflow interrupt             |      |
|-------------------------------------|---|------|---|------|
| Interrupt request flag              | PPG0:PUF0 (bit3)                        | ch.0 | PPG1:PUF1 (bit3)                        | ch.1 |
|                                     | PPG2:PUF0 (bit3)                        | ch.2 | PPG3:PUF1 (bit3)                        | ch.3 |
|                                     | PPG4:PUF0 (bit3)                        | ch.4 | PPG5:PUF1 (bit3)                        | ch.5 |
| Interrupt request output enable bit | PPG0:PUE0 (bit4)                        | ch.0 | PPG1:PUE1 (bit4)                        | ch.1 |
|                                     | PPG2:PUE0 (bit4)                        | ch.2 | PPG3:PUE1 (bit4)                        | ch.3 |
|                                     | PPG4:PUE0 (bit4)                        | ch.4 | PPG5:PUE1 (bit4)                        | ch.5 |
| Interrupt generation source         | Overflow in PPG0/PPG2/PPG4 down counter |      | Overflow in PPG1/PPG3/PPG5 down counter |      |

### ■ Interrupt of PPG Counter Underflow

- 8-bit PPG and 8 + 8-bit PPG output operation mode
  - In the 8-bit PPG 6-channel independent operation mode or the 8 + 8-bit PPG output operation mode, the interrupt can be generated independently.
  - When the value of the PPG down counter is decremented from "00<sub>H</sub>" to "FF<sub>H</sub>", an underflow occurs. When an underflow occurs, the underflow bit in the channel causing an underflow is set (PUF0 = 1, PUF1 = 1).
- 16-bit PPG output operation mode
  - In the 16-bit PPG output operation mode, when the values of the PPG0 + PPG1/PPG2 + PPG3/PPG4 + PPG5 down counters are decremented from "0000<sub>H</sub>" to "FFFF<sub>H</sub>", an underflow occurs. When an underflow occurs, the underflow generation bits in the two channels are set at one time (PIF0 = 1, PIF1 = 1).
  - When an underflow occurs with either of the two channels of the interrupt requests enabled (PIE0 = 0 + PIE1 = 1, PIE0 = 1 + PIE1 = 0), an interrupt is generated.
  - To prevent duplication of interrupt requests, disable either of the two channels of the underflow interrupt enable bits (PIE0 = 0 + PIE1 = 1, PIE0 = 1 + PIE1 = 0).
  - When the two channels of the underflow generation flag bits are set (PUF0 = 1, PUF1 = 1), clear the two channels at the same time.

## ■ Interrupt of 8/16-bit PPG Timer, DMA Transfer, and EI<sup>2</sup>OS

Table 15.4-2 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 15.4-2 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source                      | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|---------------------------------------|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|                                       |                          |                      | Number           | Address             | Number                     | Address             |
| PPG0/PPG1 counter borrow <sup>*</sup> | ×                        | ×                    | #22              | FFFFA4 <sub>H</sub> | ICR05                      | 0000B5 <sub>H</sub> |
| PPG2/PPG3 counter borrow              | ×                        | ×                    | #23              | FFFFA0 <sub>H</sub> | ICR06                      | 0000B6 <sub>H</sub> |
| PPG4/PPG5 counter borrow              | ×                        | ×                    | #24              | FFFF9C <sub>H</sub> |                            |                     |

×: Interrupt request flag is not cleared.

\*: This interrupt source shares the interrupt source and interrupt number of other peripheral function.  
For details, see Table 3.2-2.

### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing.

## ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The 8/16-bit PPG timer does not correspond to the DMA transfer function and EI<sup>2</sup>OS function.

## 15.5 Operations of 8/16-Bit PPG Timer

The 8/16-bit PPG timer contains an 8-bit PPG unit for six channels (PPG0/PPG1, PPG2/PPG3, PPG4/PPG5). In addition to independent operation mode, the channels can also be used in direct connection mode (PPG0 + PPG1, PPG2 + PPG3, and PPG4 + PPG5). In total, three types of operation modes are therefore supported: independent operation mode, 8-bit prescaler + 8-bit PPG mode and 16-bit PPG mode.

### ■ Outline of 8/16-bit PPG Timer Operation

Each 8-bit PPG unit has two 8-bit reload registers, one "L" side and the other "H" side register (PRL, PRLH).

The values in these registers are reloaded into the corresponding "L"/"H" sides of the 8-bit down counter (PCNT) alternately and decremented at every count clock. The value of the output pin is inverted at reloading when a counter borrow occurs. This operation ensures that the output pin outputs pulses with an "L"/"H" width that corresponds to the reload register values.

Operation start or restart are initiated by setting the corresponding bit in the register.

The relationship between the reload operation and pulse output is shown below.

| Reload operation | Pin output change    |
|------------------|----------------------|
| PRLH → PCNT      | PPG0/1[0 → 1] rising |

If bit4 (PIE0) of PPGC0 register is set to "1" and bit12 (PIE1) of PPGC1 is set to "1", 00<sub>H</sub> → FF<sub>H</sub> counter borrow for each counter (in 16-bit PPG mode, "0000<sub>H</sub>" to "FFFF<sub>H</sub>" counter borrow) will cause an interrupt request.

### ■ Operation Mode

The 8/16-bit PPG timer has three operation modes: two-channel independent mode, 8-bit prescaler/8-bit PPG mode, and 16-bit PPG mode (the MB90480/485 series has three channels per mode). Two-channel independent mode allows the two channels to be used independently as 8-bit PPGs. The PPG0 pin is connected to the PPG output of ch.0, and the PPG1 pin is connected to the PPG output of ch.1 (PPG2 to PPG5 correspond to ch.2 to ch.5, respectively).

8-bit prescaler/8-bit PPG mode is a mode in which ch.0 (ch.2 or ch.4) operates as 8-bit prescaler while ch.1 (ch.3 or ch.5) is counted with a borrow output of ch.0 (ch.2 or ch.4), which allows an 8-bit PPG waveform of an arbitrary interval to be output. The PPG0 (PPG2 or PPG4) pin is connected with the prescaler output of ch.0 (ch.2 or ch.4), and the PPG1 pin is connected with the PPG output of ch.1 (ch.3 or ch.5).

16-bit PPG1 channel mode (the MB90480/485 series has 3 channels) is an operation mode in which ch.0 and ch.1 are directly connected (direct connection between ch.2 and ch.3 respectively with ch.4 and ch.5) to allow 16-bit PPG operation. Both PPG 0 and PPG1 are connected with the 16-bit PPG output.

## ■ PPG Output Operation

For the 8/16-bit PPG timer, PPG operation of ch.0 (ch.2 or ch.4) is started by setting bit7 of the PPGC0 register (PEN0) to "1". Similarly, PPG operation of ch.1 (ch.3 or ch.5) is started by setting bit15 of the PPGC1 register (PEN1) to "1" to start counting. By subsequently setting bit7 of the PPGC0 register (PEN0) or bit15 of the PPGC1 register (PEN1) to "0", the count operation is stopped, and the pulse output level is fixed at "L" level.

In 8-bit prescaler/8-bit PPG mode, do not set ch.0 (ch.2 or ch.4) in stop mode and ch.1 (ch.3 or ch.5) to active mode.

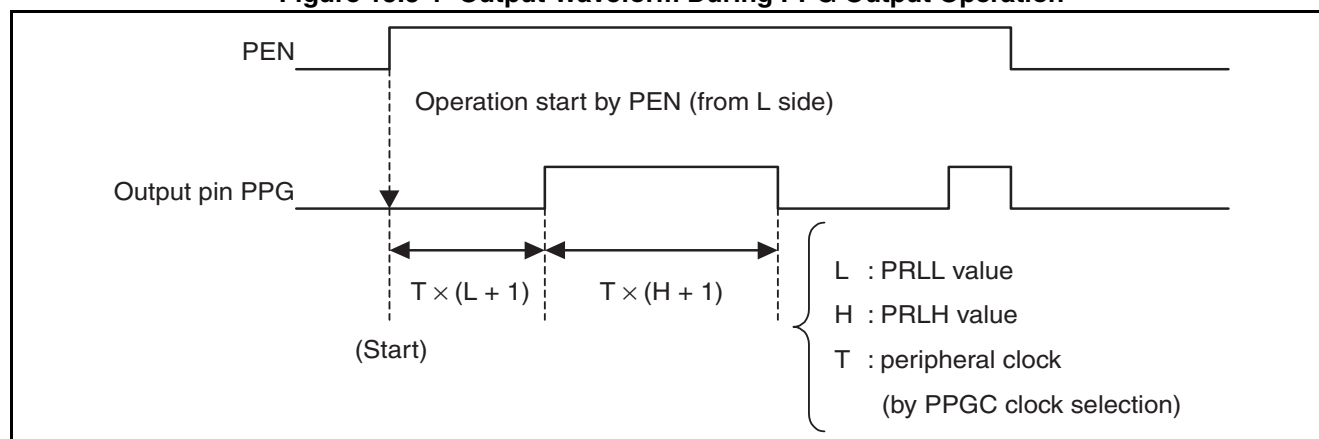
In 16-bit PPG mode, use bit7 of the PPGC0 register (PEN0) and bit15 of the PPGC1 register (PEN1) to control simultaneous start or stop of operation.

In the following, the operation for PPG output is described.

During PPG operation, a pulse wave with an arbitrary interval and duty ratio (ratio of "H" level pulse wave to "L" level pulse wave) is repeatedly output. After that, the PPG will not stop until operation stop is specified.

Figure 15.5-1 shows the output waveform during PPG output operation.

**Figure 15.5-1 Output Waveform During PPG Output Operation**



## ■ Relationship Between Reload Value and Pulse Width

The width of the output pulse can be calculated by adding 1 to the reload register value, and multiplying the result by the count clock interval. In other words, if the reload register value during 8-bit PPG operation is "00<sub>H</sub>", or that in 16-bit PPG operation is "0000<sub>H</sub>", the pulse width will be equal to one interval length of the count clock. If the reload register value during 8-bit PPG operation is "FF<sub>H</sub>", the pulse width is equal to 256 intervals of the count clock, and if the reload register value during 16-bit PPG operation is "FFFF<sub>H</sub>" the pulse width is equal to 65,536 intervals of the count clock. The pulse width is expressed with the formula below:

$$P_L = T \times (L + 1)$$

$$P_H = T \times (H + 1)$$

$P_L$  : Width of "L" pulse

$P_H$  : Width of "H" pulse

T : Input clock interval

L : PRL value

H : PRLH value

## ■ Selection of Count Clock

The 8/16-bit PPG timer uses the input from the peripheral clock and time-base counter as a counter clock, allowing a selection from six types of count clock input.

Bit4 to bit2 of the PPG01/PPG23/PPG45 registers (PCM2 to 0) are used to select the clock of ch.0 (ch.2 or ch.4), and bit7 to bit5 of the PPG01/PPG23/PPG45 registers (PCS2 to PCS0) are used to select the clock of ch.1 (ch.3, or channel5).

The clock is selected from among the machine clock (multiplied by 1/16 to 1) or the time-base counter.

### Notes:

- In 8-bit prescaler/8-bit PPG mode and in 16-bit PPG mode, the value in bit14 of the PPGC1 register (PCS1) is invalid.
- If the input from time-base timer is used, the first count cycle after a trigger or stop event may be out of sync. If the time-base counter is initialized while the 8/16-bit PPG timer is running, cycles may be out of sync.
- If, in 8-bit prescaler/8-bit PPG mode, ch.0/2/4 is in active mode and ch.1/3/5 is in stopped mode, the first count cycle may be out of sync when operation of ch.1 (ch.3 or ch.5) starts.

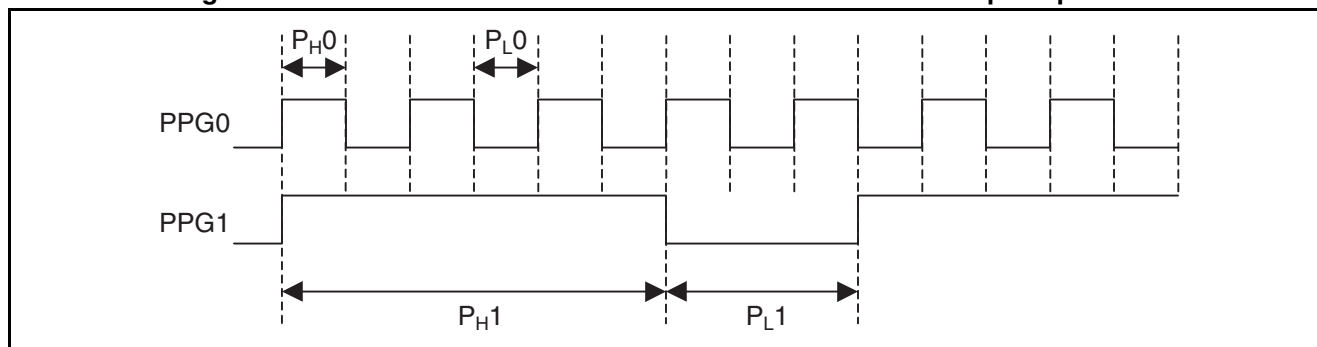
## ■ Pin Output Control of Pulses

Pulses generated by the 8/16-bit PPG timer output from the external pins (PPG0 to PPG5). To output pulses from an external pin, set the bit corresponding to the pin to "1". For enabling PPG0/PPG2/PPG4 pin output, bit5 of PPGC0 (PE0) is used, and for enabling PPG1/PPG3/PPG5 pin output, bit3 of PPGC1 (PE1) is used. If the respective bit is set to "0" (initial value), the external pin does not output pulses, but is used as a general-purpose port.

In 16-bit PPG mode, PPG0 to PPG5 output the same waveform. For this reason, it is sufficient to enable pin output for either of the corresponding pins to obtain the same output.

In 8-bit prescaler + 8-bit PPG mode, PPG0/PPG2/PPG4 output a toggle waveform of the 8-bit prescaler, and PPG1/PPG3/PPG5 output a waveform of 8-bit PPG.

Output waveform in this mode is illustrated in Figure 15.5-2.

**Figure 15.5-2 Waveform in 8-bit Prescaler + 8-bit PPG Mode Output Operation**

The pulse width shown in Figure 15.5-2 can be expressed with the following formulas.

$$P_{L0} = T \times (L0 + 1)$$

$$P_{H0} = T \times (L0 + 1)$$

$$P_{L1} = T \times (L0 + 1) \times (L1 + 1)$$

$$P_{H1} = T \times (L0 + 1) \times (H1 + 1)$$

Where

L0: PRL0 value of ch.0 and PRLH value of ch.1

L1: PRL0 value of ch.1

H1: PRLH value of ch.1

T: Input clock cycle

P<sub>H0</sub>: "H" pulse width of PPG0

P<sub>L0</sub>: "L" pulse width of PPG0

P<sub>H1</sub>: "H" pulse width of PPG1

P<sub>L1</sub>: "L" pulse width of PPG1

---

Note:

Set PRL0 of ch.0 and PRLH of ch.1 to the same value.

---

## ■ Interrupts of the 8/16-bit PPG Timer

The interrupt unit of the 8/16-bit PPG timer becomes active as soon as a counter borrow occurs after the reload value is counted out. In 8-bit PPG2 channel mode or 8-bit prescaler/9-bit PPG mode (3 channels are provided for MB90480/485 series), each borrow will cause a separate interrupt request. In 16-bit PPG mode, however, PUG0 and PUG1 will be set at the same time when a borrow from the 16-bit counter occurs. To unify interrupt sources, only one of either PIE0 or PIE1 is allowed. Interrupt sources are also cleared at the same time for PUF0 and PUF1.

## ■ Initial Value of Hardware Components

The hardware components of the 8/16-bit PPG timer are initialized to the following values at reset.

|                       |       |   |                        |
|-----------------------|-------|---|------------------------|
| < Registers >         | PPG0  | → | 0X000001               |
|                       | PPG1  | → | 00000001               |
|                       | PPG01 | → | XXXXXX00               |
| < pulse output >      | PPG0  | → | "L"                    |
|                       | PPG1  | → | "L"                    |
|                       | PE0   | → | PPG0 output prohibited |
|                       | PE1   | → | PPG1 output prohibited |
| < interrupt request > | IRQ0  | → | "L"                    |
|                       | IRQ1  | → | "L"                    |

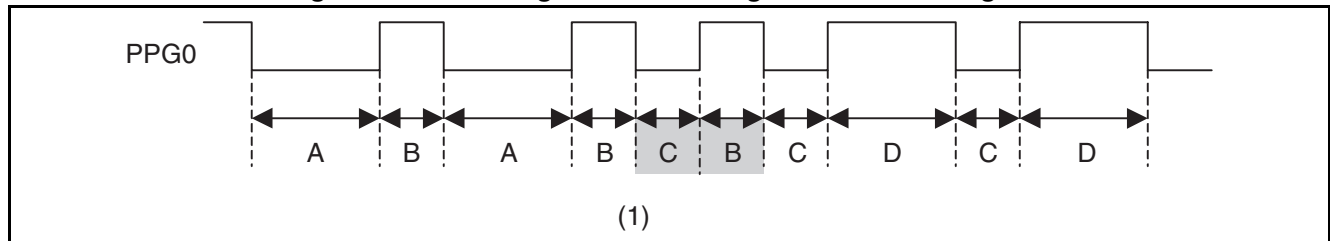
Hardware components other than above mentioned are not initialized.

## ■ Write Timing to the Reload Register

It is recommended that word transfer instructions be used for writing data to the reload registers PRL and PRLH in modes other than 16-bit PPG mode. Writing data to the register two times by separate byte transfer instructions may result in an unexpected output pulse width, depending on the timing.

Figure 15.5-3 shows the timing for writing to the reload register.

**Figure 15.5-3 Timing Chart of Writing to the Reload Register**

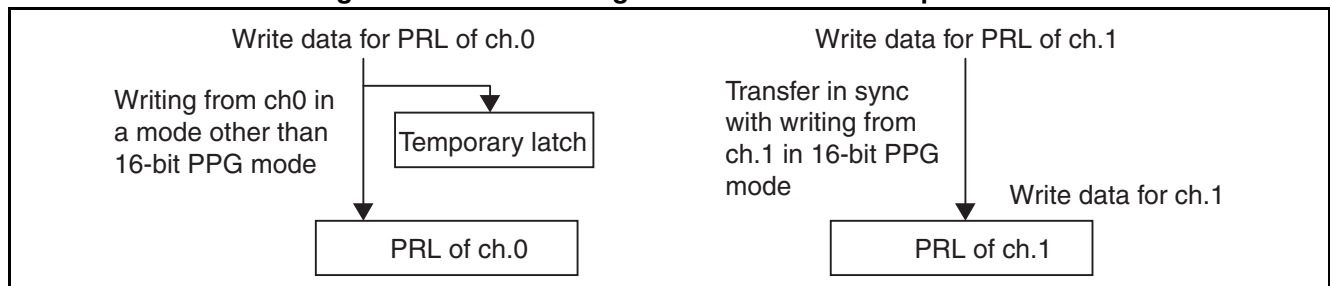


In Figure 15.5-3, PRL is changed from A to C before (1), and the PRLH value is changed from B to D after (1). However, since the PRL values at (1) are PRL = C and PRLH = B, the pulses for the "L" side count of C and the "H" side count of B are generated only once. Similarly, to write data to the PRL of ch.0/2/4 and ch.1/3/5 in the 16-bit PPG mode, use a long word transfer instruction or use a word transfer instruction in the order ch.0 → ch.1 (ch.2 → ch.3, ch.4 → ch.5 respectively). In this mode, data is temporarily written from ch.0/2/4 to the PRL; when data is then written from ch.1/3/5 to the PRL, it is actually written to the PRL of ch.0.

In modes other than 16-bit PPG mode, writing to ch.0/2/4 and ch.1/3/5 are performed independently.

Figure 15.5-4 shows a block diagram of the PRL write operation.

**Figure 15.5-4 Block Diagram of the PRL Write Operation**





## 15.6 Program Example of 8/16-Bit PPG Timer

This section describes the program example of the 8/16-bit PPG timer.

### ■ Program Example of 8/16-bit PPG Timer

| Example of setting procedure  | Program example         |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
|---|-------------------------|-------------------------|----------------------|-------|-----------------------|---------|----------------------|--------|------------------------|-------|---------------------|-------|--------------------------|--------|----------------|------|---------------------------|-------|------------|-------|--|-------------------------|------------------|-------------|-------------|-------------|----------------|--|------------------------------|-------------|---|
| <p>Generate interval interrupt and output PPG.<br/>(PPG output from PPG1 pin, software trigger)</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Control PPG01 <table border="1"> <thead> <tr> <th></th><th>Register name. bit name</th></tr> </thead> <tbody> <tr> <td>Set control register</td><td>PPG01</td></tr> <tr> <td>Select count clock &gt;&gt;</td><td>.PCS2-0</td></tr> <tr> <td>Set control register</td><td>PPGC01</td></tr> <tr> <td>Enable pulse output &gt;&gt;</td><td>.PE10</td></tr> <tr> <td>Enable interrupt &gt;&gt;</td><td>.PIE1</td></tr> <tr> <td>Select operation mode &gt;&gt;</td><td>.MD1-0</td></tr> </tbody> </table> </li> </ul> <p>• Set duty</p> <table border="1"> <tr> <td>Set PPG01 duty</td><td>PRL0</td></tr> </table> <p>• Interrupt related</p> <table border="1"> <tr> <td>Set PPG01 interrupt level</td><td>ICR05</td></tr> <tr> <td>Set I flag</td><td>(CCR)</td></tr> </table> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start PPG01 <table border="1"> <thead> <tr> <th></th><th>Register name. bit name</th></tr> </thead> <tbody> <tr> <td>Enable interrupt</td><td>PPGC01.PIE1</td></tr> <tr> <td>Start PPG01</td><td>PPGC01.PEN1</td></tr> </tbody> </table> </li> </ul> <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Interrupt processing <table border="1"> <tr> <td>Any processing</td><td></td></tr> <tr> <td>Clear interrupt request flag</td><td>PPGC01.PUF1</td></tr> </table> </li> </ul> <p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>&lt;Other&gt;</p> <p>Note:<br/>Setting related to clock and setting of _set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</p> |                         | Register name. bit name | Set control register | PPG01 | Select count clock >> | .PCS2-0 | Set control register | PPGC01 | Enable pulse output >> | .PE10 | Enable interrupt >> | .PIE1 | Select operation mode >> | .MD1-0 | Set PPG01 duty | PRL0 | Set PPG01 interrupt level | ICR05 | Set I flag | (CCR) |  | Register name. bit name | Enable interrupt | PPGC01.PIE1 | Start PPG01 | PPGC01.PEN1 | Any processing |  | Clear interrupt request flag | PPGC01.PUF1 | <pre> void PPG_sample(void) {     PPG01_initial();     PPG01_start(); }  void PPG01_initial(void) {     /*     *1     IO_PPG01.bit.PCS = 1; /* bit7-5 = 0    Select PPG1 count clock */     /*     *2     IO_PPGC01.word = 0x3101; /* bit15 = 0    Stop PPG operation */                                 /* bit13 = 1    Enable PPG1/3/5 pulse output */                                 /* bit12 = 1    Enable PPG interrupt */                                 /* bit11 = 0    Clear PUF1 interrupt request flag */                                 /* bit10-9 = 00    8-bit PPG 2ch */                                 /* bit8 = 1    Reserved bit */                                 /* bit0 = 1    Reserved bit */     *3     IO_PRL0.word = 0x7f7f; /* Set PPG duty ratio */      IO_ICR05.byte = 0x10; /* Interrupt level (arbitrary value) */     __EI(); /* Enable interrupt */ }  void PPG01_start(void) {     IO_PPGC01.bit.PIE1 = 1; /* bit12 = 1    Enable PIE1 interrupt request */     IO_PPGC01.bit.PEN1 = 1; /* bit15 = 1    Start PEN1 PPG operation */ }  __interrupt void PPG01_int(void) {     /* Any processing */     IO_PPGC01.bit.PUF1 = 0; /* bit11 = 0    PUF1 interrupt request flag */ }  #pragma intvect PPG01_int 22 </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
|   | Register name. bit name |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Set control register  | PPG01                   |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Select count clock >>   | .PCS2-0                 |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Set control register  | PPGC01                  |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Enable pulse output >>  | .PE10                   |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Enable interrupt >>   | .PIE1                   |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Select operation mode >>  | .MD1-0                  |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Set PPG01 duty  | PRL0                    |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Set PPG01 interrupt level   | ICR05                   |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Set I flag  | (CCR)                   |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
|   | Register name. bit name |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Enable interrupt  | PPGC01.PIE1             |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Start PPG01   | PPGC01.PEN1             |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Any processing  |                         |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |
| Clear interrupt request flag  | PPGC01.PUF1             |                         |                      |       |                       |         |                      |        |                        |       |                     |       |                          |        |                |      |                           |       |            |       |  |                         |                  |             |             |             |                |  |                              |             |   |

\*1: io\_PPG01 represents PPG1 register and PPG0 register.

\*2: io\_PPGC01 represents PPGC1 register and PPGC0 register.

\*3: io\_PRL0 represents PRL0 register and PRLH0 register.

## ■ Setting Method Other than Program Example

### ● Method to enable/stop PPG operation

Set by the PPG operation enable bit (PPG01/PPG23/PPG45.PEN0 or 1).

| Control                 | PPG operation enable bit (PEN0 or PEN1) |
|-------------------------|---|
| To stop PPG operation   | Set to "0"                              |
| To enable PPG operation | Set to "1"                              |

Enable the PPG operation before the PPG is started.

### ● Method to set PPG operation mode

Set by the mode selection bits (PPGC01.MD[1:0]/PPGC23.MD[1:0]/PPGC45.MD[1:0]).

### ● Type of count clock and selection method

Ch.1, ch.3, ch.5 can select using the count clock selection bits (PPG01.PCS[2:0]/PPG23.PCS[2:0]/PPG45.PCS[2:0]).

Ch.0, ch.2, ch.4 can select using the count clock selection bits (PPG01.PCM[2:0]/PPG23.PCM[2:0]/PPG45.PCM[2:0]).

### ● Interrupt related register

The relationship between channel, interrupt level, and interrupt vector is shown in the following table.

For details of the interrupt level and interrupt vector, see CHAPTER 3 INTERRUPT.

| Channel | Interrupt vector                     | Interrupt level setting register                                       |
|---------|--------------------------------------|--|
| PPG0    | #22<br>Address : FFFFA4 <sub>H</sub> | Interrupt control register 05 (ICR05)<br>Address : 0000B5 <sub>H</sub> |
| PPG1    |                                      |  |
| PPG2    | #23<br>Address : FFFFA0 <sub>H</sub> | Interrupt control register 06 (ICR06)<br>Address : 0000B6 <sub>H</sub> |
| PPG3    |                                      |  |
| PPG4    | #24<br>Address : FFFF9C <sub>H</sub> | Interrupt control register 06 (ICR06)<br>Address : 0000B6 <sub>H</sub> |
| PPG5    |                                      |  |

Clear the interrupt request flag (PPG01/PPG23/PPG45.PUF0 or PUF1) with software before returning from the interrupt processing because the flag is not cleared automatically (write "0" to PUF0 or PUF1 bit).

### ● Type of interrupt

One interrupt is provided. Caused by underflow of the PPG counter.

● Method to enable/disable/clear interrupt

Enabling/disabling interrupt is set by the interrupt request enable bit (PPG01/PPG23/PPG45. PIE0 or PIE1).

| Content of control           | Interrupt request enable bit (PIE0 or PIE1) |
|------------------------------|---|
| To disable interrupt request | Set to "0"                                  |
| To enable interrupt request  | Set to "1"                                  |

Clearing interrupt request is set by the interrupt request bit (PPG01/PPG23/PPG45.PUF0 or PUF1).

| Content of control         | Interrupt request bit (PUF0 or PUF1) |
|----------------------------|--------------------------------------|
| To clear interrupt request | Write "0"                            |

# CHAPTER 16 DTP/EXTERNAL INTERRUPTS

---

**This chapter provides an overview of the DTP/external interrupt unit, explains configuration and functions of its registers and its operation, shows the precautions on use.**

---

- 16.1 Overview of DTP/External Interrupt Unit
- 16.2 Configuration and Functions of DTP/External Interrupt Unit Registers
- 16.3 DTP/External Interrupt
- 16.4 Operations of DTP/External Interrupt Unit
- 16.5 Precautions on Use of DTP/External Interrupt Unit
- 16.6 Program Example of DTP/External Interrupt

## 16.1 Overview of DTP/External Interrupt Unit

The DTP (Data Transfer Peripheral) unit is a peripheral control section located between the peripheral units outside the device and the F<sup>2</sup>MC-16LX CPU. It is used to receive DMA request or interrupt requests from the external peripheral device and report such requests to the F<sup>2</sup>MC-16LX CPU to start  $\mu$ DMAC, EI<sup>2</sup>OS, or interrupt handling.

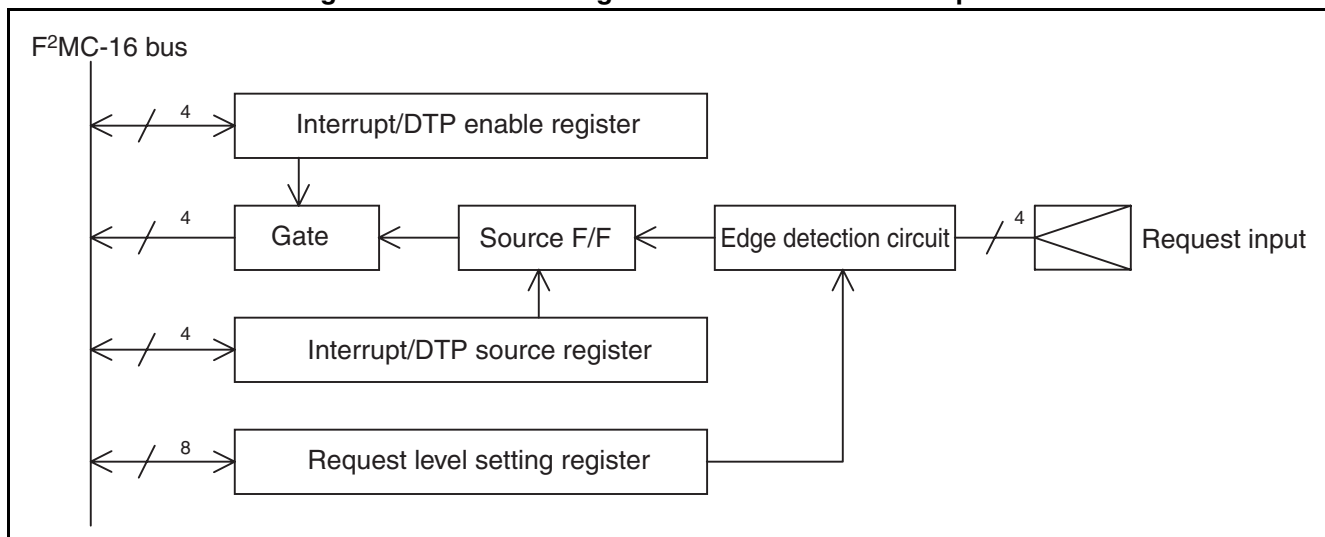
### ■ Overview of DTP/external Interrupt Unit

For  $\mu$ DMAC or EI<sup>2</sup>OS, the request level can be selected from two types, "H" and "L". For external interrupt requests, it can be selected from four types: rising, falling edges, "H" and "L" signals.

### ■ Block Diagram of DTP/external Interrupt Unit

Figure 16.1-1 shows a block diagram of the DTP/external interrupt.

Figure 16.1-1 Block Diagram of DTP/external Interrupt Unit



### ■ Pin Related to DTP/external Interrupt

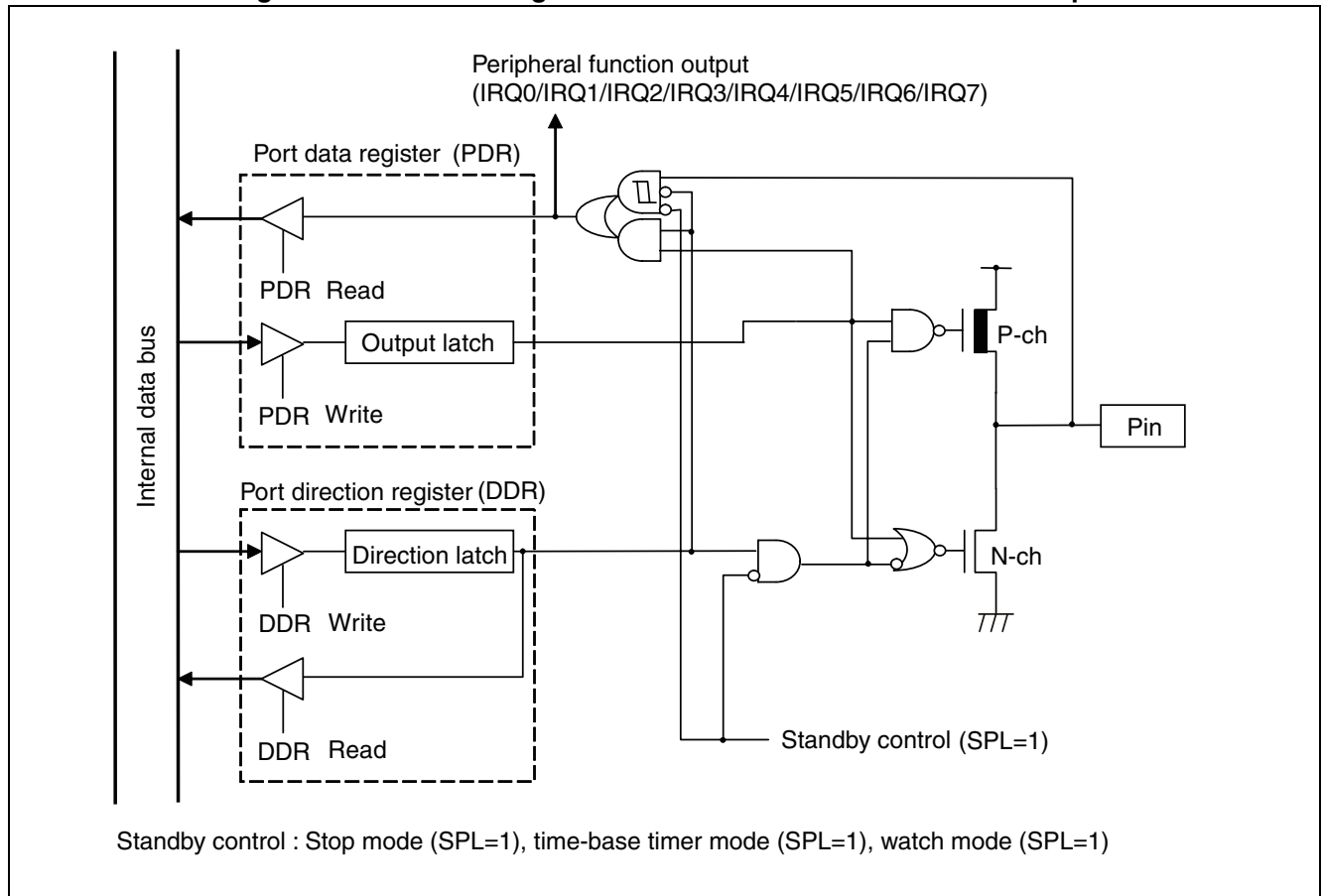
The pin related to the external interrupt pin has the IRQ0/IRQ1/IRQ2/IRQ3/IRQ4/IRQ5/IRQ6/IRQ7 pins and functions as an input port. The IRQ0/IRQ1/IRQ2/IRQ3/IRQ4/IRQ5/IRQ6/IRQ7 pins function as the general-purpose I/O port (P80/IRQ0, P81/IRQ1, P82/IRQ2, P83/IRQ3, P84/IRQ4, P85/IRQ5, P86/IRQ6, P87/IRQ7) and external interrupt input pin.

#### ● Setting when using as IRQ0/IRQ1/IRQ2/IRQ3/IRQ4/IRQ5/IRQ6/IRQ7 pins

When the P80/IRQ0, P81/IRQ1, P82/IRQ2, P83/IRQ3, P84/IRQ4, P85/IRQ5, P86/IRQ6, P87/IRQ7 pins are used as an input pin, be sure to set the port direction register to the input port (DDR8: bit0 to 7 → "0").

## ■ Block Diagram of Pin Related to DTP/external Interrupt

Figure 16.1-2 Block Diagram of Pin Related to DTP/external Interrupt



# 16.2 Configuration and Functions of DTP/External Interrupt Unit Registers

This section describes the configuration and functions of the registers used in the DTP/external interrupt unit.

## ■ List of Registers for DTP/external Interrupt Unit

Figure 16.2-1 shows a list of the registers for the DTP/external interrupt unit.

Figure 16.2-1 List of DTP/external Interrupt Unit Registers

|                              |     |     |     |     |     |     |     |     |     |                                       |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------------------------------------|
|                              | bit | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |                                       |
| Address: 00000C <sub>H</sub> |     | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 | Interrupt/DTP enable register (ENIR)  |
|                              | bit | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   |                                       |
| Address: 00000D <sub>H</sub> |     | ER7 | ER6 | ER5 | ER4 | ER3 | ER2 | ER1 | ER0 | Interrupt/DTP source register (EIRR)  |
|                              | bit | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |                                       |
| Address: 00000E <sub>H</sub> |     | LB3 | LA3 | LB2 | LA2 | LB1 | LA1 | LB0 | LA0 | Request level setting register (ELVR) |
|                              | bit | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   |                                       |
| Address: 00000F <sub>H</sub> |     | LB7 | LA7 | LB6 | LA6 | LB5 | LA5 | LB4 | LA4 | Request level setting register (ELVR) |

## ■ Interrupt/DTP Enable Register (ENIR: Enable Interrupt Request Register)

The bit configuration of the interrupt/DTP enable register (ENIR) is shown below.

|                              |     |     |     |     |     |     |     |     |     |                       |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
| ENIR                         | bit | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Initial value         |
| Address: 00000C <sub>H</sub> |     | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 | 00000000 <sub>B</sub> |
|                              |     | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |                       |

The interrupt/DTP enable register (ENIR) enables or disables an external interrupt/DTP request for an external interrupt/DTP channel.

If the interrupt/DTP enable bits (ENs) of ENIR and the interrupt/DTP request flag bits (ENs) of EIRR are all set to "1", an interrupt request for the corresponding interrupt/DTP pin is generated. Signal inputs to this register are not interrupted during standby mode.

Note:

Please clear DTP/external interrupt factor bit (EIRR: ER) corresponding to immediately before permitting DTP/external interrupt.

## ■ Interrupt/DTP Source Register (EIRR: External Interrupt Request Register)

The bit configuration of the interrupt/DTP source register (EIRR) is shown below.

|                              |     |     |     |     |     |     |     |     |     |                       |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
| EIRR                         | bit | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | Initial value         |
| Address: 00000D <sub>H</sub> |     | ER7 | ER6 | ER5 | ER4 | ER3 | ER2 | ER1 | ER0 | XXXXXXXX <sub>B</sub> |
|                              |     | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |                       |

The interrupt/DTP source register (EIRR) is set to "1" if the edge or level signal set in the detection condition selection bits (LB, LA) of the request level setting register (ELVR) is input to the external interrupt pin.

If the register is set to "1", an interrupt request for the corresponding interrupt/DTP channel is generated when the interrupt/DTP request enable bits (EN) of ENIR are set to "1".

If the register is set to "0", it is cleared.

If the register is set to "1", the interrupt request status is not affected.

### Notes:

- Reading by read-modify-write type instructions always read "1". If multiple external interrupt request outputs are enabled (ENIR: EN7 to EN0=1), only the bits for which the CPU accepts an interrupt (bits for which "1" was set in EN7 to EN0) are cleared to "0". No other bits must be cleared unconditionally.
- When corresponding DTP/external interrupt enable bit (ENIR: EN) is set to "1", the value of the DTP/external interrupt factor bit (EIRR: ER) is valid.  
When the DTP/external interrupt has disabled (ENIR: EN=0), the DTP/external interrupt factor bit might be set regardless of the existence of the DTP/external interrupt factor.
- Please clear DTP/external interrupt factor bit (EIRR: ER) corresponding to immediately before permitting DTP/external interrupt.

## ■ Request Level Setting Register (ELVR: External Level Register)

The bit configuration of the request level setting register (ELVR) is shown below.

|                              |     |     |     |     |     |     |     |     |     |                       |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
|                              | bit | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Initial value         |
| Address: 00000E <sub>H</sub> |     | LB3 | LA3 | LB2 | LA2 | LB1 | LA1 | LB0 | LA0 | 00000000 <sub>B</sub> |
|                              |     | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |                       |
|                              | bit | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | Initial value         |
| Address: 00000F <sub>H</sub> |     | LB7 | LA7 | LB6 | LA6 | LB5 | LA5 | LB4 | LA4 | 00000000 <sub>B</sub> |
|                              |     | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |                       |

The request level setting register (ELVR) is used to select a request detection level. Two bits are assigned for each pin, as shown in Table 16.2-1. If the setting for a request input indicates a level, the corresponding level will be set again when it is cleared, provided the input is active.

**Table 16.2-1 ELVR Assignment (LA0 to LA7, LB0 to LB7)**

| LBx | LAx | Operation               |
|-----|-----|-------------------------|
| 0   | 0   | Request by "L" level    |
| 0   | 1   | Request by "H" level    |
| 1   | 0   | Request by rising edge  |
| 1   | 1   | Request by falling edge |



## 16.3 DTP/External Interrupt

The interrupt related to the DTP/external interrupt occurs when the edge or level input to input pin is detected.

The DTP/external interrupt can activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ DTP/External Interrupt

The interrupt control bit and interrupt source of the DTP/external interrupt is shown in the following table.

|                                     | External interrupt<br>When ISE of ICR = 0 | DTP interrupt when ISE<br>of ICR = 1 |
|-------------------------------------|---|--------------------------------------|
| Interrupt request flag              | EIRR:ER (bit8 to 15)                      | EIRR:ER (bit8 to 15)                 |
| Interrupt request output enable bit | ENIR:EN (bit0 to 7)                       | ENIR:EN (bit0 to 7)                  |
| Interrupt generation source         | Detect external interrupt                 | Detect external interrupt            |

#### ○ Setting procedure

To use the DTP/external interrupt, set each register by using the following procedures:

1. Set the interrupt request enable bit corresponding to the DTP/external interrupt channel to be used to "0" (ENIR:EN).
2. Use the detection condition select bit corresponding to the DTP/external interrupt channel to be used to set the edge or level to be detected (ELVR:LA/LB).
3. Set the interrupt request flag corresponding to the DTP/external interrupt channel to be used to "0" (EIRR:ER).
4. Set the corresponding interrupt request enable bit to "1" (ENIR:EN).

#### Notes:

- When setting the registers for the DTP/external interrupt, the external interrupt request must be disabled (ENIR:EN=0).
- When enabling the DTP/external interrupt (ENIR:EN=1), the corresponding DTP/external interrupt request flag bit must be cleared in advance (EIRR:ER=0). These actions prevent the mistaken interrupt request from occurring when setting the register.

### ○ Selecting of DTP function or external interrupt function

Whether the DTP function or the external interrupt function is executed depends on the setting of the EI<sup>2</sup>OS enable bit in the corresponding interrupt control register (ICR:ISE) or that of the DMA enable register (DER:EN).

If the ISE bit is set to "1", the EI<sup>2</sup>OS sets the EN bit to "1" and the DTP transfer is enabled.

If the ISE and EN bits are set to "0", the EI<sup>2</sup>OS and DMA transfer are disabled and the external interrupt function is executed.

Notes:

- All interrupt requests assigned to one interrupt control register have the same interrupt level (IL2 to IL0).
- If two or more interrupt requests are assigned to one interrupt control register and EI<sup>2</sup>OS is started for any of them, other interrupt requests cannot be used.

## ■ DTP/external Interrupt, DMA Transfer, and EI<sup>2</sup>OS

Table 16.3-1 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 16.3-1 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|------------------|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|                  |                          |                      | Number           | Address             | Number                     | Address             |
| INT0 (IRQ0)      | ○                        | 0                    | #11              | FFFFD0 <sub>H</sub> | ICR00                      | 0000B0 <sub>H</sub> |
| INT1 (IRQ1)      | ○                        | ×                    | #12              | FFFFCC <sub>H</sub> |                            |                     |
| INT2 (IRQ2)      | ○                        | ×                    | #13              | FFFFC8 <sub>H</sub> | ICR01                      | 0000B1 <sub>H</sub> |
| INT3 (IRQ3)      | ○                        | ×                    | #14              | FFFFC4 <sub>H</sub> |                            |                     |
| INT4 (IRQ4)      | ○                        | ×                    | #15              | FFFFC0 <sub>H</sub> | ICR02                      | 0000B2 <sub>H</sub> |
| INT5 (IRQ5)      | ○                        | ×                    | #16              | FFFFBC <sub>H</sub> |                            |                     |
| INT6 (IRQ6)      | ○                        | ×                    | #17              | FFFFB8 <sub>H</sub> | ICR03                      | 0000B3 <sub>H</sub> |
| INT7 (IRQ7)      | ○                        | ×                    | #18              | FFFFB4 <sub>H</sub> |                            |                     |

×: Interrupt request flag is not cleared.

○: Interrupt request flag is cleared.

Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to "0" to execute the software polling processing.

## ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

EI<sup>2</sup>OS function and ch.0 in the DTP/external interrupt correspond to the DMA transfer function.

When the DMA or EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 16.4 Operations of DTP/External Interrupt Unit

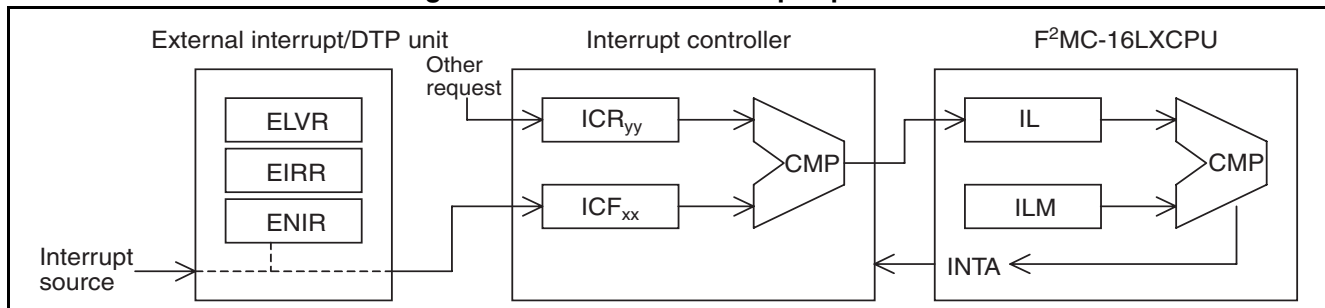
This section describes the operations of the DTP/external interrupt unit.

### ■ Operation of External Interrupt Unit

If, after an external interrupt request has been set, the interrupt request specified in the ELVR register is input to the corresponding pin, this resource will generate an interrupt request signal for the interrupt controller. Interrupts that are generated by the interrupt controller at the same time will be distinguished by priority. The interrupt controller will generate an interrupt request to the F<sup>2</sup>MC-16LX CPU if the interrupt from the corresponding resource has the highest priority. The F<sup>2</sup>MC-16LX CPU compares the interrupt level mask register (ILM) in the processor status (PS) with the interrupt request level. If the request level is found to be higher than the value expressed by the ILM bits, the hardware interrupt handling microprogram starts immediately after the currently executed instruction is completed.

Figure 16.4-1 shows the operational flow for external interrupts.

**Figure 16.4-1 External Interrupt Operation**



The interrupt handling microprogram reads data from the interrupt vector area and generates an interrupt acknowledge signal for the interrupt controller. After that, it transfers the jump destination address of the macro instruction, which is obtained from the interrupt vector, to the program counter, and execution continues with the user's interrupt handling program.

## ■ DTP Operation

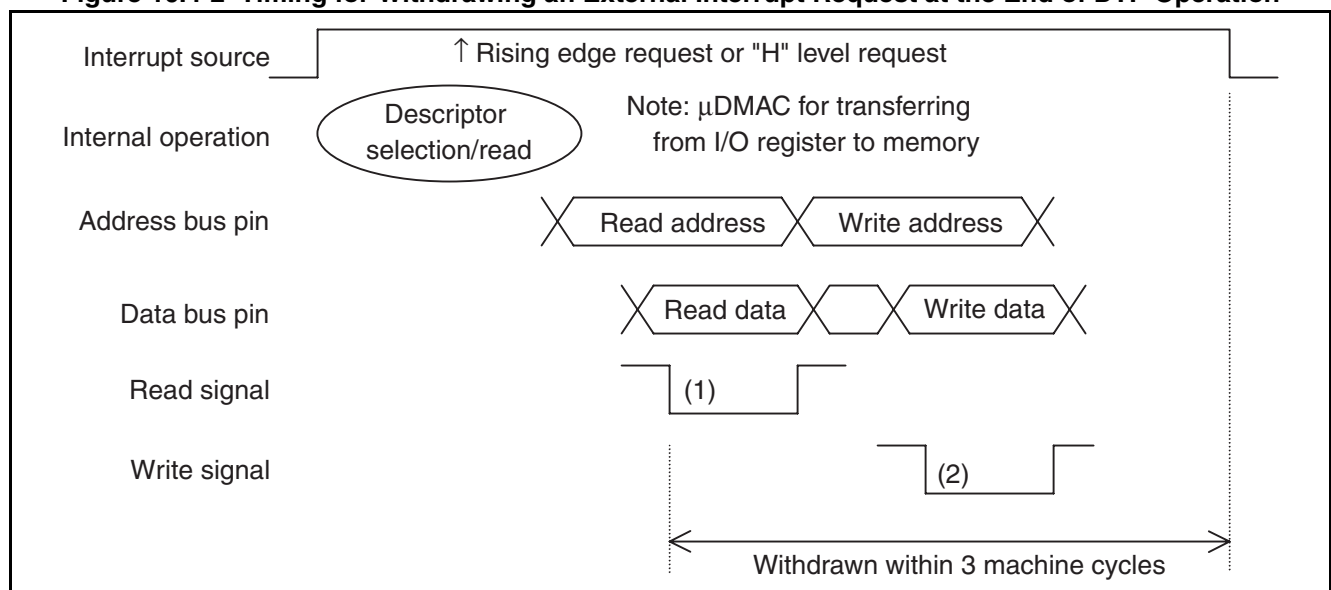
To start  $\mu$ DMAC in a user program, the following initialization operations are performed: The I/O address pointer in the  $\mu$ DMAC descriptor is set to the register address allocated in 000000<sub>H</sub> to 0000FF<sub>H</sub>, and the buffer address pointer is set to the start address of the memory buffer.

The operational sequence for DTP is almost the same as that for external interrupts. At the start of  $\mu$ DMAC, the corresponding read or write signal is transferred to the external peripheral device whose address was specified, and data transfer is performed with this chip. Ensure that the external peripheral device side is required to withdraw the interrupt request within three machine cycles after data transfer. At the end of data transfer, the descriptor is updated, and a signal to clear the interrupt source is generated by the interrupt controller. After receiving the signal, the DTP unit clears the flip-flop that retains the interrupt source and waits for the next request from the pin.

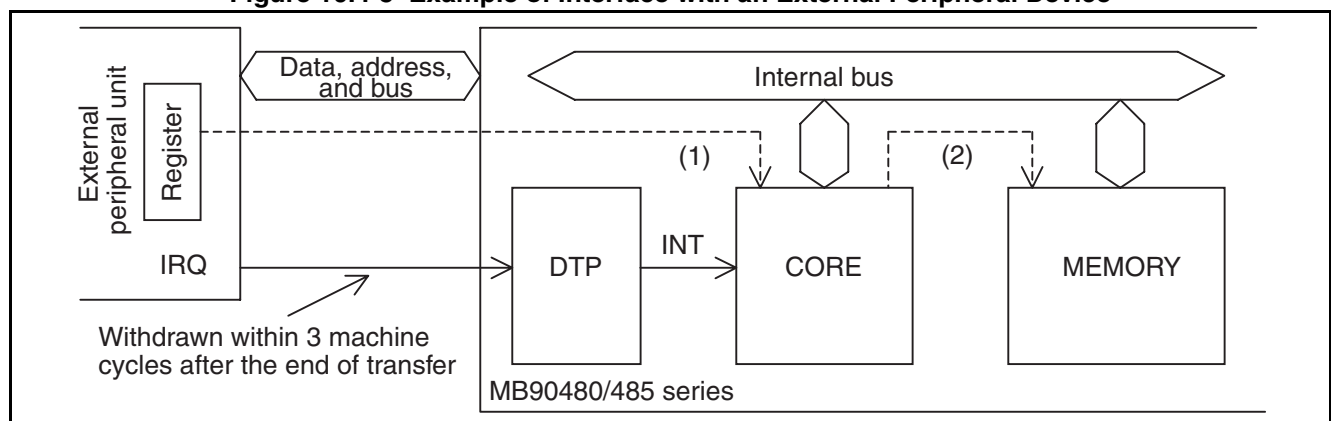
Figure 16.4-2 shows the timing for withdrawing the external interrupt request at the end of DTP operation.

Figure 16.4-3 shows an example for an interface with the external peripheral device.

**Figure 16.4-2 Timing for Withdrawing an External Interrupt Request at the End of DTP Operation**



**Figure 16.4-3 Example of Interface with an External Peripheral Device**



## 16.5 Precautions on Use of DTP/External Interrupt Unit

---

This section shows precautions on use of the DTP/external interrupt unit.

---

### ■ Conditions for External Connection of Peripheral Devices

For support by the DTP unit, external peripheral devices must be able to automatically clear a request after successful data transfer. If a transfer request fails to be withdrawn within three machine cycles after the transfer operation starts, the DTP unit will proceed as if a new transfer request had been generated.

### ■ Procedures for DTP/external Interrupt Unit Operation

Set the values of registers in the DTP/external interrupt unit as follows:

- (1) Set the pin used as an external interrupt input and the general-purpose I/O port used combinedly to the input port.
- (2) Set the bits for the registers (ENIR) to be enabled to "disable".
- (3) Set the bits of the request level setting register (ELVR).
- (4) Clear the bits in the source register (EIRR).
- (5) Set the bits for the registers (ENIR) to be enabled to "enable".

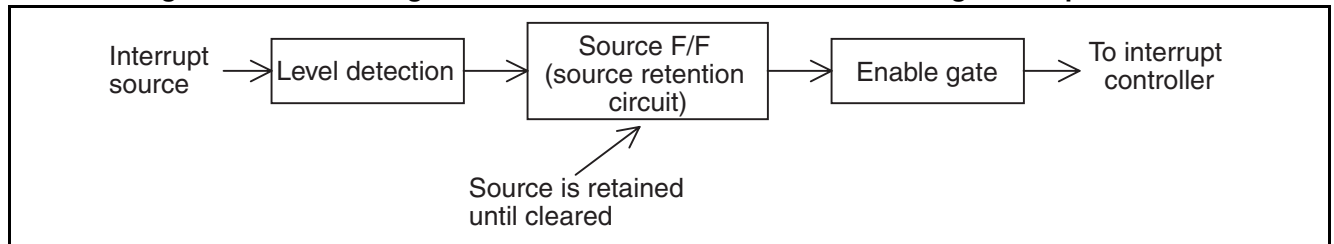
Steps (4) and (5) allow simultaneous writing by word-length specification.

To set the contents of DTP/external interrupt unit registers, first disable the registers to be enabled. Before enabling the registers to be enabled again, clear the source register in order to avoid accidental generation of an interrupt source in register setting of interrupt enabled state.

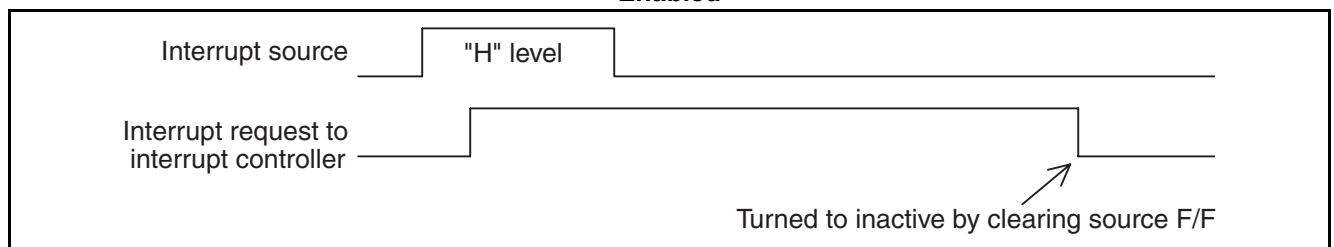
### ■ External Interrupt Request Level

- If edge request has been selected for the request level, at least a pulse width of three machine cycles is required for detecting edge.
- If level setting has been selected for the request input level, note that an external request that has been input remains active with respect to the interrupt controller even if it is later withdrawn, since the interrupt controller contains an internal source retention circuit. To withdraw a request with respect to the interrupt controller, the source retention circuit must be cleared.

**Figure 16.5-1 Clearing the Source Retention Circuit When Setting the Request Level**



**Figure 16.5-2 Interrupt Sources and Interrupt Requests to the Interrupt Controller When Interrupts are Enabled**



Note:

Edge detection cannot be used to return from watch mode.

# 16.6 Program Example of DTP/External Interrupt

This section describes the program example of the DTP/external interrupt.

## ■ Program Example of DTP/External Interrupt

|   |                        |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
|---|------------------------|-----------|-------------------------------------|------|--|-----------------|--|----------|--------------------------|-------|------------|-------|-----------------------|-----------|-----------------------|-----------|------------------------------|-----------|--|
| <div>Example of setting procedure</div> <div>Generate external interrupt at rising edge of signal input by INT0.</div> <div>&lt;Initial setting&gt;</div> <div><div>• Port</div><div>Register name. bit name</div><table><tr><td>Select INT0 port input</td><td>DDR8 .P80</td></tr></table></div> <div><div>• Control INT0</div><div>Register name. bit name</div><table><tr><td>Select external interrupt detection</td><td>ELVR</td></tr><tr><td></td><td>LB7,LA7-LB1,LA1</td></tr><tr><td></td><td>LB0, LA0</td></tr></table></div> <div><div>• Interrupt related 1</div><table><tr><td>Set INT0 interrupt level</td><td>ICR00</td></tr><tr><td>Set I flag</td><td>(CCR)</td></tr></table></div> <div><div>• Interrupt related 2</div><table><tr><td>INT0 interrupt source</td><td>EIRR. ER0</td></tr><tr><td>Enable INT0 interrupt</td><td>ENIR. EN0</td></tr></table></div> <div>&lt;Interrupt&gt;</div> <div><div>• Read conversion value</div><div>Register name. bit name</div><table><tr><td>Clear interrupt request flag</td><td>EIRR. ER0</td></tr></table></div> <div><div>• Arbitrary processing</div></div> <div>&lt;Interrupt vector&gt;</div> <div><div>• Set vector table</div></div> <div>Note:</div> <div>Setting related to clock and setting of _set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</div> | Select INT0 port input | DDR8 .P80 | Select external interrupt detection | ELVR |  | LB7,LA7-LB1,LA1 |  | LB0, LA0 | Set INT0 interrupt level | ICR00 | Set I flag | (CCR) | INT0 interrupt source | EIRR. ER0 | Enable INT0 interrupt | ENIR. EN0 | Clear interrupt request flag | EIRR. ER0 | <div>Program example</div> <div>void EX_INT_sample_1()</div> <div>{</div> <div>EX_INT0_initial();</div> <div>}</div> <div>void EX_INT0_initial(void)</div> <div>{</div> <div>IO_DDR8.bit.D80= 0; /* INT0 input */</div> <div>IO_ELVR.word= 0x0001; /* Setting value:F00000001 (bit) */</div> <div>/* Bit7-2= "000000" */</div> <div>/* Bit1-0= "01" Detect H level */</div> <div>IO_ICR00 = 0x00 /* Arbitrary value */</div> <div>__EI(); /*Enable interrupt */</div> <div>IO_EIRR.bit.ER0= 0; /* Clear ER0 interrupt flag */</div> <div>IO_ENIR.bit.EN0= 0; /* Enable EN0 interrupt */</div> <div>}</div> <div>__interrupt void INT0_int(void) /* */</div> <div>{</div> <div>IO_EIRR.bit.ER0= 0; /* Clear ER0 interrupt flag */</div> <div>}</div> <div>#pragma intvect INT0_int 11</div> <div>Note:</div> <div>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</div> |
| Select INT0 port input  | DDR8 .P80              |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
| Select external interrupt detection   | ELVR                   |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
|   | LB7,LA7-LB1,LA1        |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
|   | LB0, LA0               |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
| Set INT0 interrupt level  | ICR00                  |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
| Set I flag  | (CCR)                  |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
| INT0 interrupt source   | EIRR. ER0              |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
| Enable INT0 interrupt   | ENIR. EN0              |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |
| Clear interrupt request flag  | EIRR. ER0              |           |                                     |      |  |                 |  |          |                          |       |            |       |                       |           |                       |           |                              |           |  |

## ■ Setting Method Other than Program Example

- Type of detection level and setting method

4 types of the detection level are provided ("L" level, "H" level, rising, falling).

Set by the detection level bit (ELVR. LBx, LAx) x=0 to 7.

| Operation mode      | Detection level bit (LBx, LAx) x=0 to 7 |
|---------------------|---|
| To detect "L" level | Set to "00 <sub>B</sub> "               |
| To detect "H" level | Set to "01 <sub>B</sub> "               |
| To detect rising    | Set to "10 <sub>B</sub> "               |
| To detect falling   | Set to "11 <sub>B</sub> "               |

- Method to input IRQ pin

Set by the port 8 direction register (DDR8).

| Operation         | Direction bits (P80 to P87) | Setting    |
|-------------------|-----------------------------|------------|
| To input IRQ0 pin | DDR8. P80                   | Set to "0" |
| To input IRQ1 pin | DDR8. P81                   | Set to "0" |
| To input IRQ2 pin | DDR8. P82                   | Set to "0" |
| To input IRQ3 pin | DDR8. P83                   | Set to "0" |
| To input IRQ4 pin | DDR8. P84                   | Set to "0" |
| To input IRQ5 pin | DDR8. P85                   | Set to "0" |
| To input IRQ6 pin | DDR8. P86                   | Set to "0" |
| To input IRQ7 pin | DDR8. P87                   | Set to "0" |



## CHAPTER 16 DTP/EXTERNAL INTERRUPTS

### ● Interrupt related register

The relationship between the external interrupt pin, interrupt level, and interrupt vector is shown in the following table.

For details of the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

| External interrupt pin | Interrupt vector                     | Interrupt level setting bit  |
|------------------------|--------------------------------------|--|
| IRQ0                   | #11<br>Address : FFFFD0 <sub>H</sub> | Interrupt control register 00 (ICR00)<br>Address : 0000B0 <sub>H</sub> |
| IRQ1                   | #12<br>Address : FFFFCC <sub>H</sub> | Interrupt control register 00 (ICR00)<br>Address : 0000B0 <sub>H</sub> |
| IRQ2                   | #13<br>Address : FFFFC8 <sub>H</sub> | Interrupt control register 01 (ICR01)<br>Address : 0000B1 <sub>H</sub> |
| IRQ3                   | #14<br>Address : FFFFC4 <sub>H</sub> | Interrupt control register 01 (ICR01)<br>Address : 0000B1 <sub>H</sub> |
| IRQ4                   | #15<br>Address : FFFFC0 <sub>H</sub> | Interrupt control register 02 (ICR02)<br>Address : 0000B2 <sub>H</sub> |
| IRQ5                   | #16<br>Address : FFFFBC <sub>H</sub> | Interrupt control register 02 (ICR02)<br>Address : 0000B2 <sub>H</sub> |
| IRQ6                   | #17<br>Address : FFFF8B <sub>H</sub> | Interrupt control register 03 (ICR03)<br>Address : 0000B3 <sub>H</sub> |
| IRQ7                   | #18<br>Address : FFFF84 <sub>H</sub> | Interrupt control register 03 (ICR03)<br>Address : 0000B3 <sub>H</sub> |

### ● Type of interrupt

The interrupt source is external interrupt only. There is no bit to be selected.

### ● Method to enable/disable/clear interrupt

Enabling/disabling interrupt is set by the interrupt enable bit (ENIR.ENx x=0 to 7).

| Content of control           | Interrupt enable bit (ENx x=0 to 7) |
|------------------------------|-------------------------------------|
| To disable interrupt request | Set to "0"                          |
| To enable interrupt request  | Set to "1"                          |

Clearing interrupt request is set by the interrupt request bit (EIRR.ERx x=0 to 7).

| Content of control         | Interrupt request bit (ERx x=0 to 7) |
|----------------------------|--------------------------------------|
| To clear interrupt request | Write "0"                            |

# CHAPTER 17 8/10-BIT A/D CONVERTER

---

**This chapter provides an overview of the 8/10-bit A/D converter, explains configuration and functions of these registers, operation, conversion data protection function, and shows the precautions on use.**

---

- 17.1 Overview of 8/10-Bit A/D Converter
- 17.2 Configuration of 8/10-Bit A/D Converter
- 17.3 Configuration and Functions of 8/10-Bit A/D Converter Registers
- 17.4 Interrupt of 8/10-Bit A/D Converter
- 17.5 Operations of 8/10-Bit A/D Converter
- 17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter
- 17.7 Precautions on use of the 8/10-Bit A/D Converter
- 17.8 Program Example of 8/10-Bit A/D Converter

## 17.1 Overview of 8/10-Bit A/D Converter

---

This section describes the features of the 8/10-bit A/D converter and provides its block diagram.

---

### ■ Features of the 8/10-bit A/D Converter

The 8/10-bit A/D converter features the following functions:

- Conversion time: Minimum of 3.68  $\mu$ s per channel  
(92 machine cycles/machine clock at 25 MHz, with sampling time included)
- Sampling time: Minimum of 1.92  $\mu$ s per channel  
(48 machine cycles/machine clock at 25 MHz)
- RC-type sequential comparison conversion system with sample and hold circuit
- 8- or 10-bit resolution can be selected.
- Analog input by program via eight channels can be selected.
  - Single conversion mode: Selection and conversion of a single channel
  - Scan conversion mode: Several channels can be converted in succession.  
A maximum of eight channels can be programmed.
  - Continuous conversion mode: Conversion of the specified channels is performed repeatedly.
  - Stop conversion mode: Pauses after conversion of one channel is completed and stands by until next activation is triggered. (Conversion starts can be synchronized)

When A/D conversion ends, an interrupt request for end of A/D conversion is generated to the CPU. The occurrence of this interrupt enables the transfer of data resulting from A/D conversion to memory by starting  $\mu$ DMAC. The converter is therefore suitable for continuous processing.

As the start source, either of software, external triggers (falling edge) and the timer (rising edge) can be selected.

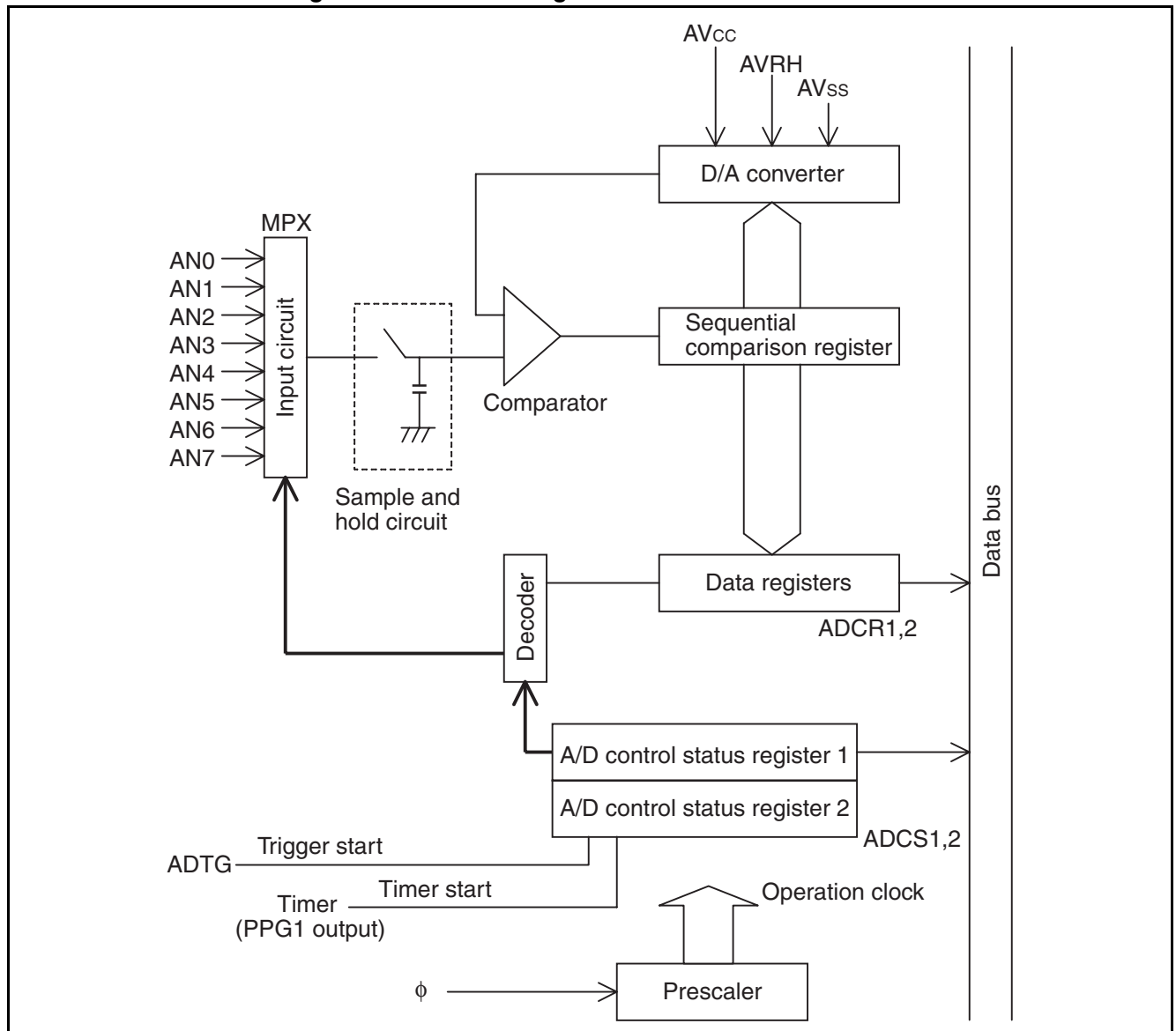
## 17.2 Configuration of 8/10-Bit A/D Converter

**This section describes the block diagram and configuration of 8/10-bit A/D converter.**

### ■ Block Diagram of 8/10-bit A/D Converter

Figure 17.2-1 shows a block diagram of the 8/10-bit A/D converter.

**Figure 17.2-1 Block Diagram of 8/10-bit A/D Converter**



## ■ Pin Related to 8/10-bit A/D Converter

The pin related to 8/10-bit A/D converter has analog input AN0/AN1/AN2/AN3/AN4/AN5/AN6/AN7 pins and input trigger ADTG pin. The general-purpose I/O port (P60/AN0, P61/AN1, P62/AN2, P63/AN3, P64/AN4, P65/AN5, P66/AN6, P67/AN7) functions as the analog input pin of A/D, and the general-purpose I/O port (P93/ADTG) functions as the trigger input of A/D.

- Setting when using as AN0/AN1/AN2/AN3/AN4/AN5/AN6/AN7 pins

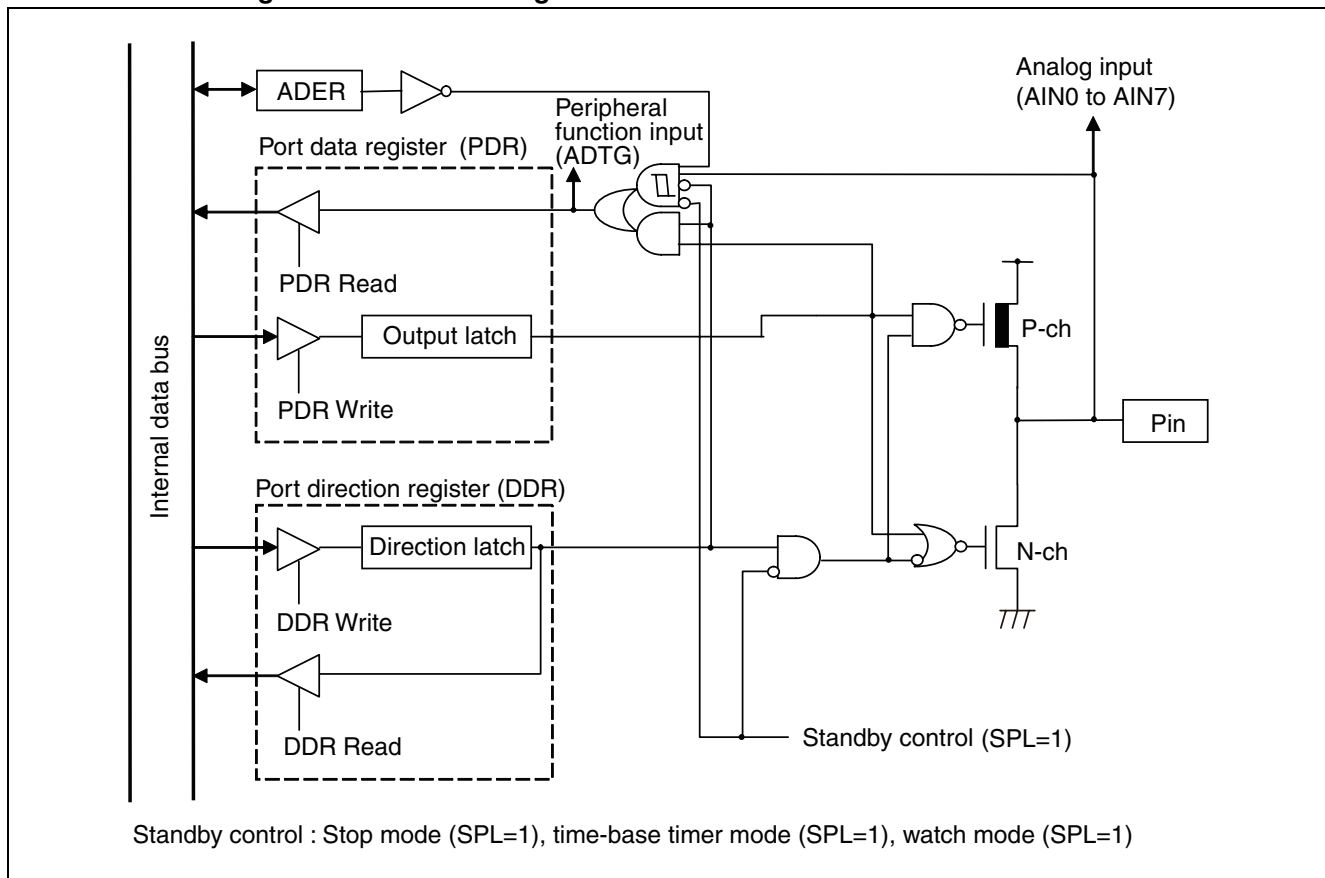
When using as an analog input, be sure to set the port direction register (DDR6: bit8, 7, 6, 5, 4, 3, 2, 1, 0 → "0") and analog enable register (ADER: bit15, 14, 13, 12, 11, 10, 9, 8 → "1").

- Setting when using as ADTG pin

When using as external trigger of the A/D converter, P93/ADTG pin should be set to the input port by port direction register (DDR9: bit11 → "0").

## ■ Block Diagram of Pin Related to 8/10-bit A/D Converter

Figure 17.2-2 Block Diagram of Pin Related to 8/10-bit A/D Converter



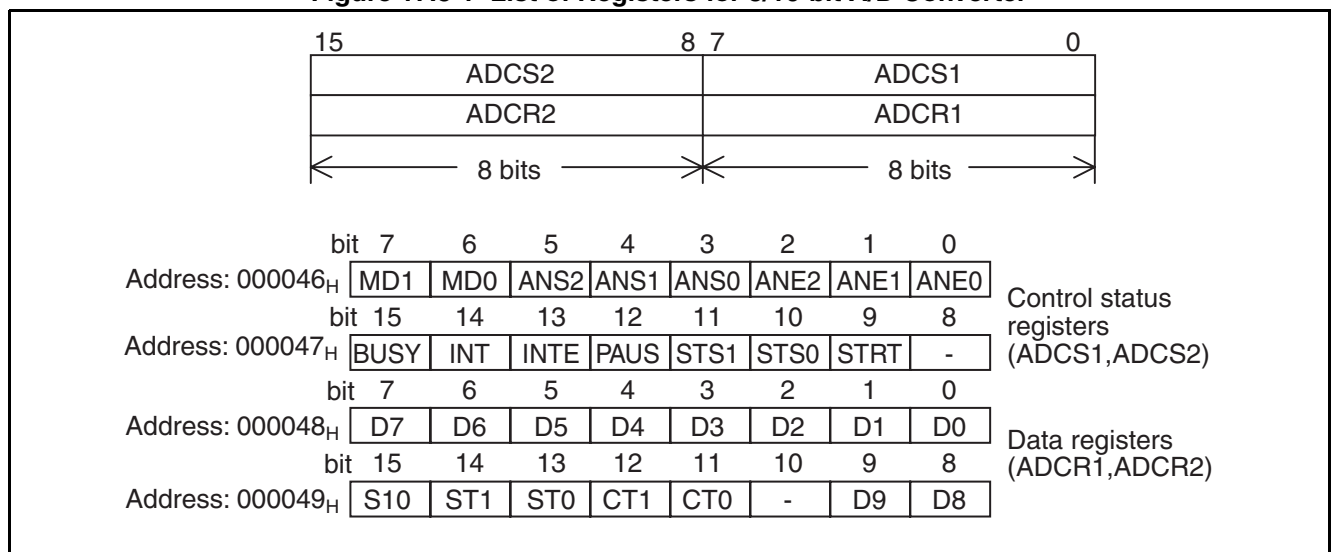
## 17.3 Configuration and Functions of 8/10-Bit A/D Converter Registers

This section describes the configuration and functions of the registers used in the 8/10-bit A/D converter.

### ■ List of Registers for 8/10-bit A/D Converter

Figure 17.3-1 illustrates the list of registers for 8/10-bit A/D Converter.

**Figure 17.3-1 List of Registers for 8/10-bit A/D Converter**



### 17.3.1 Control Status Register 1 (ADCS1)

The control status register 1 (ADCS1) controls the A/D converter and displays the status of operation.

■ Control Status Register 1 (ADCS1)

The bit configuration of the control status register 1 (ADCS1) is illustrated below.

|                                       |     |     |     |      |      |      |      |      |      |                 |
|---------------------------------------|-----|-----|-----|------|------|------|------|------|------|-----------------|
| ADCS1<br>Address: 000046 <sub>H</sub> | bit | 7   | 6   | 5    | 4    | 3    | 2    | 1    | 0    |                 |
|                                       |     | MD1 | MD0 | ANS2 | ANS1 | ANS0 | ANE2 | ANE1 | ANE0 |                 |
|                                       |     | 0   | 0   | 0    | 0    | 0    | 0    | 0    | 0    | ← Initial value |
|                                       |     | R/W | R/W | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | ← Bit attribute |

Note:

Do not rewrite control status register 1 (ADCS1) during A/D conversion.

The function of each bit of the control status register 1 (ADCS1) is described below.

**[bit7, bit6] MD1, MD0: A/D converter MoDe set (operation mode)**

These bits specify the operation mode. Operation modes that can be selected are shown below.

| MD1, MD0        | Operation mode  |
|-----------------|---|
| 00 <sub>B</sub> | Single mode, all ongoing operations can be restarted    |
| 01 <sub>B</sub> | Single mode, ongoing operations cannot be restarted     |
| 10 <sub>B</sub> | Continuous mode, ongoing operations cannot be restarted |
| 11 <sub>B</sub> | Stop mode, ongoing operations cannot be restarted       |

The functions of each mode are as follows:

- Single mode: A/D conversion is continuously performed from the specified channels in the range of ANS2 to ANS0 to the specified channels in the range of ANE2 to ANE0. Operation stops when one cycle of conversion operations ends.
- Continuous mode: A/D conversion is repeated from the specified channels in the range of ANS2 to ANS0 to the specified channels in the range of ANE2 to ANE0.
- Stop mode: A/D conversion is performed per channel from the specified channels in the range of ANS2 to ANS0 to the specified channels in the range of ANE2 to ANE0. After that, operation temporarily stops. Resuming conversion is triggered by a start source.

These bits are initialized to "00" at reset. Each operation mode operates as follows:

- After start of A/D conversion in continuous or stop mode, conversion will continue until operation is stopped via the BUSY bit.
- Operation in each operation mode is stopped by setting the BUSY bit to "0".
- "Restart disable" in the single, continuous, and stop modes affects all start factors by timer, external trigger, and software.

**[bit5, bit4, bit3] ANS2, ANS1, ANS0: ANalog Start channel set**

Set a start channel for A/D conversion using these bits.

At the startup of the A/D converter, A/D conversion starts with the channel selected by these bits.

| ANS2 | ANS1 | ANS0 | Start channel |
|------|------|------|---------------|
| 0    | 0    | 0    | AN0           |
| 0    | 0    | 1    | AN1           |
| 0    | 1    | 0    | AN2           |
| 0    | 1    | 1    | AN3           |
| 1    | 0    | 0    | AN4           |
| 1    | 0    | 1    | AN5           |
| 1    | 1    | 0    | AN6           |
| 1    | 1    | 1    | AN7           |

- During read out
  - These bits are used for reading conversion channels during A/D conversion.
  - When conversion stops in stop mode, the previous conversion channels will be read out.
- These bits are initialized to "000<sub>B</sub>" at reset.

The read values of these bits are updated during the start of A/D conversion, and before A/D conversion starts, the previous conversion channel will be read even if these bits have already been set to the new value.



**[bit2, bit1, bit0] ANE2, ANE1, ANE0: ANalog End channel set**

These bits specify the end channel for A/D conversion.

| ANE2 | ANE1 | ANE0 | End channel |
|------|------|------|-------------|
| 0    | 0    | 0    | AN0         |
| 0    | 0    | 1    | AN1         |
| 0    | 1    | 0    | AN2         |
| 0    | 1    | 1    | AN3         |
| 1    | 0    | 0    | AN4         |
| 1    | 0    | 1    | AN5         |
| 1    | 1    | 0    | AN6         |
| 1    | 1    | 1    | AN7         |

- Specifying the same channels as for ANS2 to ANS0 will result in one-channel conversion (Single conversion).
- If continuous mode or stop mode is set, the channels specified by ANS2 to ANS0 will be set again when conversion of the channels specified by these bits ends.
- If  $ANS < ANE$ , conversion will start from the channel set by ANS; after conversion has been performed up to ch.7, the setting channel will be set to ch.0, and conversion will be performed up to end channel set by ANE.
- These bits are initialized to "000<sub>B</sub>" at reset.

Example: Channel setting: Single Mode with ANS = ch.6 and ANE = ch.3

Operation conversion channel ch.6 → ch.7 → ch.0 → ch.1 → ch.2 → ch.3

---

**Note:**

Please do not set the A/D conversion mode setting bit (MD1, MD0) and the A/D conversion end channel selection bit (ANE2, ANE1 and ANE0) by the read-modify-write type instruction after setting the start channel to the A/D conversion start channel selection bit (ANS2, ANS1, ANS0). The last conversion channel is read from the ANS2, ANS1 and ANS0 bits until the A/D conversion operating starts.

Therefore, when MD1, MD0, ANE2, ANE1 and ANE0 bits are set by the read-modify-write type instruction after setting the start channel to ANS2, ANS1 and ANS0 bits, the value of the ANE2, ANE1 and ANE0 bits may be overwritten.

---

## 17.3.2 Control Status Register 2 (ADCS2)

The control status register 2 (ADCS2) is used for A/D converter control and status display.

### ■ Control Status Register 2 (ADCS2)

The bit configuration of the control status register 2 (ADCS2) is illustrated below.

|                                       |        |     |      |      |      |      |      |          |                 |
|---------------------------------------|--------|-----|------|------|------|------|------|----------|-----------------|
| ADCS2<br>Address: 000047 <sub>H</sub> | bit 15 | 14  | 13   | 12   | 11   | 10   | 9    | 8        |                 |
|                                       | BUSY   | INT | INTE | PAUS | STS1 | STS0 | STRT | Reserved |                 |
|                                       | 0      | 0   | 0    | 0    | 0    | 0    | 0    | 0        | ← Initial value |
|                                       | R/W    | R/W | R/W  | R/W  | R/W  | R/W  | W    | R/W      | ← Bit attribute |

The function of each bit of the control status register 2 (ADCS2) is described below.

#### [bit15] BUSY: busy flag and stop

- During reading: This bit indicates A/D converter operation. It is set when A/D conversion starts and cleared when A/D conversion ends.
- During writing: Setting this bit by writing "0" during A/D operation will forcibly stop operation. Use this bit to force stopping in continuous mode or stop mode.

When the bit is used for operation display, it cannot be set by writing "1". Read-modify-write (RMW) instructions will always read "1". In single mode, this bit is cleared when A/D conversion ends. In continuous and stop mode, the bit is not cleared until operation is stopped by writing "0".

This bit is initialized to "0" at reset.

Note:

Do not execute forced stop and software start simultaneously. (BUSY = 0, STRT = 1)

#### [bit14] INT: Interrupt

This bit is a data indication bit. This bit will be set when conversion data is written to the ADCR.

Setting this bit when bit5 (INTE) is "1" generates an interrupt request. If  $\mu$ DMAC start is enabled,  $\mu$ DMAC will be started. Writing "1" has no effect. Write "0" and use a  $\mu$ DMAC interrupt clear signal to clear.

Initialized to "0" at reset.

Also read the Caution when using the conversion data protection function in Section "17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter".

Note:

Clear this bit by writing "0" only while A/D conversion is not being performed.

### [bit13] INTE: interrupt enable

This bit is used to enable or disable interrupts at conversion end.

- 0: Interrupts prohibited
- 1: Interrupts allowed

Set this bit when using  $\mu$ DMAC. An interrupt request will then trigger  $\mu$ DMAC start.

This bit is initialized to "0" at reset.

Also read the Caution when using the conversion data protection function in Section "17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter".

### [bit12] PAUS: A/D converter pause

This bit is set when A/D conversion pauses.

There is only one register for storing the results of A/D conversion. When conversion is performed continuously, data stored previously is overwritten unless conversion results is transferred by  $\mu$ DMAC. To prevent overwriting data, the next conversion result cannot be stored before the contents of the data register has been transferred by  $\mu$ DMAC, and A/D conversion will be stopped in between. The A/D converter resumes conversion as soon as data transfer by  $\mu$ DMAC ends.

This register is effective only when  $\mu$ DMAC is used.

Also read the Caution when using the conversion data protection function in Section "17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter".

This bit is initialized to "0" at reset.

### [bit11, bit10] STS1, STS0: start source select

These bits select A/D start sources.

| STS1 | STS0 | Function  |
|------|------|---|
| 0    | 0    | Software start  |
| 0    | 1    | Start by an external pin trigger and software         |
| 1    | 0    | Start by timer and software                           |
| 1    | 1    | Start by an external pin trigger, timer, and software |

In a mode for which two start sources apply, the first of the sources to occur will trigger start.

Start sources become effective from the time they are rewritten. Exercise caution when rewriting during A/D operation.

- When an external pin trigger is selected, a falling edge is detected.
- When the external trigger input level is "L", A/D conversion will start as soon as this bit is rewritten to select the external trigger start.
- The output of PPG1 is selected at the time the timer is selected.

These bits are initialized to "00<sub>B</sub>" at reset.

**Note:**

When starting the A/D converter by an external trigger or an internal timer, set the input value of the internal timer and the external trigger only in inactive state.

For setting STS1 and STS0, set in the state of ADTG=1 input and internal timer (PPG1) = 0 output.

---

**[bit9] STRT: start**

8/10-bit A/D converter is started by software.

- Set this bit by writing "1" to start A/D conversion.
  - For restarting, write this bit again.
  - When stop mode is set, operation cannot be restarted by an operational function.
  - This bit is initialized to "0" at reset.
  - The byte/words instructions read "1".
  - The read-modify-write type instructions read "0".
- 

**Note:**

Do not execute forced stop and software start simultaneously. (BUSY = 0, STRT = 1)

---

**[bit8] Reserved**

In write operations, write "0".

### 17.3.3 Data Registers (ADCR2 and ADCR1)

The configurations and functions of the data registers (ADCR2 and ADCR1) are explained below.

#### ■ Data Registers (ADCR2 and ADCR1)

The function of each bit for the data registers (ADCR2 and ADCR1) is described below.

|                              |  |     |     |     |     |     |     |    |    |    |                 |
|------------------------------|--|-----|-----|-----|-----|-----|-----|----|----|----|-----------------|
|                              |  | bit | 7   | 6   | 5   | 4   | 3   | 2  | 1  | 0  |                 |
| ADCR1                        |  |     | D7  | D6  | D5  | D4  | D3  | D2 | D1 | D0 |                 |
| Address: 000048 <sub>H</sub> |  |     | X   | X   | X   | X   | X   | X  | X  | X  | ← Initial value |
|                              |  |     | R   | R   | R   | R   | R   | R  | R  | R  | ← Bit attribute |
|                              |  | bit | 15  | 14  | 13  | 12  | 11  | 10 | 9  | 8  |                 |
| ADCR2                        |  |     | S10 | ST1 | ST0 | CT1 | CT0 | -  | D9 | D8 |                 |
| Address: 000049 <sub>H</sub> |  |     | 0   | 0   | 0   | 0   | 0   | X  | X  | X  | ← Initial value |
|                              |  |     | W   | W   | W   | W   | W   | R  | R  | R  | ← Bit attribute |

The data registers (ADCR2 and ADCR1) are registers for storing conversion results as digital values. The upper two bits of conversion values are stored in ADCR2, and the lower eight bits are stored in ADCR1. The values of these registers are updated at the end of each conversion. Normally, the last conversion value is stored. The S10-bits must be rewritten if A/D operation stops before conversion ends. Rewriting after conversion may result in an undefined ADCR data. To read ADCR registers with 10-bit mode specified, always use a word instruction. The ADCR registers have a conversion data protection function. Refer to Section "17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter".

Do not write data to these registers during A/D operation. When S10 bit is set to "0", conversion results will be output in units of ten bits; when S10 bit is set to "1", conversion results will be output in units of eight bits.

| ST1 | ST0 | Sampling time setting bit                  | CT1 | CT0 | Compare time setting bit                   |
|-----|-----|--|-----|-----|--|
| 0   | 0   | 20 machine cycles<br>(0.8 $\mu$ s@25MHz)   | 0   | 0   | 44 machine cycles<br>(1.76 $\mu$ s@25MHz)  |
| 0   | 1   | 32 machine cycles<br>(1.28 $\mu$ s@25MHz)  | 0   | 1   | 66 machine cycles<br>(2.64 $\mu$ s@25MHz)  |
| 1   | 0   | 48 machine cycles<br>(1.92 $\mu$ s@25MHz)  | 1   | 0   | 88 machine cycles<br>(3.52 $\mu$ s@25MHz)  |
| 1   | 1   | 128 machine cycles<br>(5.12 $\mu$ s@25MHz) | 1   | 1   | 176 machine cycles<br>(7.04 $\mu$ s@25MHz) |

Note:

Setting ST1 and ST0 = 00<sub>B</sub> or 01<sub>B</sub> during operation at 25 MHz may prevent the proper analog voltages from being obtained.

## 17.4 Interrupt of 8/10-Bit A/D Converter

The 8/10-Bit A/D converter generates the interrupt request when the A/D conversion is terminated, and the conversion result is stored to the A/D data register (ADCR). Also, it can activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt of 8/10-bit A/D Converter

The interrupt control bit and the interrupt source of the 8/10-bit A/D converter is shown in the following table.

|                                     |  |
|-------------------------------------|--|
| Interrupt request flag              | ADCS:INT (bit14)   |
| Interrupt request output enable bit | ADCS:INTE (bit13)  |
| Interrupt generation source         | A/D conversion result is stored to A/D data register (ADCR). |

### ■ Interrupt of A/D Converter

When A/D conversion of the analog input voltage is terminated and its results are stored in the data register (ADCR), the interrupt request flag bit in the A/D control status register (ADCS: INT) is set to "1". Interrupt request is generated when the interrupt request flag bit (ADCS: INT=1) is set with interrupt request output enabled (ADCS: INTE=1).

### ■ Interrupt of 8/10-bit A/D Converter, DMA Transfer, and EI<sup>2</sup>OS

Table 17.4-1 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 17.4-1 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|------------------|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|                  |                          |                      | Number           | Address             | Number                     | Address             |
| A/D converter*   | ○                        | 15                   | #40              | FFFF5C <sub>H</sub> | ICR14                      | 0000BE <sub>H</sub> |

○: Interrupt request flag is cleared.

\*: This interrupt source shares the interrupt source and interrupt number of other peripheral function.

For details, see Table 3.2-2.

Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing..

### ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The interrupt of the A/D converter corresponds to the DMA transfer function and EI<sup>2</sup>OS function. When the DMA or EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 17.5 Operations of 8/10-Bit A/D Converter

---

The 8/10-bit A/D converter operates based on a sequential comparison method and has a 10-bit resolution.

The 8/10-bit A/D converter has only one register for storing results of conversion (10-bit). Data register (ADCR1 and ADCR2) is updated whenever one conversion operation ends. The converter therefore does not support continuous conversion processing. It is recommended that conversion be performed while transferring conversion data to memory using the  $\mu$ DMAC function of F<sup>2</sup>MC-16LX.

This section describes the operations of the 8/10-bit A/D converter.

---

### ■ Operation Modes

#### ○ Single Mode

In this mode, analog input set by the bits ANS and ANE is sequentially converted. The converter stops conversion operation when conversion up to the end channel set by the bit ANE is finished. If the start and end channels are the same (ANS = ANE), one-channel conversion will be performed.

Example:

ANS = 000<sub>B</sub>, ANE = 011<sub>B</sub>

Start → AN0 → AN1 → AN2 → AN3 → End

ANS = 010<sub>B</sub>, ANE = 010<sub>B</sub>

Start → AN2 → End

---

Note:

The A/D conversion ends without restarting when the restart and the end of the A/D conversion occur at the same time, and "300<sub>H</sub>" is stored in data register (ADCR1/0). Therefore, please restart so that neither the A/D conversion comeback movement nor the end may occur at the same time.

---

#### ○ Continuous Mode

In this mode, analog input set by the bits ANS and ANE is sequentially converted. The converter continues conversion operation by returning to ANS analog input when conversion up to the end channel set by the bit ANE is finished. One-channel conversion operation will continue if the start and end channels are the same. (ANS = ANE)

Example:

ANS = 000<sub>B</sub>, ANE = 011<sub>B</sub>

Start → AN0 → AN1 → AN2 → AN3 → AN0 ..... Repeat

ANS = 010<sub>B</sub>, ANE = 010<sub>B</sub>

Start → AN2 → AN2 → AN2 ..... Repeat

When conversion is performed in continuous mode, conversion operation will be repeated until the BUSY bit is set by writing "0".

Set the BUSY bit to "0" to forcibly stop operation.

When operation is stopped forcibly, data before the completion of conversion will be stored in the conversion registers.

Note:

When operation is forcibly stopped in continuous mode, conversion data retains data before operation of forced stop.

### ○ Stop Mode

In this mode, analog input specified by the bits ANS and ANE is sequentially converted. However, conversion operation stops temporarily after conversion of each channel. To release the temporary stop, it is necessary to start A/D conversion again.

Analog input of channel set by ANS will be resumed and A/D conversion will continue when conversion up to the end channel specified by the bit ANE ends.

One-channel conversion will be performed if the start and end channels are the same. (ANS = ANE)

Example:

ANS = 000<sub>B</sub>, ANE = 011<sub>B</sub>

Start → AN0 → Stop → Activate → AN1 → Stop → Activate →

→ AN2 → Stop → Activate → AN3 → Stop → Activate → AN0 ..... Repeat

ANS = 010<sub>B</sub>, ANE = 010<sub>B</sub>

Start → AN2 → Stop → Activate → AN2 → Stop → Activate → AN2 ..... Repeat

Only A/D startup sources specified by STS1 and 0 can be used in this mode.

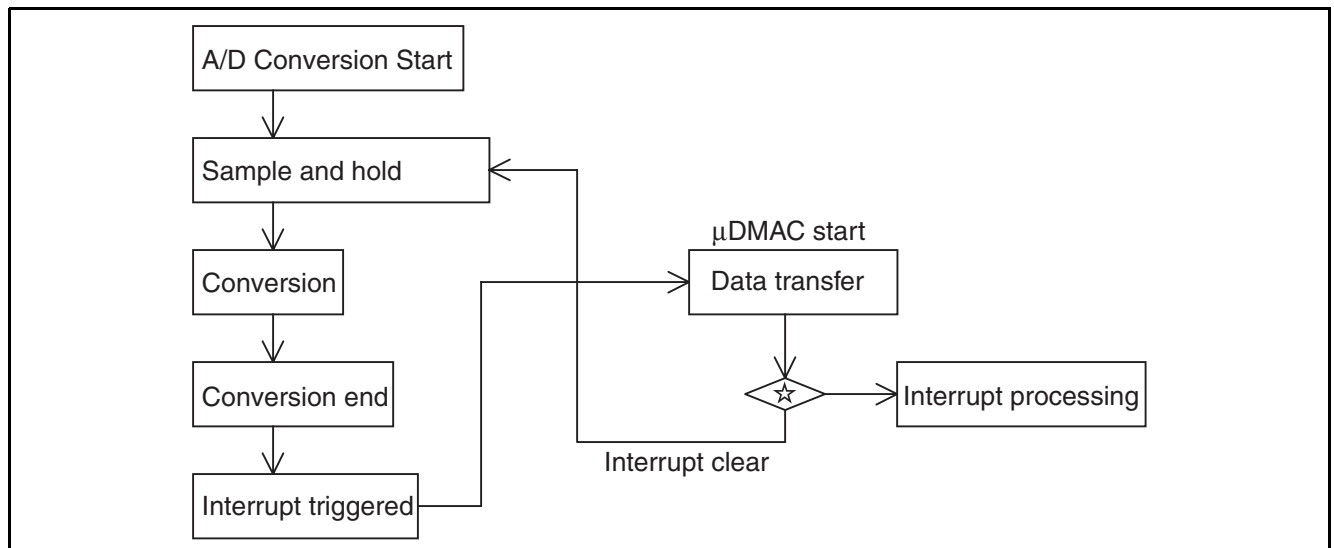
Operation in this mode enables synchronizing the start of multiple conversions.

### ■ Conversion Operation Using $\mu$ DMAC

Figure 17.5-1 shows an example of the operational flow (in continuous mode) from start of A/D conversion to transfer of conversion data.

The result of the decision operation marked by "☆" in the diagram is decided based on the settings of  $\mu$ DMAC.

**Figure 17.5-1 Example of Operation Flow (in Continuous Mode) from Start of A/D Conversion to Transfer of Conversion Data**





## 17.5.1 Example of $\mu$ DMAC Start in Single Mode

An example of  $\mu$ DMAC start in the single mode is described below.

### ■ Example of $\mu$ DMAC Start in Single Mode

An example of start operation is based on the conditions described below:

- Conversion finishes after conversion up to analog input (AN1 to AN3)
- Conversion data is transferred to addresses 200<sub>H</sub> to 206<sub>H</sub> sequentially
- Start by software
- The highest interrupt level is used

| Setting item            | Sample program   | Operation   |
|-------------------------|------------------|---|
| $\mu$ DMAC setting      | MOV ICR14,#00H   | Sets highest interrupt level and enables interrupts   |
|                         | MOV BAPL,#00H    | Address to transfer conversion data   |
|                         | MOV BAPM,#02H    |   |
|                         | MOV BAPH,#00H    |   |
|                         | MOV DMACS,#18H   | Set DMA control status register (Transfers word data and increments the destination address after transfer) |
|                         | MOV IOA,#48H     | Stores A/D conversion results in registers  |
|                         | MOV DCT,#03H     | Performs three transfers, matching the number of conversions  |
|                         | MOVW DERL,#8000H | Setting for the $\mu$ DMAC enable register (EN15)   |
| A/D converter setting   | MOV ADCS1,#0BH   | Single mode, start channel AN1, end channel AN3   |
|                         | MOV ADCS2,#A2H   | Software start, A/D conversion start  |
| $\mu$ DMAC end sequence | WBTC io ADCS2:7  | Determines end of A/D conversion  |
|                         | MOV ADCS2,#000H  | Resource interrupt clear  |
|                         | MOVW DSRL,#0000H | $\mu$ DMAC status register clear  |
|                         | RETI             | Reset from interrupt  |

ICR14: Interrupt control register

BAPL: Buffer address pointer lower

BAPM: Buffer address pointer middle

BAPH: Buffer address pointer higher

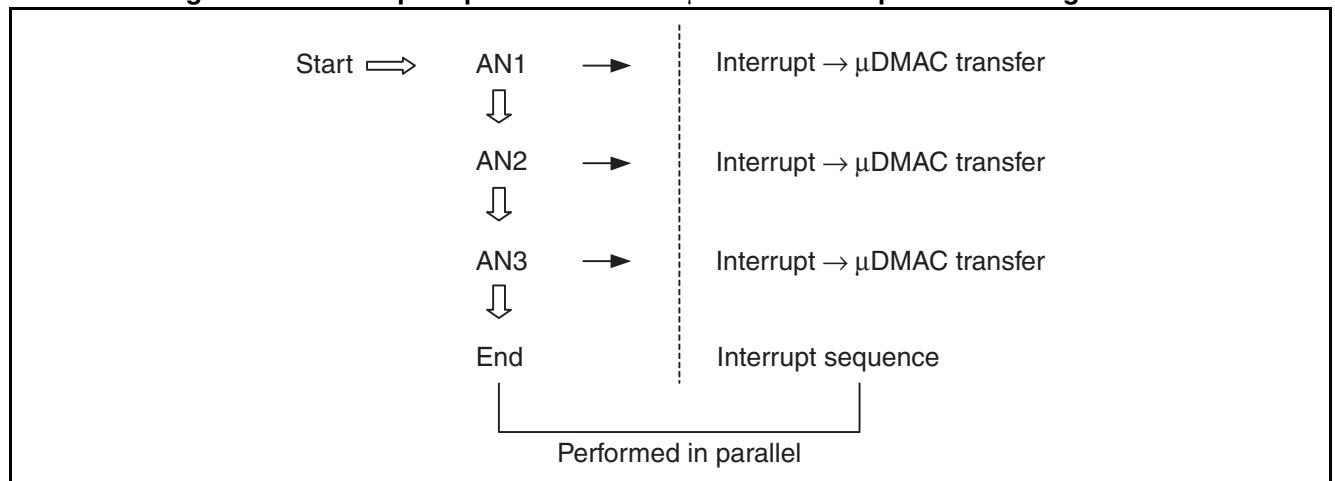
ISCS: Status register

IOA: Address register

DCT: Data counter

Figure 17.5-2 shows an example of the operational flow for start of conversion.

**Figure 17.5-2 Sample Operation Flow for  $\mu$ DMAC Start Operation in Single Mode**



## 17.5.2 Example of $\mu$ DMAC Start in Continuous Mode

An example of  $\mu$ DMAC start in the continuous mode is described below.

### ■ Example of $\mu$ DMAC Start in Continuous Mode

Examples of start operation are based on the following conditions:

- Analog input (AN3 to AN5) is converted, and two items of data for conversion are obtained from each channel
- Conversion data is transferred to addresses 600<sub>H</sub> to 60C<sub>H</sub> sequentially
- Start by an external edge
- The highest interrupt level is used.

| Setting item            | Sample program   | Operation   |
|-------------------------|------------------|---|
| $\mu$ DMAC setting      | MOV ICR14,#00H   | Sets highest interrupt level and enables interrupts   |
|                         | MOV BAPL,#00H    | Address to transfer conversion data   |
|                         | MOV BAPM,#06H    |   |
|                         | MOV BAPH,#00H    |   |
|                         | MOV DMACS,#18H   | Set DMA control status register (Transfers word data and increments the destination address after transfer) |
|                         | MOV IOA,#48H     | Stores A/D conversion results in registers  |
|                         | MOV DCT,#06H     | Performs six transfers, matching the number of conversions  |
|                         | MOVW DERL,#8000H | Setting for the $\mu$ DMAC enable register (EN15)   |
| A/D converter setting   | MOV ADCS1,#9DH   | Single Mode, start channel AN3, end channel AN5   |
|                         | MOV ADCS2,#A4H   | Start by external edge, start of A/D conversion   |
| $\mu$ DMAC end sequence | WBTC io ADCS2:7  | Determines end of A/D conversion  |
|                         | MOV ADCS2,#000H  | Resource interrupt clear  |
|                         | MOVW DSRL,#0000H | $\mu$ DMAC status register clear  |
|                         | RETI             | Reset from interrupt  |

ICR14: Interrupt control register

BAPL: Buffer address pointer lower

BAPM: Buffer address pointer middle

BAPH: Buffer address pointer higher

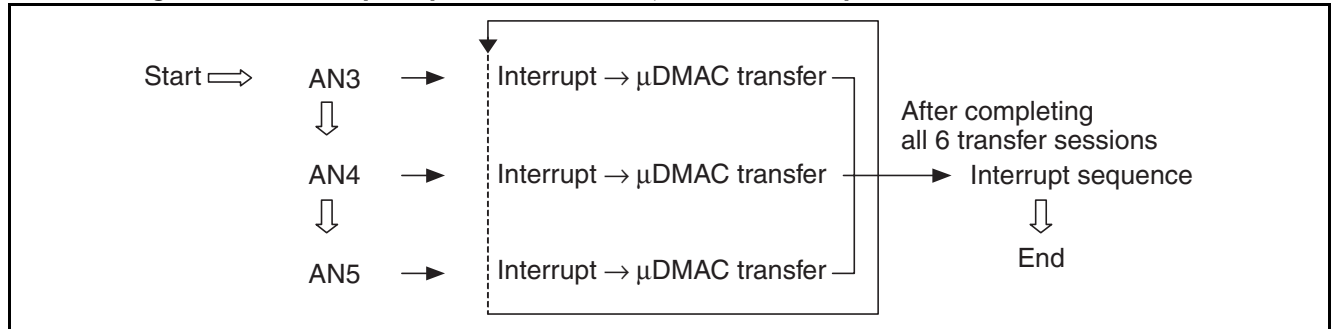
ISCS: Status register

IOA: Address register

DCT: Data counter

Figure 17.5-3 shows a sample operation flow for start processing.

**Figure 17.5-3 Sample Operation Flow for  $\mu$ DMAC Start Operation in Continuous Mode**



### 17.5.3 Example of $\mu$ DMAC Start in Stop Mode

An example of  $\mu$ DMAC start in stop mode is described below.

#### ■ Example of $\mu$ DMAC Start in Stop Mode

Examples of start operation are based on the following conditions:

- Analog input (AN3) is converted 12 times in preset intervals
- Conversion data is transferred to addresses 600<sub>H</sub> to 618<sub>H</sub> sequentially
- Conversion is started by an external edge
- Interrupts have the highest interrupt level

| Setting item            | Sample program   | Operation   |
|-------------------------|------------------|---|
| $\mu$ DMAC setting      | MOV ICR14,#00H   | Sets highest interrupt level and enables interrupts   |
|                         | MOV BAPL,#00H    | Address to transfer conversion data   |
|                         | MOV BAPM,#06H    |   |
|                         | MOV BAPH,#00H    |   |
|                         | MOV DMACS,#18H   | Set DMA control status register (Transfers word data and increments the destination address after transfer) |
|                         | MOV IOA,#48H     | Stores A/D conversion results in registers  |
|                         | MOV DCT,#0CH     | Performs twelve transfers, matching the number of conversions   |
|                         | MOVW DERL,#8000H | Setting for the $\mu$ DMAC enable register (EN15)   |
| A/D converter setting   | MOV ADCS1,#DBH   | Continuous mode, start channel AN3, end channel AN3 (one-channel conversion)                                |
|                         | MOV ADCS2,#A4H   | Start by external edge, start of A/D conversion   |
| $\mu$ DMAC end sequence | WBTC io ADCS2:7  | Determines end of A/D conversion  |
|                         | MOV ADCS2,#000H  | Resource interrupt clear  |
|                         | MOVW DSRL,#0000H | $\mu$ DMAC status register clear  |
|                         | RETI             | Reset from interrupt  |

ICR14: Interrupt control register

BAPL: Buffer address pointer lower

BAPM: Buffer address pointer middle

BAPH: Buffer address pointer higher

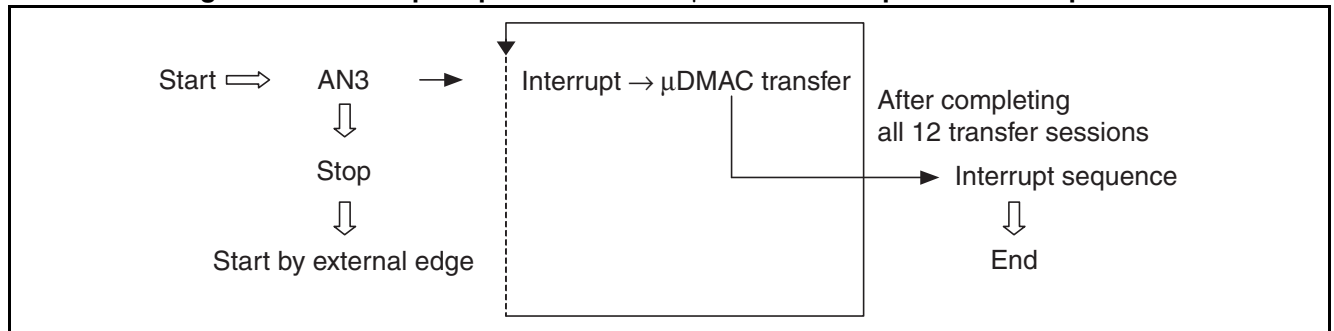
ISCS: Status register

IOA: Address register

DCT: Data counter

Figure 17.5-4 shows a sample operation flow for the start operation.

**Figure 17.5-4 Sample Operation Flow of  $\mu$ DMAC Start Operation in Stop Mode**



## 17.6 Conversion Data Protection Function of 8/10-Bit A/D Converter

---

**This 8/10-bit A/D converter has a conversion data protection function to enable continuous conversion and saving of multiple data items by  $\mu$ DMAC.**

---

### ■ Conversion Data Protection Function

The 8/10-bit A/D converter has only a single conversion data register. In continuous A/D conversion, conversion data is lost when the next conversion operation ends and its result is stored in the register. To prevent this, the A/D converter stops A/D conversion without storing a conversion result to the register in cases when a conversion has been completed, but the result of the previous conversion has not been transferred yet to memory via  $\mu$ DMAC.

The temporary stop is canceled after the conversion data has been transferred to memory by  $\mu$ DMAC. Provided conversion data is transferred normally, A/D conversion continuously performs without pausing.

### ■ Caution When Using the Conversion Data Protection Function

This function corresponds to the INT and INTE bits of ADCS2 register.

The conversion data protection function operates only in the interrupt enabled state (INTE = 1).

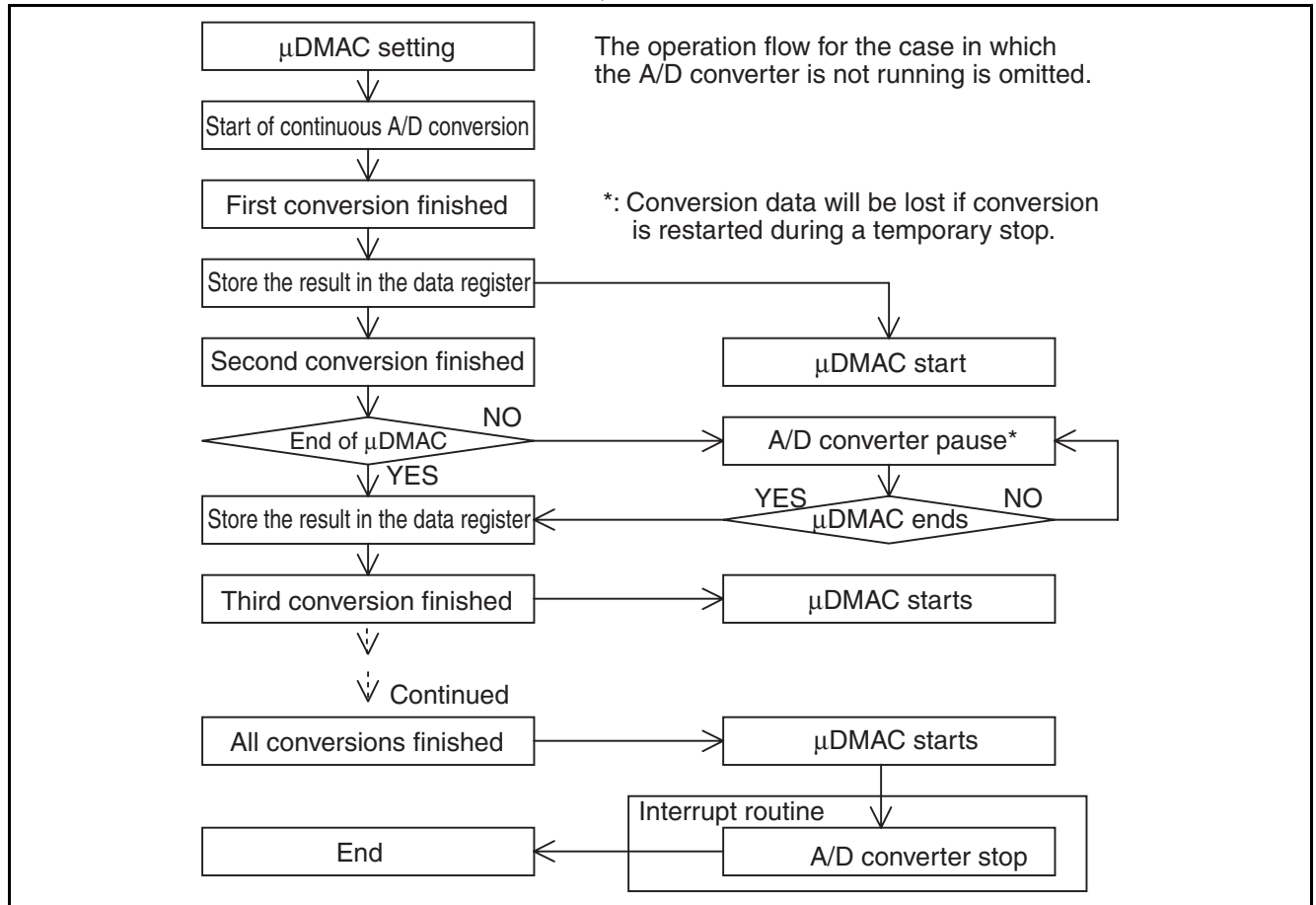
In interrupt disabled state (INTE = 0), this function does not operate. In continuous A/D conversion, conversion results will be stored to the register in succession and older results are lost. If  $\mu$ DMAC is not used in the interrupt enabled state (INTE = 1), the INT bit will not be cleared and the data protection function becomes effective, stopping the operation of the 8/10-bit A/D converter temporarily. Clear the INT bit by the interrupt sequence in this case to cancel the stop.

When interrupts are disabled during DMA operation while A/D conversion is temporarily stopped, data in the conversion data registers will sometimes change before A/D conversion starts and data is transferred. Data that was stored during the stop will be lost if data conversion is restarted during operation stop.

## ■ Operation Flow of Conversion Data Protection Function (when $\mu$ DMAC is Used)

Figure 17.6-1 shows the operation flow of the conversion data protection function.

**Figure 17.6-1 Operation Flow of Conversion Data Protection Function  
(when  $\mu$ DMAC is Used)**





# 17.7 Precautions on use of the 8/10-Bit A/D Converter

This section explains the precautions required when the 8/10-bit A/D converter is used.

## ■ Precautions When Starting by External Trigger/Internal Trigger

To start the A/D converter by an external trigger or the internal timer, specify the input values of the external trigger and internal timer only in the inactive state.

When STS1 and STS0 are set, perform conversion with ADTG = 1 input and internal timer (PPG1) = 0 output.

## ■ Handling of Analog Input Pins

Be sure to set the ADER bits corresponding to the pins used in analog input to "1".

| bit                          | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    | Initial value         |
|------------------------------|------|------|------|------|------|------|------|------|-----------------------|
| Address: 00001F <sub>H</sub> | ADE7 | ADE6 | ADE5 | ADE4 | ADE3 | ADE2 | ADE1 | ADE0 | 11111111 <sub>B</sub> |
|                              | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |                       |

The settings for pin control of the pins of port 6 are as follows:

- 0: Port input/output mode
- 1: Analog input mode

At reset, "1" will be set.

### ● Precautions on use of the 8/10-Bit A/D Converter

The A/D conversion ends without restarting when the restart and the end of the A/D conversion occur at the same time, and 300<sub>H</sub> is stored in data register (ADCR1/0). Therefore, please restart so that neither the A/D conversion comeback movement nor the end may occur at the same time.

## 17.8 Program Example of 8/10-Bit A/D Converter

This section describes the program example of the 8/10-bit A/D converter.

### ■ Program Example of 8/10-bit A/D Converter

| <p>Example of setting procedure</p> <p>An example that A/D-converts the level input by AN0 (single conversion, software trigger) is shown below.</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>• Enable AN0 input <table border="1" data-bbox="228 701 716 758"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Set port input</td> <td>DDR6, P60</td> </tr> <tr> <td>Enable A/D input of AN0</td> <td>ADER.ADE0</td> </tr> </tbody> </table> </li> <li>• A/D0 conversion time <table border="1" data-bbox="228 783 716 863"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Set conversion time</td> <td>ADCR2</td> </tr> <tr> <td>Set sampling time&gt;&gt;</td> <td>.ST1-0</td> </tr> <tr> <td>Set compare time&gt;&gt;</td> <td>.CT1-0</td> </tr> </tbody> </table> </li> <li>• Control A/D0 <table border="1" data-bbox="228 911 716 1262"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Control status register 1</td> <td>ADCS1</td> </tr> <tr> <td>Select operation mode&gt;&gt;</td> <td>.MD1-0</td> </tr> <tr> <td>Set start channel&gt;&gt;</td> <td>.ANS2-0</td> </tr> <tr> <td>Set end channel&gt;&gt;</td> <td>.ANE2-0</td> </tr> <tr> <td>Control status register 2</td> <td>ADCS2</td> </tr> <tr> <td></td> <td>.BUSY</td> </tr> <tr> <td>Clear interrupt request flag&gt;&gt;</td> <td>.INT</td> </tr> <tr> <td>Disable interrupt&gt;&gt;</td> <td>.INTE</td> </tr> <tr> <td></td> <td>.PAUS</td> </tr> <tr> <td>Select start trigger&gt;&gt;</td> <td>.STS1-0</td> </tr> <tr> <td></td> <td>.STRT</td> </tr> <tr> <td></td> <td>.Reserved bit</td> </tr> </tbody> </table> </li> <li>• Interrupt related <table border="1" data-bbox="228 1287 716 1344"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Set A/D interrupt level</td> <td>ICR14</td> </tr> <tr> <td>Set I flag</td> <td>(CCR)</td> </tr> </tbody> </table> </li> </ul> <p>&lt;Start A/D&gt;</p> <ul style="list-style-type: none"> <li>• Start AN0 <table border="1" data-bbox="228 1486 716 1566"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Enable A/D0 interrupt</td> <td>ADCS2</td> </tr> <tr> <td></td> <td>.INT</td> </tr> <tr> <td></td> <td>.INTE</td> </tr> </tbody> </table> </li> <li>Start A/D0 software <table border="1" data-bbox="228 1591 716 1648"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Start A/D0 software</td> <td>ADCS2</td> </tr> <tr> <td></td> <td>.STRT</td> </tr> </tbody> </table> </li> </ul> |                         | Register name, bit name | Set port input | DDR6, P60 | Enable A/D input of AN0 | ADER.ADE0 |  | Register name, bit name | Set conversion time | ADCR2 | Set sampling time>> | .ST1-0 | Set compare time>> | .CT1-0 |  | Register name, bit name | Control status register 1 | ADCS1 | Select operation mode>> | .MD1-0 | Set start channel>> | .ANS2-0 | Set end channel>> | .ANE2-0 | Control status register 2 | ADCS2 |  | .BUSY | Clear interrupt request flag>> | .INT | Disable interrupt>> | .INTE |  | .PAUS | Select start trigger>> | .STS1-0 |  | .STRT |  | .Reserved bit |  | Register name, bit name | Set A/D interrupt level | ICR14 | Set I flag | (CCR) |  | Register name, bit name | Enable A/D0 interrupt | ADCS2 |  | .INT |  | .INTE |  | Register name, bit name | Start A/D0 software | ADCS2 |  | .STRT | <p>Program example</p> <pre> void AD_sample() {     AD0_INITIAL();     AD0_ch0_start(); }  void AD0_INITIAL(void) {     IO_DDR6.bit.D60 = 0; /* DDR6 AN0(P60) input */     IO_ADER.bit.ADE0 = 1; /* AN0 only, A/D input */      IO_ADCR2.byte = 0x70; /* Value is recommended value */                         /* 11 */                         /* 10 */      IO_ADCS1.byte = 0x00; /* Setting value : 00000000 (bit) */                         /* Bit7-6=00: single mode */                         /* Bit5-3=000: AN0 */                         /* Bit2-0=000: AN0 */      IO_ADCS2.byte = 0x00; /*Setting value : 00000000 (bit) */                         /* Bit15=0: (no effect) */                         /* Bit14=0: Clear interrupt request */                         /* Bit13=0: Disable interrupt */                         /* Bit12=0: */                         /* Bit11-10=00: Software trigger */                         /* Bit9=0: */                         /* Bit8=0: "0" Write */      IO_ICR14.byte = 0x00; /* arbitrary value */     __EI();               /* Enable interrupt */ }  void AD0_ch0_start(void) {     IO_ADCS2.byte = 0x20;                         /* Bit14=0: Clear AD0 interrupt flag */                         /* Bit13=1: Enable AD0 interrupt */      IO_ADCS2.byte = 0xA2;                         /* Bit9=1: Start software */ } </pre> |
|---|-------------------------|-------------------------|----------------|-----------|-------------------------|-----------|--|-------------------------|---------------------|-------|---------------------|--------|--------------------|--------|--|-------------------------|---------------------------|-------|-------------------------|--------|---------------------|---------|-------------------|---------|---------------------------|-------|--|-------|--------------------------------|------|---------------------|-------|--|-------|------------------------|---------|--|-------|--|---------------|--|-------------------------|-------------------------|-------|------------|-------|--|-------------------------|-----------------------|-------|--|------|--|-------|--|-------------------------|---------------------|-------|--|-------|---|
|   | Register name, bit name |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set port input  | DDR6, P60               |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Enable A/D input of AN0   | ADER.ADE0               |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | Register name, bit name |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set conversion time   | ADCR2                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set sampling time>>   | .ST1-0                  |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set compare time>>  | .CT1-0                  |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | Register name, bit name |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Control status register 1   | ADCS1                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Select operation mode>>   | .MD1-0                  |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set start channel>>   | .ANS2-0                 |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set end channel>>   | .ANE2-0                 |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Control status register 2   | ADCS2                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .BUSY                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Clear interrupt request flag>>  | .INT                    |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Disable interrupt>>   | .INTE                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .PAUS                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Select start trigger>>  | .STS1-0                 |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .STRT                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .Reserved bit           |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | Register name, bit name |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set A/D interrupt level   | ICR14                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Set I flag  | (CCR)                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | Register name, bit name |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Enable A/D0 interrupt   | ADCS2                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .INT                    |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .INTE                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | Register name, bit name |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
| Start A/D0 software   | ADCS2                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |
|   | .STRT                   |                         |                |           |                         |           |  |                         |                     |       |                     |        |                    |        |  |                         |                           |       |                         |        |                     |         |                   |         |                           |       |  |       |                                |      |                     |       |  |       |                        |         |  |       |  |               |  |                         |                         |       |            |       |  |                         |                       |       |  |      |  |       |  |                         |                     |       |  |       |   |

(Continued)

(Continued)

|  |   |                        |                       |         |                  |                |  |
|--|---|------------------------|-----------------------|---------|------------------|----------------|--|
| <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>• Read conversion value</li> </ul> <table border="1"> <tr> <td>Disable interrupt, clear interrupt request flag</td> <td>ADCS2<br/>.INT<br/>.INTE</td> </tr> </table><br><table border="1"> <tr> <td>Read conversion value</td> <td>ADCR1-2</td> </tr> </table><br><table border="1"> <tr> <td>Enable interrupt</td> <td>ADCS2<br/>.INTE</td> </tr> </table><br><p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>• Set vector table</li> </ul><br><p>Note:<br/>Setting related to clock and setting of _set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</p> | Disable interrupt, clear interrupt request flag | ADCS2<br>.INT<br>.INTE | Read conversion value | ADCR1-2 | Enable interrupt | ADCS2<br>.INTE | <pre> __interrupt void AD0_ch0_int(void) /* */ {     IO_ADCS2.bit.INT = 0; /* Bit14=0: Clear AD0 interrupt flag */     IO_ADCS2.bit.INTE = 0; /* Bit13=0: Disable AD0 interrupt */      [Any storage location] = IO_ADCR1-2; /* Store conversion value */      IO_ADCS2.bit.INTE = 1; /* Bit13=1: Enable AD0 interrupt */ }  #pragma intvect AD0_ch0_int 40 </pre><br><p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
| Disable interrupt, clear interrupt request flag  | ADCS2<br>.INT<br>.INTE                          |                        |                       |         |                  |                |  |
| Read conversion value  | ADCR1-2   |                        |                       |         |                  |                |  |
| Enable interrupt   | ADCS2<br>.INTE                                  |                        |                       |         |                  |                |  |

## ■ Setting Method Other than Program Example

### ● Type of conversion mode and setting method

The following three conversion modes are available.

- Single conversion mode to convert once and terminate
- Continuous conversion mode to convert specified channels
- Stop mode to temporarily stop after specified channels are converted

The conversion mode is set by the conversion mode selection bits (ADCS1. MD[1:0]).

| Operation mode                       | Conversion mode selection bits (MD[1:0])      |
|--------------------------------------|---|
| To set to single conversion mode     | Set to "00 <sub>B</sub> ", "01 <sub>B</sub> " |
| To set to continuous conversion mode | Set to "10 <sub>B</sub> "                     |
| To set to stop mode                  | Set to "11 <sub>B</sub> "                     |

### ● Method to specify bit length

Set by the number of storage bits of conversion result (ADCR.S10).

| Operation mode                            | Number of storage bits of conversion result (S10) |
|---|---|
| To store to ADCR1, 2 register with 10-bit | Set to "0"  |
| To store to ADCR1 register with 8-bit     | Set to "1"  |

### ● Method to select channel

Channel to start the conversion is set by the A/D conversion start channel bits (ADCS1.ANS[2:0]).

Channel to end the conversion is set by the A/D conversion end channel bits (ADCS1.ANE[2:0]).

● Method to enable analog pin input

Set by the analog input enable register (ADER).

| Operation        | Control bit | Setting    |
|------------------|-------------|------------|
| To input AN0 pin | (ADER.ADE0) | Set to "1" |
| To input AN1 pin | (ADER.ADE1) | Set to "1" |
| To input AN2 pin | (ADER.ADE2) | Set to "1" |
| To input AN3 pin | (ADER.ADE3) | Set to "1" |
| To input AN4 pin | (ADER.ADE4) | Set to "1" |
| To input AN5 pin | (ADER.ADE5) | Set to "1" |
| To input AN6 pin | (ADER.ADE6) | Set to "1" |
| To input AN7 pin | (ADER.ADE7) | Set to "1" |

● Method to select start of A/D converter

The following three start triggers are provided.

- Software trigger
- Rising signal of reload timer
- Falling signal of external trigger input

The start trigger is set by the start source select bits (ADCS2.STS[1:0]).

| A/D start source  | Start source select bits (STS[1:0]) |
|---|-------------------------------------|
| To specify software trigger                               | Set to "00 <sub>B</sub> "           |
| To specify external trigger/software trigger              | Set to "01 <sub>B</sub> "           |
| To specify reload timer/software trigger                  | Set to "10 <sub>B</sub> "           |
| To specify external trigger/reload timer/software trigger | Set to "11 <sub>B</sub> "           |

The A/D converter starts with the first source among the selected sources.

● Method to start A/D converter

- Method to generate software trigger

The software trigger is set by the A/D conversion software trigger bit (ADCS2.STRT).

| Operation                    | A/D conversion software trigger bit (STRT) |
|------------------------------|--|
| To generate software trigger | Write "1"                                  |

- Method to start by reload timer 0, reload timer 1

It is necessary to set and start the reload timer. For details, see "CHAPTER 14 16-BIT RELOAD TIMER".

## CHAPTER 17 8/10-BIT A/D CONVERTER

The start trigger occurs due to underflow of the reload timer when output signal of the reload timer is set to rising edge.

- Method to start by external trigger

The external trigger is set by the external trigger input pin (ADTG). The external trigger input pin is set by the data direction bit (DDR9.P93).

| Operation                        | Setting   |
|----------------------------------|---|
| To set ADTG pin to trigger input | Set bit in data direction register (DDR9.P93) to "0". |

- Method to confirm termination of conversion

There are two methods to confirm the termination of conversion.

- Method to confirm using A/D conversion end interrupt request bit (ADCS2.INT).

| (INT)             | Meaning   |
|-------------------|---|
| Read value is "0" | No interrupt request upon termination of A/D conversion |
| Read value is "1" | Interrupt request upon termination of A/D conversion    |

- Method to confirm using operation confirmation bit (ADCS2.BUSY)

| (BUSY)            | Setting                                  |
|-------------------|--|
| Read value is "0" | A/D conversion is terminated (stopping). |
| Read value is "1" | A/D conversion is in progress.           |

- Method to read conversion value

The conversion value can be read by the data registers (ADCR1, ADCR2).

- Method to stop A/D conversion operation forcibly

Set by the forced stop bit (ADCS2.BUSY).

| Operation                                 | Forced stop bit (BUSY) |
|---|------------------------|
| To stop A/D conversion operation forcibly | Write "0"              |

Writing "1" to forced stop bit (BUSY) does not affect the A/D operation.

- Interrupt related register

The relationship between the interrupt level and interrupt vector is shown in the following table.  
For details of the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

| Interrupt vector                    | Interrupt level setting bit   |
|-------------------------------------|---|
| #40<br>Address: FFFF5C <sub>H</sub> | Interrupt control register 14 (ICR14)<br>Address: 0000BE <sub>H</sub> |

- Type of interrupt

The interrupt source is used only when the A/D conversion is ended. There is no bit to be selected.

- Method to enable/disable/clear interrupt

Enabling interrupt is set by the interrupt request enable bit (ADCS2.INTE).

| Content of control           | Interrupt request enable bit (INTE) |
|------------------------------|-------------------------------------|
| To disable interrupt request | Set to "0"                          |
| To enable interrupt request  | Set to "1"                          |

Clearing interrupt request is set by the interrupt request bit (ADCS2.INT).

| Content of control         | Interrupt request bit (INT) |
|----------------------------|-----------------------------|
| To clear interrupt request | Write "0" or start A/D.     |



# CHAPTER 18 EXPANDED I/O SERIAL INTERFACE

---

**This chapter provides an overview, of the expanded I/O serial interface, explains the configuration, interrupt, and its operation, the configuration and functions of its registers.**

---

18.1 Overview of Expanded I/O Serial Interface

18.2 Configuration of Expanded I/O Serial Interface

18.3 Configuration and Functions of Expanded I/O Serial Interface Registers

18.4 Interrupt of Expanded I/O Serial Interface

18.5 Operation of Expanded I/O Serial Interface

18.6 Program Example of Expanded I/O Serial Interface



## 18.1 Overview of Expanded I/O Serial Interface

---

The expanded I/O serial interface is a serial I/O interface with an 8-bit/1-channel configuration that is used to transfer data by clock synchronization. For data transfer, LSB first or MSB first can be selected.

---

### ■ Overview of Expanded I/O Serial Interface

The expanded I/O serial interface has the following two operation modes:

- Internal shift clock mode: This mode transfers data by synchronization with the internal clock.
- External shift clock mode: This mode transfers data by synchronization with a clock that is supplied via an external pin (SCK). Data can also be transferred with instruction of CPU in this mode by using a general-purpose port that shares the external pin (SCK).

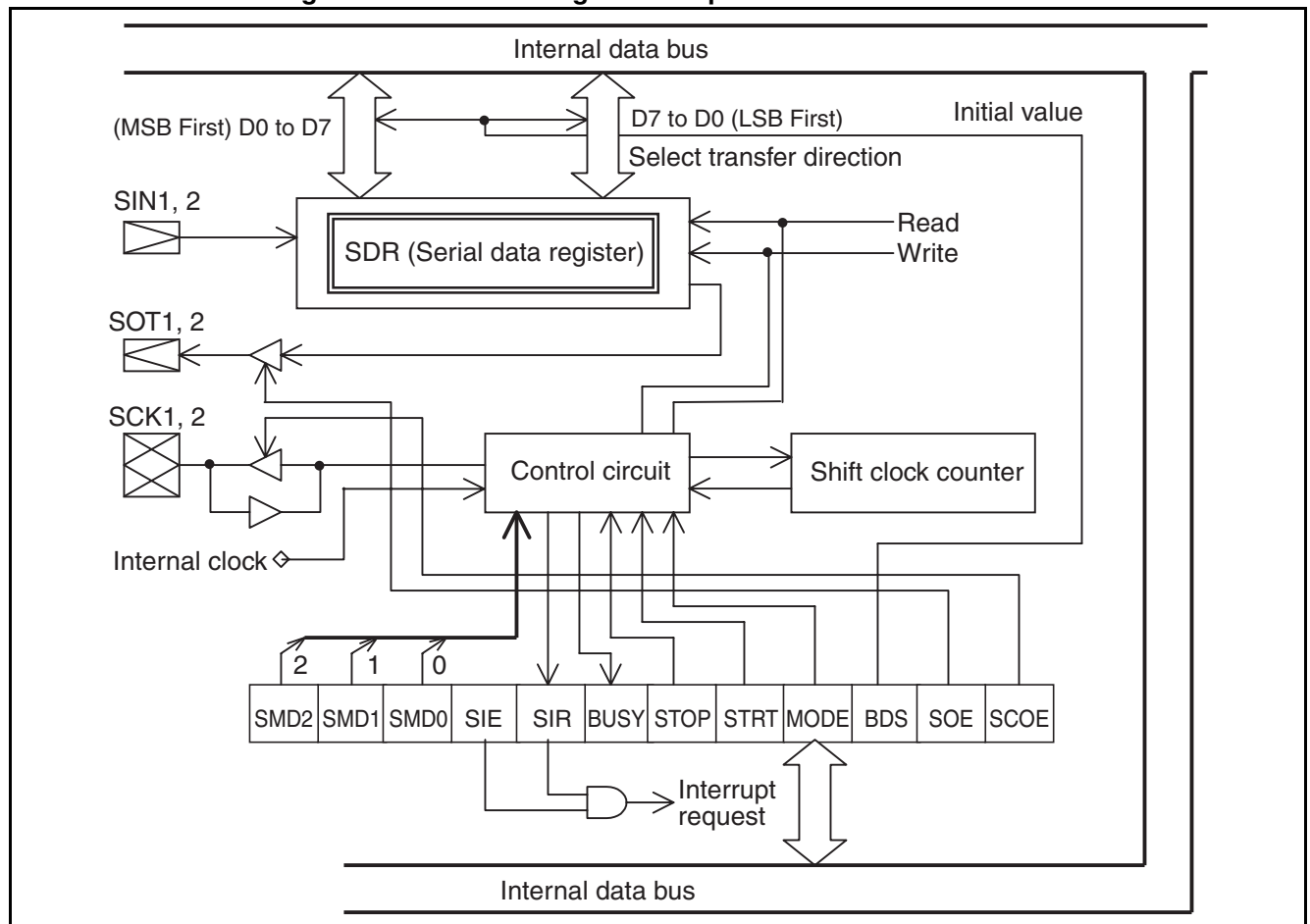
## 18.2 Configuration of Expanded I/O Serial Interface

The expanded I/O serial interface consists of the serial mode control status register and serial data register.

### ■ Block Diagram of Expanded I/O Serial Interface

Figure 18.2-1 shows a block diagram of the expanded I/O serial interface.

**Figure 18.2-1 Block Diagram of Expanded I/O Serial Interface**



### ■ Pin Related to Expanded I/O Serial Interface

The pin related to the expanded I/O serial interface has 2ch of SIN1/SOT1/SCK1, SIN2/SOT2/SCK2 pins. The SIN1/SIN2 pins function as the serial input port, the SOT1/2 pins function as the serial output port, and the SCK1/SCK2 pins function as the external clock input port. The SIN1, SCK1, SIN2, SCK2 pins function as the general-purpose I/O port (P90/SIN1, P92/SCK1, P40/SIN2, P41/SCK2) and the input pin of the expanded I/O serial interface, and the SOT1, SOT2 pins function as the general-purpose I/O port (P91/SOT1, P41/SOT2) and the output pin of the expanded I/O serial interface.

- Setting when using as SIN1/SCK1/SIN2/SCK2 pins

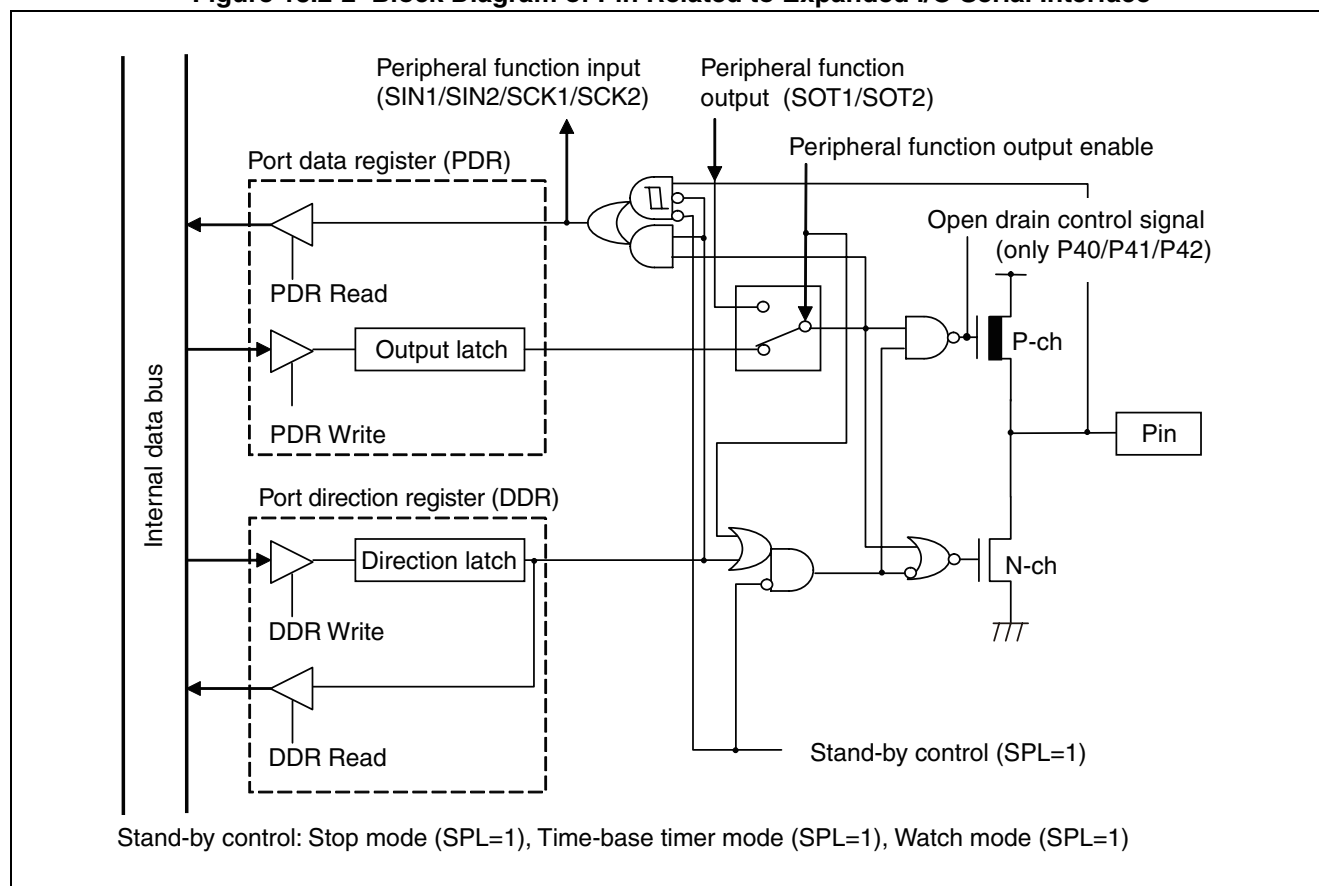
When the SIN1/SCK1/SIN2/SCK2 pins are used as input by the expanded I/O serial interface, P90/SIN1, P92/SCK1, P40/SIN2, P42/SCK2 pins should be set to the input port by the port direction register (DDR9 bit8, 10→ "0" DDR4 bit0, 2→ "0").

- Setting when using as SOT1/SOT2 pins

When the SOT1/SOT2 pins are used as the data output pin by the expanded I/O serial interface, be sure to set the serial mode control status registers 0/1 (SMCS0/SMCS1) to enable serial output (SOE bit1→ "1").

### ■ Block Diagram of Pin Related to Expanded I/O Serial Interface

**Figure 18.2-2 Block Diagram of Pin Related to Expanded I/O Serial Interface**



## 18.3 Configuration and Functions of Expanded I/O Serial Interface Registers

This section describes the configuration and functions of the registers used by the expanded I/O serial interface.

### ■ List of Registers for Expanded I/O Serial Interface

Figure 18.3-1 shows a list of the registers used by the expanded I/O serial interface.

**Figure 18.3-1 List of Registers for the Expanded I/O Serial Interface**

|  |       |      |      |     |       |       |       |       |   |
|--|-------|------|------|-----|-------|-------|-------|-------|---|
| ch.0 Address: 000027 <sub>H</sub><br>ch.1        00002B <sub>H</sub> | 15    | 14   | 13   | 12  | 11    | 10    | 9     | 8     | Serial mode control<br>status register0/1<br>(SMCS0/SMCS1)      |
|  | SMD2  | SMD1 | SMD0 | SIE | SIR   | BUSY  | STOP  | STRT  |   |
| ch.0 Address: 000026 <sub>H</sub><br>ch.1        00002A <sub>H</sub> | 7     | 6    | 5    | 4   | 3     | 2     | 1     | 0     | Serial data register0/1<br>(SDR0/SDR1)                          |
|  | -     | -    | -    | -   | MODE  | BDS   | SOE   | SCOE  |   |
| ch.0 Address: 000028 <sub>H</sub><br>ch.1        00002C <sub>H</sub> | 7     | 6    | 5    | 4   | 3     | 2     | 1     | 0     | Communication prescaler<br>control register0/1<br>(SDCR0,SDCR1) |
|  | D7    | D6   | D5   | D4  | D3    | D2    | D1    | D0    |   |
| ch.0 Address: 000029 <sub>H</sub><br>ch.1        00002D <sub>H</sub> | 15    | 14   | 13   | 12  | 11    | 10    | 9     | 8     | Read/write  |
|  | MD    | -    | -    | -   | DIV3  | DIV2  | DIV1  | DIV0  |   |
|  | (R/W) | (-)  | (-)  | (-) | (R/W) | (R/W) | (R/W) | (R/W) |   |

### 18.3.1 Serial Mode Control Status Register 0/1 (SMCS0/SMCS1)

This section describes the configuration and functions of the serial mode control status register0/1 (SMCS0/SMCS1).

■ Serial Mode Control Status Register0/1 (SMCS0/SMCS1)

The serial mode control status register 0/1 (SMCS0/SMCS1) controls the data transfer mode of serial I/O operations.

The bit configuration of the serial mode control status register 0/1 (SMCS0/SMCS1) is illustrated below.

|                                   |  |      |      |      |     |     |      |      |      |                       |
|-----------------------------------|--|------|------|------|-----|-----|------|------|------|-----------------------|
| SMCS                              |  | 15   | 14   | 13   | 12  | 11  | 10   | 9    | 8    | Initial value         |
| ch.0 address: 000027 <sub>H</sub> |  | SMD2 | SMD1 | SMD0 | SIE | SIR | BUSY | STOP | STRT |                       |
| ch.1 address: 00002B <sub>H</sub> |  | R/W  | R/W  | R/W  | R/W | R/W | R    | R/W  | R/W  | 00000010 <sub>B</sub> |

|                                   |  |   |   |   |   |      |     |     |      |                        |
|-----------------------------------|--|---|---|---|---|------|-----|-----|------|------------------------|
| SMCS                              |  | 7 | 6 | 5 | 4 | 3    | 2   | 1   | 0    | Initial value          |
| ch.0 address: 000026 <sub>H</sub> |  | - | - | - | - | MODE | BDS | SOE | SCOE |                        |
| ch.1 address: 00002A <sub>H</sub> |  |   |   |   |   | R/W  | R/W | R/W | R/W  | --- -0000 <sub>B</sub> |

The functions of the bits of the serial mode control status register 0/1 (SMCS0/SMCS1) are described below.

[bit15, bit14, bit13] SMD2, SMD1 and SMD0: Serial Shift Clock Mode (Shift clock select)

These bits select the serial shift clock mode. Table 18.3-1 shows the settings for the serial shift clock mode.

Table 18.3-1 Settings of Serial Shift Clock Mode

Selection of serial shift clock mode

| SMD2 | SMD1 | SMD0 | $\phi$ =16MHz<br>div=8    | $\phi$ =8MHz<br>div=4 | $\phi$ =4MHz<br>div=4 | Division<br>value |
|------|------|------|---------------------------|-----------------------|-----------------------|-------------------|
| 0    | 0    | 0    | 1MHz                      | 1MHz                  | 500kHz                | 2                 |
| 0    | 0    | 1    | 500kHz                    | 500kHz                | 250kHz                | 4                 |
| 0    | 1    | 0    | 125kHz                    | 125kHz                | 62.5kHz               | 16                |
| 0    | 1    | 1    | 62.5kHz                   | 62.5kHz               | 31.2kHz               | 32                |
| 1    | 0    | 0    | 31.2kHz                   | 31.2kHz               | 15.6kHz               | 64                |
| 1    | 0    | 1    | External shift clock mode |                       |                       |                   |
| 1    | 1    | 0    | reserved                  |                       |                       |                   |
| 1    | 1    | 1    | reserved                  |                       |                       |                   |

Setting of communication prescaler (SDCR)

| Div | (Machine clock) |      |      |      |      | Recommended<br>machine clock<br>cycle |
|-----|-----------------|------|------|------|------|---------------------------------------|
|     | MD              | DIV3 | DIV2 | D1IV | DIV0 |                                       |
| 1   | 1               | 0    | 0    | 0    | 0    | 2MHz                                  |
| 2   | 1               | 0    | 0    | 0    | 1    | 4MHz                                  |
| 3   | 1               | 0    | 0    | 1    | 0    | 6MHz                                  |
| 4   | 1               | 0    | 0    | 1    | 1    | 8MHz                                  |
| 5   | 1               | 0    | 1    | 0    | 0    | 10MHz                                 |
| 6   | 1               | 0    | 1    | 0    | 1    | 12MHz                                 |
| 7   | 1               | 0    | 1    | 1    | 0    | 14MHz                                 |
| 8   | 1               | 0    | 1    | 1    | 1    | 16MHz                                 |

At reset, the settings will be initialized to "000<sub>B</sub>". These bits cannot be rewritten while data transfer is in progress.

The shift clock can be selected from among five internal clocks and one external clock. The combinations of SMD2, SMD1, SMD0 = 110<sub>B</sub> and 111<sub>B</sub> are reserved and must not be set.

Shifting can also be performed for individual instructions by setting SCOE = 0 to select the clock and using a port that shares the pins SCK1 and SCK2.

## [bit12] SIE: Serial I/O Interrupt Enable (Serial I/O interrupt enable)

This bit controls serial I/O interrupt requests as shown below.

|   |  |
|---|--|
| 0 | Serial I/O interrupts prohibited (initial value) |
| 1 | Serial I/O interrupts allowed                    |

- This bit is initialized to "0" at reset
- This bit can be read and written.

## [bit11] SIR: Serial I/O Interrupt Request (Serial I/O interrupt request)

This bit is set to "1" when serial data transfer ends. When this bit becomes "1" in the interrupt enabled state (SIE = 1), an interrupt request to the CPU will be generated. The condition for clearing this bit depends on the setting of the MODE bit:

- Cleared by setting the SIR bit to "0" in a write operation when the MODE bit is "0".
- Cleared by reading or writing to the SDR, when the MODE bit is "1".
- Cleared by reset or writing "1" for the STOP bit regardless of the value of the MODE bit.
- Writing "1" for this bit has no effect.
- Reading by read-modify-write instructions always read "1".

## [bit10] BUSY (transfer status display)

This bit indicates whether serial transfer is currently being executed.

| BUSY | Operation   |
|------|---|
| 0    | Operation is stopped or the serial data register is in the R/W wait state (initial value) |
| 1    | Serial transfer take place  |

- This bit is initialized to "0" at reset.
- This bit can only be read.

**[bit9] STOP (Stop bit)**

This bit is used to forcibly interrupt serial transfer. Setting this bit to "1" will result in operation stop.

| STOP | Operation  |
|------|--|
| 0    | Normal operation                                     |
| 1    | Stop of transfer because of STOP = 1 (initial value) |

- This bit is initialized to "1" at reset.
- This bit can be read and written.

**[bit8] STRT: Start (start bit)**

This bit is used to start serial transfer. Write "1" in the stopped state to start transfer. Writing "1" will be ignored during serial transfer operation or in serial shift register R/W wait state.

- Writing "0" has no effect.
- Read operations always read "0".

**[bit3] MODE (serial mode selection)**

This bit selects the condition to start from the stopped state. Rewriting this bit during operation is prohibited.

| MODE | Operation   |
|------|---|
| 0    | STRT = 1 will start operation (initial value)       |
| 1    | Started by reading/writing the serial data register |

- This bit is initialized to "0" at reset.
- This bit can be read and written.
- This bit is set to "1" at  $\mu$ DMAC start.

**[bit2] BDS: Bit Direction Select (selection of transfer direction)**

This bit selects whether to start transfer with the LSB side (LSB first) or MSB side (MSB first) during input and output of serial data.

|   |                           |
|---|---------------------------|
| 0 | LSB first (initial value) |
| 1 | MSB first                 |

- This bit is initialized to "0" at reset.
- This bit can be read and written.

---

Note:

Specify the transfer direction before writing data to the SDR.

---

**[bit1] SOE: Serial Out Enable (enable serial output)**

This bit controls the output of the external output pins (SOT1 and SMD2) for serial I/O.

|   |  |
|---|--|
| 0 | General-purpose port pin (initial value) |
| 1 | Serial data output                       |

- This bit is initialized to "0" at reset.
- This bit can be read and written.

**[bit0] SCOE: SCK1 Output Enable (enable shift clock output)**

This bit controls output of the external input/output pins (SCK1 and SCK2) for the shift clock.

|   |   |
|---|---|
| 0 | Use of general-purpose port pins, transfer for each instruction (initial value) |
| 1 | Shift clock output pin  |

Set to "0" when transferring data for each instruction in external shift clock mode.

- This bit is initialized to "0" at reset.
- This bit can be read and written.



18.3.2 Serial Data Register 0/1 (SDR0/SDR1)

This section describes the configuration and functions of the serial data register 0/1 (SDR0/SDR1).

Serial Data Register 0/1 (SDR0/SDR1)

The bit configuration of the serial data register 0/1 (SDR0/SDR1) is illustrated below.

|                                   |                     |     |     |     |     |     |     |     |     |                       |
|-----------------------------------|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
| SDR0/SDR1                         |                     | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Initial value         |
| ch.0 address: 000028 <sub>H</sub> |                     | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  | XXXXXXXX <sub>B</sub> |
| ch.1                              | 00002C <sub>H</sub> | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | (Undefined)           |

The serial data register 0/1 (SDR0/SDR1) stores transfer data of the serial I/O unit.

The SDR cannot be written or read during data transfer.

### 18.3.3 Communication Prescaler Control Register0/1 (SDCR0/SDCR1)

This section describes the configuration and functions of the communication prescaler control register0/1 (SDCR0/SDCR1).

#### ■ Communication Prescaler Control Register0/1 (SDCR0/SDCR1)

The bit configuration of the communication prescaler control register0/1 (SDCR0/SDCR1) is illustrated below.

|                                   |     |   |   |   |      |      |      |      |                       |
|-----------------------------------|-----|---|---|---|------|------|------|------|-----------------------|
| SDCR0/SDCR1                       | 7   | 6 | 5 | 4 | 3    | 2    | 1    | 0    | Initial value         |
| ch.0 address: 000029 <sub>H</sub> | MD  | - | - | - | DIV3 | DIV2 | DIV1 | DIV0 | 0---0000 <sub>B</sub> |
| ch.1 address: 00002D <sub>H</sub> | R/W | - | - | - | R/W  | R/W  | R/W  | R/W  |                       |

The functions of the bits for the communication prescaler control register0/1 (SDCR0/SDCR1) are described below.

#### [bit15] MD: Machine clock divide moDe select

This bit is used to enable operation of the communication prescaler.

|   |                                       |
|---|---------------------------------------|
| 0 | The communication prescaler stops.    |
| 1 | The communication prescaler operates. |

#### [bit11, bit10, bit9, bit8] DIV3, DIV2, DIV1, DIV0: DIVide3 to 0

These bits determine the division ratio of the machine clocks.

| DIV3 to DIV0      | Division Ratio |
|-------------------|----------------|
| 0000 <sub>B</sub> | Division by 1  |
| 0001 <sub>B</sub> | Division by 2  |
| 0010 <sub>B</sub> | Division by 3  |
| 0011 <sub>B</sub> | Division by 4  |
| 0100 <sub>B</sub> | Division by 5  |
| 0101 <sub>B</sub> | Division by 6  |
| 0110 <sub>B</sub> | Division by 7  |
| 0111 <sub>B</sub> | Division by 8  |

#### Note:

When changing the clock division ratio, wait for time of 2 division as a clock stabilization time before the communication is performed.

## 18.4 Interrupt of Expanded I/O Serial Interface

The interrupt of the expanded I/O serial interface occurs when the data transfer is terminated.

The interrupt of the expanded I/O serial interface can activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt of Expanded I/O Serial Interface

The following table shows the interrupt control bit and interrupt source of the expanded I/O serial interface.

|                                     | Interrupt of serial I/O                          |
|-------------------------------------|--|
| Interrupt request flag              | SMCS0:SIR (bit11) ch.0<br>SMCS1:SIR (bit11) ch.1 |
| Interrupt request output enable bit | SMCS0:SIE (bit12) ch.0<br>SMCS1:SIE (bit12) ch.1 |
| Interrupt generation source         | Terminate transfer of serial data                |

### ■ Interrupt Source Related to Expanded I/O Serial Interface

The interrupt of the expanded I/O serial interface occurs when the data transfer is terminated. The interrupt request to CPU is executed when the SIE (bit12) flag in the serial mode control status register (SMCS) is set and SIE (bit11): interrupt enable is "1".

### ■ Interrupt of Expanded I/O Serial Interface, DMA Transfer, and EI<sup>2</sup>OS

Table 18.4-1 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 18.4-1 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|------------------|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|                  |                          |                      | Number           | Address             | Number                     | Address             |
| SIO1             | ○                        | 13                   | #37              | FFFF68 <sub>H</sub> | ICR13                      | 0000BD <sub>H</sub> |
| SIO2             | ○                        | 14                   | #38              | FFFF64 <sub>H</sub> |                            |                     |

○: Interrupt request flag is cleared.

Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing.

### ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The expanded I/O serial interface corresponds to the DMA transfer function and EI<sup>2</sup>OS function. When the DMA or EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 18.5 Operation of Expanded I/O Serial Interface

---

The expanded I/O serial interface consists of the serial mode control status register (SMCS) and shift register (SDR). This interface is used for input and output of 8-bit serial data.

This section describes the operations of the expanded I/O serial interface.

---

### ■ Overview of Operation for Expanded I/O Serial Interface

Input and output with serial data are individually performed as follows:

#### ○ Serial data input

By synchronizing with the rising edge of a serial shift clock (external or internal clock), data is input to the SDR (serial data register) from a serial input pin (Pin SIN1).

The shift direction (data transfer beginning with the MSB or LSB) is specified by the direction specify bit (BDS) of the serial mode control status register (SMCS).

After data transfer is completed, the operation enters the stop state or data register R/W wait state as determined by the MODE bit of the serial mode control status register (SMCS). To change the state from each state to transfer state, to do the following setting.

- To return from the stop state, write "0" to the STOP bit and write "1" to the STRT bit to set it. (STOP and STRT can be set simultaneously)
- To return from the wait state, perform a read or write operation for the data register.

#### ○ Serial data output

By synchronizing the shift register with the falling edges of a serial shift clock (external or internal clock), data is output from the serial output pin (SOT1 pin).

## 18.5.1 Shift Clock Modes

The shift clock has two modes, the internal shift clock mode and the external shift clock mode. These two modes are specified by the setting of the SMCS. Change the mode only when the serial I/O interface is not operating. This condition can be determined by reading the BUSY bit.

### ■ Internal Shift Clock Mode

This mode uses an internal clock, and a shift clock with a duty ratio of 50% is output via the SCK pin for synchronous timing output.

One bit of data is transferred for each clock. The data transfer speed can be calculated as follows:

$$\text{Transfer speed (S)} = \frac{A}{\text{Internal clock machine cycle (A)}}$$

"A" is the following division ratio selected by the SMD bit of SMCS.

$(\phi \div \text{div})/2$ ,  $(\phi \div \text{div})/2^2$ ,  $(\phi \div \text{div})/2^4$ ,  $(\phi \div \text{div})/2^5$ ,  $(\phi \div \text{div})/2^6$

### ■ External Shift Clock Mode

In synchronization with an external shift clock supplied via the SCK pin, one bit of data is transferred for each clock.

Data can be transferred at a speed up to 1/(8 machine cycles) from DC. For example, data can be transferred at a speed of up to 2 MHz when one machine cycle is 62.5 ns.

Transfer for individual instructions can be achieved making the following settings:

- Select the external shift clock mode and set the SCOE bit of SMCS to "0".
- Write "1" to the direction register of the port that shares the SCK pin, then set the port to output mode.

After making the above settings, write "1" and "0" to the port data register (PDR) to obtain the value of the port that is output to the SCK pin for supplying the external clock for data transfer. Start the shift clock as soon as the "H" level is input.

Note:

Writing to the SMCS and SDR during serial I/O operation is prohibited.

## 18.5.2 Operational States of Serial I/O Units

Four serial I/O states are used, namely, STOP, Halt, SDR R/W Wait, and Transfer.

### ■ Operational States of Serial I/O Units

#### ○ STOP State

The shift counter is initialized to SIR=0 at reset or by writing "1" to the STOP bit of SMCS.

To return from the STOP state, set STOP = 0 and STRT = 1 (these bits can be set simultaneously). Because the STOP bit has a higher priority than the STRT bit, data transfer will not be executed even when STRT = 1 is set while STOP = 1.

#### ○ Halt State

When the MODE bit is set to "0", the BUSY and SIR bits of SMCS will become BUSY = 0 and SIR = 1 after data transfer ends. The counter will then be initialized and set to the Halt state. For returning from the Halt state, set STRT = 1 to resume data transfer operation.

#### ○ Serial Data Register R/W Wait State

When the MODE bit of SMCS is "1", serial transfer ends, this will result in BUSY = 0 and SIR = 1 and the serial data register R/W wait state will be entered. If the interrupt enable register is set to "enable", the applicable block will issue an interrupt signal.

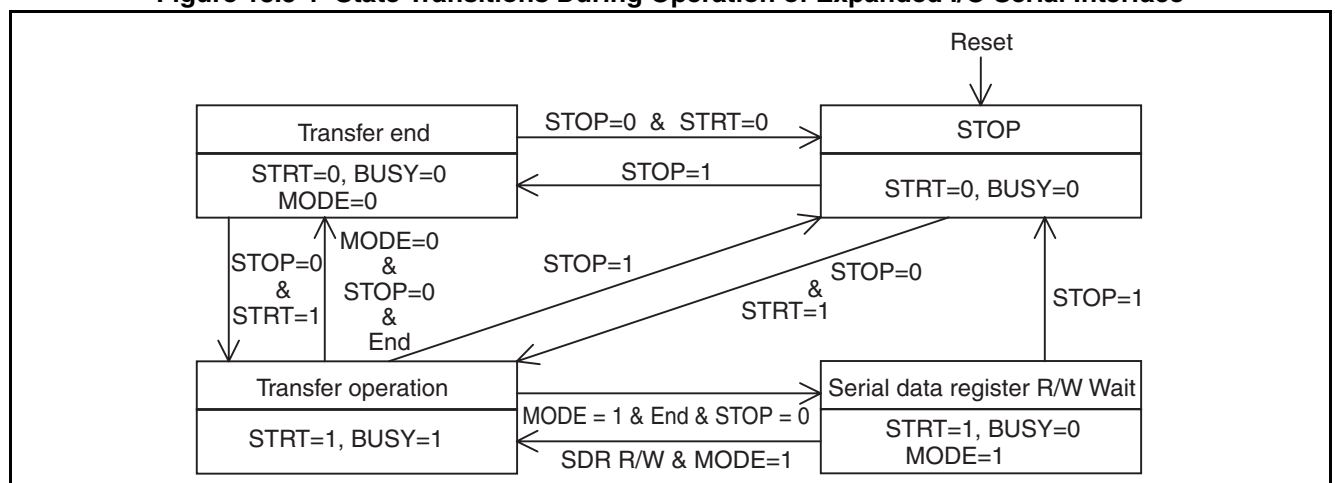
When returning from the R/W wait state, the BUSY becomes BUSY = 1 and data transfer operation will be resumed as soon as a read or write operation is performed for the serial data register.

#### ○ Transfer State

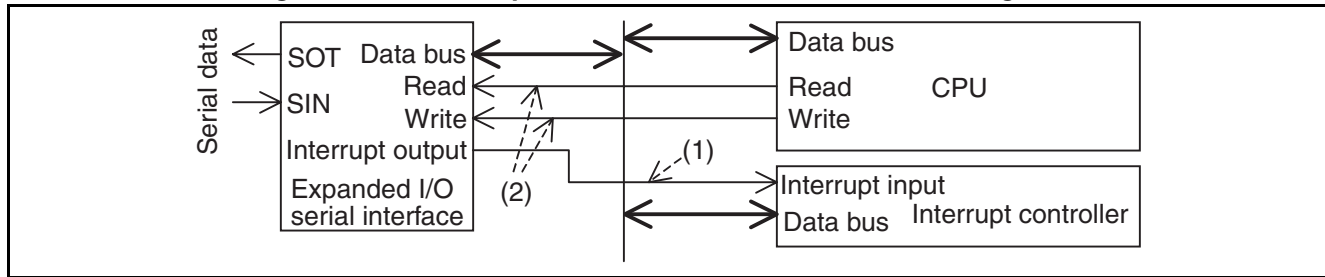
In this state, serial transfer is performed if BUSY = 1. Depending on the setting of the MODE bit, the Halt or R/W wait state will be entered.

Figure 18.5-1 shows the state transitions during operation. Figure 18.5-2 illustrates the concept of reading and writing the serial data register.

**Figure 18.5-1 State Transitions During Operation of Expanded I/O Serial Interface**



**Figure 18.5-2 Concept of Read and Write for Serial Data Registers**



- (1) For MODE = 1, data transfer is ended by the shift clock counter. A read/write wait state will be entered after SIR is set to 1. If the SIE bit is "1", an interrupt signal is generated. An interrupt signal will not be generated; however, if the SIE is inactive or when data transfer is stopped by setting the STOP bit by writing "1".
- (2) As soon as the serial data register is read or written, the interrupt request will be cleared and serial transfer will start.

### 18.5.3 Start/Stop Timing and Input/Output Timing of Shift Operation

Start/stop timing and input/output timing of the shift operation are described below.

■ Start/Stop Timing and Input/Output Timing of Shift Operation

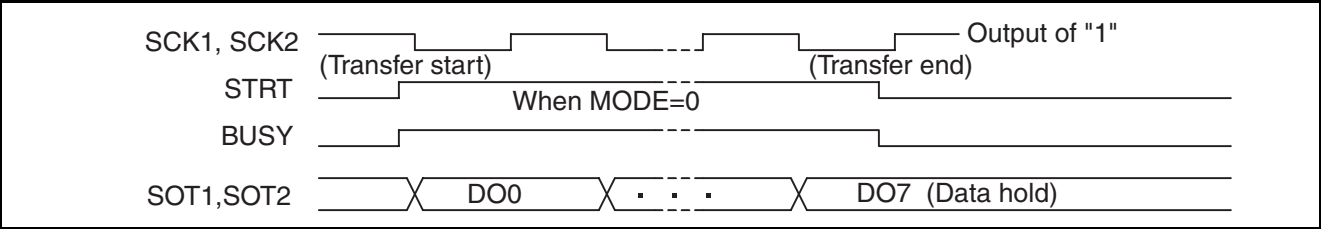
- Start  
Set the STOP bit and STRT bit of SMCS to "0" and "1", respectively.
- Stop
  - Operation is stopped by the end of data transfer or as soon as STOP = 1.
  - If operation stopped because of STOP = 1: Stop is performed while SIR is "0", regardless of the MODE bit.
  - For an operation stop because of the end of data transfer: SIR is set to "1" and data transfer is stopped regardless of the MODE bit.

Irrespective of the MODE bit, the BUSY bit becomes "1" in the serial data transfer state and "0" during the stop state or R/W wait state. Read this bit for checking the data transfer state.

Timing charts illustrating the transfer operation in various modes are provided below. D07 to D00 in the diagram represent output data.

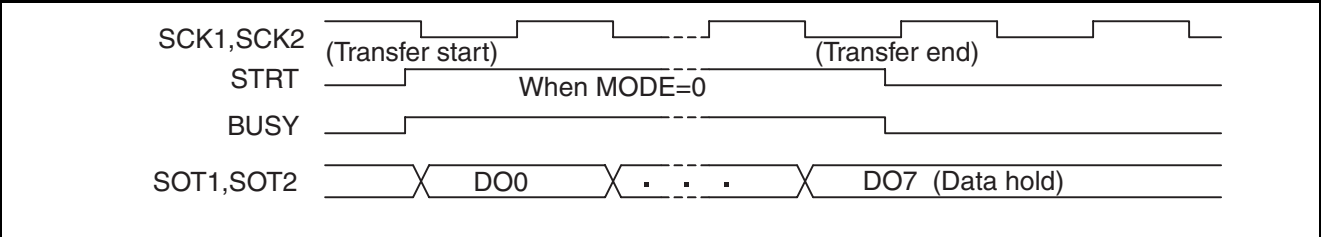
○ Internal shift clock mode (LSB First)

Figure 18.5-3 Start/stop Timing in Shift Operation (Using the Internal Clock)



○ External shift clock mode (LSB first)

Figure 18.5-4 Start/stop Timing in Shift Operation (Using the External Clock)

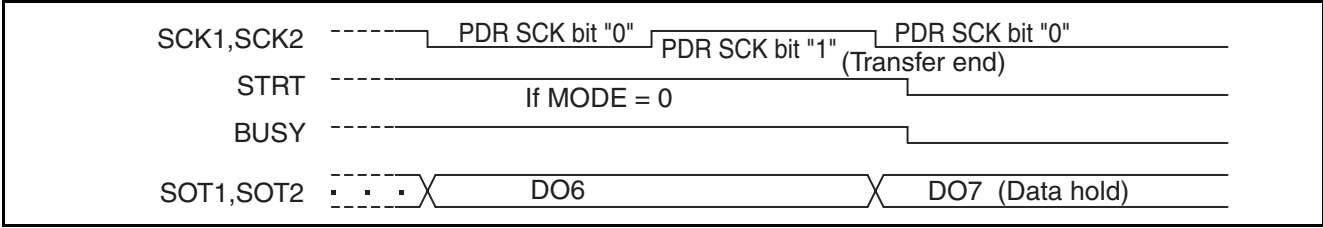




○ **Instruction shift in external shift clock mode (LSB first)**

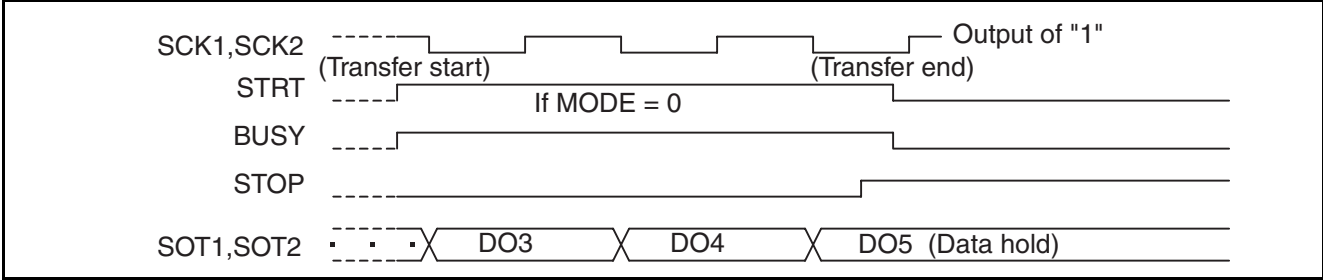
During instruction shift, "H" will be output if the bit corresponding to SCK in PDR is set to "1" and "L" will be output if the bit is set to "0". (If SCOE = 0 when external shift clock mode is selected.)

**Figure 18.5-5 Instruction Shift in External Shift Clock Mode**



○ **Stop by STOP = 1 (LSB first, internal clock used)**

**Figure 18.5-6 Stop Timing When the STOP Bit is Set to "1"**

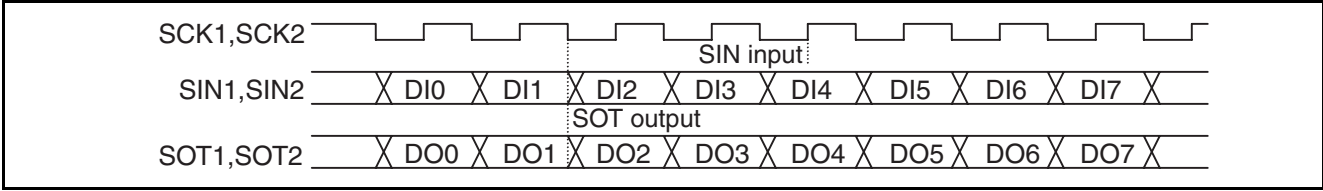


■ **Operation During Serial Data Transfer**

During serial data transfer, data from the serial output pin (SOT2) is output at a falling edge of the shift clock. Data from the serial input pin (SIN) is input at a rising edge.

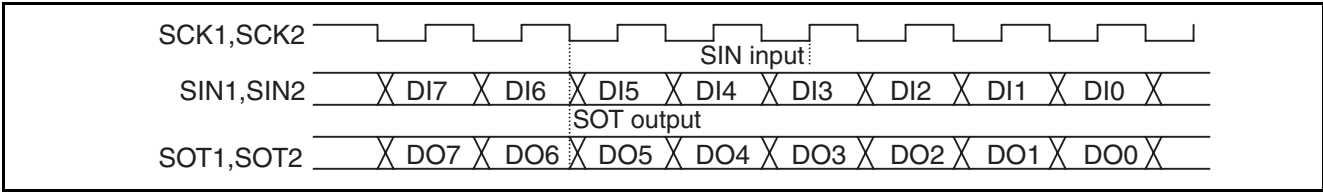
○ **LSB first (if the BDS bit is "0")**

**Figure 18.5-7 Input and Output Shift Timing (LSB First)**



○ **MSB first (If the BDS bit is "1")**

**Figure 18.5-8 Input and Output Shift Timing (MSB First)**



### 18.5.4 Interrupt Function

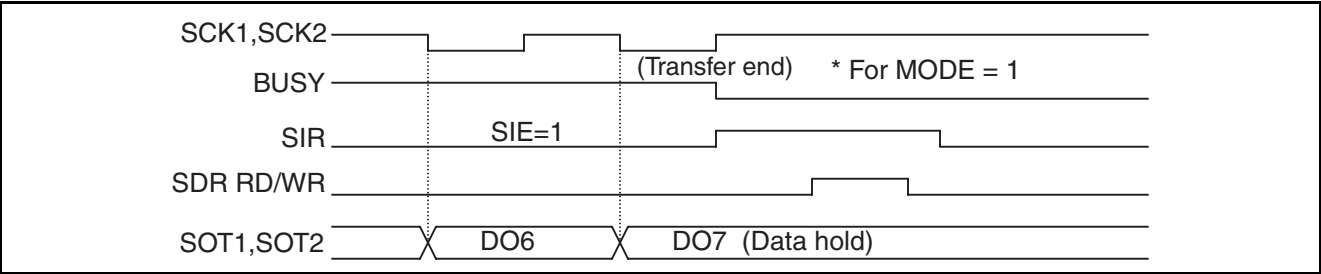
The expanded I/O serial interface generates the interrupt requests for the CPU.

■ Interrupt Function of Expanded I/O Serial Interface

An interrupt request is output to the CPU when the SIR bit, which acts as an interrupt flag, is set at the end of data transfer provided that the SIE bit of the SMCS, which enables interrupts, is "1".

Figure 18.5-9 shows the timing for output of interrupt signals.

Figure 18.5-9 Timing for Interrupt Signal Output



## 18.6 Program Example of Expanded I/O Serial Interface

This section describes the program example of the expanded I/O serial interface.

### ■ Program Example of Expanded I/O Serial Interface

| <p>Example of setting procedure</p> <p>Send data of 1 byte with sio1 (ch.0)</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Control SIO</li> </ul> <table border="1"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Set mode register</td> <td>SMCS0</td> </tr> <tr> <td>Select shift clock &gt;&gt;</td> <td>.SMD2-0</td> </tr> <tr> <td>Set interrupt enable &gt;&gt;</td> <td>.SIE</td> </tr> <tr> <td>Set interrupt request &gt;&gt;</td> <td>.SIR</td> </tr> <tr> <td>Display transfer state &gt;&gt;</td> <td>.BUSY</td> </tr> <tr> <td>Set SOT1, 2 pins &gt;&gt;</td> <td>.SOE</td> </tr> <tr> <td>Set SCK1, 2 pins &gt;&gt;</td> <td>.SCOE</td> </tr> <tr> <td>Set control register</td> <td>SDCR0</td> </tr> <tr> <td>Set communication prescaler &gt;&gt;</td> <td>.MD</td> </tr> <tr> <td>Set division ratio &gt;&gt;</td> <td>.DIV3-0</td> </tr> </tbody> </table> <ul style="list-style-type: none"> <li>Interrupt related</li> </ul> <table border="1"> <tbody> <tr> <td>Set interrupt level</td> <td>ICR13</td> </tr> <tr> <td>Set I flag</td> <td>(CCR)</td> </tr> </tbody> </table> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start SIO</li> </ul> <table border="1"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Send any data</td> <td>SDR0</td> </tr> <tr> <td>Start SIO operation</td> <td>SMCS0 .STRT</td> </tr> </tbody> </table> <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Transmission interrupt processing</li> </ul> <table border="1"> <thead> <tr> <th></th> <th>Register name, bit name</th> </tr> </thead> <tbody> <tr> <td>Initialize interrupt flag</td> <td>SMCS0 .SIR</td> </tr> </tbody> </table> <p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>Note:<br/>Setting related to clock and setting of __set_il (numeric value) are required in advance.<br/>See the chapter of clock and interrupt.</p> |                         | Register name, bit name | Set mode register | SMCS0 | Select shift clock >> | .SMD2-0 | Set interrupt enable >> | .SIE | Set interrupt request >> | .SIR | Display transfer state >> | .BUSY | Set SOT1, 2 pins >> | .SOE | Set SCK1, 2 pins >> | .SCOE | Set control register | SDCR0 | Set communication prescaler >> | .MD | Set division ratio >> | .DIV3-0 | Set interrupt level | ICR13 | Set I flag | (CCR) |  | Register name, bit name | Send any data | SDR0 | Start SIO operation | SMCS0 .STRT |  | Register name, bit name | Initialize interrupt flag | SMCS0 .SIR | <p>Program example</p> <pre> void sio_sample(void) {     sio_initial();     sio_start(); }  void sio_initial(void) {     IO_SMCS0.word = 0x3003; /* Setting value = 0011_0000_0000_0011 */                                 /* bit15-13 = 001 4 division */                                 /* bit12 = 1 Enable interrupt */                                 /* bit11 = 0 Clear interrupt request */                                 /* bit10 = 0 R/W wait state */                                 /* bit1 = 1 Enable SOE serial output */                                 /* bit0 = 1 Enable SCOE shift clock output */     IO_SDCR0.byte = 0x83; /* Setting value =1000_0011 */                                 /* bit15 = 1 Enable communication prescaler */                                 /* bit11-8 = 11 4 division */      IO_ICR13.byte = 0x10; /* Set SIO transmission completion interrupt level                                 (arbitrary value) */     __EI(); /* Enable interrupt */ }  void sio_start(void) {     IO_SDR0 = 0xaa; /* Send any value of data */     IO_SMCS0.bit.STRT = 1; /* bit1 = 1 Enable SIO operation */ }  __interrupt void sio_int(void) {     IO_SMCS0.bit.SIR = 0; /* bit0 = 0 Initialize SIR interrupt flag */ }  #pragma intvect sio_int 37 </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
|---|-------------------------|-------------------------|-------------------|-------|-----------------------|---------|-------------------------|------|--------------------------|------|---------------------------|-------|---------------------|------|---------------------|-------|----------------------|-------|--------------------------------|-----|-----------------------|---------|---------------------|-------|------------|-------|--|-------------------------|---------------|------|---------------------|-------------|--|-------------------------|---------------------------|------------|---|
|   | Register name, bit name |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set mode register   | SMCS0                   |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Select shift clock >>   | .SMD2-0                 |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set interrupt enable >>   | .SIE                    |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set interrupt request >>  | .SIR                    |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Display transfer state >>   | .BUSY                   |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set SOT1, 2 pins >>   | .SOE                    |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set SCK1, 2 pins >>   | .SCOE                   |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set control register  | SDCR0                   |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set communication prescaler >>  | .MD                     |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set division ratio >>   | .DIV3-0                 |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set interrupt level   | ICR13                   |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Set I flag  | (CCR)                   |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
|   | Register name, bit name |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Send any data   | SDR0                    |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Start SIO operation   | SMCS0 .STRT             |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
|   | Register name, bit name |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |
| Initialize interrupt flag   | SMCS0 .SIR              |                         |                   |       |                       |         |                         |      |                          |      |                           |       |                     |      |                     |       |                      |       |                                |     |                       |         |                     |       |            |       |  |                         |               |      |                     |             |  |                         |                           |            |   |

## ■ Setting Method Other than Program Example

### ● Type of operation clock and selection method

There are two operation clocks: internal timer and external clock.

Set by the shift clock select bits (SMCS0.SMD[2:0], SMCS1.SMD[2:0]).

| Content of control       | Shift clock select bits (SMD[2:0])                |
|--------------------------|---|
| To select internal timer | Set to "000 <sub>B</sub> " to "100 <sub>B</sub> " |
| To select external clock | Set to "101 <sub>B</sub> "                        |

### ● Method to control SCK, SIN, SOT pins

Set by SIO1, SIO2.

| Operation         | SIO1           | SIO2           |
|-------------------|----------------|----------------|
| To input SCK pin  | DDR9.P92 = 0   | DDR4.P42 = 0   |
| To output SCK pin | SMCS0.SCOE = 1 | SMCS1.SCOE = 1 |
| To input SIN pin  | DDR9.P90 = 0   | DDR4.P40 = 0   |
| To output SOT pin | SMCS0.SOE = 1  | SMCS1.SOE = 1  |

### ● Method to enable/stop SIO operation

Set by the start bit (SMCS0.STRT, SMCS1.STRT).

| Content of control    | Start bit (STRT) |
|-----------------------|------------------|
| Start serial transfer | Set to "1"       |

Set by the stop bit (SMCS0.STOP, SMCS1.STOP).

| Content of control   | Stop bit (STOP) |
|----------------------|-----------------|
| Stop serial transfer | Set to "1"      |

### ● Method to set transfer direction

Set by the setting direction selection bit (SMCS0.BDS, SMCS1.BDS).

The transfer direction of LSB/MSB can be selected from any operation mode.

| Content of control                                  | Setting direction selection bit (BDS) |
|---|---------------------------------------|
| To set to LSB transfer (from least significant bit) | Set to "0"                            |
| To set to MSB transfer (from most significant bit)  | Set to "1"                            |

## CHAPTER 18 EXPANDED I/O SERIAL INTERFACE

- Interrupt related register

The relationship between the SIO number, interrupt level, and interrupt vector is shown in the following table.

For details of the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

|      | Interrupt vector                     | Interrupt level setting register                                  |
|------|--------------------------------------|---|
| SIO1 | #37<br>Address : FFFF68 <sub>H</sub> | Interrupt level register (ICR13)<br>Address : 0000BD <sub>H</sub> |
| SIO2 | #38<br>Address : FFFF64 <sub>H</sub> | Interrupt level register (ICR13)<br>Address : 0000BD <sub>H</sub> |

- Type of interrupt

One interrupt is provided. It occurs when transfer of the serial data is terminated.

- Method to enable/disable/clear interrupt

Enabling/disabling interrupt is set by the interrupt request enable bit (SMCS0.SIE, SMCS1.SIE).

| Content of control           | Interrupt request enable bit (SIE) |
|------------------------------|------------------------------------|
| To disable interrupt request | Set to "0"                         |
| To enable interrupt request  | Set to "1"                         |

Clearing interrupt request is set by the interrupt request flag (SMCS0.SIR, SMCS1.SIR).

| Content of control         | Interrupt request flag (SIR) |
|----------------------------|------------------------------|
| To clear interrupt request | Set to "0"                   |

# CHAPTER 19    UART

---

**This chapter provides an overview, of the UART, explains the configuration, interrupt, its operation, the configuration and functions of its registers shows the precautions on use, and program example of the UART.**

---

- 19.1 Overview of the UART
- 19.2 Configuration of UART
- 19.3 Configuration and Functions of UART Registers
- 19.4 Interrupt of UART
- 19.5 UART Operations
- 19.6 Precautions on use of the UART
- 19.7 Program Example of UART

## 19.1 Overview of the UART

---

**The UART is a serial I/O port for asynchronous (start-stop synchronization) communications or CLK synchronous communication.**

---

### ■ UART Features

The UART has the following features:

- Built-in full-duplex double buffer
- Both asynchronous (start-stop synchronization) and CLK synchronous communication (no start bit and stop bit) are available
- Support of multiprocessor mode
- Built-in dedicated baud rate generator
  - In asynchronous: 76923/38461/19230/9615/500K/250Kbps
  - In CLK synchronous: 16M/8M/4M/2M/1M/500Kbps
- Free baud rate setting via external clock
- Internal clock supplied by PPG1 can be used
- Data length: 7 bits (asynchronous normal mode only)/8 bits
- Master/slave communication function (in multiprocessor mode): 1 (master) to n (slaves) communication enabled
- Error detection function (parity, framing, overrun)
- Transfer signal: NRZ code
- DMA support (reception/transmission)

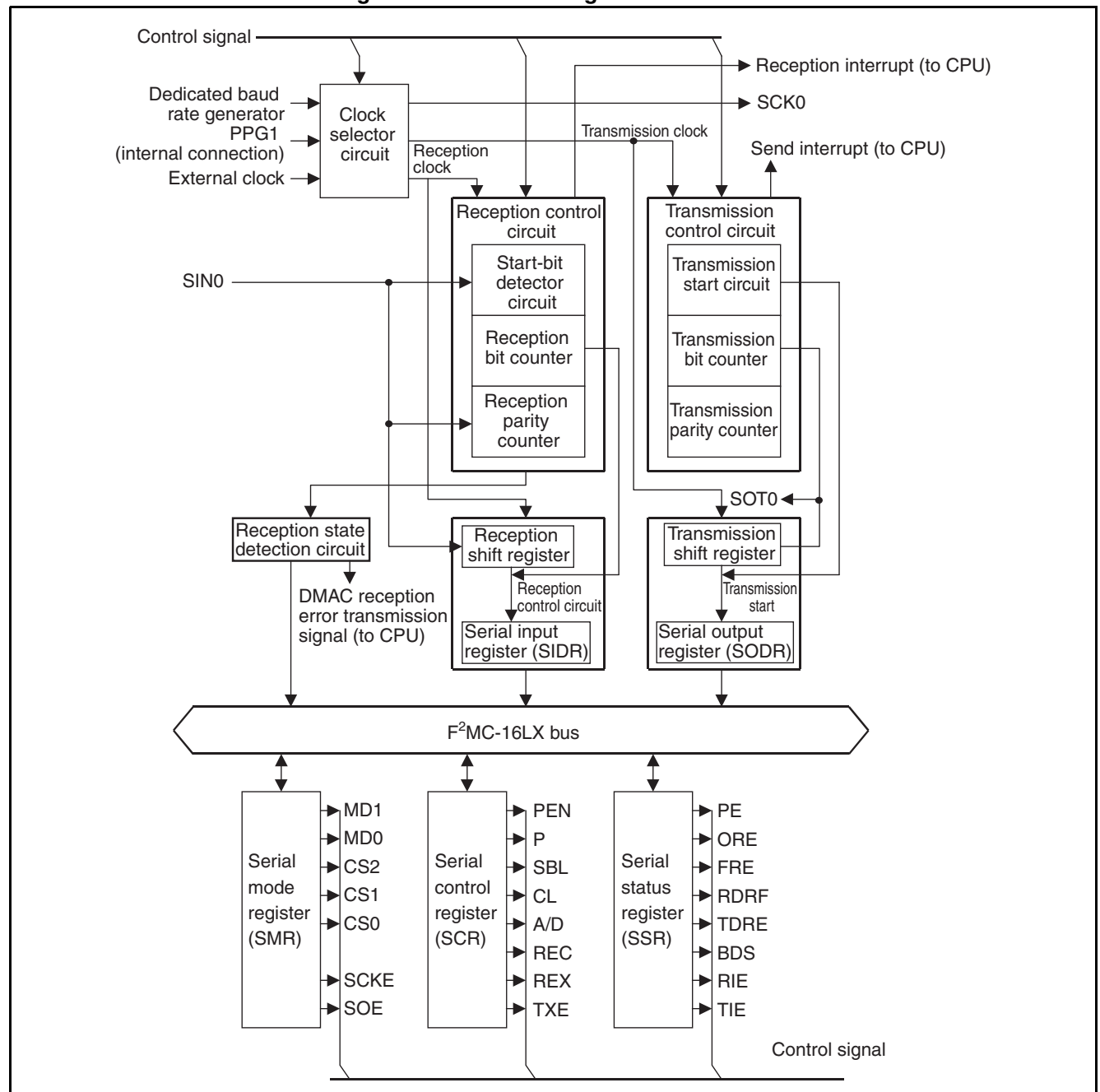
## 19.2 Configuration of UART

The UART consists of the serial mode register, serial control register, serial status register, communication prescaler control register and serial input/output register.

### ■ UART Block Diagram

Figure 19.2-1 shows a block diagram of the UART.

**Figure 19.2-1 Block Diagram of the UART**





## ■ Pin Related to UART

The pin related to the UART has the SIN0/SOT0/SCK0 pins. The SIN0 pin functions as the serial input port, the SOT0 pin functions as the serial output port, and the SCK0 pin functions as the external clock input port. The SIN0, SCK0 pins function as the general-purpose I/O port (P70/SIN0, P72/SCK0) and the input pin of UART, and the SOT0 pin functions as the general-purpose I/O port (P71/SOT0) and the output pin of UART.

- Setting when using as SIN0/SCK0 pins

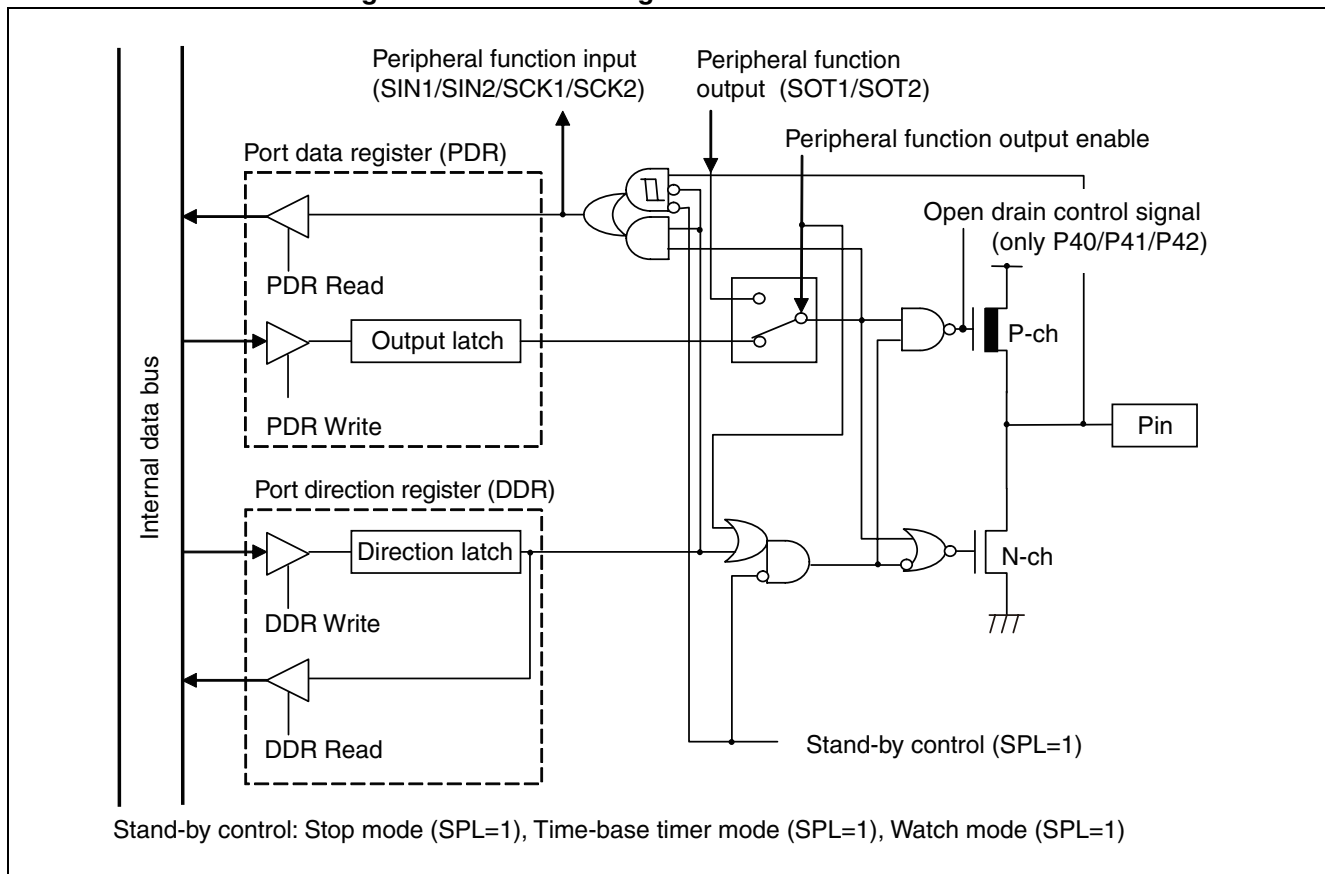
When the SIN0/SCK0 are used as the input pin by the UART, the P70/SIN0, P72/SCK0 pins should be set to the input port by the port direction register (DDR7 bit8, 10→ "0").

- Setting when using as SOT0 pin

When the SOT0 is used as the data output pin by the UART, be sure to set the serial mode register (SMR) to the serial data output (SOE bit0→ "1").

## ■ Block Diagram of Pin Related to UART

**Figure 19.2-2 Block Diagram of Pin Related to UART**



## 19.3 Configuration and Functions of UART Registers

This section describes the configuration and functions of the registers used by the UART.

### ■ List of UART Registers

Figure 19.3-1 lists the UART registers.

**Figure 19.3-1 List of UART Registers**

|  |          |     |  |  |  |  |  |                 |          |  |  |  |  |  |       |  |
|--|----------|-----|--|--|--|--|--|-----------------|----------|--|--|--|--|--|-------|--|
|  | 15       | 8 7 |  |  |  |  |  | 0               |          |  |  |  |  |  |       |  |
|  | SCR      |     |  |  |  |  |  | SMR             |          |  |  |  |  |  | (R/W) |  |
|  | SSR      |     |  |  |  |  |  | SIDR(R)/SODR(W) |          |  |  |  |  |  | (R/W) |  |
|  | CDCR     |     |  |  |  |  |  | -               |          |  |  |  |  |  | (R/W) |  |
|  | ← 8 bits |     |  |  |  |  |  | ✕               | 8 bits → |  |  |  |  |  |       |  |

|                     |       |       |       |       |       |          |       |       |                            |
|---------------------|-------|-------|-------|-------|-------|----------|-------|-------|----------------------------|
|                     | 7     | 6     | 5     | 4     | 3     | 2        | 1     | 0     |                            |
| 000020 <sub>H</sub> | MD1   | MD0   | CS2   | CS1   | CS0   | Reserved | SCKE  | SOE   | Serial mode register (SMR) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (R/W) | (R/W) | Read/write                 |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (X)      | (0)   | (0)   | Initial value              |

|                     |       |       |       |       |       |     |       |       |                               |
|---------------------|-------|-------|-------|-------|-------|-----|-------|-------|-------------------------------|
|                     | 15    | 14    | 13    | 12    | 11    | 10  | 9     | 8     |                               |
| 000021 <sub>H</sub> | PEN   | P     | SBL   | CL    | A/D   | REC | RXE   | TXE   | Serial control register (SCR) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (W) | (R/W) | (R/W) | Read/write                    |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (1) | (0)   | (0)   | Initial value                 |

|                     |       |       |       |       |       |       |       |       |  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|--|
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |  |
| 000022 <sub>H</sub> | D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    | Serial input register (SIDR)/<br>serial output register (SODR) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write   |
|                     | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value  |

|                     |     |     |     |      |      |       |       |       |                              |
|---------------------|-----|-----|-----|------|------|-------|-------|-------|------------------------------|
|                     | 15  | 14  | 13  | 12   | 11   | 10    | 9     | 8     |                              |
| 000023 <sub>H</sub> | PE  | ORE | FRE | RDRF | TDRE | BDS   | RIE   | TIE   | Serial status register (SSR) |
|                     | (R) | (R) | (R) | (R)  | (R)  | (R/W) | (R/W) | (R/W) | Read/write                   |
|                     | (0) | (0) | (0) | (0)  | (1)  | (0)   | (0)   | (0)   | Initial value                |

|                     |       |       |     |     |       |       |       |       |  |
|---------------------|-------|-------|-----|-----|-------|-------|-------|-------|--|
|                     | 15    | 14    | 13  | 12  | 11    | 10    | 9     | 8     |  |
| 000025 <sub>H</sub> | MD    | SRST  | -   | -   | DIV3  | DIV2  | DIV1  | DIV0  | Communication prescaler<br>control register (CDCR) |
|                     | (R/W) | (R/W) | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write   |
|                     | (0)   | (0)   | (-) | (-) | (0)   | (0)   | (0)   | (0)   | Initial value                                      |

Note:

When setting a UART register, set the communication mode while the UART is not in operation. When the communication mode is changed while operation is in progress, the sent/received data cannot be assured.

### 19.3.1 Serial Mode Register (SMR)

This section describes the configuration and functions of the serial mode register (SMR).

■ Serial Mode Register (SMR)

The bit configuration of the serial mode register (SMR) is illustrated below.

|                     |       |       |       |       |       |          |       |       |                            |
|---------------------|-------|-------|-------|-------|-------|----------|-------|-------|----------------------------|
|                     | 7     | 6     | 5     | 4     | 3     | 2        | 1     | 0     |                            |
| 000020 <sub>H</sub> | MD1   | MD0   | CS2   | CS1   | CS0   | Reserved | SCKE  | SOE   | Serial mode register (SMR) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | (R/W) | (R/W) | Read/write                 |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (X)      | (0)   | (0)   | Initial value              |

The functions of the bits for the serial mode register (SMR) are as follows.

[bit7, bit6] MD1, MD0: Mode Select

These bits are used to select the UART operation mode.

| Mode | MD1 | MD0 | Operation mode  |
|------|-----|-----|---|
| 0    | 0   | 0   | Asynchronous (start-stop synchronization) normal mode         |
| 1    | 0   | 1   | Asynchronous (start-stop synchronization) multiprocessor mode |
| 2    | 1   | 0   | CLK synchronous mode  |
| -    | 1   | 1   | Setting prohibited  |

In Mode 1, the asynchronous (start-stop synchronization) multiprocessor mode is used when several slave CPUs are connected to one host CPU. UART cannot distinguish the data format of reception data; therefore, it only supports the master in multiprocessor mode. The parity check function cannot be used. Set the PEN bit of the SCR Register to "0".

**[bit5, bit4, bit3] CS2, CS1, CS0: Clock Select**

These bits select the baud rate clock sources. When a dedicated baud rate generator is selected, the baud rate will be determined at the same time.

| CS2 to CS0                           | Clock input                   |
|--------------------------------------|-------------------------------|
| 000 <sub>B</sub> to 101 <sub>B</sub> | Dedicated baud rate generator |
| 110 <sub>B</sub>                     | Internal clock                |
| 111 <sub>B</sub>                     | External clock                |

- PPG1 will be selected in the MB90480/485 series if an internal clock is selected.
- Please do not use the following settings when dedicated baud rate generator is used at synchronous transfer.
  - 1) CS2 to CS0 = 000<sub>B</sub>
  - 2) CS2 to CS0 = 001<sub>B</sub>, DIV3 to DIV0 = 0000<sub>B</sub>

**[bit2] Reserved**

This bit is reserved.

**[bit1] SCKE: SCLK Enable**

This bit specifies whether to use SCK0 as a clock input pin or as a clock output pin during communication in CLK synchronous mode (Mode 2). Set this bit to "0" in CLK asynchronous mode or external clock mode.

- 0: The pin functions as clock input pin.
- 1: The pin functions as clock output pin.

---

Note:

An external clock source must be selected in advance for using as a clock input pin via this bit.

---

**[bit0] SOE: Serial Output Enable**

This bit specifies whether to use the external pin (SOT0), which is used also as a general-purpose I/O port pin, as a serial output pin or as an I/O port pin.

- 0: The pin functions as a general-purpose I/O port pin.
- 1: The pin functions as the serial data output pin (SOT0).

### 19.3.2 Serial Control Register (SCR)

This section describes the configuration and functions of the serial control register (SCR).

■ Serial Control Register (SCR)

The bit configuration of the serial control register (SCR) is illustrated below.

|                     |       |       |       |       |       |     |       |       |                               |
|---------------------|-------|-------|-------|-------|-------|-----|-------|-------|-------------------------------|
|                     | 15    | 14    | 13    | 12    | 11    | 10  | 9     | 8     |                               |
| 000021 <sub>H</sub> | PEN   | P     | SBL   | CL    | A/D   | REC | RXE   | TXE   | Serial control register (SCR) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (W) | (R/W) | (R/W) | Read/write                    |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (1) | (0)   | (0)   | Initial value                 |

The functions of the bits for the serial control register (SCR) are as follows:

[bit15] PEN: Parity ENable

This bit specifies whether to add a parity bit during transmission and to detect it during receiving when processing serial data.

|   |                 |
|---|-----------------|
| 0 | No parity       |
| 1 | Parity provided |

Note:

Parity can be added only in normal mode (Mode 0) in asynchronous (start-stop synchronization) communication mode. Parity cannot be added in multiprocessor mode (Mode 1) or in CLK synchronous communication (Mode 2).

[bit14] P: Parity

This bit specifies even or odd parity in data communications with parity.

|   |             |
|---|-------------|
| 0 | Even parity |
| 1 | Odd parity  |

[bit13] SBL: Stop Bit Length

This bit specifies the bit length of the stop bit, which is a frame end mark in asynchronous (start-stop synchronization) communication.

|   |             |
|---|-------------|
| 0 | 1 stop bit  |
| 1 | 2 stop bits |

**[bit12] CL: Character Length**

This bit specifies the data length of one frame to be sent or received.

|   |            |
|---|------------|
| 0 | 7-bit data |
| 1 | 8-bit data |

**Note:**

Only the normal mode (Mode 0) in asynchronous (start-stop synchronization) communications can handle 7-bit data. Specify 8-bit data in multiprocessor mode (Mode 1) or CLK synchronous mode (Mode 2).

**[bit11] A/D: Address/Data**

This bit specifies the data format for frames to be sent and received in multiprocessor mode (Mode 1) during asynchronous (start-stop synchronization) communication.

|   |               |
|---|---------------|
| 0 | Data frame    |
| 1 | Address frame |

**[bit10] REC: Receiver Error Clear**

Writing "0" to this bit clears the error flags (PE, ORE, FRE) of the SSR register.

Writing "1" has no effect. Read operations always read "1".

**[bit9] RXE: Receiver Enable**

This bit controls the reception state of the UART.

|   |                              |
|---|------------------------------|
| 0 | Disables reception operation |
| 1 | Enables reception operation  |

If reception operation becomes disabled while reception is in progress (while data is input to the reception shift register), reception operation will only be disabled after reception of the frame completes and the reception data is stored in the SDR register from the reception data buffer.

**[bit8] TXE: Transmitter Enable**

This bit controls the UART transmission states.

|   |                                 |
|---|---------------------------------|
| 0 | Disables transmission operation |
| 1 | Enables transmission operation  |

If transmission operation becomes disabled while transmission is in progress (while data is output from the transmission register), transmission operation will only be disabled after the serial output register (SODR) no longer contains transmission data. Transmission is resumed by synchronization with an internal serial clock after writing a value to the serial output register (SODR). Disabling of transmission (TXE = 0) is invalid when the TDRE flag is "0".

### 19.3.3 Serial Input/Output Register (SIDR/SODR)

This section describes the configuration and functions of the serial input/output register (SIDR/SODR).

■ Serial Input/Output Register (SIDR/SODR)

The bit configuration of the serial input/output register (SIDR/SODR) is illustrated below.

|                     |       |       |       |       |       |       |       |       |                               |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------|
|                     | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | Serial input register (SIDR)/ |
| 000022 <sub>H</sub> | D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    | Serial output register (SODR) |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/write                    |
|                     | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | Initial value                 |

The upper bit (D7) of the serial input/output register (SIDR/SODR) will become invalid when the stored data elements are 7 bits long. Always set the TDRE of the SSR register to "1" when writing data elements to the SODR register.

Note:

Write data elements at this address by the same method as that in writing data elements in the SODR register. This address is read by the same method as that used in reading the SIDR register.

## 19.3.4 Serial Status Register (SSR)

This section describes the configuration and functions of the serial status register (SSR).

### ■ Serial Status Register (SSR)

The bit configuration of the serial status register (SSR) is illustrated below.

|                     |     |     |     |      |      |       |       |       |                              |
|---------------------|-----|-----|-----|------|------|-------|-------|-------|------------------------------|
|                     | 15  | 14  | 13  | 12   | 11   | 10    | 9     | 8     |                              |
| 000023 <sub>H</sub> | PE  | ORE | FRE | RDRF | TDRE | BDS   | RIE   | TIE   | Serial status register (SSR) |
|                     | (R) | (R) | (R) | (R)  | (R)  | (R/W) | (R/W) | (R/W) | Read/write                   |
|                     | (0) | (0) | (0) | (0)  | (1)  | (0)   | (0)   | (0)   | Initial value                |

The SSR provides a flag that represents the UART status.

The functions of the serial status register (SSR) bits are described below.

#### [bit15] PE: Parity Error

This bit is an interrupt request flag that is set when a parity error occurs during reception. Set the REC bit (bit10) of the SCR register to "0" to clear the flag that has been set. The data in the SDR becomes invalid when this bit is set.

|   |                       |
|---|-----------------------|
| 0 | No parity error       |
| 1 | Parity error detected |

#### [bit14] ORE: Over Run Error

This bit is an interrupt request flag that is set when an overrun error occurs during reception. Set the REC bit (bit10) of the SCR register to "0" to clear the flag that has been set. The data in the SDR becomes invalid when this bit is set.

|   |                        |
|---|------------------------|
| 0 | No overrun error       |
| 1 | Overrun error detected |

#### [bit13] FRE: FRaming Error

This bit is an interrupt request flag that is set when a framing error occurs during reception. Set the REC bit (bit10) of the SCR register to "0" to clear the flag that has been set. The data in the SDR becomes invalid when this bit is set.

|   |                        |
|---|------------------------|
| 0 | No framing error       |
| 1 | Framing error detected |

#### [bit12] RDRF: Receiver Data Register Full

This bit is an interrupt request flag that indicates that the SDR register has stored reception data. This flag is set when reception data is loaded into the SDR register and is cleared automatically when the SDR register is reads.

|   |                       |
|---|-----------------------|
| 0 | No reception data     |
| 1 | Reception data loaded |



**[bit11] TDRE: Transmitter Data Register Empty**

This bit is an interrupt request flag that indicates that transmission data can be written to the SODR register. This flag is cleared when transmission data is written to the SODR register. The flag will be set again to indicate that the next item of transmission data can be written when written data is loaded into the transmission shift register unit and data transfer starts.

|   |                                       |
|---|---------------------------------------|
| 0 | Prohibit writing of transmission data |
| 1 | Allow writing of transmission data    |

**[bit10] BDS: Bit Direction Select**

This bit controls the selection of a data transfer direction.

|   |  |
|---|--|
| 0 | Serial data is transferred starting with the LSB side. (LSB first) |
| 1 | Serial data is transferred starting with the MSB side. (MSB first) |

**Note:**

When this bit is rewritten after writing data to the SDR register to switch between the upper side and lower side of data, the data in the serial status register (SDR) becomes invalid in read and write operations.

**[bit9] RIE: Receiver Interrupt Enable**

This bit controls reception interrupts.

|   |                      |
|---|----------------------|
| 0 | Prohibit interrupts. |
| 1 | Allow interrupts.    |

In addition to PE, ORE, and FRE errors, normal reception by RDRF also acts as reception interrupt source.

**[bit8] TIE: Transmitter Interrupt Enable**

This bit controls transmission interrupts.

|   |                      |
|---|----------------------|
| 0 | Prohibit interrupts. |
| 1 | Allow interrupts.    |

**Note:**

If transmission operation becomes disabled during transmission, the transmission operation stops after no more data remains in the serial output register (SODR). For writing "0", wait a predefined time after data has been written to SODR register. In clock asynchronous transfer mode, the term "predefined time" means 1/16 the time corresponding to the baud rate. In clock synchronous transfer mode, this term refers to the time corresponding to the baud rate.

## 19.3.5 Communication Prescaler Control Register (CDCR)

This section describes the configuration and functions of the communication prescaler control register (CDCR).

### ■ Communication Prescaler Control Register (CDCR)

The bit configuration of the communication prescaler control register (CDCR) is illustrated below.

|                     |       |       |     |     |       |       |       |       |   |
|---------------------|-------|-------|-----|-----|-------|-------|-------|-------|---|
|                     | 15    | 14    | 13  | 12  | 11    | 10    | 9     | 8     | Communication prescaler control register (CDCR) |
| 000025 <sub>H</sub> | MD    | SRST  | -   | -   | DIV3  | DIV2  | DIV1  | DIV0  | Read/write                                      |
|                     | (R/W) | (R/W) | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value                                   |
|                     | (0)   | (0)   | (-) | (-) | (0)   | (0)   | (0)   | (0)   |   |

The CDCR register controls machine clock division. The UART operation clocks are obtained by dividing the machine clock. This communication prescaler is allowed to obtain the constant baud rates with respect to various machine clocks.

#### [bit15] MD: Machine clock divide moDe select

This bit enables the operation of the communication prescaler.

|   |                                       |
|---|---------------------------------------|
| 0 | The communication prescaler stops.    |
| 1 | The communication prescaler operates. |

#### [bit14] SRST: Set ReSeT

This bit resets all operations of the UART. It initializes all data and register values.

|   |                               |
|---|-------------------------------|
| 0 | Initial value (has no effect) |
| 1 | Forced reset                  |

#### Note:

Setting this bit will forcibly clear all data and register values of the UART. Set all data and registers again to return to their initial values. Data being transferred as well as saved data will be invalid until the respective settings have been made again.

**[bit11, bit10, bit9, bit8] DIV3, DIV2, DIV1, DIV0**

These bits are determines the division ratios of the machine clocks.

| <b>DIV3 to DIV0</b> | <b>Division Ratio</b> |
|---------------------|-----------------------|
| 0000 <sub>B</sub>   | Division by 1         |
| 0001 <sub>B</sub>   | Division by 2         |
| 0010 <sub>B</sub>   | Division by 3         |
| 0011 <sub>B</sub>   | Division by 4         |
| 0100 <sub>B</sub>   | Division by 5         |
| 0101 <sub>B</sub>   | Division by 6         |
| 0110 <sub>B</sub>   | Division by 7         |
| 0111 <sub>B</sub>   | Division by 8         |

## Notes:

- When changing the clock division ratio, wait for time of 2 division as a clock stabilization time before the communication is performed.
- Please do not use the following settings when the dedicated baud rate generator is used at the synchronous transmission.
  - 1) CS2 to CS0 = 000<sub>B</sub>
  - 2) CS2 to CS0 = 001<sub>B</sub>, DIV3 to DIV0 = 0000<sub>B</sub>

## 19.4 Interrupt of UART

The UART has the reception and transmission interrupts.

The interrupt of the UART can activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt of UART

The following table shows the interrupt control bit and interrupt source of the UART.

|                                     | UART reception interrupt   | UART transmission interrupt |
|-------------------------------------|--|-----------------------------|
| Interrupt request flag              | Data reception completion SSR: RDRF (bit12)<br>Framing error SSR:FRE (bit13)<br>Overrun error SSR:ORE (bit14)<br>Parity error SSR:PE (bit15) | SSR:TDRE (bit11)            |
| Interrupt request output enable bit | SSR:RIE (bit9)   | SSR:TIE (bit8)              |
| Interrupt generation source         | At receiving UART  | At transmitting UART        |

### ■ Interrupt Source Related to UART

The interrupts occurs at receiving and transmitting UART.

The interrupt request occurs with the sources as shown below.

- When the reception data is loaded to the serial input register (SIDR)
- When the reception error (parity, overrun, framing error) occurs
- When the transmission data is transferred from the serial output register (SODR) to the transmission shift register

## ■ Interrupt of UART, DMA Transfer, and EI<sup>2</sup>OS

Table 19.4-1 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 19.4-1 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source               | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|--------------------------------|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|                                |                          |                      | Number           | Address             | Number                     | Address             |
| UART transmission completion * | ○                        | 11                   | #34              | FFFF74 <sub>H</sub> | ICR11                      | 0000BB <sub>H</sub> |
| UART reception completion *    | ◎                        | 7                    | #36              | FFFF6C <sub>H</sub> | ICR12                      | 0000BC <sub>H</sub> |

○: Interrupt request flag is cleared.

◎: Interrupt request flag is cleared (stop request).

\*: This interrupt source shares the interrupt number with interrupt source of other peripheral functions. For details, see Table 3.2-2.

---

### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing.

---

## ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The UART corresponds to the DMA transfer function and EI<sup>2</sup>OS function. When the DMA or EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 19.5 UART Operations

---

This section describes the operations of the UART.

---

### ■ Operation Modes

UART has the operation modes shown below. The modes can be changed by setting values in the SMR and SCR registers.

| Mode | Parity                | Data length              | Operation mode  | Stop bit length               |
|------|-----------------------|--------------------------|---|-------------------------------|
| 0    | Provided/not provided | 7 bits                   | Asynchronous (start-stop synchronization) normal mode         | 1 bit or 2 bits <sup>*2</sup> |
|      | Provided/not provided | 8 bits                   |   |                               |
| 1    | None                  | 8 + 1 bits <sup>*1</sup> | Asynchronous (start-stop synchronization) multiprocessor mode |                               |
| 2    | None                  | 8 bits                   | CLK synchronous mode  | None                          |

<sup>\*1</sup>: The "+1" term represents the address/data selection bit (A/D) used in communication control.

<sup>\*2</sup>: Only one bit can be detected as stop bit during reception.

The stop bit length in asynchronous (start-stop synchronization) mode can only be specified for send operations. The bit length in reception operations is always one bit. Do not set the stop bit length other than asynchronous (start-stop synchronization) mode.

UART operation mode 1 is used only for the master in master/slave connection.

### ○ Connection between CPUs

1:1 connection (normal mode) or master/slave connection (multiprocessor mode) can be selected. The data length, whether add or not to add a parity bit and synchronization etc. for these two systems must be the same across all CPUs. The following operation modes can be selected:

- In 1:1 connection (normal mode), the same operation mode, either operation mode 0 or operation mode 2, must be selected for both CPUs.  
Select operation mode 0 for asynchronous operation. Select operation mode 2 for synchronous operation.
- In master/slave connection (multiprocessor mode), use operation mode 1. Select operation mode 1 and use this device as the master. For this type of connection, select "no parity".

## ■ UART Clock Selection

### ○ Dedicated Baud Rate Generator

- Asynchronous baud rate =  $\phi$  / (prescaler division ratio) / (asynchronous transfer clock division ratio)
- Synchronous baud rate =  $\phi$  / (prescaler division ratio) / (synchronous transfer clock division ratio)

$\phi$ : Machine clock

- The division ratios provided by the prescaler (common for asynchronous/synchronous operation) are listed in Table 19.5-1.

**Table 19.5-1 Division Ratios by the Prescaler**

| MD | DIV3 to DIV0      | DIV  |
|----|-------------------|------|
| 0  | f                 | Stop |
| 1  | 0000 <sub>B</sub> | 1    |
| 1  | 0001 <sub>B</sub> | 2    |
| 1  | 0010 <sub>B</sub> | 3    |
| 1  | 0011 <sub>B</sub> | 4    |
| 1  | 0100 <sub>B</sub> | 5    |
| 1  | 0101 <sub>B</sub> | 6    |
| 1  | 0110 <sub>B</sub> | 7    |
| 1  | 0111 <sub>B</sub> | 8    |

- For the division ratios of the synchronous transfer clock, see Table 19.5-2.

**Table 19.5-2 Division Ratios of the Synchronous Transfer Clock**

| CS2 | CS1 | CS0 | CLK synchronous | Calculation formula      | SCK0                     |
|-----|-----|-----|-----------------|--------------------------|--------------------------|
| 0   | 0   | 1   | 4M              | $(\phi / \text{DIV})/2$  | $(\phi / \text{DIV})/2$  |
| 0   | 1   | 0   | 2M              | $(\phi / \text{DIV})/4$  | $(\phi / \text{DIV})/4$  |
| 0   | 1   | 1   | 1M              | $(\phi / \text{DIV})/8$  | $(\phi / \text{DIV})/8$  |
| 1   | 0   | 0   | 500K            | $(\phi / \text{DIV})/16$ | $(\phi / \text{DIV})/16$ |
| 1   | 0   | 1   | 250K            | $(\phi / \text{DIV})/32$ | $(\phi / \text{DIV})/32$ |

$\phi$ : Calculated based on the machine clock (internal frequency f=16 MHz) for DIV=2.

#### Note:

Please do not use the following settings when the dedicated baud rate generator is used at the synchronous transmission.

- 1) CS2 to CS0 = 000<sub>B</sub>
- 2) CS2 to CS0 = 001<sub>B</sub>, DIV3 to DIV0 = 0000<sub>B</sub>

- For the division ratios of the asynchronous transfer clock, see Table 19.5-3.

**Table 19.5-3 Division Ratios of the Asynchronous Transfer Clock**

| CS2 | CS1 | CS0 | Non-CLK<br>synchronous | Calculation formula                             | SCK0                                   |
|-----|-----|-----|------------------------|---|--|
| 0   | 0   | 0   | 76923                  | $(\phi / \text{DIV}) / (8 \times 13 \times 2)$  | $(\phi / \text{DIV}) / (13 \times 2)$  |
| 0   | 0   | 1   | 38461                  | $(\phi / \text{DIV}) / (8 \times 13 \times 4)$  | $(\phi / \text{DIV}) / (13 \times 4)$  |
| 0   | 1   | 0   | 19230                  | $(\phi / \text{DIV}) / (8 \times 13 \times 8)$  | $(\phi / \text{DIV}) / (13 \times 8)$  |
| 0   | 1   | 1   | 9615                   | $(\phi / \text{DIV}) / (8 \times 13 \times 16)$ | $(\phi / \text{DIV}) / (13 \times 16)$ |
| 1   | 0   | 0   | 500K                   | $(\phi / \text{DIV}) / (8 \times 2 \times 2)$   | $(\phi / \text{DIV}) / 2$              |
| 1   | 0   | 1   | 250K                   | $(\phi / \text{DIV}) / (8 \times 2 \times 4)$   | $(\phi / \text{DIV}) / 4$              |

$\phi$ : Calculated based on the machine clock (internal frequency  $f=16$  MHz) for DIV=1.



### ○ Internal timer

The applicable baud rate when CS2 to CS0 are set to "110" and the internal timer (PPG1) is selected can be calculated by the following expressions:

**Asynchronous (start-stop synchronization):**  $(\phi / N) / (16 \times 2 \times (n + 1))$

**CLK synchronous:**  $(\phi / N) / (2 \times (n + 1))$

N: Count clock source of the timer (PPG1)

n: Reload value of the timer (PPG1)

Table 19.5-4 shows the relationship between the baud rate and reload value when the machine clock frequency is 7.3728 MHz.

**Table 19.5-4 Relationship Between Baud Rate and Reload Value  
(Machine Clock Frequency: 7.3728 MHz)**

| Baud rate<br>(bps) | Reload value  |   |   |   |
|--------------------|---|---|---|---|
|                    | Clock asynchronous<br>(Start-Stop synchronization)      |   | Clock synchronous                                       |   |
|                    | N = 2 <sup>1</sup><br>(divide-by-2 of<br>machine clock) | N = 2 <sup>3</sup><br>(divide-by-8 of<br>machine clock) | N = 2 <sup>1</sup><br>(divide-by-2 of<br>machine clock) | N = 2 <sup>3</sup><br>(divide-by-8 of<br>machine clock) |
| 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  | 3071  | 767   |
| 300                | 383   | 95  | 6143  | 1535  |

- : Indicates that a setting is prohibited

### Note:

Please do not use the following setting at the clock synchronization.

N = 1, n = 0

### ○ External Clock

The baud rate when CS2 to CS0 are set to "111" can be calculated by the following expressions:

**Asynchronous (start-stop synchronization):**  $f / 16$

**CLK synchronous:**  $f'$

f can be up to 1/2 of the machine clock. f' can be up to 1/8 of the machine clock.

## 19.5.1 Operation in Asynchronous Mode (Operation Modes 0 and 1)

Transfer operation becomes asynchronous when the UART is used in operation mode 0 (normal mode) or in operation mode 1 (multiprocessor mode).

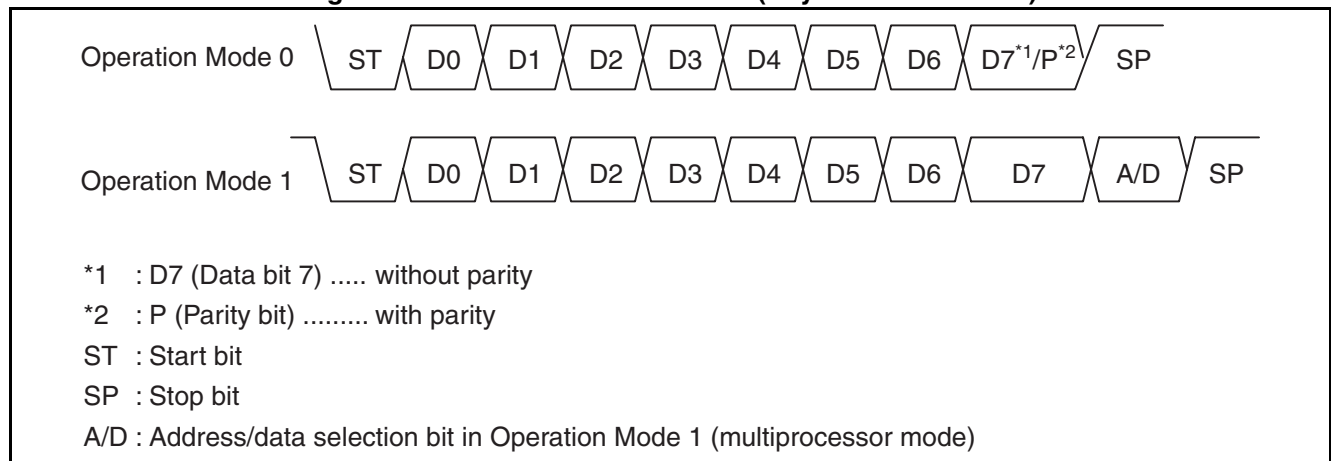
### ■ Operation in Asynchronous Mode (Operation Modes 0 and 1)

#### ○ Transfer data format

Transfer data always starts with a start bit ("L" level), a transfer operation of a specified data bit length is performed LSB first, and transfer ends with a stop bit ("H" level).

- In operation mode 0, data items without parity are fixed at a length of 7 bits, while data items with parity have a length of 8 bits.
- In operation mode 1, the data length is fixed at 8 bits and a parity bit is not added. Instead, an A/D (address/data selection bit) is added.

**Figure 19.5-1 Transfer Data Format (Asynchronous Mode)**



#### ○ Transmission Operation

When the transmission data empty flag bit (SSR: TDRE) is "1", transmission data is written to the output data register (SODR). The data is sent if send operation is enabled (SCR: TXE = 1) at that time. Transmission data is sent to the send shift register and sending starts. The TDRE flag is then reset to "1" to enable setting of the next item of transmission data.

If transmission interrupt requests are enabled (SSR: TIE = 1), a transmission interrupt request is output to request writing of the transmission data to the SODR. The TDRE flag is cleared to "0" when transmission data has been written to the SODR.

### ○ Reception Operation

Reception is always performed if reception operation is enabled (SCR: RXE = 1). When the start bit is detected, one frame of data is received in accordance with the data format determined by the serial control register (SCR). When the error flag is set at the time of error after one frame has been received, the reception data full flag bit (SSR: RDRF) is set to "1". If reception interrupt requests are enabled (SSR: RIE = 1), a reception interrupt request is output in this case. Each flag of the serial status register (SSR) is checked. If reception was performed normally, the serial input register (SIDR) is read; if an error is detected, perform error processing.

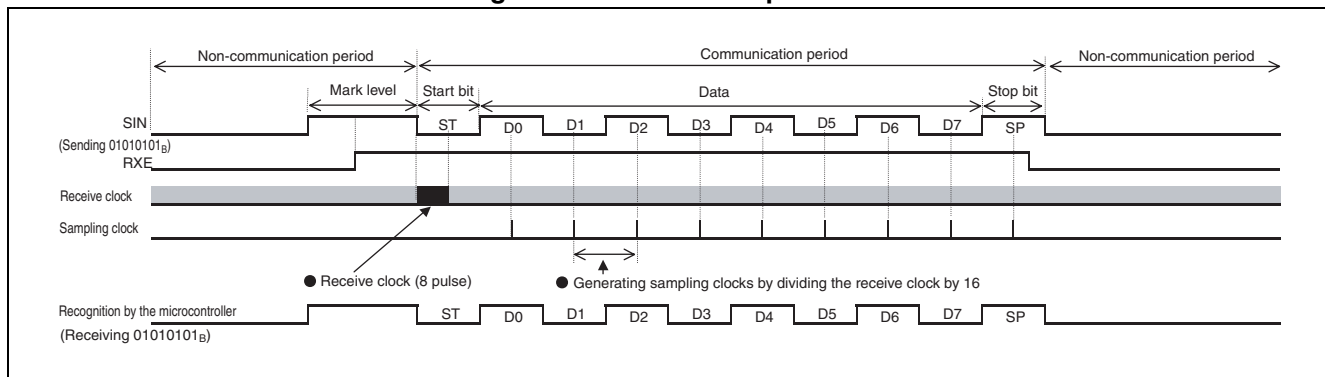
The RDRF flag is cleared to "0" after reception data is read from the SIDR.

### ○ Detecting the start bit

Implement the following settings to detect the start bit:

- Set the communication line level to "H" (attach the mark level) before the communication period.
- Specify reception permission (RXE="H") while the communication line level is "H" (mark level).
- Do not specify reception permission (RXE="H") for the periods other than the communication period (without mark level).
- After the stop bit is detected (the RDRF flag is set to "1"), specify reception inhibition (RXE = "L") while the communication line level is "H" (mark level).

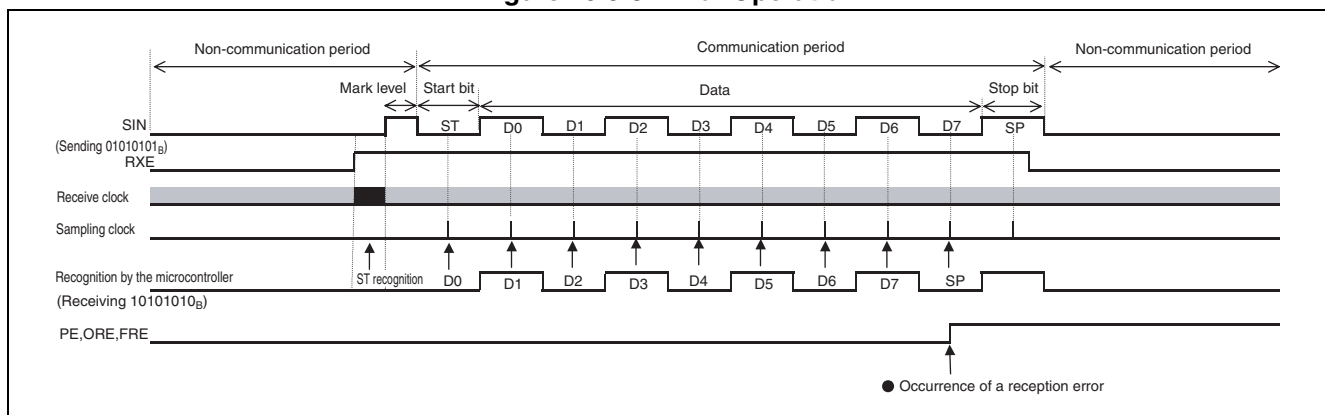
**Figure 19.5-2 Normal Operation**



Note that specifying reception permission at the timing shown below obstructs the correct recognition of the input data (SIN) by the microcontroller.

- Example of operation if reception permission (RXE="H") is specified while the communication line level is "L".

**Figure 19.5-3 Error Operation**



### ○ Stop bit

One or two stop bits can be selected for sending. The receiving unit, however, will only identify the first stop bit.

### ○ Error detection

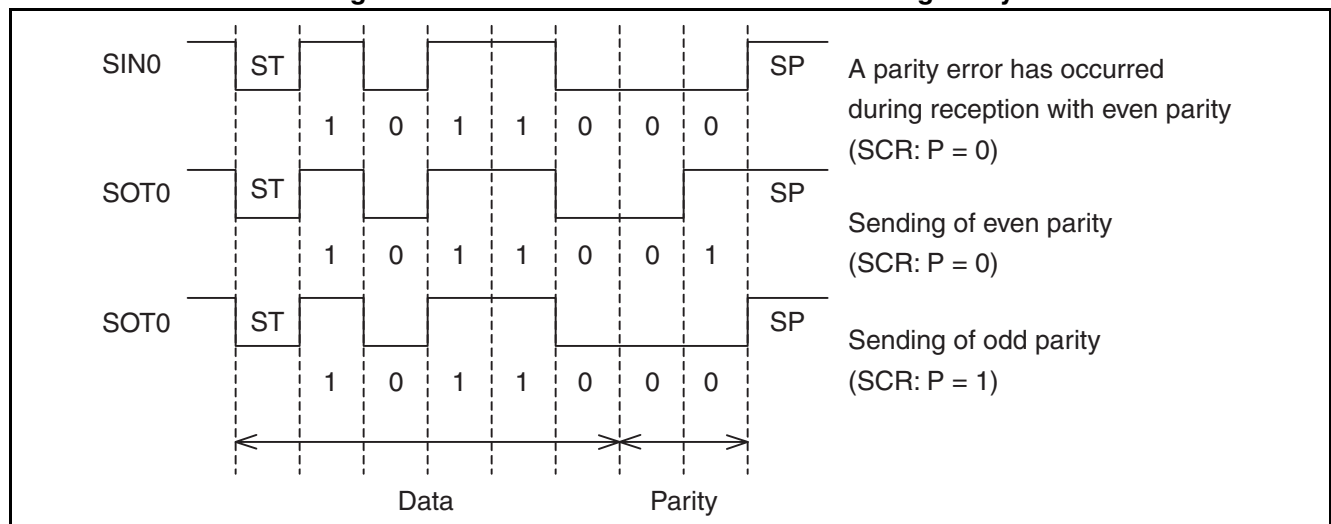
- Mode 0: Parity errors, overrun errors, and frame errors can be detected.
- Mode 1: Overrun and frame errors can be detected. Parity errors cannot be detected.

### ○ Parity

Parity can be used only in operation mode 0 (asynchronous and normal modes). Whether to use parity can be set with the PEN bit, while use of even or odd parity can be selected with the P bit of the serial control register (SCR). Parity cannot be used in operation mode 1 (asynchronous and multiprocessor modes) and operation mode 2 (CLK synchronous mode). Figure 19.5-4 shows the data format for sending and receiving data when parity is used.

The items "ST" and "SP" in the diagram indicate the "start bit" and "stop bit" respectively.

**Figure 19.5-4 Transfer Data Format When Using Parity**



Note:

Parity cannot be used in operation modes 1 and 2.

## 19.5.2 Operation in Synchronous Mode (Operation Mode 2)

The transfer operation becomes clock-synchronous when the UART operates in operation mode 2 (CLK synchronous mode).

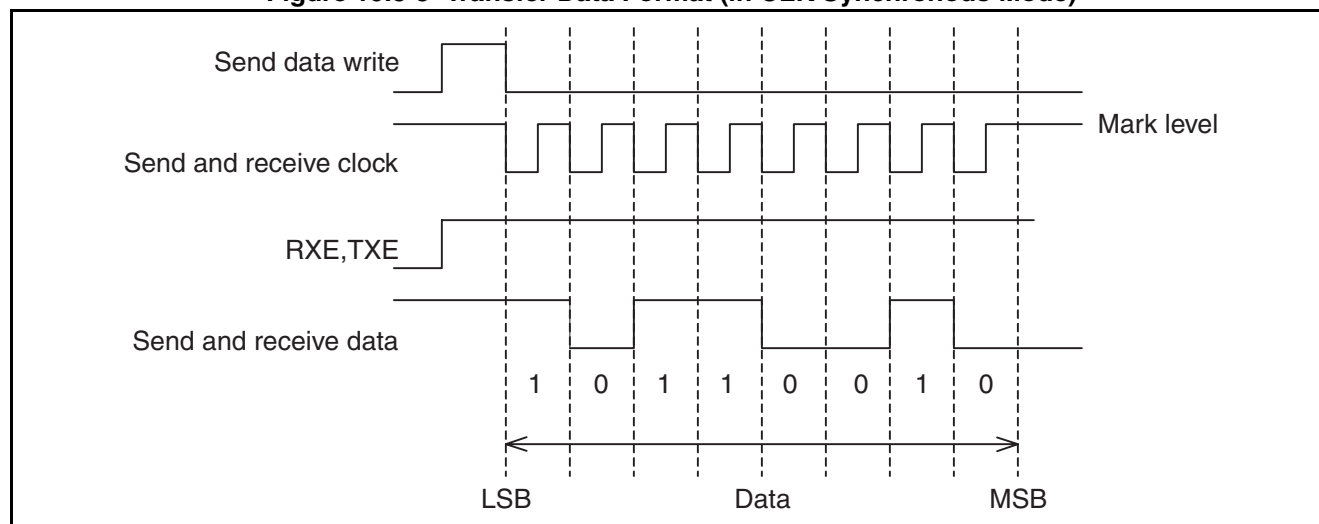
### ■ Operation in CLK Synchronous Mode (Operation Mode 2)

#### ○ Transfer Data Format

In synchronous mode, 8-bit data is transferred with LSB first, and no start bit or no stop bit is added.

Figure 19.5-5 shows the data format in synchronous mode.

**Figure 19.5-5 Transfer Data Format (in CLK Synchronous Mode)**



#### ○ Clock Supply

In a clock synchronous (expanded I/O serial) operation, a number of clock equivalent to the number of bits in the transmission/reception data must be supplied.

When an internal clock (dedicated baud rate generator or internal timer) is selected, a data reception synchronous clock will be supplied automatically when data is sent.

If an external clock is selected, the serial output register (SODR) in the UART of the transmission side system must contain data. After confirmation that TDRE of the SSR is "0", clock for one byte must be supplied correctly from the outside.

Always set the mark level "H" before and after sending data.

#### ○ Error Detection

Only overrun errors can be detected. Parity and framing errors cannot be detected.

### ○ Initialization

The appropriate setting values for the control registers when using synchronous mode are shown below.

#### [Serial mode register (SMR)]

- MD1 and MD0: "10"
- CS2, CS1, CS0: Specify the clock source determined by the clock selector.
- SCKE: "1" if the dedicated baud rate generator or internal clock is used, "0" when the external clock is used.
- SOE: "1" for sending. "0" for only receiving.

#### [Serial control register (SCR)]

- PEN: "0"
- P, SBL, A/D: These bits have no effect.
- CL: "1" (8-bit data)
- REC: "0" (Error flag clear for initialization)
- RXE: TXE: Ensure that at least one of RXE and TXE is "1".

#### [Serial status Register (SSR)]

- RIE: "1" when interrupts are used. "0" when no interrupts are used.
- TIE: "0"

### ○ Communication start

Start communication by writing to the serial output register (SODR). Note that temporary data must be written to the SODR before starting communication, even when receiving data.

### ○ Communication end

After the end of transmission and reception of one data frame, the RDRF flag of the serial status register (SSR) is set to "1". During reception, check the overrun error flag bit (SSR: ORE) and decide whether communication has been performed normally.

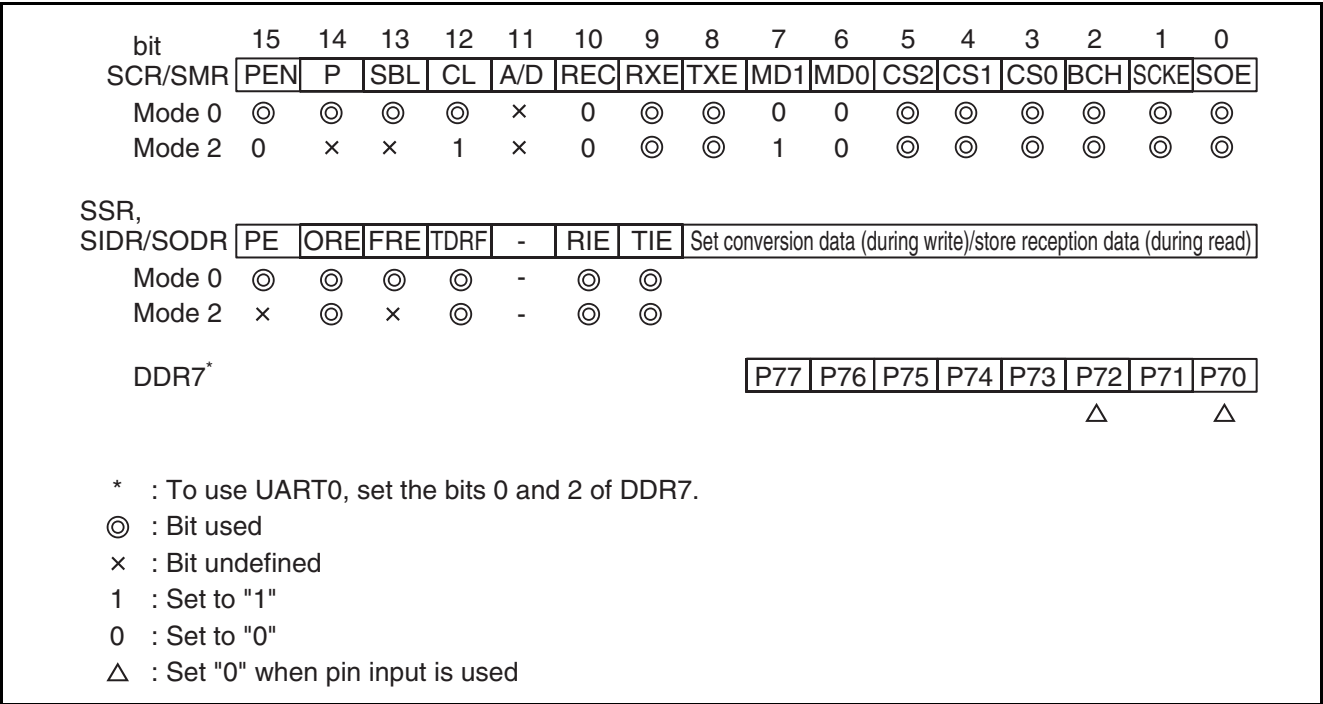
### 19.5.3 Two-Way Communication Function (Normal Mode)

Normal serial two-way communication in a 1:1 connection can be performed in operation modes 0 and 2. The synchronization type is "asynchronous" for operation mode 0 and "synchronous" for operation mode 2.

■ Register Settings in Two-way Communication

Set the registers as shown in Figure 19.5-6 before starting two-way communication.

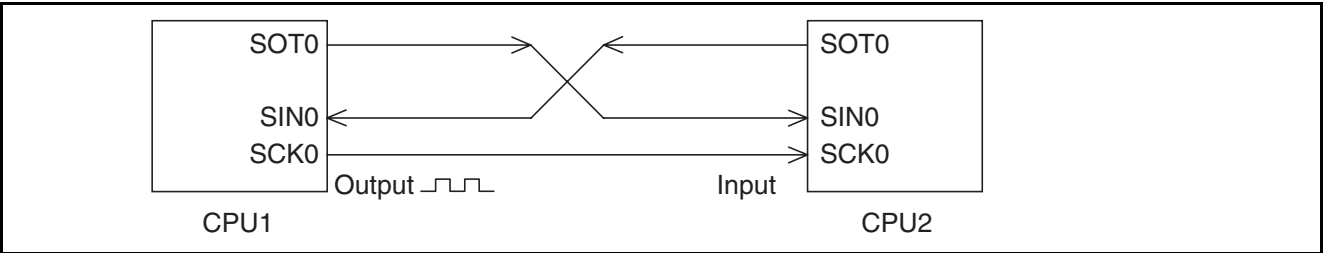
Figure 19.5-6 Register Settings in Two-way Communication



■ Connection Between CPUs in Two-way Communication

Figure 19.5-7 shows the connection between CPUs in two-way communication.

Figure 19.5-7 Connection Between CPUs in Two-way Communication



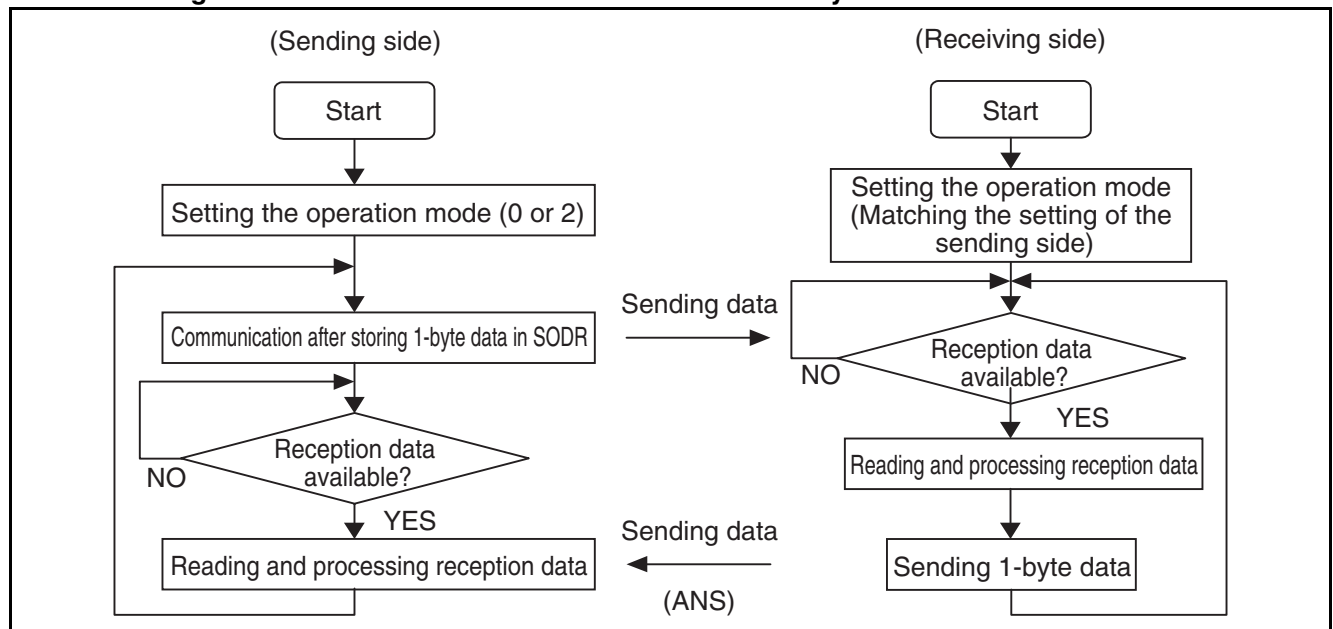
## ■ Communication Procedure for Two-way Communication Function

Communication is started from the sending side with an arbitrary timing when data for transmission is ready.

When the receiving side receives the transmission data, returns ANS (in this example, separately for each byte) periodically.

Figure 19.5-8 illustrates the procedure for communication with the two-way communication function.

**Figure 19.5-8 Communication Procedure for Two-way Communication Function**





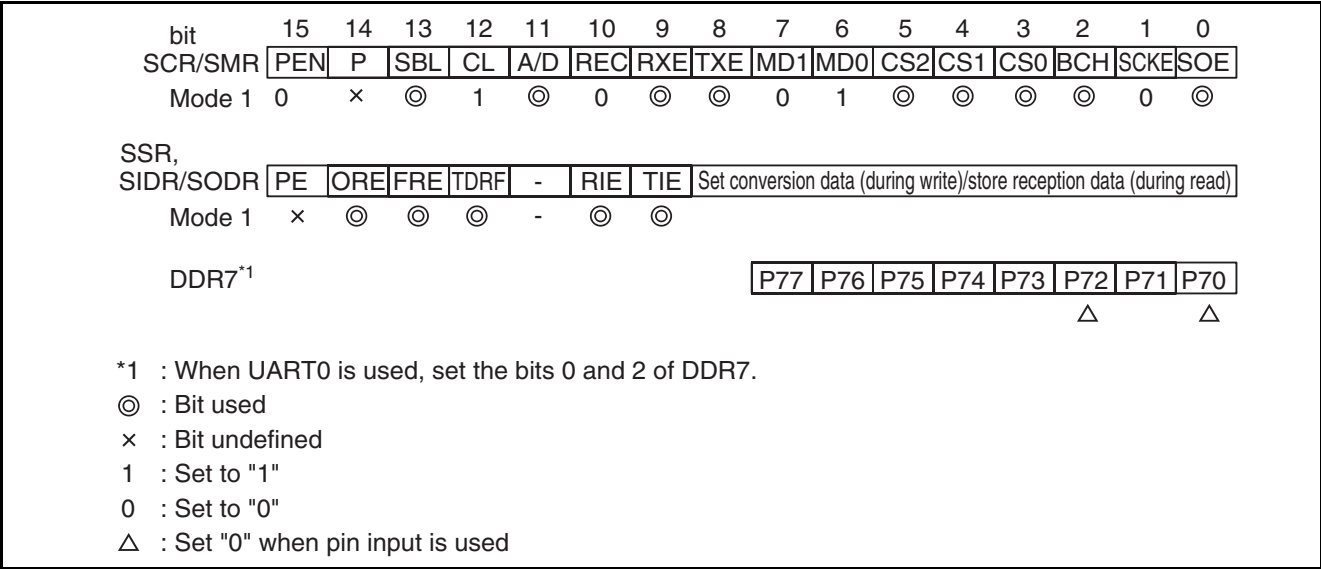
### 19.5.4 Master/Slave Communication Function (Multiprocessor Mode)

The UART enables communication in a master/slave connection in which more than one CPU is connected. Operation mode 1 is used in this case. The UART itself can be used only as the master system.

■ Register Settings in Master/Slave Communication

Set the registers as shown in Figure 19.5-9 before starting master-slave communication.

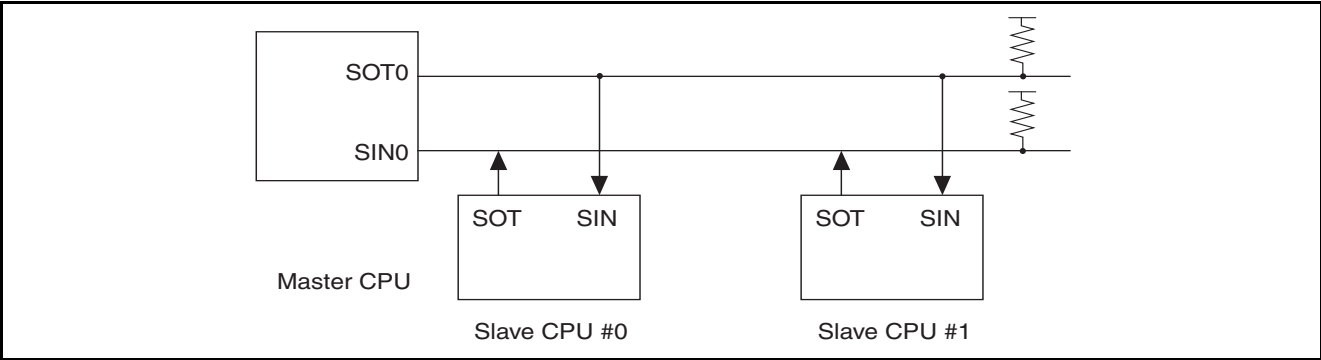
Figure 19.5-9 Register Settings in Master/Slave Communication



■ Connection Between CPUs in Master/Slave Communication

Figure 19.5-10 shows the connection between CPUs in master/slave communication.

Figure 19.5-10 Connection Between CPUs in Master/Slave Communication



## ■ Function Selection

Table 19.5-5 lists settings for selecting the communication method in master/slave communication.

**Table 19.5-5 Function Selection in Master/Slave Communication**

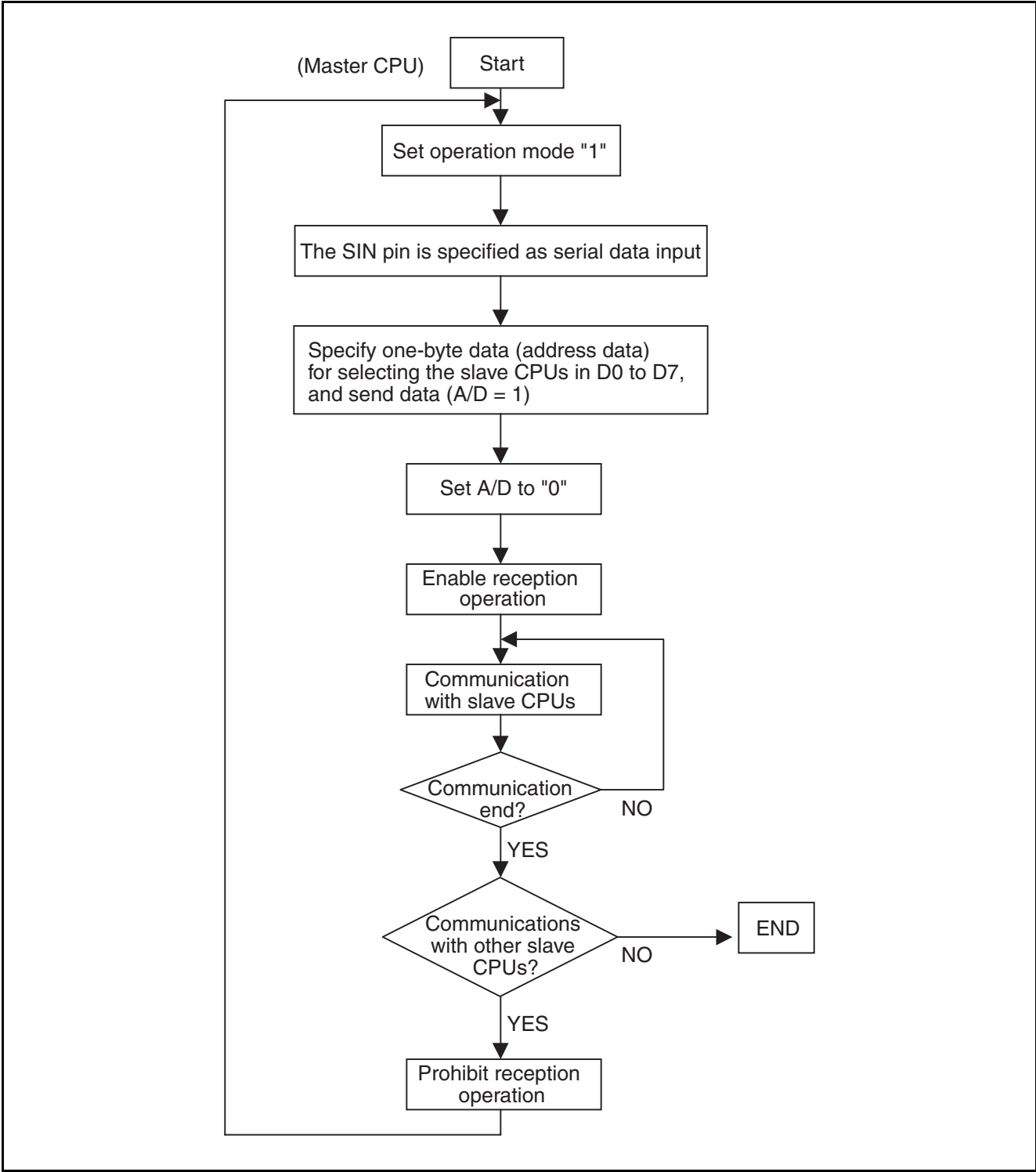
|                                     | Operation mode |           | Data                    | Parity | Synchronous operation | Stop bit    |
|-------------------------------------|----------------|-----------|-------------------------|--------|-----------------------|-------------|
|                                     | Master CPU     | Slave CPU |                         |        |                       |             |
| Sending and receiving the addresses | Mode 1         | -         | A/D = 1 + 8-bit address | None   | Asynchronous          | 1 or 2 bits |
| Sending and receiving the data      |                |           | A/D = 0 + 8-bit address |        |                       |             |

## ■ Communication Procedure of Master/Slave Communication Function

Communication starts when the master CPU sends address data. Address data is data for which the A/D bit is "1"; the address is used to select the slave CPU that becomes the communication destination. Each slave CPU interprets the address data by a program. If the address data matches the address assigned to the system, communication (transfer of ordinary data) with the master CPU is established.

Figure 19.5-11 shows the procedure for communication using the master/slave communication function.

Figure 19.5-11 Procedure for Communication Using the Master/Slave Communication Function



## 19.6 Precautions on use of the UART

---

Notes the following points when using the UART.

---

### ■ Precautions on Use of the UART

#### ○ Enabling operation

The serial control register (SCR) of the UART contains operation enable bits for enabling sending and receiving, namely, TXE (sending) and RXE (reception).

By default (initial value), both sending and receiving are disabled. If required, enable the operation.

#### ○ Communication mode setting

Set the communication mode while the UART is not in operation. Transmission and reception data cannot be assured if the mode is changed during data transmission or reception.

#### ○ Synchronous mode

The clock synchronous mode (operation mode 2) of the UART uses a clock control (expanded I/O serial) operation, and a start bit or stop bit is not added to data.

#### ○ Transmission data empty flag bit

By default (initial value), the transmission data empty flag bit (TDRE of SSR) is set to "1" (no transmission data, transmission data can be written). Therefore, when transmission interrupt requests are enabled (TIE = 1 of SSR), a transmission interrupt request will be triggered immediately. Be sure that the sending data is available before setting the TIE flag to "1".

#### ○ Clock setting in synchronous transfer

- Please do not use the following settings when the dedicated baud rate generator is used at the synchronous transmission.

1) CS2 to CS0 = 000<sub>B</sub>

2) CS2 to CS0 = 001<sub>B</sub>, DIV3 to DIV0 = 0000<sub>B</sub>

- Please do not use the following settings when the internal timer (PPG1) is used at the synchronous transmission.

N = 1, n = 0

## 19.7 Program Example of UART

This section describes the program example of the UART.

### ■ Program Example of UART

|   |                     |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
|---|---------------------|-------|------------|-------|-------------------|-----|-----------------------|----------|------------------------|--------|-----------------|-------|-----------------|------|----------------------|-----|----------------------------|------|------------------------|----|---------------------|------|---------------------------------|-----|-----------------------------|------|---------------|------|-----------------------------------|------|--------------------------------------|------|-------------------|----------|-----------------------|------|------------------------------------|-----------|---------------|------|---|
| <p>Example of setting procedure</p> <p>Send data of 1 byte by UART.<br/>Asynchronous normal mode</p> <p>&lt;Initial setting&gt;</p> <ul style="list-style-type: none"> <li>Interrupt related</li> </ul> <table border="1"> <tr> <td>Set interrupt level</td><td>ICR11</td></tr> <tr> <td>Set I flag</td><td>(CCR)</td></tr> </table> <ul style="list-style-type: none"> <li>Control UART</li> </ul> <table border="1"> <tr> <td>Set mode register</td><td>SMR</td></tr> <tr> <td>Set operation mode &gt;&gt;</td><td>.MD1,MD0</td></tr> <tr> <td>Set operation clock &gt;&gt;</td><td>.CS2-0</td></tr> <tr> <td>Set SCK0 pin &gt;&gt;</td><td>.SCKE</td></tr> <tr> <td>Set SOT0 pin &gt;&gt;</td><td>.SOE</td></tr> <tr> <td>Set control register</td><td>SCR</td></tr> <tr> <td>Set parity or no parity &gt;&gt;</td><td>.PEN</td></tr> <tr> <td>Set even/odd parity &gt;&gt;</td><td>.P</td></tr> <tr> <td>Set stop bit length</td><td>.SBL</td></tr> <tr> <td>Set data length of one frame &gt;&gt;</td><td>.CL</td></tr> <tr> <td>Set data format of frame &gt;&gt;</td><td>.A/D</td></tr> <tr> <td>Error flag &gt;&gt;</td><td>.REC</td></tr> <tr> <td>Set reception operation enable &gt;&gt;</td><td>.RXE</td></tr> <tr> <td>Set transmission operation enable &gt;&gt;</td><td>.TXE</td></tr> <tr> <td>Control interrupt</td><td>SSR .TIE</td></tr> <tr> <td>Set transmission data</td><td>SIDR</td></tr> </table> <p>&lt;Start&gt;</p> <ul style="list-style-type: none"> <li>Start UART</li> </ul> <table border="1"> <tr> <td>Start UART0 transmission operation</td><td>SCR0 .TXE</td></tr> </table> <p>&lt;Interrupt&gt;</p> <ul style="list-style-type: none"> <li>Transmission interrupt processing</li> </ul> <table border="1"> <tr> <td>Send any data</td><td>SIDR</td></tr> </table> <p>&lt;Interrupt vector&gt;</p> <ul style="list-style-type: none"> <li>Set vector table</li> </ul> <p>Note:<br/>Setting related to clock and setting of __set_il (numeric value) are required in advance. See the chapter of clock and interrupt.</p> | Set interrupt level | ICR11 | Set I flag | (CCR) | Set mode register | SMR | Set operation mode >> | .MD1,MD0 | Set operation clock >> | .CS2-0 | Set SCK0 pin >> | .SCKE | Set SOT0 pin >> | .SOE | Set control register | SCR | Set parity or no parity >> | .PEN | Set even/odd parity >> | .P | Set stop bit length | .SBL | Set data length of one frame >> | .CL | Set data format of frame >> | .A/D | Error flag >> | .REC | Set reception operation enable >> | .RXE | Set transmission operation enable >> | .TXE | Control interrupt | SSR .TIE | Set transmission data | SIDR | Start UART0 transmission operation | SCR0 .TXE | Send any data | SIDR | <p>Program example</p> <pre> void Asynch_uart_sample(void) {     INT_initial();     Asynch_uart_initial();     Uart_start(); }  void INT_initial(void) {     IO_ICR11.byte = 0x10; /* Set UART transmission completion interrupt level                            (arbitrary value) */     __EI(); /* Enable interrupt */ }  void Asynch_uart_initial(void) {     IO_SMR.byte = 0x19; /* Setting value =0001_1001 */     ; /* bit7-6 = 00 MD1,MD0 asynchronous normal mode */     /* bit5-3 = 011 CS2-0 dedicated baud rate generator */     /* bit2 = 0 Undefined bit */     /* bit1 = 0 SCKE clock input pin */     /* bit0 = 1 SOE external pin */     IO_SCR.byte = 0x10; /* Setting value =0001_0000 */     /* bit15 = 0 PEN no parity */     /* bit14 = 0 P even parity */     /* bit13 = 0 SBL 1 stop bit */     /* bit12 = 1 CL 8-bit data */     /* bit11 = 0 A/D data frame */     /* bit10 = 0 REC error flag clear */     /* bit9 = 0 RXE reception operation disable */     /* bit8 = 0 TXE transmission operation enable */      IO_SSR.bit.TIE = 1; /* bit8 = 1 TIE transmission interrupt enable */     IO_SIDR = 0x0aa; /* Send any value of data */ }  void Uart_start(void) {     IO_SCR.bit.TXE = 1; /* bit8 = 1 Enable TXE transmission operation */ }  __interrupt void uart_tx_int(void) {     IO_SIDR = 0x0aa; /* Send any value of data */ }  #pragma intvect uart_tx_int 34 </pre> <p>Note:<br/>For the description form of the register, see "SAMPLE I/O REGISTER FILES FOR F<sup>2</sup>MC-16LX FAMILY MB90480/485 SERIES".</p> |
| Set interrupt level   | ICR11               |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set I flag  | (CCR)               |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set mode register   | SMR                 |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set operation mode >>   | .MD1,MD0            |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set operation clock >>  | .CS2-0              |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set SCK0 pin >>   | .SCKE               |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set SOT0 pin >>   | .SOE                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set control register  | SCR                 |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set parity or no parity >>  | .PEN                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set even/odd parity >>  | .P                  |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set stop bit length   | .SBL                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set data length of one frame >>   | .CL                 |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set data format of frame >>   | .A/D                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Error flag >>   | .REC                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set reception operation enable >>   | .RXE                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set transmission operation enable >>  | .TXE                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Control interrupt   | SSR .TIE            |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Set transmission data   | SIDR                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Start UART0 transmission operation  | SCR0 .TXE           |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |
| Send any data   | SIDR                |       |            |       |                   |     |                       |          |                        |        |                 |       |                 |      |                      |     |                            |      |                        |    |                     |      |                                 |     |                             |      |               |      |                                   |      |                                      |      |                   |          |                       |      |                                    |           |               |      |   |

## ■ Setting Method Other than Program Example

- Combination that can be set

The combination is shown below.

| Operation mode<br>(MD[1:0]) |  | Data length<br>(CL) | Parity (PEN)        | Parity<br>selection<br>(P) | Data format<br>(A/D) | Selection of<br>STOP bit length<br>(SBL) * |             | Presence or absence of error flags |                  |                |
|-----------------------------|--|---------------------|---------------------|----------------------------|----------------------|--|-------------|------------------------------------|------------------|----------------|
|                             |  |                     |                     |                            |                      |  |             | Overrun<br>(ORE)                   | Framing<br>(FRE) | Parity<br>(PE) |
| 0                           | Asynchronous-<br>normal mode (00)            | 7bit<br>(0)         | Not provided<br>(0) | --                         | --                   | 1bit<br>(0)                                | 2bit<br>(1) | ○                                  | ○                | ○              |
|                             |  |                     | Provided<br>(1)     | Even (0)                   |                      | 1bit<br>(0)                                | 2bit<br>(1) |                                    |                  |                |
|                             |  |                     |                     | Odd (1)                    |                      | 1bit<br>(0)                                | 2bit<br>(1) |                                    |                  |                |
|                             |  | 8bit<br>(1)         | Not provided<br>(0) | --                         |                      | 1bit<br>(0)                                | 2bit<br>(1) |                                    |                  |                |
|                             |  |                     | Provided<br>(1)     | Even (0)                   |                      | 1bit<br>(0)                                | 2bit<br>(1) |                                    |                  |                |
|                             |  |                     |                     | Odd (1)                    |                      | 1bit<br>(0)                                | 2bit<br>(1) |                                    |                  |                |
| 1                           | Asynchronous-<br>multiprocessor mode<br>(01) | 8bit<br>(1)         | Not provided<br>(0) | --                         | Address (1)          | 1bit<br>(0)                                | 2bit<br>(1) | ○                                  | ○                | --             |
|                             |  |                     |                     |                            | Address (0)          | 1bit<br>(0)                                | 2bit<br>(1) |                                    |                  |                |
| 2                           | CLK synchronous<br>mode (10)                 | 8bit<br>(1)         | Not provided<br>(0) | --                         | --                   | --   |             | ○                                  | --               | --             |

\*: STOP bit can be selected only at transmission. First bit is detected only at reception (second bit is ignored).

- Method to select operation modes

Set by the operation mode bits (SMR.MOD[1:0])

| Operation mode |                                  | Operation mode bit (MOD[1:0]) |
|----------------|----------------------------------|-------------------------------|
| Mode 0         | Asynchronous-normal mode         | Set to "00 <sub>B</sub> "     |
| Mode 1         | Asynchronous-multiprocessor mode | Set to "01 <sub>B</sub> "     |
| Mode 2         | CLK synchronous mode             | Set to "10 <sub>B</sub> "     |
| -----          |                                  | Setting "11" is disabled.     |

- Type of operation clock and selection method

Set by the operation clock select bit (SMR.CS[2:0]).

| Clock input                   | Operation clock select bits (CS[2:0])             |
|-------------------------------|---|
| Dedicated baud rate generator | Set to "000 <sub>B</sub> " to "101 <sub>B</sub> " |
| To select internal clock      | Set to "110 <sub>B</sub> "                        |
| To select external clock      | Set to "111 <sub>B</sub> "                        |

## CHAPTER 19 UART

- Method to control SCK, SIN and SOT pins

Set as follows.

|                   | UART register                |
|-------------------|------------------------------|
| To input SCK pin  | DDR7.P72 = 0<br>SMR.SCKE = 0 |
| To output SCK pin | SMR.SCKE = 1                 |
| To input SIN pin  | DDR7.P70 = 0                 |
| To output SOT pin | SMR.SOE = 1                  |

- Method to enable/stop UART operation

Set by the reception control bit (SCR.RXE).

| Content of control                    | Reception control bit (RXE) |
|---------------------------------------|-----------------------------|
| To disable reception operation (stop) | Set to "0"                  |
| To enable reception operation         | Set to "1"                  |

Set by the transmission control bit (SCR.TXE).

| Content of control                       | Transmission control bit (TXE) |
|--|--------------------------------|
| To disable transmission operation (stop) | Set to "0"                     |
| To enable transmission operation         | Set to "1"                     |

- Method to set parity

Set by the parity setting bit (SCR.PEN) and parity select bit (SCR.P).

| Operation             | Parity setting bit, parity select bit (PEN, P)  |
|-----------------------|---|
| To set no parity      | Set to "00 <sub>B</sub> " or "01 <sub>B</sub> " |
| To set to even parity | Set to "10 <sub>B</sub> "                       |
| To set to odd parity  | Set to "11 <sub>B</sub> "                       |

- Method to set data length

Set by the data length select bit (SCR.CL).

| Operation              | Data length select bit (CL) |
|------------------------|-----------------------------|
| To set to 7-bit length | Set to "0"                  |
| To set to 8-bit length | Set to "1"                  |

- Method to select stop bit length

Set by the stop bit length select bit (SCR, SBL).

| Operation                | Stop bit length select bit (SBL) |
|--------------------------|----------------------------------|
| To set STOP bit to 1 bit | Set to "0"                       |
| To set STOP bit to 2 bit | Set to "1"                       |

- Method to clear error flag

Set by the error flag clear bit (SCR.REC).

| Content of control                 | Error flag clear bit (REC) |
|------------------------------------|----------------------------|
| To clear error flag (PE, OFE, PRE) | Write "0"                  |

- Method to set transfer direction

Set by the setting direction selection bit (SSR.BDS).

The transfer direction of LSB/MSB can be selected from any operation mode.

| Content of control                               | Setting direction selection bit (BDS) |
|--|---------------------------------------|
| To set LSB transfer (from least significant bit) | Set to "0"                            |
| To set MSB transfer (from most significant bit)  | Set to "1"                            |

- Method to clear reception completion flag

Perform in the following method.

| Content of control                 | Serial input register (SIDR) |
|------------------------------------|------------------------------|
| To clear reception completion flag | Read SIDR register           |

Reading the SIDR register for the first time starts the reception.

- Type of error flag and meaning

There are three error flags, and they have the following meanings.

| Error flag          | Meaning   |
|---------------------|---|
| Parity error (PE)   | Error is found in the received value (numeric value)    |
| Overrun error (ORE) | Next data is received before the reception data is read |
| Framing error (FRE) | Error is found in format of the received data           |

- Storage register of reception data

The reception data is stored in the serial input register (SIDR).

- Status confirm method of write timing of transmission data

The transmission data can be checked by the transmission buffer empty flag (SSR.TDRE).

- Write register of transmission data

The transmission data is written to the serial output register (SODR).



- Method to clear the transmission buffer empty flag

Perform in the following method.

| Content of control                      | Serial output register (SODR) |
|---|-------------------------------|
| To clear transmission buffer empty flag | Write to SODR register        |

Writing to the SODR register for the first time starts the transmission.

- Method to select data format (address/data)

Set by the data length select bit (SCR.A/D) .

| Operation                           | Data length select bit (A/D) |
|-------------------------------------|------------------------------|
| To set 8-bit to data ("L" level)    | Set to "0"                   |
| To set 8-bit to address ("H" level) | Set to "1"                   |

It is valid only at the transmission. A/D bit is ignored at the reception.

- Method to start reception/transmission

Perform in the following method.

- Mode 0/1, transmission :
  1. Enable the transmission operation.
  2. Write data to the serial output register (SODR).  
( = start transmission)
- Mode 0/1, reception :
  1. Enable reception.
  2. Read the serial input register (SIDR).  
( Dummy read = start reception)
- Mode 2, transmission/  
reception :
  1. Enable transmission. (Enabling reception can be omitted.)
  2. Write data to the serial output register (SODR).  
( = start transmission/reception)  
(It is necessary to write to the serial output register (SODR) only in the reception operation.)

- Operation stop and state

- Mode 0/1, transmission :  
When the transmission operation is disabled, last transmission data is transmitted after the transmission buffer is empty. Thus, the operation is stopped after the stop bit is transmitted.
- Mode 0/1, reception :  
After the currently received data is completed (after reception of stop bit) when the reception operation is disabled, the reception data is transferred from shifter to register and then the operation is stopped.
- Mode 0/1, transmission/reception :  
When both reception and transmission operations are disabled, the operation is stopped after data to be transmitted/received is completed and then the reception data is transferred from shifter to register.

● Method to check completion of operation

Perform in the following method.

- Mode 0/1, transmission : Check the SSR register empty flag after next transmission data is written. (completion can be checked when next transmission data is transferred from register to shifter and the empty flag in the serial output register is set to "1".)
- Mode 0/1, reception : Check register full flag. (Completion can be checked when the full flag in the serial input register is set to "1".)
- Mode 2, transmission/reception : Check register full flag. (Completion can be checked when the full flag in the serial input register is set to "1".)

● Method to set baud rate

See section "19.5 UART Operations".

● Interrupt related register

The relationship between the UART interrupt vector and UART interrupt level setting register is shown in the following table.

For details of the interrupt level and interrupt vector, see "CHAPTER 3 INTERRUPT".

|                   | Interrupt vector                    | Interrupt level setting register                                      |
|-------------------|-------------------------------------|---|
| UART reception    | #36<br>Address: FFFF6C <sub>H</sub> | Interrupt control register 12 (ICR12)<br>Address: 0000BC <sub>H</sub> |
| UART transmission | #34<br>Address: FFFF74 <sub>H</sub> | Interrupt control register 11 (ICR11)<br>Address: 0000BB <sub>H</sub> |

● Type of interrupt

4 interrupt sources are provided at reception side and one at transmission side.

|                   |  |
|-------------------|--|
| UART reception    | The interrupt request occurs due to the sources that occurred first among reception completion (serial input register full), parity error, overrun error, and framing error. |
| UART transmission | The interrupt request occurs at the transmission buffer empty.   |

● Method to enable/disable/clear interrupt

Enabling/disabling interrupt is set by the interrupt request enable bit (SSR.RIE, SSR.TIE).

|                              | UART reception                     | UART transmission                  |
|------------------------------|------------------------------------|------------------------------------|
|                              | Interrupt request enable bit (RIE) | Interrupt request enable bit (TIE) |
| To disable interrupt request | Set to "0"                         |                                    |
| To enable interrupt request  | Set to "1"                         |                                    |

The interrupt request is cleared in the following method.

|                            | UART reception   | UART transmission  |
|----------------------------|--|--|
| To clear interrupt request | The reception completion flag (RDRF) is set to "0" by reading the serial input register (SIDR).  | The transmission buffer empty flag (TDRE) is set to "0" when data is written to the serial output register (SODR). |
|                            | The error flag (PE,ORE,FRE) is set to "0" when "0" is written to the error flag clear bit (REC). | -  |

# CHAPTER 20    CHIP SELECTION FACILITY

---

**This chapter provides an overview, of the chip selection facility explains the configuration, and its operation, the configuration and functions of its registers.**

---

- 20.1 Overview of Chip Selection Facility
- 20.2 Configuration of Chip Selection Facility
- 20.3 Configuration and Functions of Chip Selection Facility Registers
- 20.4 Operation of the Chip Selection Facility

## 20.1 Overview of Chip Selection Facility

---

The chip selection facility is a module used to generate a chip selection signal for simplified memory connection to the outside. It contains four chip selection output pins. The chip selection facility has four chip select output pins and enables an area within the hardware to be specified via an output setting register, and if the device detects an access to that external address, it outputs a selection signal via the corresponding pin.

---

### ■ Overview of the Chip Selection Facility

The chip selection facility contains two 8-bit registers for each output pin. One register (CARx) is used to specify the upper 8 bits of the compared address, allowing area within 64 K bytes to be specified. Another register (CMRx) is used to mask the corresponding bits of the compared address so that values above 64 K bytes can be specified for the area to be detected.

In external bus hold mode, CS output is set to high impedance mode.

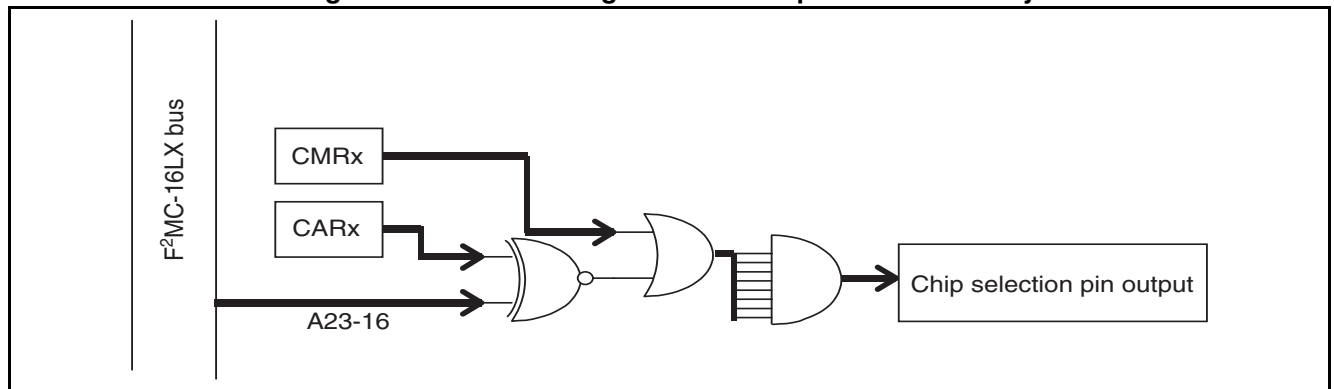
## 20.2 Configuration of Chip Selection Facility

This section describes the block diagram and pin related to the chip selection facility.

### ■ Block Diagram of the Chip Selection Facility

Figure 20.2-1 shows a block diagram of the chip selection facility.

**Figure 20.2-1 Block Diagram of the Chip Selection Facility**



### ■ Pin Related to Chip Selection Facility

The pin related to the chip selection facility has four CS0/CS1/CS2/CS3 output pins. The CS0/CS1/CS2/CS3 pins function as the general-purpose I/O port (P90/CS0, P91/CS1, P92/CS2, P93/CS3) and the output pin of the chip selection facility.

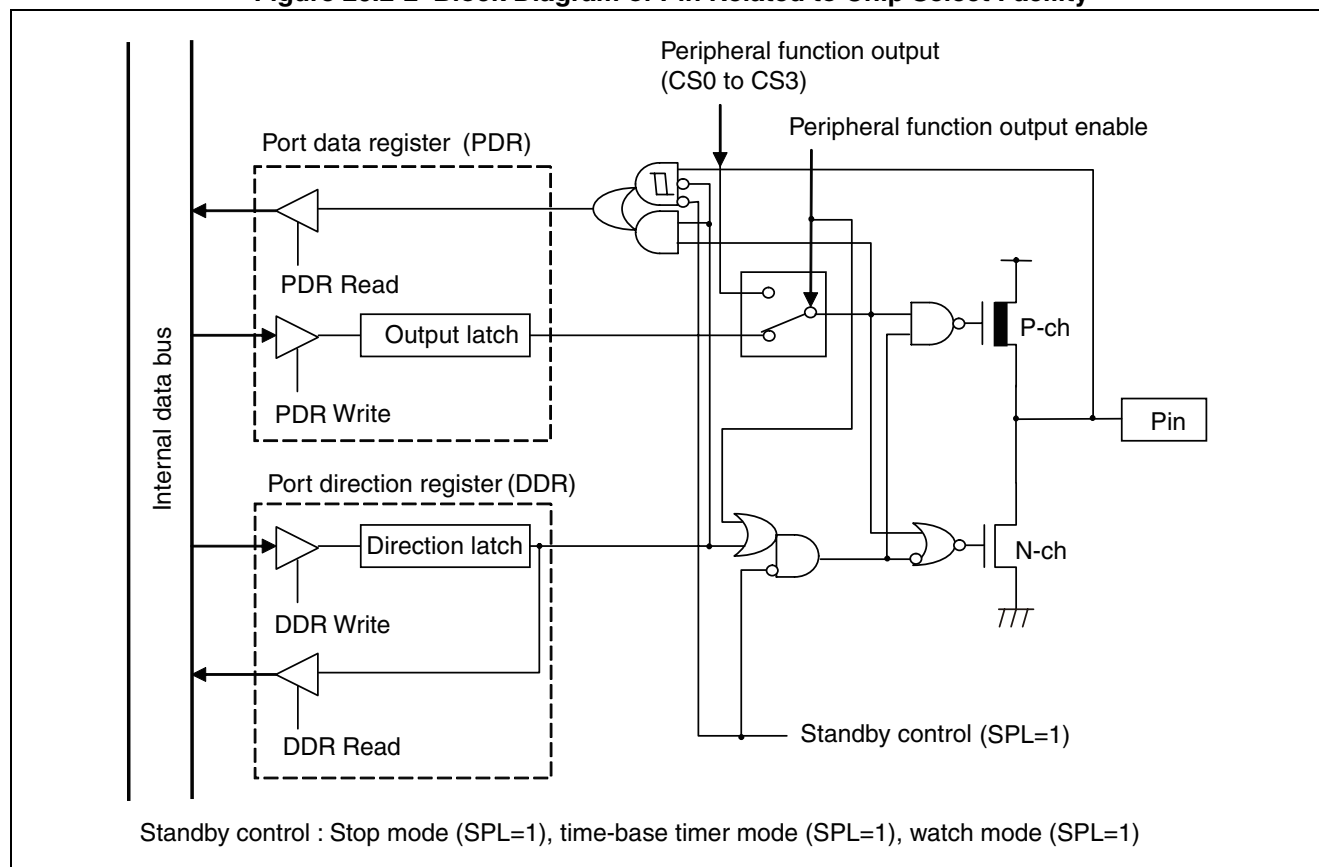
#### ○ Setting when using as CS0/CS1/CS2/CS3 pins

When the CS0/CS1/CS2/CS3 are used as output by the chip selection facility, be sure to set the chip selection control register (CSCR) to enable output (OPL [3:0] → "1").

Also, other resource that is assigned to these pins cannot be used when the chip selection facility is used.

### ■ Block Diagram of Pin Related to Chip Select Facility

**Figure 20.2-2 Block Diagram of Pin Related to Chip Select Facility**



## 20.3 Configuration and Functions of Chip Selection Facility Registers

This section describes the configuration and functions of the registers used by the chip selection facility.

### ■ List of Registers Used for the Chip Selection Facility

Figure 20.3-1 lists the registers for the chip selection facility.

**Figure 20.3-1 List of Registers for the Chip Selection Facility**

|      |      |       |   |  |
|------|------|-------|---|--|
| 15   | 8    | 7     | 0 |  |
| CAR0 | CMR0 | (R/W) |   |  |
| CAR1 | CMR1 | (R/W) |   |  |
| CAR2 | CMR2 | (R/W) |   |  |
| CAR3 | CMR3 | (R/W) |   |  |
| CALR | CSCR | (R/W) |   |  |

|                     |       |       |       |       |       |       |       |       |                                   |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|-----------------------------------|
| 0000C0 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | CMRx                              |
| 0000C2 <sub>H</sub> | M7    | M6    | M5    | M4    | M3    | M2    | M1    | M0    | Chip selection area MASK register |
| 0000C4 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/Write                        |
| 0000C6 <sub>H</sub> | (0)   | (0)   | (0)   | (0)   | (1)   | (1)   | (1)   | (1)   | Initial value                     |

|                     |       |       |       |       |       |     |       |       |                              |
|---------------------|-------|-------|-------|-------|-------|-----|-------|-------|------------------------------|
| 0000C1 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10  | 9     | 8     | CARx                         |
| 0000C3 <sub>H</sub> | A7    | A6    | A5    | A4    | A3    | A2  | A1    | A0    | Chip selection area register |
| 0000C5 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (W) | (R/W) | (R/W) | Read/Write                   |
| 0000C7 <sub>H</sub> | (1)   | (1)   | (1)   | (1)   | (1)   | (1) | (1)   | (1)   | Initial value                |

|                     |     |     |     |     |       |       |       |       |                                 |
|---------------------|-----|-----|-----|-----|-------|-------|-------|-------|---------------------------------|
| 0000C8 <sub>H</sub> | 7   | 6   | 5   | 4   | 3     | 2     | 1     | 0     | CSCR                            |
|                     | -   | -   | -   | -   | OPL3  | OPL2  | OPL1  | OPL0  | Chip selection control register |
|                     | (-) | (-) | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | Read/Write                      |
|                     | (-) | (-) | (-) | (-) | (0)   | (0)   | (0)   | (*)   | Initial value                   |

|                     |     |     |     |     |       |       |       |       |                                      |
|---------------------|-----|-----|-----|-----|-------|-------|-------|-------|--------------------------------------|
| 0000C9 <sub>H</sub> | 15  | 14  | 13  | 12  | 11    | 10    | 9     | 8     | CALR                                 |
|                     | -   | -   | -   | -   | ACTL3 | ACTL2 | ACTL1 | ACTL0 | Chip selection active level register |
|                     | (-) | (-) | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | Read/Write                           |
|                     | (-) | (-) | (-) | (-) | (0)   | (0)   | (0)   | (0)   | Initial value                        |



### 20.3.1 Chip Select Area MASK Register (CMRx)

This section describes the configuration and functions of the chip selection area MASK register (CMRx).

■ Chip Selection Area MASK Register (CMRx)

The diagram below shows the bit configuration of the chip selection area MASK register (CMRx).

|                     |       |       |       |       |       |       |       |       |                                   |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|-----------------------------------|
| 0000C0 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | CMRx                              |
| 0000C2 <sub>H</sub> | M7    | M6    | M5    | M4    | M3    | M2    | M1    | M0    | Chip selection area MASK register |
| 0000C4 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Read/Write                        |
| 0000C6 <sub>H</sub> | (0)   | (0)   | (0)   | (0)   | (1)   | (1)   | (1)   | (1)   | Initial value                     |
| [bit7 to 0]M7 to M0 |       |       |       |       |       |       |       |       |                                   |

[bit7 to bit0] M7 to M0

These bits are used to specify an address decode area for the chip selection pin. Set the corresponding bit to "1" for masking.

These bits are used to specify an area of 128 K bytes or more.

Note:

If all bits are masked, the CS pin becomes active using all external accessible areas.

## 20.3.2 Chip Selection Area Register (CARx)

This section describes the configuration and functions of the chip selection area register (CARx).

### ■ Chip Selection Area Register (CARx)

The diagram below shows the bit configuration of the chip selection area register (CARx).

|                     |       |       |       |       |       |     |       |       |                           |
|---------------------|-------|-------|-------|-------|-------|-----|-------|-------|---------------------------|
| 0000C1 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10  | 9     | 8     | CARx                      |
| 0000C3 <sub>H</sub> | A7    | A6    | A5    | A4    | A3    | A2  | A1    | A0    | Chip select area register |
| 0000C5 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (W) | (R/W) | (R/W) | Read/Write                |
| 0000C7 <sub>H</sub> | (1)   | (1)   | (1)   | (1)   | (1)   | (1) | (1)   | (1)   | Initial value             |
| [bit7 to 0]A7 to A0 |       |       |       |       |       |     |       |       |                           |

#### [bit7 to bit0] A7 to A0

These bits are used to set the address decode area for the chip select pin. They specify the upper 8 bits of the address value, allowing an area within 64 K bytes to be specified.

Note:

The CS pin is not set to active while CPU is performing internal access (such as built-in RAM, built-in ROM, and I/O).

### 20.3.3 Chip Selection Control Register (CSCR)

This section describes the configuration and functions of the chip selection control register (CSCR).

■ Chip Selection Control Register (CSCR)

The diagram below shows the bit configuration of the chip selection control register (CSCR).

|   |     |     |     |     |       |       |       |       |                                 |
|---|-----|-----|-----|-----|-------|-------|-------|-------|---------------------------------|
|   | 7   | 6   | 5   | 4   | 3     | 2     | 1     | 0     | CSCR                            |
| 0000C8 <sub>H</sub>   | -   | -   | -   | -   | OPL3  | OPL2  | OPL1  | OPL0  | Chip selection control register |
|   | (-) | (-) | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | Read/Write                      |
|   | (-) | (-) | (-) | (-) | (0)   | (0)   | (0)   | (*)   | Initial value                   |
| * : The initial value of this bit is "1" or "0". The value depends on the mode pins (MD2, MD1, MD0 pins). |     |     |     |     |       |       |       |       |                                 |

[bit7 to bit4] Unused bits

These bits are unused. In read operations, the return value for these bits is undefined.

[bit3 to bit0] OPL3 to OPL0

These bits are used to specify whether CS3 to CS0 are output to the external pin.

The operational settings are as follows:

- "0": Decode output from each CS3 to CS0 pin is prohibited
- "1": Decode output from each CS3 to CS0 pin is allowed

Notes:

- The initial value of OPL0 is set to "1" in external vector mode, and set to "0" in internal vector mode.
- Enabling CS3 to CS0 output must be performed after all settings have been made.
- Change settings during operation only after prohibiting OPL3 to OPL0 to output.

## 20.3.4 Chip Selection Active Level Register (CALR)

This section describes the configuration and functions of the chip selection active level register (CALR).

### ■ Chip Selection Active Level Register (CALR)

The diagram below shows the bit configuration of the chip selection active level register (CALR).

|                     | 15  | 14  | 13  | 12  | 11    | 10    | 9     | 8     | CALR                                |
|---------------------|-----|-----|-----|-----|-------|-------|-------|-------|-------------------------------------|
| 0000C9 <sub>H</sub> | -   | -   | -   | -   | ACTL3 | ACTL2 | ACTL1 | ACTL0 | Chip selector active level register |
|                     | (-) | (-) | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | Read/Write                          |
|                     | (-) | (-) | (-) | (-) | (0)   | (0)   | (0)   | (0)   | Initial value                       |

#### [bit15 to bit12] Unused bits

These bits are unused. In read operations, the return value for these bits is undefined.

#### [bit11 to bit8] ACTL3 to ACTL0

These bits set the active level of each CS3 to CS0 pin.

The followings are set.

- "0": Each CS3 to CS0 pin outputs "L" after decoding.
- "1": Each CS3 to CS0 pin outputs "H" after decoding.

#### Notes:

- Before changing the active level, prohibit output via the chip selection control register.
- Writing these bits in units of words is prohibited. Always write these bits in units of bytes. This will prevent the enabling of output with the write operation at the same time that the active level is changed.

# 20.4 Operation of the Chip Selection Facility

This section describes the operations of the chip selection facility.

## ■ Outline of Operations

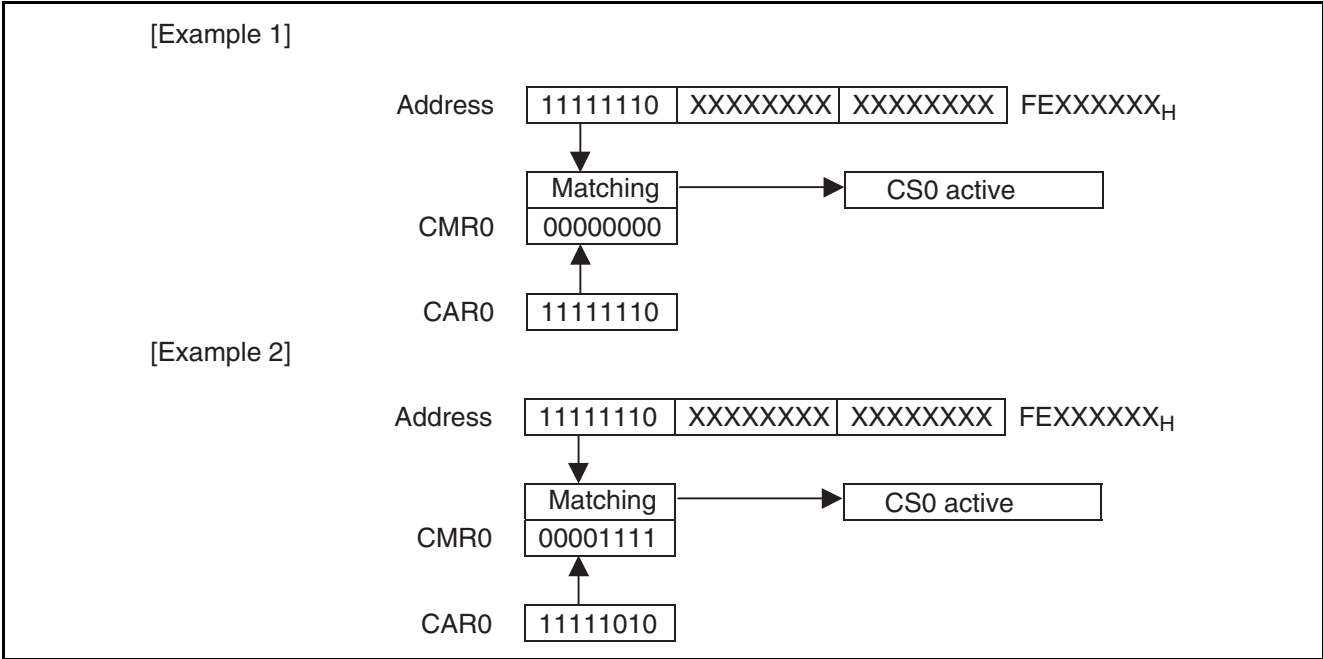
When the CPU accesses program or data, the chip selection facility is activated if a match between the upper 8 bits of an address and CAR0/1/2/3 is detected. Addresses for which the corresponding bits in CMR0/CMR1/CMR2/CMR3 are set to "1" are ignored, and decoding becomes possible for an area from 64 K bytes to 16 MB.

The CS pin is not set to active while CPU is performing internal access (such as built-in RAM, built-in ROM, and built-in I/O).

## ■ Example of Using the Chip Selection Facility

Figure 20.4-1 shows an example of using the chip selection facility.

Figure 20.4-1 Example of Using the Chip Selection Facility



### ■ Notes on Using the Chip Selection Facility

- The CS0 pin always becomes active to read the reset vector if used in external vector mode. In the address space F00000<sub>H</sub> to FFFFFFF<sub>H</sub> (1 MB: initial value), always use this pin only for the access to program ROM, since the corresponding decode signal will be output immediately after a reset. In this case, the active level of the CS0 pin is set to "L", and "H" is output at reset. In internal vector mode, this pin, like the CS3 to CS1 pins, is used as a general-purpose port. Therefore, switch the CS0 pin to the output state after the corresponding settings have been made.
- Only enable output after the settings of the chip selection area register, chip selection area MASK register, and chip selection active level register have been made.
- Note that, since the chip selection output is shared with pins P90 to P93, chip selection output will not be available while the resource assigned to this pin is in use.
- If the pin is set to the hold state when the external bus is used, output is disabled and the pin is set to high impedance state. In these cases, always set the shared general-purpose port to act as an input.
- In sleep or stop mode, the CS pin is not active.
- The chip selection facility cannot be used during access to built-in DMA.



# CHAPTER 21 ADDRESS MATCH DETECTION FUNCTION

---

**This chapter explains the functions and operations of the address match detection.**

---

- 21.1 Overview of Address Match Detection Function
- 21.2 Block Diagram of Address Match Detection Function
- 21.3 Configuration of Registers for Address Match Detection Function
- 21.4 Explanation of Operation of Address Match Detection Function
- 21.5 Program Example of Address Match Detection Function



## 21.1 Overview of Address Match Detection Function

---

If the address of the instruction to be processed next to the instruction currently processed by the program matches the address set in the program address detection registers, the address match detection function forcibly replaces the next instruction to be processed by the program with the INT9 instruction to branch to the interrupt processing program. Since the address match detection function can use the INT9 interrupt, the program can be corrected by patch processing.

---

### ■ Overview of Address Match Detection Function

- There are two program address detection registers (PADR0 and PADR1), each of which has an interrupt enable bit. The generation of an interrupt due to a match between the address held in the address latch and the address set in the detection address setting registers can be enabled and disabled for each register.

## 21.2 Block Diagram of Address Match Detection Function

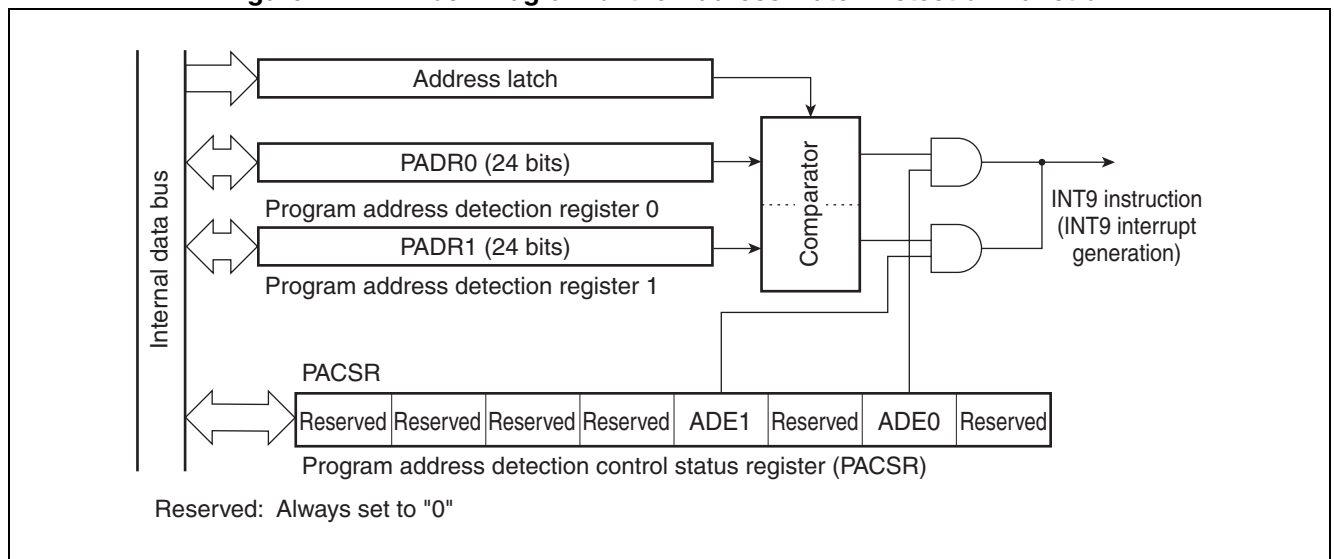
The address match detection module consists of the following blocks:

- Address latch
- Program address detection control status register (PACSR)
- Program address detection registers (RADR)

### ■ Block Diagram of Address Match Detection Function

Figure 21.2-1 shows the block diagram of the address match detection function.

**Figure 21.2-1 Block Diagram of the Address Match Detection Function**



#### ○ Address latch

The address latch stores the value of the address output to the internal data bus.

#### ○ Program address detection control status register (PACSR)

The program address detection control status register enables or disables output of an interrupt at an address match.

#### ○ Program address detection registers (PADR0, PADR1)

The program address detection registers set the address that is compared with the value of the address latch.

Note:

The addresses of the program address detection register are "1FF0<sub>H</sub>" to "1FF5<sub>H</sub>" and are included in the RAM area. Therefore, the access to the RAM area should not be performed during the use of this function.

## 21.3 Configuration of Registers for Address Match Detection Function

This section details the registers used by the address match detection function.

### ■ List of Registers and Initial Values of Address Match Detection Function

Figure 21.3-1 List of Registers and Initial Values of Address Match Detection Function

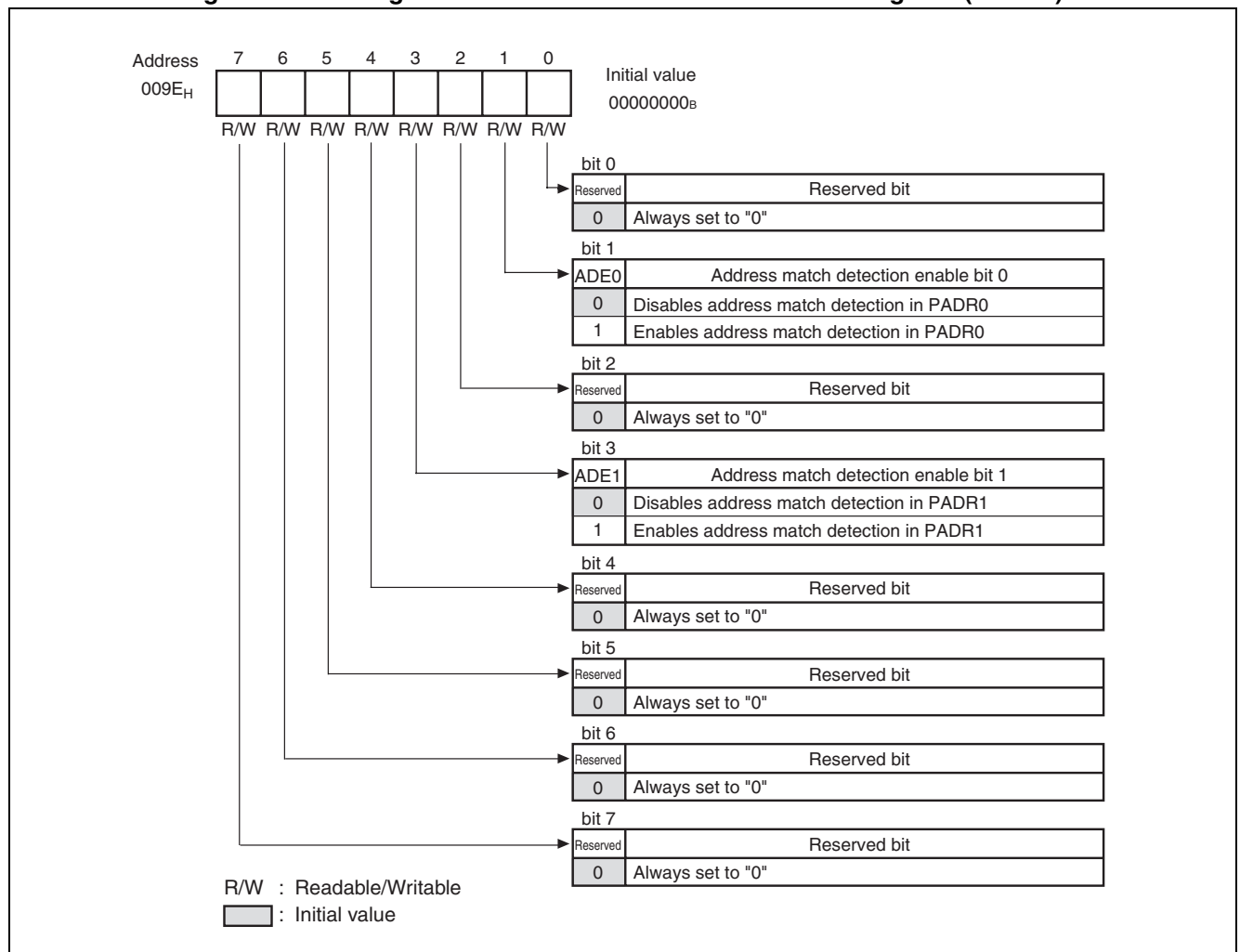
|   |     |    |    |    |    |    |    |   |   |
|---|-----|----|----|----|----|----|----|---|---|
| Program address detection control status register<br>(PACSR): Address 009E <sub>H</sub> | bit | 7  | 6  | 5  | 4  | 3  | 2  | 1 | 0 |
|   |     | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 |
| Program address detection register 0<br>(PADR0): High Address 1FF2 <sub>H</sub>         | bit | 7  | 6  | 5  | 4  | 3  | 2  | 1 | 0 |
|   |     | ×  | ×  | ×  | ×  | ×  | ×  | × | × |
| Program address detection register 0<br>(PADR0): Middle Address 1FF1 <sub>H</sub>       | bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|   |     | ×  | ×  | ×  | ×  | ×  | ×  | × | × |
| Program address detection register 0<br>(PADR0): Low Address 1FF0 <sub>H</sub>          | bit | 7  | 6  | 5  | 4  | 3  | 2  | 1 | 0 |
|   |     | ×  | ×  | ×  | ×  | ×  | ×  | × | × |
| Program address detection register 1<br>(PADR1): High Address 1FF5 <sub>H</sub>         | bit | 7  | 6  | 5  | 4  | 3  | 2  | 1 | 0 |
|   |     | ×  | ×  | ×  | ×  | ×  | ×  | × | × |
| Program address detection register 1<br>(PADR1): Middle Address 1FF4 <sub>H</sub>       | bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|   |     | ×  | ×  | ×  | ×  | ×  | ×  | × | × |
| Program address detection register 1<br>(PADR1): Low Address 1FF3 <sub>H</sub>          | bit | 7  | 6  | 5  | 4  | 3  | 2  | 1 | 0 |
|   |     | ×  | ×  | ×  | ×  | ×  | ×  | × | × |
| ×: Undefined  |     |    |    |    |    |    |    |   |   |

### 21.3.1 Program Address Detection Control Status Register (PACSR)

The program address detection control status register (PACSR) enables or disables output of an interrupt at an address match. If an address match is detected when output of an interrupt at an address match is enabled, the INT9 interrupt is output.

#### ■ Program Address Detection Control Status Register (PACSR)

Figure 21.3-2 Program Address Detection Control Status Register (PACSR)



## CHAPTER 21 ADDRESS MATCH DETECTION FUNCTION

**Table 21.3-1 Functions of Program Address Detection Control Status Register (PACSR)**

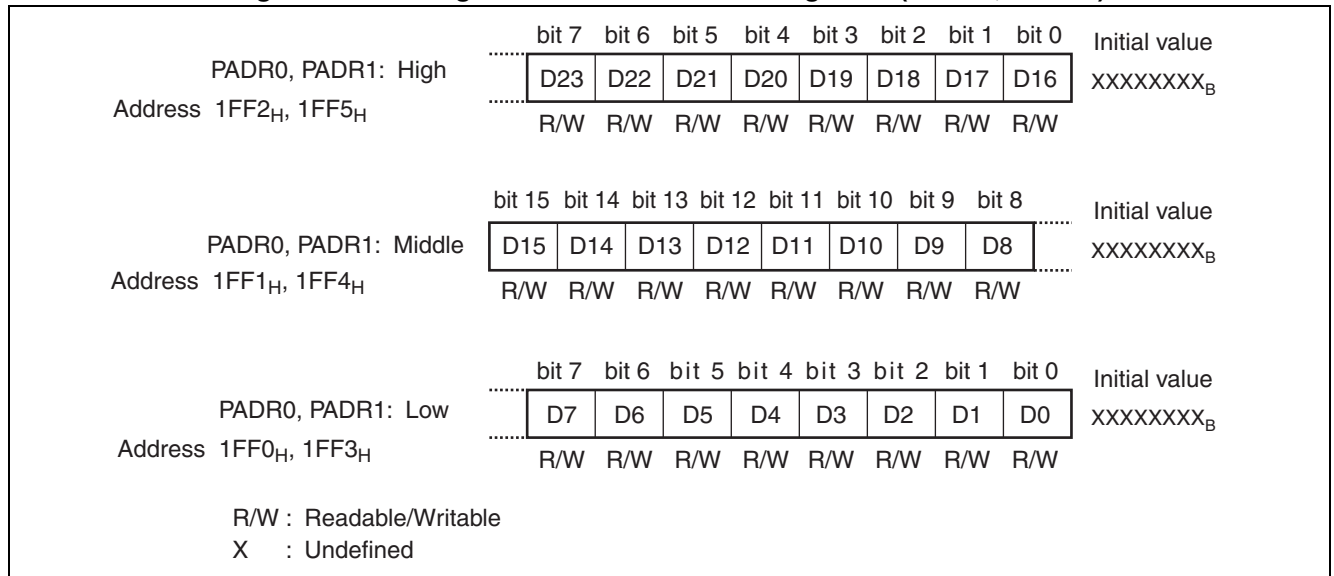
| Bit Name     |   | Function  |
|--------------|---|---|
| bit7 to bit4 | reserved: reserved bit                        | Always set to "0".  |
| bit3         | ADE1:<br>Address match detection enable bit 1 | <p>The address match detection operation with the program address detection register 1 (PADR1) is enabled or disabled.</p> <p><b>When set to "0":</b> Disables the address match detection operation.</p> <p><b>When set to "1":</b> Enables the address match detection operation.</p> <ul style="list-style-type: none"> <li>When the value of program address detection registers 1 (PADR1) matches with the value of address latch at enabling the address match detection operation (AD1E = 1), the INT9 instruction is immediately executed.</li> </ul> |
| bit2         | reserved: reserved bit                        | Always set to "0".  |
| bit1         | ADE0:<br>Address match detection enable bit 0 | <p>The address match detection operation with the program address detection register 0 (PADR0) is enabled or disabled.</p> <p><b>When set to "0":</b> Disables the address match detection operation.</p> <p><b>When set to "1":</b> Enables the address match detection operation.</p> <ul style="list-style-type: none"> <li>When the value of program address detection register 0 (PADR0) matches with the value of address latch at enabling the address match detection operation (AD0E = 1), the INT9 instruction is immediately executed.</li> </ul>  |
| bit0         | reserved: reserved bit                        | Always set to "0".  |

## 21.3.2 Program Address Detection Registers (PADR0, PADR1)

The value of an address to be detected is set in the program address detection registers. When the address of the instruction processed by the program matches the address set in the program address detection registers, the next instruction is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.

### ■ Program Address Detection Registers (PADR0, PADR1)

Figure 21.3-3 Program Address Detection Registers (PADR0, PADR1)



## ■ Functions of Program Address Detection Registers (PADR0 and PADR1)

There are two program address detection registers (PADR0 and PADR1) that consist of a high byte, middle byte, and low byte, totaling 24 bits.

**Table 21.3-2 Address Setting of Program Address Detection Registers**

| Register Name                                | Interrupt Output Enable | Address Setting |   |
|--|-------------------------|-----------------|---|
| Program address detection register 0 (PADR0) | PACSR: AD0E             | High            | Set the upper 8 bits of program address 0 (bank). |
|  |                         | Middle          | Set the middle 8 bits of program address 0.       |
|  |                         | Low             | Set the lower 8 bits of program address 0.        |
| Program address detection register 1 (PADR1) | PACSR: AD1E             | High            | Set the upper 8 bits of program address 1 (bank). |
|  |                         | Middle          | Set the middle 8 bits of program address 1.       |
|  |                         | Low             | Set the lower 8 bits of program address 1.        |

In the program address detection registers (PADR0 and PADR1), starting address (first byte) of instruction to be replaced by INT9 instruction should be set.

**Figure 21.3-4 Setting of Starting Address of Instruction Code to be Replaced by INT9 Instruction**

| Address | Instruction code | Mnemonic |            |
|---------|------------------|----------|------------|
| FF001C: | A8 00 00         | MOVW     | RW0, #0000 |
| FF001F: | 4A 00 00         | MOVW     | A, #0000   |
| FF0022: | 4A 80 08         | MOVW     | A, #0880   |

Set to detection address (High : FF<sub>H</sub>, Middle : 00<sub>H</sub>, Low : 1F<sub>H</sub>)

**Notes:**

- When an address of other than the first byte is set to the program address detection register (PADR0 and PADR1), the instruction code is not replaced by INT9 instruction and a program of an interrupt processing is not performed. When the address is set to the second byte or subsequent, the address set by the instruction code is replaced by "01<sub>H</sub>" (INT9 instruction code) and, which may cause malfunction.
- The program address detection registers (PADR0 and PADR1) should be set after disabling the address match detection (PACSR: AD0E = 0 or AD1E = 0) of corresponding program address detection control status registers. If the program address detection registers are changed without disabling the address match detection, the address match detection function will work immediately after an address match occurs during writing address, which may cause malfunction.
- The address match detection function can be used only for addresses of the internal ROM area. If addresses of the external memory area are set, the address match detection function will not work and the INT9 instruction will not be executed.

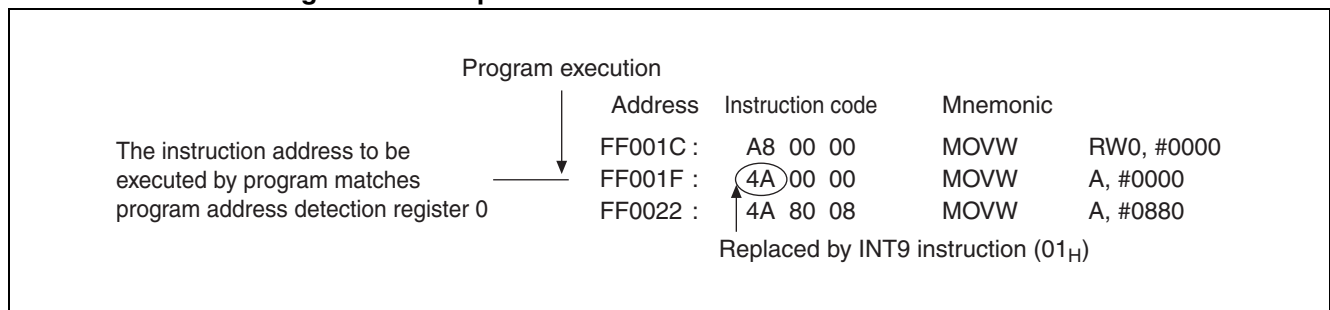
## 21.4 Explanation of Operation of Address Match Detection Function

If the addresses of the instructions executed in the program match those set in the program address detection registers (PADR0 and PADR1), the address match detection function will replace the first instruction code executed by CPU with the INT9 instruction (01<sub>H</sub>) to branch to the interrupt processing program.

### ■ Operation of Address Match Detection Function

Figure 21.4-1 shows the operation of the address match detection function.

**Figure 21.4-1 Operation of Address Match Detection Function**



### ■ Setting Detection Address

1. Disable the program address detection register 0 (PADR0) where the detect address is set for address match detection (PACSR: AD0E = 0).
2. Set the detected address in the program address detection register 0 (PADR0). Set "FF<sub>H</sub>" at the higher bits of the program address detection register 0 (PADR0), "00<sub>H</sub>" at the middle bits, and "1F<sub>H</sub>" at the lower bits.
3. Enable the program address detection register 0 (PADR0) where the detection address is set for address match detection (PACSR: AD0E = 1).

### ■ Program Execution

1. If the address of the instruction to be executed in the program matches the set detect address, the first instruction code at the matched address is replaced by the INT9 instruction code ("01<sub>H</sub>").
2. When the INT9 instruction is executed, INT9 interrupt is generated and then interrupt processing program is executed.



## 21.4.1 Example of using Address Match Detection Function

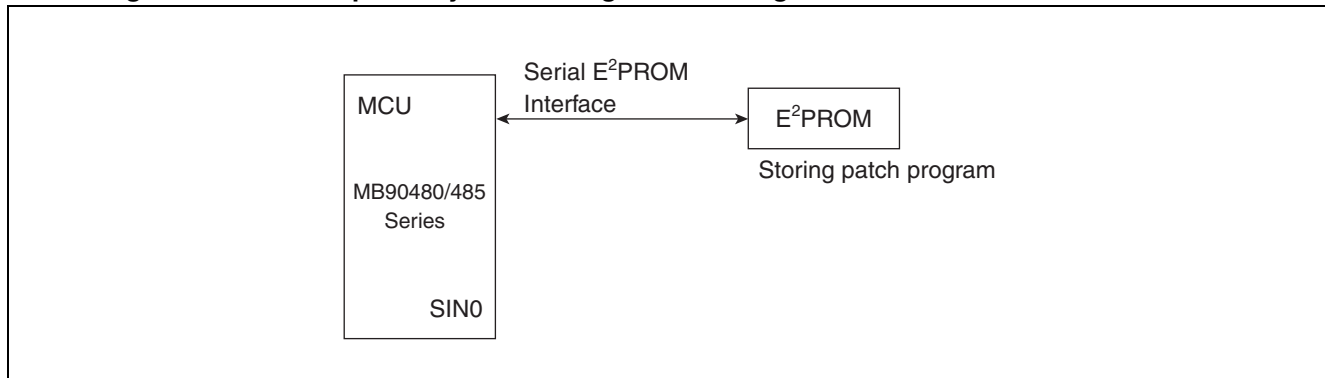
This section gives an example of patch processing for program correction using the address match detection function.

### ■ System Configuration and E<sup>2</sup>PROM Memory Map

#### ○ System configuration

Figure 21.4-2 gives an example of the system configuration using the address match detection function.

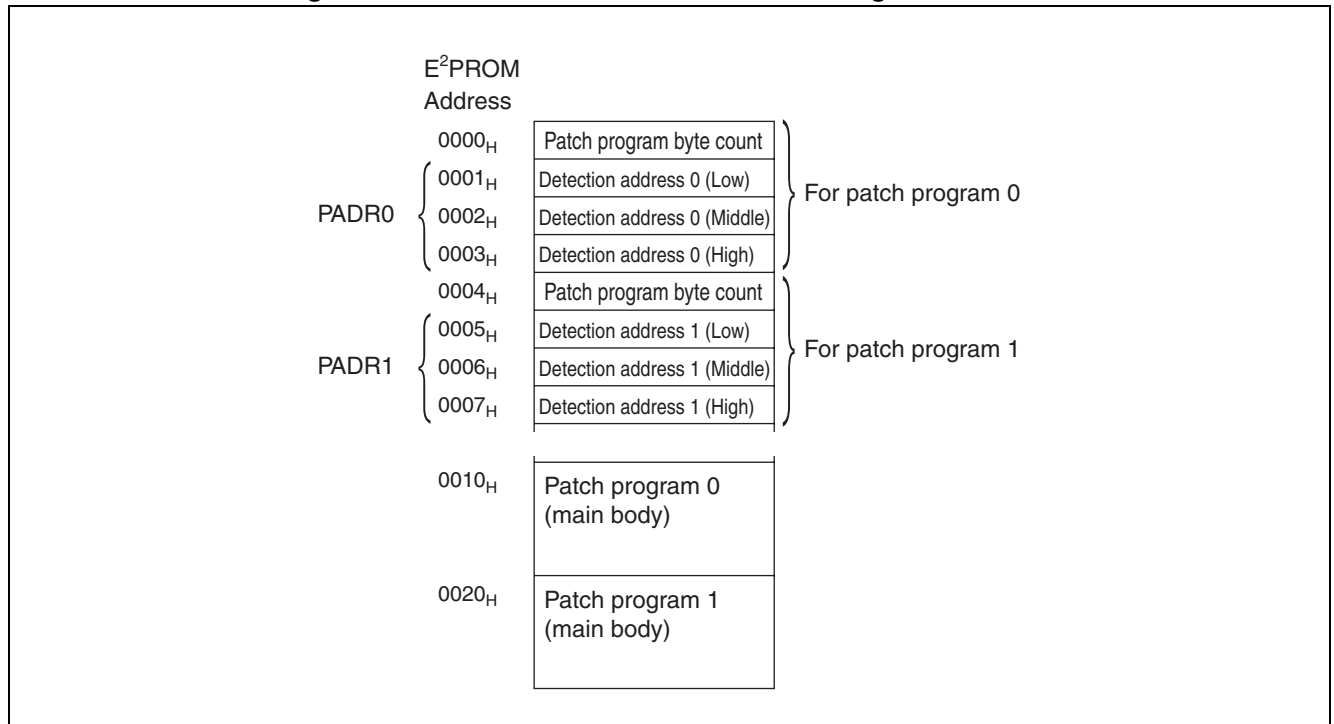
**Figure 21.4-2 Example of System Configuration Using Address Match Detection Function**



## ■ E<sup>2</sup>PROM Memory Map

Figure 21.4-3 shows the allocation of the E<sup>2</sup>PROM patch program data.

**Figure 21.4-3 Allocation of E<sup>2</sup>PROM Patch Program and Data**



### ○ Patch program byte count

The total byte count of the patch program (main body) is stored. If the byte count is "00<sub>H</sub>", it indicates that no patch program is provided.

### ○ Detection address (24 bits)

The address where the instruction code is replaced by the INT9 instruction code due to program error is stored. This address is set in the program address detection registers (PADR0 and PADR1).

### ○ Patch program (main body)

The program executed by the INT9 interrupt processing when the program address matches the detect address is stored. Patch program 0 is allocated from any predetermined address. Patch program 1 is allocated from the address indicating <starting address of patch program 0 + total byte count of patch program 0>.

### ■ Setting and Operating State

#### ○ Initialization

- E<sup>2</sup>PROM data are all cleared to "00<sub>H</sub>".

#### ○ Occurrence of program error

- By using the connector (UART), information about the patch program is transmitted to the MCU (MB90480/485 series) from the outside according to the allocation of the E<sup>2</sup>PROM patch program and data.
- The MCU (MB90480/485 series) stores the information received from outside in the E<sup>2</sup>PROM.

#### ○ Reset sequence

- After reset, the MCU (MB90480/485 series) reads the byte count of the E<sup>2</sup>PROM patch program to check the presence or absence of the correction program.
- If the byte count of the patch program is not "00<sub>H</sub>", the higher, middle and lower bits at detection addresses 0 and 1 are read and set in the program address detection registers 0 and 1 (PADR0 and PADR1). The patch program (main body) is read according to the byte count of the patch program and written to RAM in the MCU (MB90480/485 series).
- The patch program (main body) is allocated to the address where the patch program is executed in the INT9 interrupt processing by the address match detection function.
- Address match detection is enabled (PACSR: AD0E = 1, AD1E = 1).

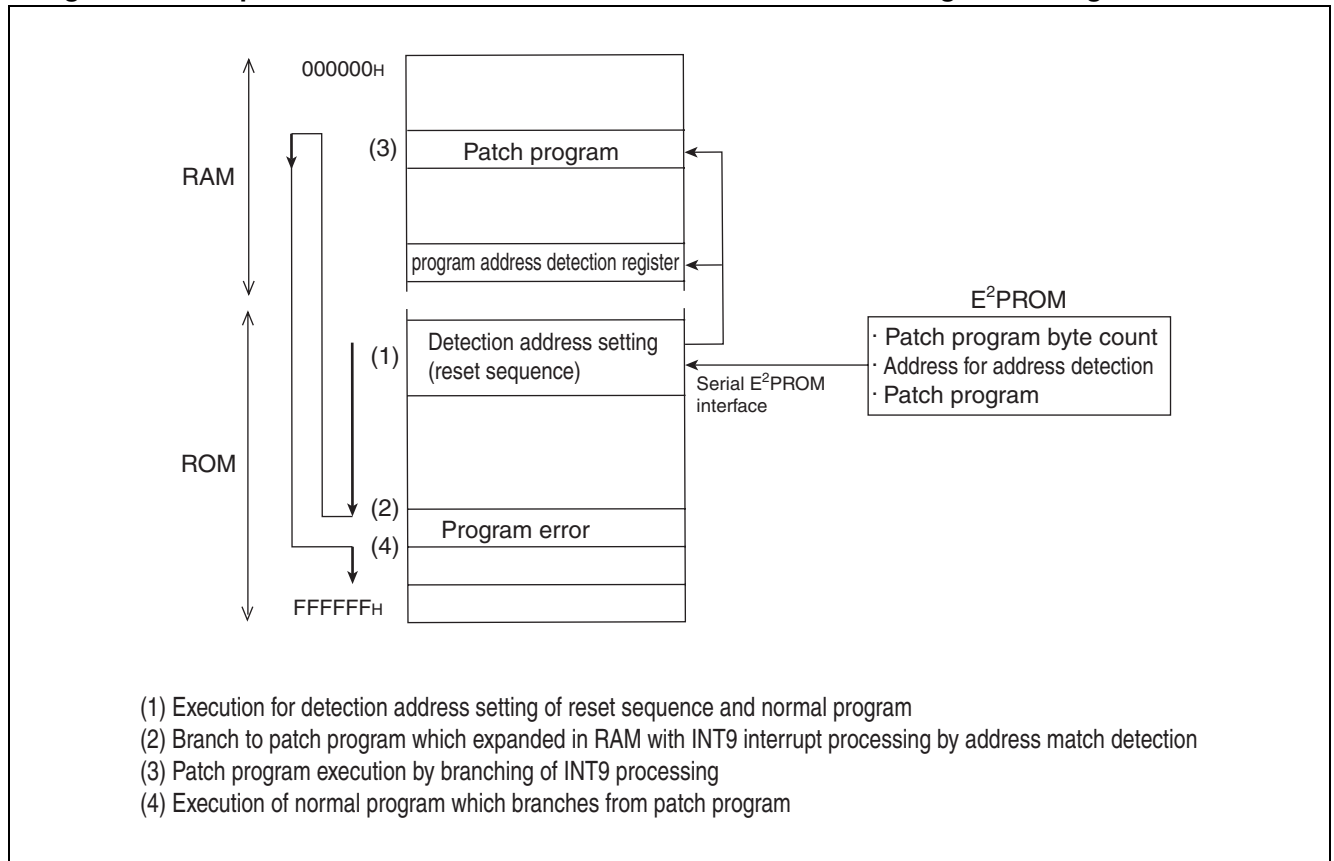
#### ○ INT9 Interrupt processing

- Interrupt processing is performed by the INT9 instruction. The MB90480/485 series has no interrupt request flag by address match detection. Therefore, if the stack information in the program counter is discarded, the detect address cannot be checked. When checking the detect address, check the value of program counter stacked in the interrupt processing routine.
- The patch program is executed, branching to the normal program.

## ■ Operation of Address Match Detection Function at Storing Patch Program in E<sup>2</sup>PROM

Figure 21.4-4 shows the operation of the address match detection function at storing the patch program in E<sup>2</sup>PROM.

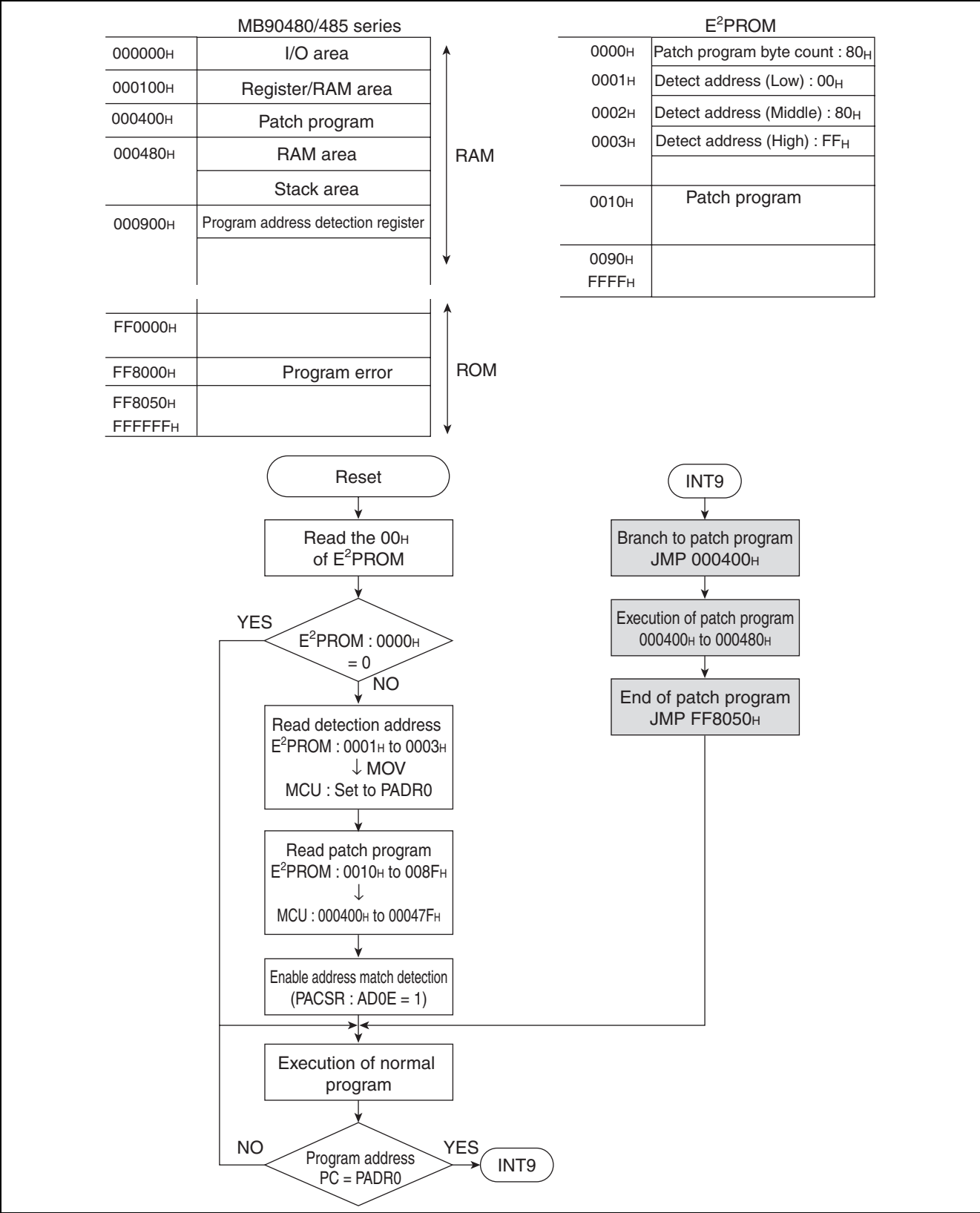
**Figure 21.4-4 Operation of Address Match Detection Function at Storing Patch Program in E<sup>2</sup>PROM**



■ Flow of Patch Processing

Figure 21.4-5 shows the flow of patch processing.

Figure 21.4-5 Flow of Patch Processing



## 21.5 Program Example of Address Match Detection Function

---

This section gives a program example for the address match detection function.

---

### ■ Program Example for Address Match Detection Function

#### ○ Processing specifications

If the address of the instruction to be executed by the program matches the address set in the program address detection register (PADR0), the INT9 instruction is executed.

#### ○ Coding example

```

PACSR    EQU    00009EH          ; Program address detection control status register
PADRL    EQU    000001H          ; Program address detection register 0 (Low)
PADRM    EQU    000002H          ; Program address detection register 0 (Middle)
PADRH    EQU    000003H          ; Program address detection register 0 (High)
;
;-----Main program-----
CODE     CSEG
START:
; Stack pointer (SP), etc.,
; already initialized
MOV     PADRL,#00H               ; Set Program address detection register 0 (Low)
MOV     PADRM,#00H               ; Set Program address detection register 0 (Middle)
MOV     PADRH,#00H               ; Set Program address detection register 0 (High)
;
MOV     I:PACSR,#00000010B      ; Enable address match detection
:
processing by user
:
LOOP:
:
processing by user
:
BAR     LOOP
;-----Interrupt program-----
WARI:
:
processing by user
:
RETI                                ; Return from interrupt processing
CODE     ENDS
;-----Vector setting-----
VECT     CSEG ABS=0FFH
ORG     00FFDCH                  ; Set reset vector
DSL     START
DB     00H                       ; Set to single-chip mode
VECT     ENDS
END     START

```



## CHAPTER 22 ROM MIRROR FUNCTION SELECTION MODULE

---

**This chapter provides an overview of the ROM mirror function selection module and explains its registers.**

---

22.1 Overview of ROM Mirror Function Selection Module

22.2 ROM Mirror Function Selection Register (ROMM)



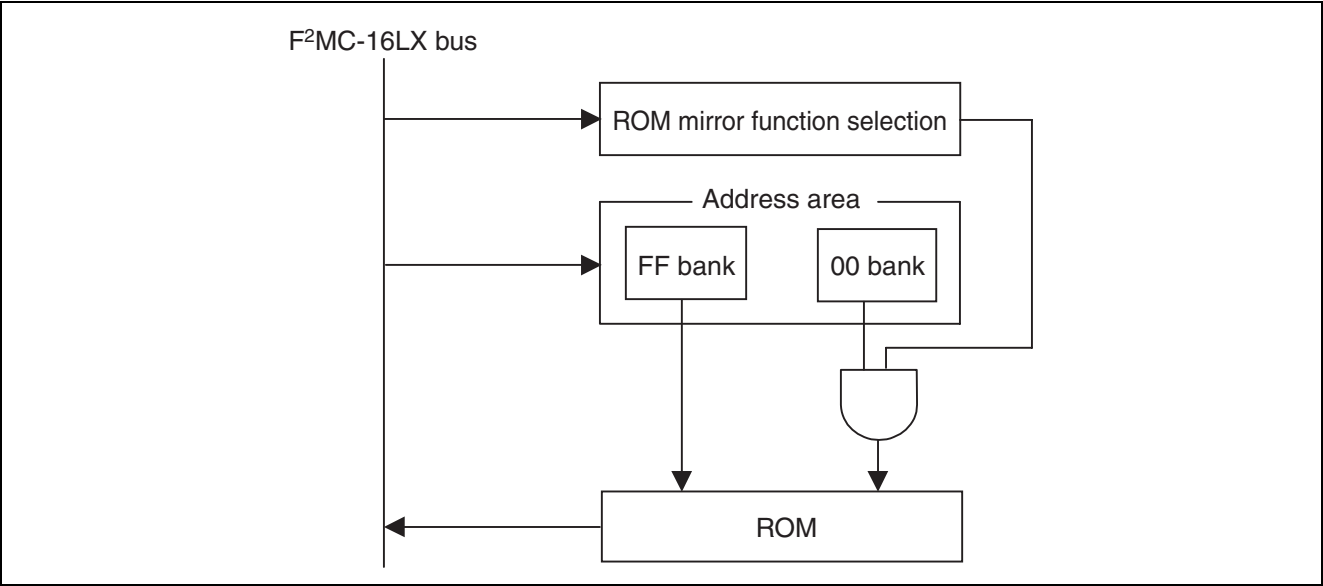
22.1 Overview of ROM Mirror Function Selection Module

The ROM mirror function selection module is set to read the ROM data arranged in FF bank with access to 00 bank.

■ Block Diagram of the ROM Mirror Function Selection Module

Figure 22.1-1 shows a block diagram of the ROM mirror function selection module.

Figure 22.1-1 Block Diagram of the ROM Mirror Function Selection Module



■ Registers of the ROM Mirror Function Selection Module

The following diagram shows configuration of the ROM mirror function selection module.

ROM mirror function selection register (ROMM)

| bit                          | 15                              | 14 | 13 | 12 | 11 | 10 | 9   | 8   | Initial value           |
|------------------------------|---------------------------------|----|----|----|----|----|-----|-----|-------------------------|
| address: 00006F <sub>H</sub> | -                               | -  | -  | -  | -  | -  | MS  | MI  | ----- (+)1 <sub>B</sub> |
|                              |                                 |    |    |    |    |    | R/W | R/W |                         |
|                              |                                 |    |    |    |    |    | (+) |     |                         |
| (+): MB90F489B               | : Read only, fix to "1"         |    |    |    |    |    |     |     |                         |
| Other                        | : Selectable, initial value "0" |    |    |    |    |    |     |     |                         |

## 22.2 ROM Mirror Function Selection Register (ROMM)

This section describes the configuration and functions of the ROM mirror function selection register (ROMM).

### ■ ROMM (ROM Mirror Function Selection Register)

The diagram below shows the bit configuration of the ROM mirror function selection register (ROMM).

|   |    |    |    |    |    |    |     |     |                         |
|---|----|----|----|----|----|----|-----|-----|-------------------------|
| bit                                     | 15 | 14 | 13 | 12 | 11 | 10 | 9   | 8   | Initial value           |
| ROMM address: 00006F <sub>H</sub>       | -  | -  | -  | -  | -  | -  | MS  | MI  | ----- (+)1 <sub>B</sub> |
|   |    |    |    |    |    |    | R/W | R/W |                         |
|   |    |    |    |    |    |    | (+) |     |                         |
| (+) : MB90F489B : Read only, fix to "1" |    |    |    |    |    |    |     |     |                         |
| Other : Selectable, initial value "0"   |    |    |    |    |    |    |     |     |                         |

#### [bit9] MS

This bit selects the ROM mirror area.

- "1": The ROM mirror area is 32 K bytes (008000<sub>H</sub> to 00FFFF<sub>H</sub>).
- "0": The ROM mirror area is 48 K bytes (004000<sub>H</sub> to 00FFFF<sub>H</sub>).

Note:

In the MB90F489B, this bit is fixed to "1", and reading only can be used. Other device can be selected.

#### [bit8] MI

This bit sets whether to enable or disable the ROM mirror function.

- "1": Enable the mirror function.
- "0": Disable the mirror function.

Notes:

- Do not access this register during accesses to address 004000<sub>H</sub> to 00FFFF<sub>H</sub> (008000<sub>H</sub> to 00FFFF<sub>H</sub>).
- If the ROM mirror function is started, addresses FF4000<sub>H</sub> to FFFFFFF<sub>H</sub> (FF8000<sub>H</sub> to FFFFFFF<sub>H</sub>) are mirrored at addresses 004000<sub>H</sub> to 00FFFF<sub>H</sub> (008000<sub>H</sub> to 00FFFF<sub>H</sub>) in Bank 00. ROM addresses of FF3FFF<sub>H</sub> (FF7FFF<sub>H</sub>) or below are not mirrored in Bank 00 even if the ROM mirror function is enabled.  
The value in ( ) is used if the MS bit is set to "1".



# CHAPTER 23 2M/3M BIT FLASH MEMORY

---

**This chapter explains the functions and operations of the 2M/3M bit flash memory.**

---

- 23.1 Overview of 2M/3M Bit Flash Memory
- 23.2 Sector Configuration of 2M/3M Bit Flash Memory
- 23.3 Flash memory Control Status Register (FMCS)
- 23.4 Method for Starting the Flash Memory's Automatic Algorithm
- 23.5 Verifying the Execution State of the Automatic Algorithm
- 23.6 Flash Memory Write/Erase Operations
- 23.7 Flash Security Function

## 23.1 Overview of 2M/3M Bit Flash Memory

In the CPU memory map, the 2M/3M bit flash memory is allocated in banks FC to FF, and the operations for using the flash memory interface circuit, read access and program access from the CPU are provided just as they are for mask ROM. Writing data to flash memory and erasing data from flash memory is executed with instructions from the CPU via the flash memory interface circuit. Thus, the contents of the flash memory can be rewritten in implementation mode under control from built-in CPU, enabling the efficient tuning of programs and data. Selector operations, such as enable sector protect, are not available.

### ■ Features of the 2M/3M Bit Flash Memory

The 2M/3M bit flash memory has the following features:

- 2M: 256K words × 8 bits/128K words × 16 bits (16K + 8K + 8K + 32K + 64K + 64K + 64K) sector configuration
- 3M: 384K words × 8 bits/192K words × 16 bits (16K + 8K + 8K + 32K + 64K + 64K + 64K + 64K + 64K) sector configuration
- Automatic program algorithm (Embedded Algorithm, which is the same as that for the MBM29F400TA)
- Built-in erasure suspend/erasure resume functions
- Detection of completion for write/erase operations using data polling and a toggle bit
- Detection of completion for write/erase operations using CPU interrupts
- Compatibility with JEDEC-standard commands
- Sector-level erasure is available (any combination of sectors is allowed)
- Minimum write/erase count: up to 10000

Embedded Algorithm is a trademark of Advanced Micro Devices, Inc.

### ■ Methods for Writing/Erasing Flash Memory

Flash memory cannot be written and read at the same time. To perform write/erase operation, copy the program on the flash memory to RAM, then the program operates on the RAM data, and finally the result is written back to flash memory. In other words, program execution of flash memory must be performed without read accesses.

### ■ Flash Memory Control Status Register (FMCS)

The following diagram shows the bit configuration of the flash memory control status register (FMCS) used by the flash memory.

| 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) | (W) | (W)      | (R/W) | (W)      | (R/W) |
| Initial value                | (0)   | (0)    | (0)   | (X) | (0)      | (0)   | (0)      | (0)   |

## 23.2 Sector Configuration of 2M/3M Bit Flash Memory

This section describes the sector configuration of the 2M/3M bit flash memory.

### ■ Sector Configuration

Figure 23.2-1 shows the sector configuration of the 2M/3M bit flash memory. The addresses in the figure indicate the upper and lower addresses of each sector.

To access from the CPU, SA0 is allocated in the FC bank register, SA1 is allocated in the FD bank register, SA2 is allocated in the FE bank register, and SA3 to 6 are allocated in the FF bank register.

**Figure 23.2-1 Sector Configuration of 2M/3M Bit Flash Memory**

| 2M bits flash memory |                    |                     | 3M bits flash memory |                    |                     |
|----------------------|--------------------|---------------------|----------------------|--------------------|---------------------|
|                      | Writer address     | CPU address         |                      | Writer address     | CPU address         |
| SA6 (16K bytes)      | 7FFFF <sub>H</sub> | FFFFFF <sub>H</sub> | SA8 (16K bytes)      | 7FFFF <sub>H</sub> | FFFFFF <sub>H</sub> |
|                      | 7C000 <sub>H</sub> | FFC000 <sub>H</sub> |                      | 7C000 <sub>H</sub> | FFC000 <sub>H</sub> |
| SA5 (8K bytes)       | 7BFFF <sub>H</sub> | FFBFFF <sub>H</sub> | SA7 (8K bytes)       | 7BFFF <sub>H</sub> | FFBFFF <sub>H</sub> |
|                      | 7A000 <sub>H</sub> | FFA000 <sub>H</sub> |                      | 7A000 <sub>H</sub> | FFA000 <sub>H</sub> |
| SA4 (8K bytes)       | 79FFF <sub>H</sub> | FF9FFF <sub>H</sub> |                      | 79FFF <sub>H</sub> | FF9FFF <sub>H</sub> |
|                      | 78000 <sub>H</sub> | FF8000 <sub>H</sub> | SA6 (8K bytes)       | 78000 <sub>H</sub> | FF8000 <sub>H</sub> |
| SA3 (32K bytes)      | 77FFF <sub>H</sub> | FF7FFF <sub>H</sub> |                      | 77FFF <sub>H</sub> | FF7FFF <sub>H</sub> |
|                      | 70000 <sub>H</sub> | FF0000 <sub>H</sub> | SA5 (32K bytes)      | 70000 <sub>H</sub> | FF0000 <sub>H</sub> |
| SA2 (64K bytes)      | 6FFFF <sub>H</sub> | FEFFFF <sub>H</sub> |                      | 6FFFF <sub>H</sub> | FEFFFF <sub>H</sub> |
|                      | 60000 <sub>H</sub> | FE0000 <sub>H</sub> | SA4 (64K bytes)      | 60000 <sub>H</sub> | FE0000 <sub>H</sub> |
| SA1 (64K bytes)      | 5FFFF <sub>H</sub> | FDFFFF <sub>H</sub> |                      | 5FFFF <sub>H</sub> | FDFFFF <sub>H</sub> |
|                      | 50000 <sub>H</sub> | FD0000 <sub>H</sub> | SA3 (64K bytes)      | 50000 <sub>H</sub> | FD0000 <sub>H</sub> |
| SA0 (64K bytes)      | 4FFFF <sub>H</sub> | FCFFFF <sub>H</sub> |                      | 4FFFF <sub>H</sub> | FCFFFF <sub>H</sub> |
|                      | 40000 <sub>H</sub> | FC0000 <sub>H</sub> | Unused               | 40000 <sub>H</sub> | FC0000 <sub>H</sub> |
|                      |                    |                     | SA2 (64K bytes)      | 3FFFF <sub>H</sub> | FBFFFF <sub>H</sub> |
|                      |                    |                     |                      | 30000 <sub>H</sub> | FB0000 <sub>H</sub> |
|                      |                    |                     | SA1 (64K bytes)      | 2FFFF <sub>H</sub> | FAFFFF <sub>H</sub> |
|                      |                    |                     |                      | 20000 <sub>H</sub> | FA0000 <sub>H</sub> |
|                      |                    |                     | SA0 (64K bytes)      | 1FFFF <sub>H</sub> | F9FFFF <sub>H</sub> |
|                      |                    |                     |                      | 10000 <sub>H</sub> | F90000 <sub>H</sub> |
|                      |                    |                     | Unused               | 0FFFF <sub>H</sub> | F8FFFF <sub>H</sub> |
|                      |                    |                     |                      | 00000 <sub>H</sub> | F80000 <sub>H</sub> |

#### ○ Writer address

The writer address shown in Figure 23.2-1 is the address corresponding to the CPU address when data is written to the flash memory with a parallel programmer. When using a general-purpose programmer for write/erase operations, this address is used for write/erase operations.

### 23.3 Flash memory Control Status Register (FMCS)

The flash memory control status register (FMCS) is used for write/erase operations on flash memory via the registers in the flash memory interface circuit.

■ Flash Memory Control Status Register (FMCS)

The diagram below shows the bit configuration of the 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) | (W) | (W)      | (R/W) | (W)      | (R/W) |
| Initial value                | (0)   | (0)    | (0)   | (X) | (0)      | (0)   | (0)      | (0)   |

The bits in the flash memory control status register (FMCS) have the following functions.

**[bit7] INTE: INTerrupt Enable**

This bit is used to enable or disable an interrupt request to the CPU due to the end of a flash memory write/erase access.

If the INTE bit is set to "1" and the RDYINT bit is set to "1", an interrupt is issued to the CPU. If the INTE bit is set to "0", no interrupt is issued.

|   |   |
|---|---|
| 0 | Interrupt at the end of write/erase operations prohibited |
| 1 | Interrupt at the end of write/erase operations allowed    |

**[bit6] RDYINT: ReaDY INTerrupt**

This bit indicates the operation state of the flash memory.

At the end of a flash memory write/erase operation, this bit is normally set to "1". When this bit remains "0" after the end of a flash memory write/erase operation, further flash memory write/erase operations are not allowed. Only after this bit has been set to "1" at the end of the write/erase operations is the next write/erase operation for flash memory allowed.

This bit is cleared by writing "0". Writing "1" has no effect. This bit is set to "1" according to the end timing of the flash memory automatic algorithm (Refer to Section "23.4 Method for Starting the Flash Memory's Automatic Algorithm"). Read-modify-write (RMW) instructions always read "1" for this bit.

|   |   |
|---|---|
| 0 | Write/erase operation in progress                           |
| 1 | End of write/erase operation (interrupt request generation) |

**[bit5] WE: Write Enable**

This bit is a write-enable bit for the flash memory area.

If this bit is "1", a command sequence that targets banks FC to FF (refer to Section "23.4 Method for Starting the Flash Memory's Automatic Algorithm") will result in a write operation in the flash memory area. If this bit is "0", no write/erase signals are generated. This bit is used for initiating write/erase commands with respect to the flash memory.

In order to prevent accidental writing of any data to the flash memory, Fujitsu recommends always setting this bit to "0" whenever no write/erase operations are to be executed.

|   |                                     |
|---|-------------------------------------|
| 0 | Flash memory write/erase prohibited |
| 1 | Flash memory write/erase allowed    |

**[bit4] RDY: ReadDY**

This bit is used to allow flash memory write/erase operations to be performed.

If this bit is "0", write/erase operations with respect to the flash memory are not allowed. However, the command of read/reset and sector erase suspend can be accepted in this state.

|   |   |
|---|---|
| 0 | Write/erase operation in progress   |
| 1 | End of write/erase operation (next data write/erase operation is enabled) |

**[bit3] Reserved bit**

This bit is reserved. Always set this bit to "0" for ordinary use.

**[bit1] Reserved bit**

This bit is reserved. Always set this bit to "0" for ordinary use.

**[bit2, bit0] LPM1, LPM0: Low Power Mode**

These bits are used to control flash memory power consumption. If these bits are set to "00", flash memory operations are performed normally. If these bits are set to "01", "10" or "11", however, access to flash memory according to the select signal will be performed in low power consumption mode. In this case, access time increases significantly compared to access with LPM = "00", and memory access will be disabled during high-speed operation of the CPU. To use this mode, use a CPU with a frequency 4 MHz, 8 MHz, or 10 MHz.

| LPM1 | LPM0 | Power consumption mode  |
|------|------|---|
| 0    | 0    | Normal power consumption mode   |
| 0    | 1    | Power saving mode (operation with an internal operation frequency of 4MHz)  |
| 1    | 0    | Power saving mode (operation with an internal operation frequency of 8MHz)  |
| 1    | 1    | Power saving mode (operation with an internal operation frequency of 10MHz) |

---

Note:

For a CPU operation frequency of 10 MHz or higher, always use normal mode.

---

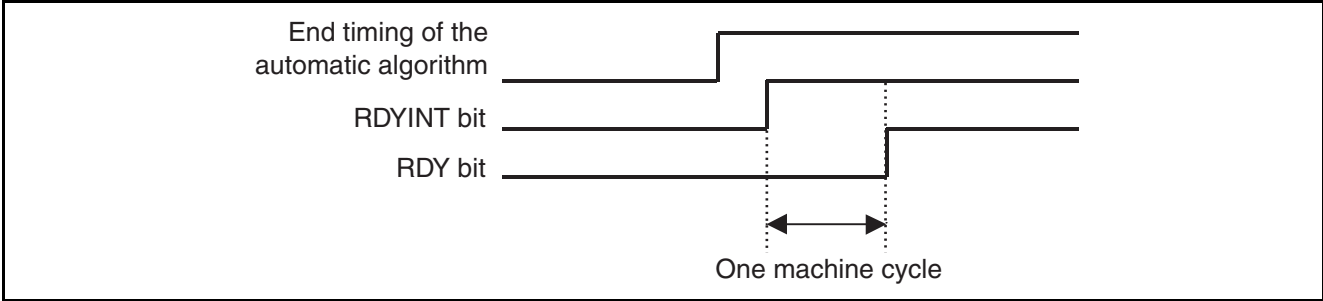


■ End Timing of the Automatic Algorithm

Figure 23.3-1 shows the relationship among the end timing of the automatic algorithm, the RDYINT bit, and the RDY bit.

The RDYINT bit and RDY bit do not change at the same time. Write programs so as to determine the end of automatic algorithm using either of one bit.

Figure 23.3-1 Relationship Among Automatic Algorithm End Timing, RDYINT Bit and RDY Bit



■ Precaution on Use of the Sub Clock Mode

When the sub clock mode is used, the low-power consumption mode selection bits (LPM1, LPM0) of the flash memory control status register (address AE<sub>FF</sub>: FMCS register) must be set to select other than the normal power consumption mode (LPM1, LPM0 = 0, 0), as shown in Table 23.3-1. Therefore, when the device is operating at an internal clock frequency above 10 MHz, the mode cannot be changed to sub clock mode.

Table 23.3-1 Low-power Consumption Mode Selection Bits

| LPM1 | LPM0 | Low-power consumption mode selection bit                                 |                           |
|------|------|--|---------------------------|
|      |      | Low-power consumption mode   | Setting in sub clock mode |
| 0    | 0    | Normal power consumption mode  | × setting prohibited      |
| 0    | 1    | Low-power consumption mode at internal clock frequency of 4 MHz or less  | ○ setting allowed         |
| 1    | 0    | Low-power consumption mode at internal clock frequency of 8 MHz or less  | ○ setting allowed         |
| 1    | 1    | Low-power consumption mode at internal clock frequency of 10 MHz or less | ○ setting allowed         |

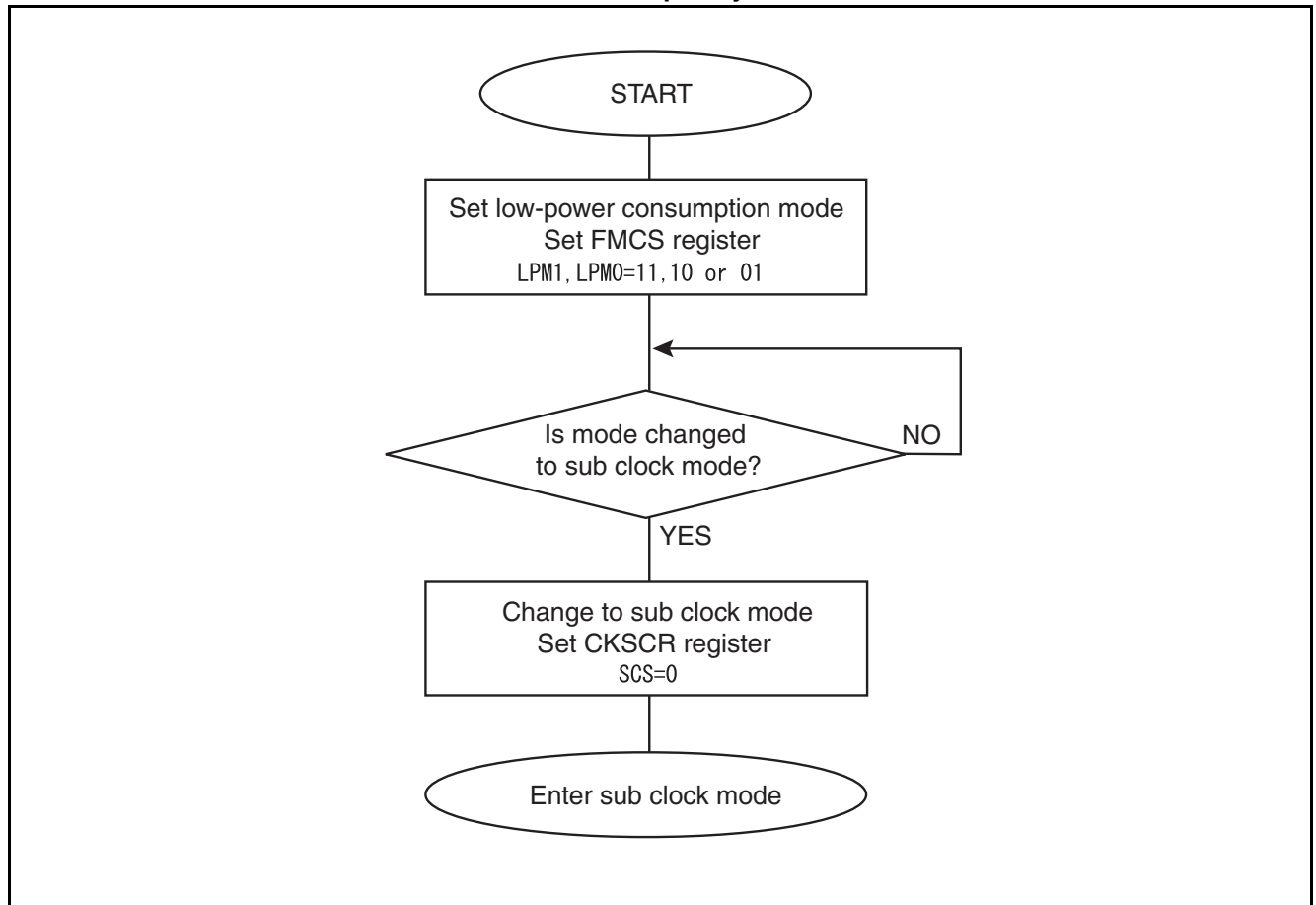
## ■ Setting Sub Clock Mode

If the sub clock mode is used, the following settings are required:

### ○ If sub clock mode is used at an internal clock frequency of 10 MHz or less

To use the sub clock mode, change the low-power consumption mode selection bits (LPM1, LPM0) of the flash memory control status register to other than 0, 0 at initialization. Figure 23.3-2 shows a flowchart example.

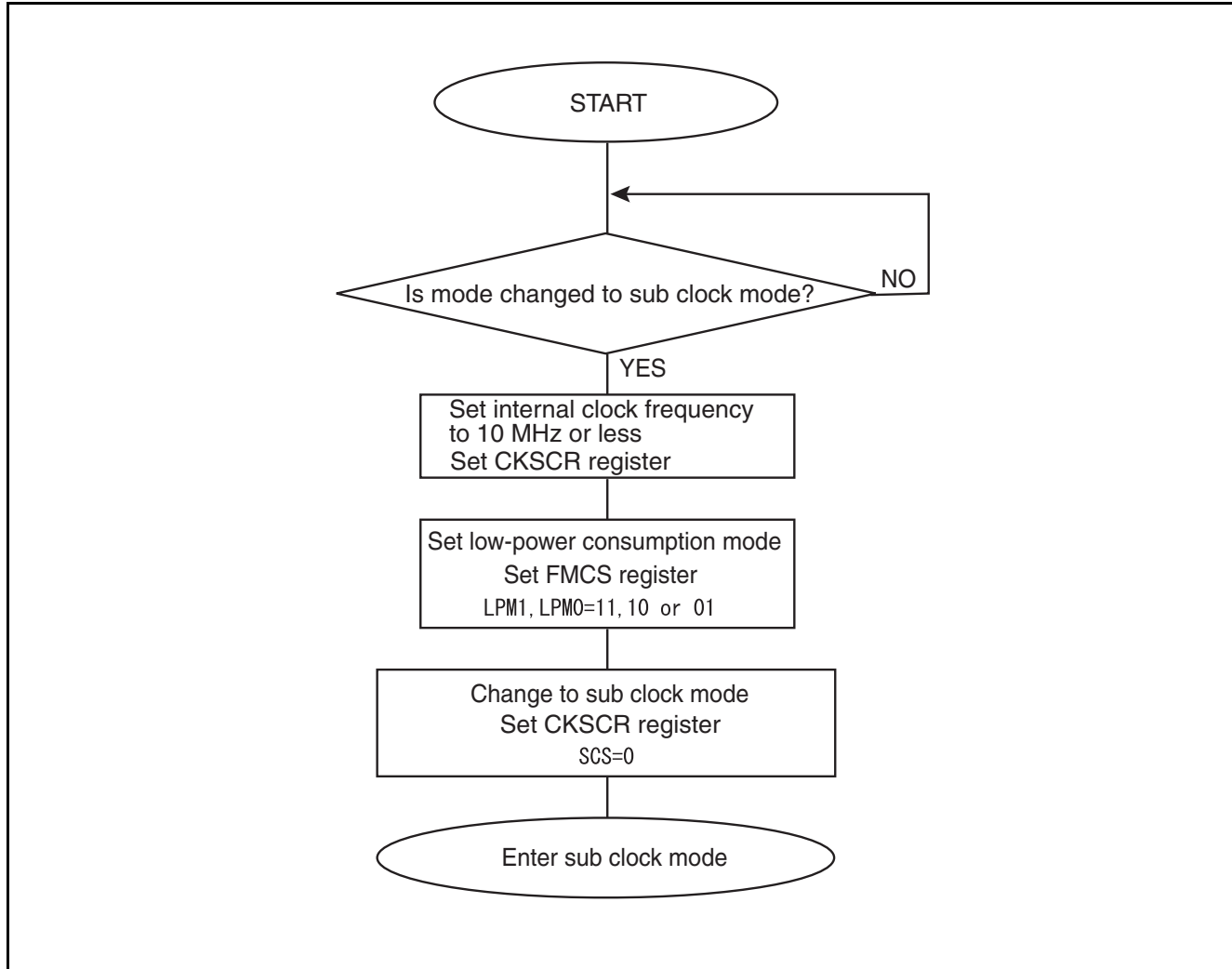
**Figure 23.3-2 Flowchart Example in which Sub Clock Mode is Used at an Internal Clock Frequency of 10 MHz or Less**



○ If sub clock mode is used at an internal clock frequency of more than 10 MHz

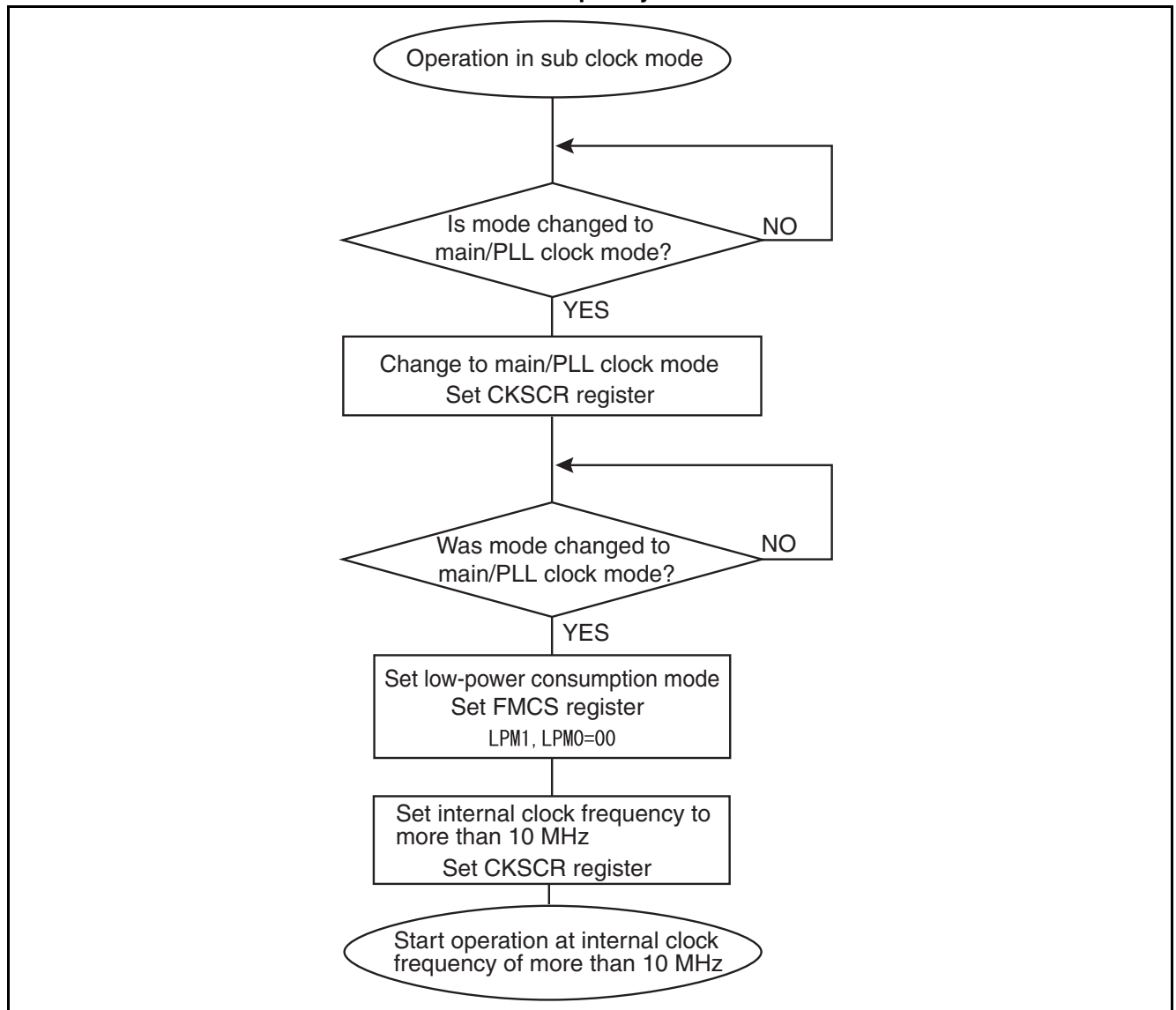
To use sub clock mode, first set the internal clock frequency to 10 MHz or less and next change the low-power consumption mode selection bits (LPM1, LPM0) to other than 0, 0 before changing to sub clock mode. Figure 23.3-3 shows a flowchart example.

**Figure 23.3-3 Flowchart Example in which Sub Clock Mode is Used at an Internal Clock Frequency of More than 10 MHz**



For operation at an internal clock frequency of more than 10 MHz after sub clock mode is canceled, change the mode to main clock mode or PLL clock mode (internal clock frequency becomes 10 MHz or less), change the low-power consumption mode selection bits (LPM1, LPM0) to 0, 0, and then set the internal clock frequency to more than 10 MHz. Figure 23.3-4 shows a flowchart example.

**Figure 23.3-4 Flowchart Example in which Sub Clock Mode is Used at an Internal Clock Frequency of More than 10 MHz**



**Note:**

To use the sub clock mode, input a clock signal of 20 MHz or less to the main clock. If a clock signal of more than 20 MHz is input, the internal clock frequency cannot be set to 10 MHz or less and the sub clock mode cannot be used.

## 23.4 Method for Starting the Flash Memory's Automatic Algorithm

There are four kinds of commands for starting the automatic algorithm for flash memory: read/reset, write, chip erase, and sector erase. For sector erase operations, control of suspension and resuming is provided.

### ■ Command Sequence Table

Table 23.4-1 lists the commands used for flash memory write/erase operations. Although the data for writing to the command register is indicated in units of bytes, use word access during actual operations. The contents of the upper byte are ignored in this case.

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              | Address             | Data              | Address             | Data              | Address             | Data              |
| Read/reset*          | 1               | FxXXXX <sub>H</sub>  | XXF0 <sub>H</sub> | -                   | -                 | -                   | -                 | -                   | -                 | -                   | -                 | -                   | -                 |
| Read/reset*          | 4               | FxAAAA <sub>H</sub>  | XXAA <sub>H</sub> | Fx5554 <sub>H</sub> | XX55 <sub>H</sub> | FxAAAA <sub>H</sub> | XXF0 <sub>H</sub> | RA                  | RD                | -                   | -                 | -                   | -                 |
| Write program        | 4               | FxAAAA <sub>H</sub>  | XXAA <sub>H</sub> | Fx5554 <sub>H</sub> | XX55 <sub>H</sub> | FxAAAA <sub>H</sub> | XXA0 <sub>H</sub> | PA (even)           | PD (word)         | -                   | -                 | -                   | -                 |
| Chip erase           | 6               | FxAAAA <sub>H</sub>  | XXAA <sub>H</sub> | Fx5554 <sub>H</sub> | XX55 <sub>H</sub> | FxAAAA <sub>H</sub> | XX80 <sub>H</sub> | FxAAAA <sub>H</sub> | XXAA <sub>H</sub> | Fx5554 <sub>H</sub> | XX55 <sub>H</sub> | FxAAAA <sub>H</sub> | XX10 <sub>H</sub> |
| Sector erase         | 6               | FxAAAA <sub>H</sub>  | XXAA <sub>H</sub> | Fx5554 <sub>H</sub> | XX55 <sub>H</sub> | FxAAAA <sub>H</sub> | XX80 <sub>H</sub> | FxAAAA <sub>H</sub> | XXAA <sub>H</sub> | Fx5554 <sub>H</sub> | XX55 <sub>H</sub> | SA (even)           | XX30 <sub>H</sub> |
| Sector erase suspend |                 | Entering data (xxB0 <sub>H</sub> ) at address "FxXXXX <sub>H</sub> " will suspend a erasure in sector erase mode.      |                   |                     |                   |                     |                   |                     |                   |                     |                   |                     |                   |
| Sector erase resume  |                 | Entering data (xx30 <sub>H</sub> ) at address "FxXXXX <sub>H</sub> " will resume erasure in sector erase suspend mode. |                   |                     |                   |                     |                   |                     |                   |                     |                   |                     |                   |

RA: Read address

PA: Write address. Only even addresses can be specified.

SA: Sector address (Refer to Section "23.2 Sector Configuration of 2M/3M Bit Flash Memory")

RD: Read data

PD: Write data; only words can be specified

\* : Both read and reset commands allow the flash memory to be reset to read mode.

#### Notes:

- The address Fx in the table represents FF, FE, FD or FC. Specify the actual value corresponding to the bank to be accessed in each operation.
- The address in the table indicates the value in the CPU memory map. Addresses and data are indicated in hexadecimal representation; "X" indicates an arbitrary value.

## 23.5 Verifying the Execution State of the Automatic Algorithm

The flash memory contains dedicated hardware indicating the internal operation state of the flash memory and whether operations have been completed that can be used to control the operational flow of write/erase operations via the automatic algorithm. The operation state of built-in flash memory can verify using the hardware sequence flag.

### ■ Hardware Sequence Flags

The hardware sequence flags consist of the four bits DQ7, DQ6, DQ5, and DQ3. These bits have the following functions: DQ7 is the data polling flag, DQ6 is the toggle bit flag, DQ5 is the timing limit excess flag, and DQ3 is the sector erase timer flag. The hardware sequence flags is therefore used to confirm that writing or chip sector erase has been completed or that erase code write is valid.

Table 23.5-1 shows the bit assignments of the hardware sequence flags.

**Table 23.5-1 Bit Assignments of Hardware Sequence Flags**

| Bit number             | 7   | 6   | 5   | 4 | 3   | 2 | 1 | 0 |
|------------------------|-----|-----|-----|---|-----|---|---|---|
| Hardware sequence flag | DQ7 | DQ6 | DQ5 | - | DQ3 | - | - | - |

To refer to the hardware sequence flag, read the address of the sector for the internal flash memory after the corresponding command sequence has been set (refer to Table 23.4-1).

The execution state of automatic algorithm can be checked in the following methods.

- Check by referring to hardware sequence flag
- Check by referring to RDY bit of flash memory control status register (FMCS)

When creating actual program, perform the subsequent instruction after completion of executing automatic algorithm is checked using the above verification method. The hardware sequence flags are described below.

## CHAPTER 23 2M/3M BIT FLASH MEMORY

Table 23.5-2 lists the hardware sequence flag functions.

**Table 23.5-2 List of Hardware Sequence Flag Functions**

| State                                 |  | DQ7           | DQ6              | DQ5         | DQ3         |
|---------------------------------------|--|---------------|------------------|-------------|-------------|
| State change in normal operation mode | Write operation → write completed (specifying the write address) | DQ7 → DATA: 7 | Toggle → DATA: 6 | 0 → DATA: 5 | 0 → DATA: 3 |
|                                       | Chip sector erase operation → erase completed                    | 0 → 1         | Toggle → Stop    | 0 → 1       | 1           |
|                                       | Sector erase wait → erase start                                  | 0             | Toggle           | 0           | 0 → 1       |
|                                       | Erase operation → sector erase suspend (sector being erased)     | 0 → 1         | Toggle → 1       | 0           | 1 → 0       |
|                                       | Sector erase suspend → erase resume (sector being erased)        | 1 → 0         | 1 → Toggle       | 0           | 0 → 1       |
|                                       | Sector erase suspend mode (sector not being erased)              | DATA: 7       | DATA: 6          | DATA: 5     | DATA: 3     |
| Operational error                     | During write operation   | DQ7           | Toggle           | 1           | 0           |
|                                       | During chip sector erase operation                               | 0             | Toggle           | 1           | 1           |

## 23.5.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) is a flag that is used to indicate via the data polling function whether execution of the automatic algorithm is in progress or has ended.

### ■ State Transitions of the Data Polling Flag (DQ7)

#### ○ State changes during normal operation

| Operation state | Write operation → completed                        | Chip sector erase → completed | Sector erase wait → start | Sector erase → erase suspend (sector being erased) | Sector erase suspend → resume (sector being erased) | Sector erased Suspend mode Sector not being erased |
|-----------------|--|-------------------------------|---------------------------|--|---|--|
| DQ7             | $\overline{\text{DQ7}} \rightarrow \text{DATA: 7}$ | $0 \rightarrow 1$             | 0                         | $0 \rightarrow 1$                                  | $1 \rightarrow 0$                                   | DATA: 7  |

#### ○ State changes in operation error mode

| Operation state | Write operation         | Chip sector erase operation |
|-----------------|-------------------------|-----------------------------|
| DQ7             | $\overline{\text{DQ7}}$ | 0                           |

### ■ Write Operation

During read accesses while the automatic write algorithm is executed, the reverse data for bit7 of the last data item written to flash memory is referred and output, regardless of the specified address. If the read operation is performed at the time when the automatic write algorithm has ended, the read value of bit7 for the specified address in flash memory is output.

### ■ Chip/Sector Erase Operation

When the chip erase/sector erase algorithm is being executed, "0" is output in read operations of the flash memory, either for the sector currently erased in sector erase mode or independently of addressing in chip erase mode. "1" is output when the chip erase/sector erase algorithm has ended.



### ■ Sector Erase Suspend

Read operations of the flash memory while sector erase is suspended will output "1" if an address for the sector being erased has been specified, or will output the read value of bit7 (DATA: 7) for the specified data item will be output in other cases. Which sector is being erased, and whether that sector is in sector suspend state, can be identified by referring to the toggle bit flag (DQ6).

---

#### Note:

If the automatic algorithm starts, read accesses to the specified address are not effective. In data read operations, the other bits can be output provided the end of data polling flag (DQ7) is set. Thus, read data after the end of the automatic algorithm only after the read access that confirms the end of data polling.

---

## 23.5.2 Toggle Bit Flag (DQ6)

Like the data polling flag (DQ7), the toggle bit flag (DQ6) is a flag mainly used to indicate whether the automatic algorithm is being executed or has ended. In the case of the toggle bit flag, a toggle bit function is used for that purpose.

### ■ State Transitions of the Toggle Bit Flag (DQ6)

#### ○ State transitions in normal operation

| Operation state | Write operation → completed | Chip sector erase → completed | Sector erase wait → start | Sector erase → erase suspend (sector being erased) | Sector erase suspend → resume (sector being erased) | Sector erased Suspend mode Sector not being erased |
|-----------------|-----------------------------|-------------------------------|---------------------------|--|---|--|
| DQ6             | Toggle → DATA: 6            | Toggle → Stop                 | Toggle                    | Toggle → 1   | 1 → Toggle  | DATA: 6  |

#### ○ State transitions in operation error mode

| Operation state | Write operation | Chip sector erase operation |
|-----------------|-----------------|-----------------------------|
| DQ6             | Toggle          | Toggle                      |

### ■ Write/Chip Sector Erase Operations

In the case of repeated read accesses when the automatic write algorithm and the chip sector erase algorithm are being executed, the flash memory toggles between output of 1 and 0 in toggle mode for each read operation independent of addressing. When repeated read accesses are performed when the automatic write algorithm and chip/sector erase algorithm have ended, the flash memory stops with the toggle operation of bit6, and instead outputs bit6 (DATA: 6) of the read value for the specified address.

### ■ Sector Erase Suspend

If read access is performed when a sector erase operation is suspended, the flash memory outputs "1" when the specified address belongs to the sector being erased. In other cases, bit6 (DATA: 6) of the read value for the specified address will be output.

#### Reference:

If, in write operations, the sector to be written is rewrite-protected, toggle operation is performed for about 2  $\mu$ s before the operation ends without any data being rewritten.

If, in erase operation, all selected sectors are rewrite-protected, toggle operation is performed for about 100  $\mu$ s before the device returns to the read/reset state without any data being rewritten.

### 23.5.3 Timing Limit Excess Flag (DQ5)

The timing limit excess flag (DQ5) is used to indicate when the execution of the automatic algorithm exceeds the time (internal pulse count) specified in the internal flash memory.

#### ■ State Transitions of the Timing Limit Excess Flag (DQ5)

##### ○ State transitions in normal operation

| Operation state | Write operation → completed | Chip sector erase → completed | Sector erase wait → start | Sector erase → erase suspend (sector being erased) | Sector erase suspend → resume (sector being erased) | Sector erased Suspend mode Sector not being erased |
|-----------------|-----------------------------|-------------------------------|---------------------------|--|---|--|
| DQ5             | 0 → DATA: 5                 | 0 → 1                         | 0                         | 0  | 0   | DATA: 5  |

##### ○ State transitions in operation error mode

| Operation state | Write operation | Chip sector erase operation |
|-----------------|-----------------|-----------------------------|
| DQ5             | 1               | 1                           |

#### ■ Write/Chip Sector Erase Operation

When a read access is performed after a writing operation occurs or the chip sector erase automatic algorithm starts, "0" is output if the specified time (for write/erase operations) is not exceeded, while "1" is output if this time is exceeded. This operation is independent of whether the automatic algorithm is being executed or has ended, thus allowing whether a failure has occurred during a write/erase operation to be determined. If this flag indicates "1" while the automatic algorithm is still being executed according to the data polling function or toggle bit function, it can be assumed that a failure during a write operation has occurred.

For example, if there is an attempt to write "1" to a flash memory address whose corresponding value has already been set to "0", a failure occurs. In this case, the flash memory will be locked, and the automatic algorithm will not end. Occasionally, it is likely to end normally as be able to be written "1". As a result, the data polling flag (DQ7) will output that there is no valid data. The toggle bit flag (DQ6) will continue with the toggle operation until the time limit is exceeded, and the timing limit excess flag (DQ5) will output "1". This indicates that the flash memory is not defective, but was used incorrectly. If this state occurs, execute the reset command.

## 23.5.4 Sector Erase Timer Flag (DQ3)

The sector erase timer flag (DQ3) indicates whether the device is waiting for the end of sector erase after starting of the sector erase command.

### ■ State Transitions of Sector Erase Timer Flag (DQ3)

#### ○ State changes during normal operation

| Operation state | Write operation → completed | Chip sector erase → completed | Sector erase wait → start | Sector erase → erase suspend (sector being erased) | Sector erase suspend → resume (sector being erased) | Sector erased Suspend mode Sector not being erased |
|-----------------|-----------------------------|-------------------------------|---------------------------|--|---|--|
| DQ3             | 0 → DATA: 3                 | 1                             | 0 → 1                     | 1 → 0  | 0 → 1   | DATA: 3  |

#### ○ State changes in operation error mode

| Operation state | Write operation | Chip sector erase operation |
|-----------------|-----------------|-----------------------------|
| DQ3             | 0               | 1                           |

### ■ Sector Erase Operation

In read accesses after the sector erase command starts, the flash memory outputs "0" if the sector erase waiting time is in progress, irrespective of the address specified by the address signal for the sector that issued the command, or it outputs "1" if the sector erase waiting time was exceeded.

If the data polling function or toggle bit function shows that the erase algorithm is being executed and this flag is set to "1", erasure under internal control will start. Subsequent commands other than writing the sector erase code or erase suspend are ignored until the erasure is completed.

If this flag is set to "0", the flash memory will accept writing of the additional sector erase code. Fujitsu recommends checking the state of the flag before subsequent sector erase codes are written to verify the operational state of the device. If a second state check returns the flag to "1", the erase code for an additional sector may not have been accepted.

### ■ Sector Erase Suspend

If read access is performed when a sector erase operation is suspended, the flash memory outputs "1" for an address of a sector being erased. Otherwise, the read value of bit3 (DATA: 3) for the corresponding address will be output.

## 23.6 Flash Memory Write/Erase Operations

---

**This section describes various operation procedures after issuing the automatic algorithm start command, including flash memory read/reset, write, chip erase, sector erase, sector erase suspend and sector erase resume.**

---

### ■ Flash Memory Write/Erase

The automatic algorithm can be started by writing one of read/reset, write, chip erase, sector erase, sector erase suspend, and erase resume in the command sequence (see Table 23.4-1) from CPU to the flash memory. Write cycle from CPU to the flash memory must be executed consecutively. The end of the automatic algorithm can be detected with such functions as the data polling function. After a normal end, the operational state returns to the read/reset state.

This section describes the following items related to flash memory write/erase operations.

- Setting the flash memory to read/reset state
- Writing data
- Erasing all data (entire chip erase)
- Erasing arbitrary data (sector erase)
- Suspending sector erase
- Resuming sector erase
- Flash memory performs a bus write cycle according to for operations such as

## 23.6.1 Setting the Flash Memory to Read/Reset State

---

**This section describes the procedures for issuing read/reset commands and setting the flash memory to read/reset state.**

---

### ■ Setting the Flash Memory to the Read/Reset State

To set the flash memory to the read/reset state, continuously send the read/reset command in the command sequence table (see Table 23.4-1) to the relevant sector in the flash memory.

Read/reset commands use two types of command sequences: execution in one bus operation and execution in three bus operations. There are no basic differences in between command sequences.

The read/reset state is the initial state of the flash memory. It occurs at power-on or at normal end of a command. The read/reset state is a state in which the device waits for input of other commands.

The read/reset state enables data to be read with normal read accesses. Program access from the CPU is performed in the same way that it is for mask ROM. However, this command does not require that a data read operation ends normally: This command is mainly used to initialize the automatic algorithm if a command does not end for some reason.

## 23.6.2 Writing Data to Flash Memory

---

This section describes the procedures for issuing a write command to write data to the flash memory.

---

### ■ Writing Data to Flash Memory

To start the automatic data write algorithm for the flash memory, repeatedly send the write command in the command sequence table (see Table 23.4-1) to the relevant sector in the flash memory. When data writing to the target address is completed in the 4th cycle, the automatic algorithm, and therefore automatic writing, will be started.

#### ○ Address specification method

Only an even write address can be specified in the write data cycle. Specifying an odd address will cause a failure during writing. It is necessary to write to even addresses in word units.

Any order of addresses or even addresses exceeding the sector boundary are acceptable in write operations. However, a single write command can only write one word of data.

#### ○ Notes on writing data

Data polling flag (DQ7) or toggle bit flag (DQ6) doesn't enter the state of the end if data in the flash memory is written from "0" in "1". So, it is judged that the flash memory element is defective, and falls into the following states. Therefore, please do not return data in the flash memory from "0" to "1" by write.

- Time limit over flag (DQ5) becomes the error state by exceeding the write regulation time.
- In the appearance, it becomes the state that "1" seems to be written in data on the flash memory. (Data is "0" when read the data at read/reset state. Please erase it when you return data from "0" to "1".)

During execution of automatic write, all other commands are ignored. Note that if a hardware reset starts while a write operation is in progress, the data that is written to the address is not assured.

## ■ Operation for Writing to Flash Memory

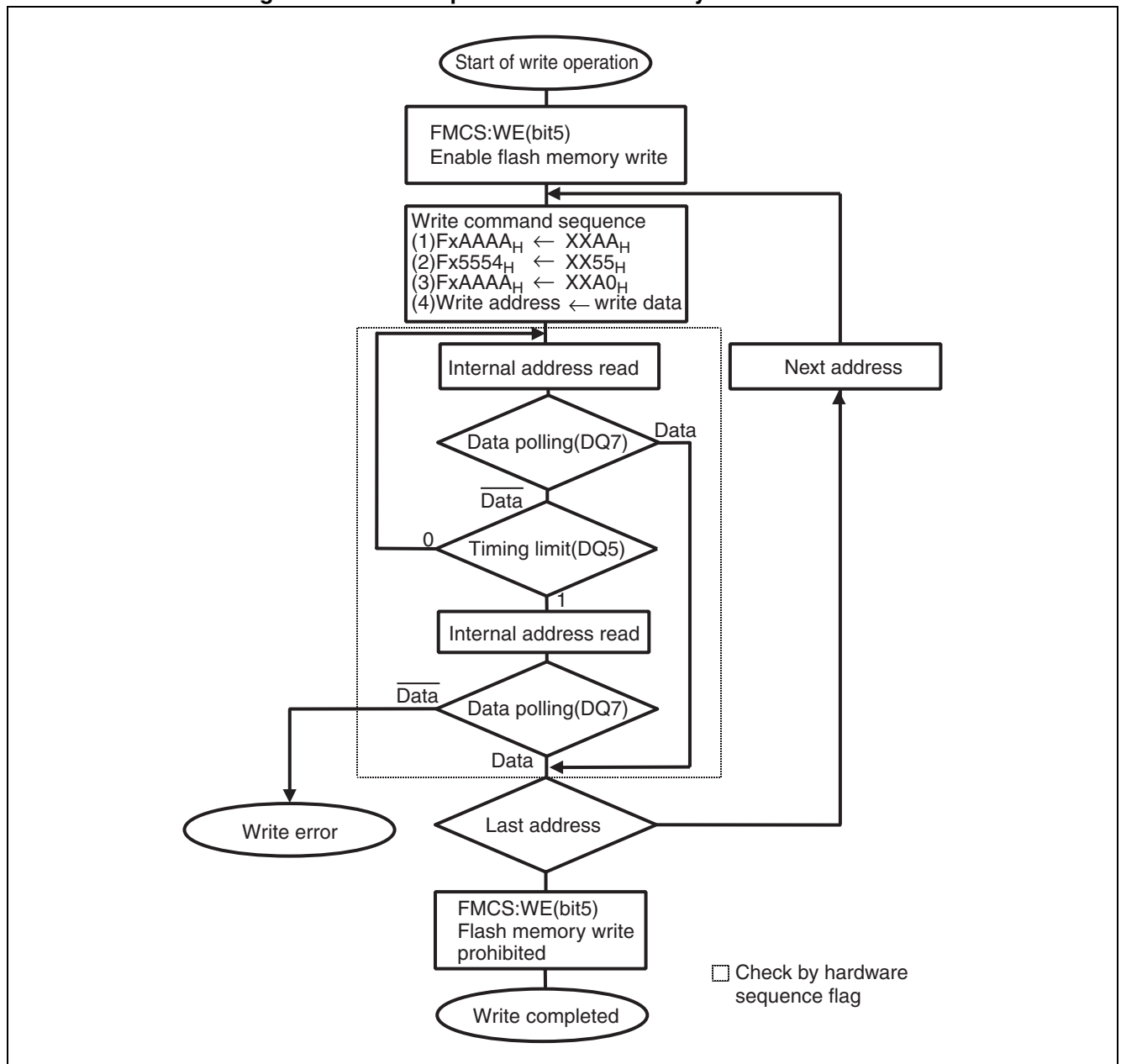
Figure 23.6-1 shows an example of the procedure for writing to flash memory. Using the hardware sequence flag (see Section "23.5 Verifying the Execution State of the Automatic Algorithm"), the operational state of the automatic algorithm operating on the flash memory can be determined. In this example, the data polling flag (DQ7) is used to indicate the end of writing.

During the flag check, data is read from the last address data was written to.

The data polling flag (DQ7) changes at the same time as the timing limit excess flag (DQ5). Even if the timing limit excess flag (DQ5) is "1", the data polling flag bit (DQ7) must be checked again.

Since the toggle bit flag (DQ6) also stops the toggle operation when the timing limit excess flag bit (DQ5) is set to "1", the toggle bit flag (DQ6) must be checked again in this case.

### Figure 23.6-1 Example of the Flash Memory Write Procedure





### 23.6.3 Erasing All Data in the Flash Memory (Chip Erase)

---

This section describes the procedure for issuing the chip erase command to erase all data in the flash memory.

---

#### ■ Erasing All Data in the Flash Memory (Chip Erase)

To erase all data in the flash memory, repeatedly send the chip erase command in the command sequence table (see Table 23.4-1) to the relevant sector in the flash memory.

The chip erase command is executed in six bus operations. When the write operation is completed in the 6th cycle, the chip erase operation will start. During the chip erase operation, the user does not need to write to the flash memory before erasing; during execution of the automatic erase algorithm, "0" will automatically be written to all cells of the flash memory for verification before erasure.

## 23.6.4 Erasing Arbitrary Data in Flash Memory (Sector Erase)

---

This section describes the procedure for issuing a sector erase command to erase an arbitrary sector in flash memory. This procedure allows either erasure of individual sectors or erasure of multiple sectors at the same time to be specified.

---

### ■ Erasing Arbitrary Data in the Flash Memory (Sector Erase)

To erase an arbitrary sector in flash memory, repeatedly send the sector erase command in the command sequence table (see Table 23.4-1) to the relevant sector in the flash memory.

#### ○ Method for specifying a sector

The sector erase command is performed in six bus operations. To start a sector erase wait of 50 $\mu$ s, write the sector erase code (30<sub>H</sub>) in the 6th cycle to an arbitrary even address in the target sector that can be accessed. To erase multiple sectors using the above procedure, write the erase code (30<sub>H</sub>) sequentially to the addresses of the target sector to be erased.

#### ○ Notes on specifying multiple sectors

Erase operation starts when the sector erase wait time of 50  $\mu$ s has elapsed after the last sector erase code has been written. In other words, to erase multiple sectors at the same time, enter the next erase sector address and erase code (which must be entered in the 6th cycle of the command sequence) within 50  $\mu$ s. After this time limit is exceeded, the sector address or erase code may not be accepted. Whether the next sector erase code can be written can be checked using the sector erase timer (hardware sequence flag:DQ3). In this case, the address for reading the sector erase timer must also specify the sector to be erased.

### ■ Procedure for Sector Erasure

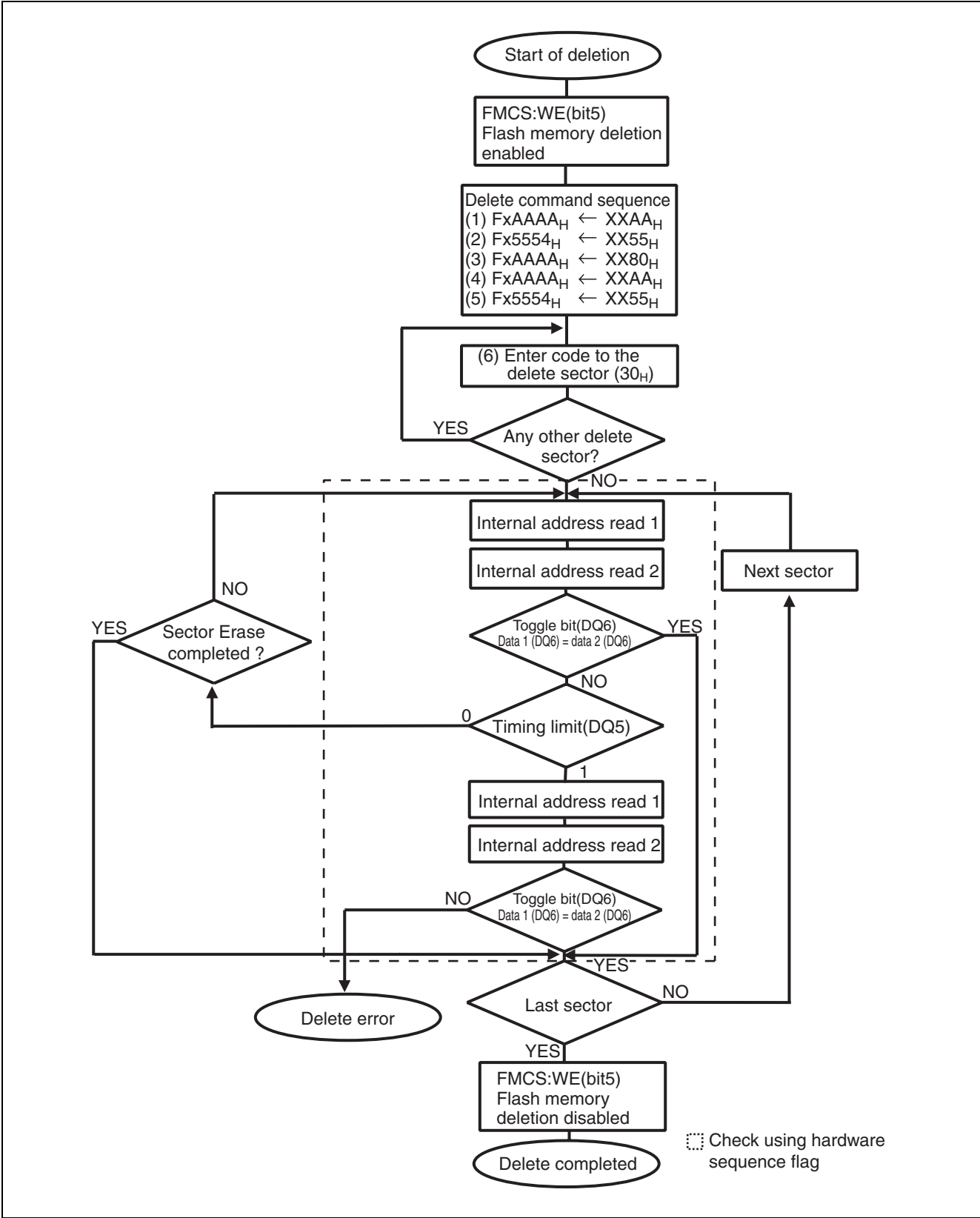
The hardware sequence flag (see Section "23.5 Verifying the Execution State of the Automatic Algorithm") can be used to identify the operational state of the automatic algorithm operating on the internal flash memory. Figure 23.6-2 shows an example of the procedure for sector erasure of the flash memory. In this example, the toggle bit flag (DQ6) is used to check for the end of erasure.

Be sure that the data read in the flag check is read from the sector to be erased.

The toggle bit flag (DQ6) stops the toggle operation when the timing limit excess flag (DQ5) changes to "1". Therefore, even if DQ5 is "1", check the toggle bit flag (DQ6) again (processing in the dotted line in Figure 23.6-2).

The data polling flag (DQ7) also changes when the timing limit excess flag (DQ5) changes. Therefore, check the data polling flag (DQ7) again.

Figure 23.6-2 Example of Sector Erase Procedure for Flash Memory



## 23.6.5 Suspending Sector Erasure for the Flash Memory

---

**This section describes the procedure for issuing the sector erase suspend command to suspend a sector erase operation for the flash memory. During erase suspension, data can be read from any sector that is not subject to erasure.**

---

### ■ Suspending Sector Erasure for the Flash Memory

To suspend sector erasure for the flash memory, send the sector erase suspend command in the command sequence table (see Table 23.4-1) from CPU to the internal flash memory.

The sector erase suspend command is used to suspend the erasure during a sector erase operation, allowing data from a sector that is not being erased to be read. In this state, only reading is allowed; writing is prohibited. This command is enabled only in sector erase mode, including within the erase wait time, and ignored in chip erase mode or during write operations.

This operation is executed by writing the erase suspend code (B0<sub>H</sub>). To do so, specify an arbitrary address in flash memory. During the erase suspend state, repeatedly issued erase suspend commands are ignored.

If a sector erase suspend command is entered during the sector erase wait time, sector erase wait ends immediately, the erase operation is interrupted, and the operational state changes to erase stop.

If a erase suspend command is entered during a sector erase operation after the sector erase wait time, the system enters the erase suspend mode after 20  $\mu$ s have elapsed or earlier. Please execute the sector erase stop command after sector erase command or sector erase resume command issuing and 20  $\mu$ s or more.

## 23.6.6 Resuming the Sector Erasure of Flash Memory

---

**This section describes the procedure for issuing the sector erase resume command and resuming a suspended flash memory sector erase operation.**

---

### ■ Resuming the Sector Erasure of Flash Memory

To resume a suspended sector erase operation, send the sector erase resume command in the command sequence table (see Table 23.4-1) to the internal flash memory.

The sector erase resume command is used to resume a sector erasure from the sector erase suspend mode caused by a sector erase suspend command. This command is executed by writing the erase restart code (30<sub>H</sub>) while specifying an arbitrary address in the flash memory area.

Issuing the sector erase restart command during a sector erase operation will be ignored.

## 23.7 Flash Security Function

---

The flash security function can preserve the contents of flash memory.

---

### ■ Overview

Writing the protection code, 01<sub>H</sub>, to the security bit of flash memory can limit access to flash memory. If flash memory is protected once, it is impossible to release the protected condition until the chip deletion is completed. Unless the protected condition is released, it is impossible to read/write data of flash memory from an external pin. This function is suitable for the application which requires security of the self-conclusion-type program and data stored in flash memory.

The address of a security bit depends on the size of installed flash memory. Table 23.7-1 shows a list of the flash security bit address.

**Table 23.7-1 Flash Security Bit Address**

|           | Flash memory size             | Security bit address |
|-----------|-------------------------------|----------------------|
| MB90F488B | built-in 2M bits flash memory | FC0001 <sub>H</sub>  |
| MB90F489B | built-in 3M bits flash memory | F90001 <sub>H</sub>  |

### ■ Setting of Security

The security function is set after writing the protection code "01<sub>H</sub>" to a security bit and resumption of the external reset or the power supply.

### ■ Cancel of Security

Execution of the chip deletion

### ■ Operation in Security Permission

- Read : The invalid data is read out from the external pin.
- Write : Unable to write.

### ■ the Others

- These setting for the general-purpose parallel writer should depend on the specification of the using parallel writer.
- It is recommended to write the protection code at the end of flash programming.  
The purpose is to prevent from being protected unintentionally during programming.

---

Note:

A security bit is located within an area of flash memory. Writing the protection code, "01<sub>H</sub>", to a security bit guarantees security. Therefore, do not write "01<sub>H</sub>" to this address when not using the security function.

For the address of a security bit, see Table 23.7-1 mentioned above.

---



## **CHAPTER 24    EXAMPLES OF MB90F481B/MB90F482B/ MB90F488B/MB90F489B SERIAL PROGRAMMING CONNECTION**

---

**This chapter shows an example of a serial programming connection using the AF220/AF210/AF120/AF110 Flash Microcontroller Programmer by Yokogawa Digital Computer Corporation.**

---

- 24.1 Basic Configuration of Serial Programming Connection with MB90F481B/MB90F482B/MB90F488B/MB90F489B
- 24.2 Example of Connection in Single-Chip Mode (When Using the User Power Supply)
- 24.3 Example of Minimum Connection with Flash Microcontroller Programmer (When Using the User Power Supply)



## 24.1 Basic Configuration of Serial Programming Connection with MB90F481B/MB90F482B/MB90F488B/MB90F489B

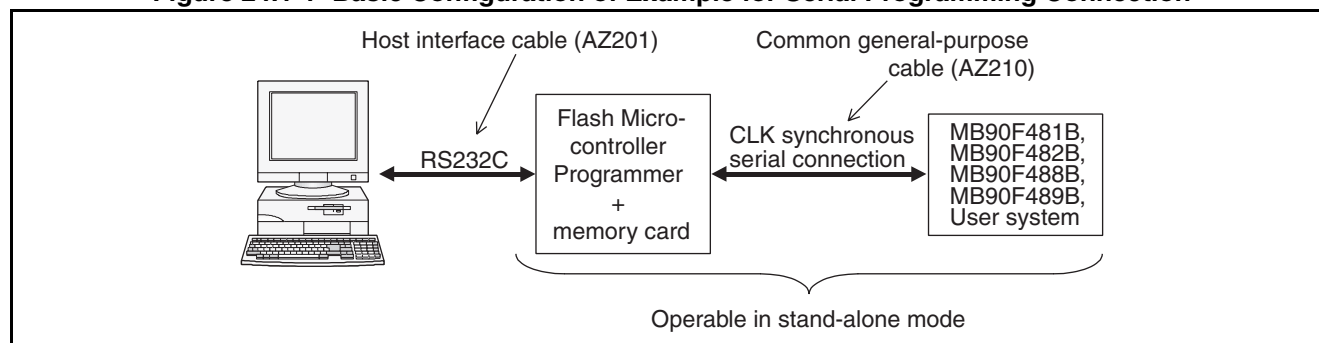
The MB90F481B/MB90F482B/MB90F488B/MB90F489B supports serial on-board writing (Fujitsu standard) of the flash ROM. This section provides the related specifications.

### ■ Basic Configuration of Serial Programming Connection with MB90F481B/MB90F482B/MB90F488B/MB90F489B

Fujitsu standard serial on-board writing uses the Yokogawa Digital Computer Corporation flash microcontroller programmer. It is possible to write it by selecting either of the program that operates by the single chip mode or the internal ROM external ROM bus mode.

Figure 24.1-1 shows the basic configuration for the example for serial programming connection.

**Figure 24.1-1 Basic Configuration of Example for Serial Programming Connection**



For information on the functions of and operational procedures related to the flash microcontroller programmer (AF220/AF210/AF120/AF110), the general-purpose common cable (AZ210) for connection, and the connector, contact Yokogawa Digital Computer Corporation.

## ■ Pins Used for Fujitsu Standard Serial On-board Writing

Table 24.1-1 shows the functions of the related pins used for Fujitsu Standard serial on-board writing.

**Table 24.1-1 Function of Pins**

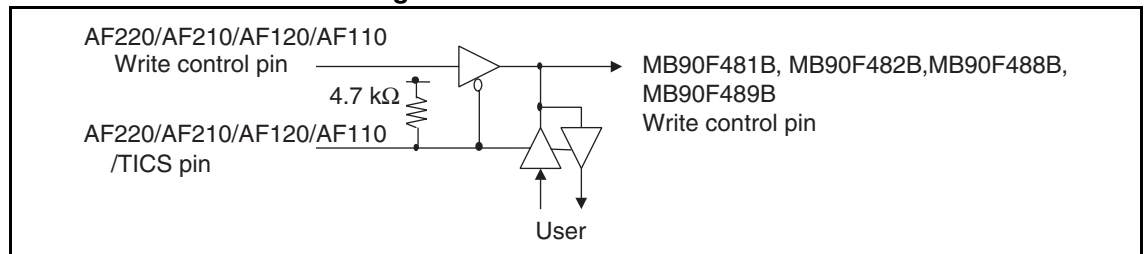
| Pin                     | Function                      | Description   |
|-------------------------|-------------------------------|---|
| MD2, MD1, MD0           | Mode pin                      | Setting MD2=1, MD1=1, and MD0=0 to enter the serial programming mode.   |
| X0, X1                  | Oscillation pin               | As, in the serial programming mode, CPU internal operation clock is the PLL clock multiplied-by-1, the internal operation clock frequency is equal to the oscillation clock frequency. Consequently, the frequencies that can be input to the high-speed oscillation input pin for serial writing are from 4.5 to 25 MHz. |
| P80, P81                | Programming program start pin | Input "L" level to P80, and "H" level to P81  |
| $\overline{\text{RST}}$ | Reset pin                     | -   |
| SIN0                    | Serial data input pin         | Use UART0 for CLK sync mode.  |
| SOT0                    | Serial data output pin        |   |
| SCK0                    | Serial clock input pin        |   |
| V <sub>CC</sub>         | Power voltage supply pin      | Programming voltage (V <sub>CC</sub> = 3.3 V ± 5%)  |
| V <sub>SS</sub>         | GND pin                       | Must be shared with GND of the flash microcontroller programmer.  |

Note:

To use the P80, P81, SIN0, SOT0, and SCK0 pins within the user system as well, the control circuit shown in the Figure 24.1-2 is required.

Using the flash microcontroller programmer's "/TICS" signal for outputting "L", the user circuit can be disconnected in serial programming mode. Refer to the connection example.

**Figure 24.1-2 Pin Control Circuit**



## ■ Oscillation Clock Frequency and Serial Clock Input Frequency

The serial clock frequencies that can be used for input to the MB90F481B, MB90F482B, MB90F488B, and MB90F489B can be calculated from the following formulas.

Use the flash microcontroller programmer settings to set the serial clock input frequency for the required oscillation clock frequency.

Serial clock frequency to be input =  $0.125 \times$  oscillation clock frequency

Table 24.1-2 shows a serial clock frequency that can be input.

**Table 24.1-2 Example of Serial Clock Frequency That can be Input**

| Oscillation clock frequency | Maximum serial clock frequency that can be input to microcontroller | Maximum serial clock frequency that can be set for AF220/AF210/AF120/AF110 | Maximum serial clock frequency that can be set for AF200 |
|-----------------------------|---|--|--|
| 8 MHz                       | 1MHz  | 850kHz   | 500kHz   |
| 16 MHz                      | 2MHz  | 1.25MHz  | 500kHz   |

## ■ System Configuration of Flash Microcontroller Programmer (Yokogawa Digital Computer Corporation)

Table 24.1-3 shows the system configuration of the flash microcontroller programmer.

**Table 24.1-3 System Configuration of the Flash Microcontroller Programmer**

| Type      |            | Function   |                               |
|-----------|------------|--|-------------------------------|
| Main body | AF220/AC4P | Model with built-in Ethernet interface   | /100 V to 220 V power adapter |
|           | AF210/AC4P | Standard model   | /100 V to 220 V power adapter |
|           | AF120/AC4P | Model with built-in single key Ethernet interface                              | /100 V to 220 V power adapter |
|           | AF110/AC4P | Single key model   | /100 V to 220 V power adapter |
| AZ221     |            | Programmer dedicated RS232C cable for PC/AT                                    |                               |
| AZ210     |            | Standard target probe (a) length: 1 m  |                               |
| FF201     |            | Fujitsu F <sup>2</sup> MC-16LX flash microcontroller control module            |                               |
| AZ290     |            | Remote controller  |                               |
| /P2       |            | 2M bytes PC Card (Option) FLASH memory capacity of up to 128 K bytes supported |                               |
| /P4       |            | 4M bytes PC Card (Option) FLASH memory capacity of up to 512 K bytes supported |                               |

Inquiries: Yokogawa Digital Computer Corporation  
Telephone number: (81)-42-333-6224

## ■ Examples of Serial Programming Connections

Examples for the following two types of connections are shown below.

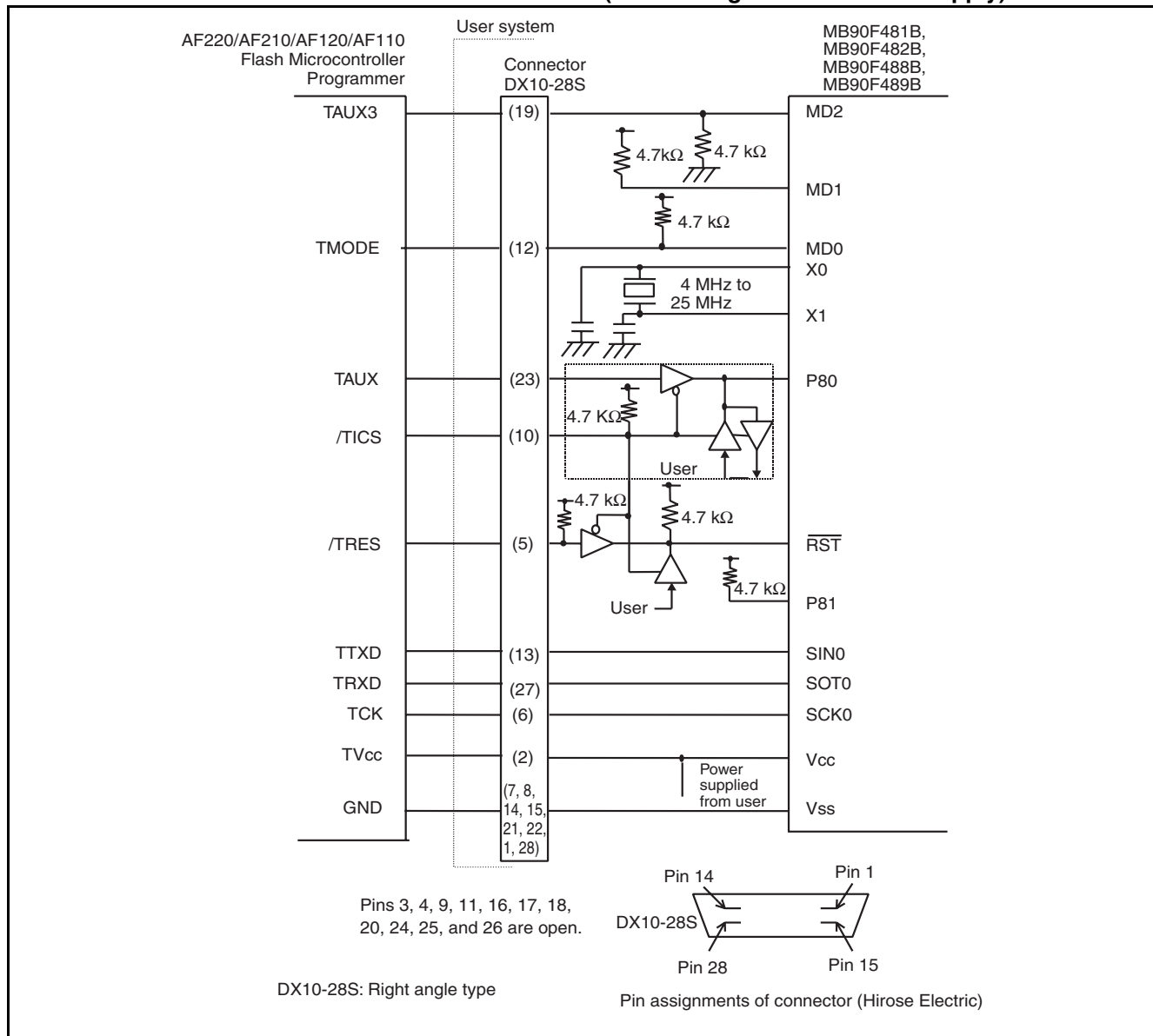
- Example of connection in single-chip mode (When Using the User Power Supply)
- Example of minimum connection with flash microcontroller programmer (When Using the User Power Supply)

## 24.2 Example of Connection in Single-Chip Mode (When Using the User Power Supply)

In the user system, mode pins MD2 and MD0, which are set to single-chip mode, are supplied with the inputs MD2=1 and MD0=0 by TAUX3 and TMODE of AF220/AF210/AF120/AF110, and the system is set to serial programming mode (serial programming mode: MD2, MD1, MD0=110).

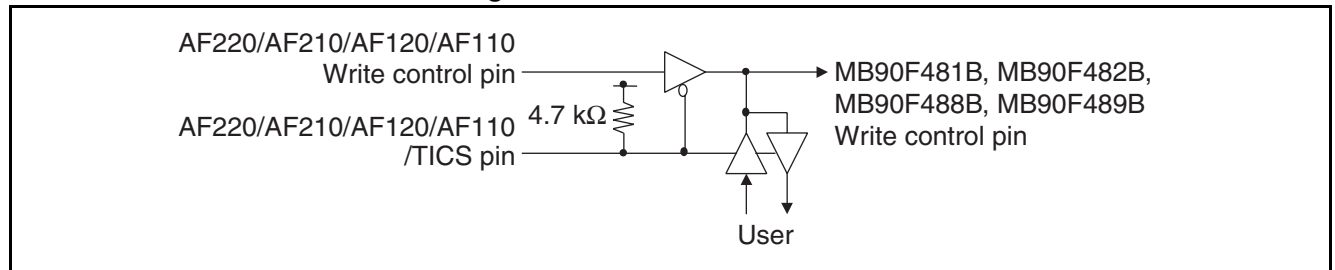
### ■ Example of Connection in Single-Chip Mode (When Using the User Power Supply)

Figure 24.2-1 Example of Serial Programming Connection in Single Chip Mode for MB90F481B/MB90F482B/MB90F488B/MB90F489B (when Using the User Power Supply)



Similarly to P80, using the SIN0, SOT0, and SCK0 pins in the user system requires a control circuit as shown in Figure 24.2-2. The user circuit is disconnected in serial programming mode by the flash microcontroller programmer's "/TICS" signal.

**Figure 24.2-2 Pin Control Circuit**

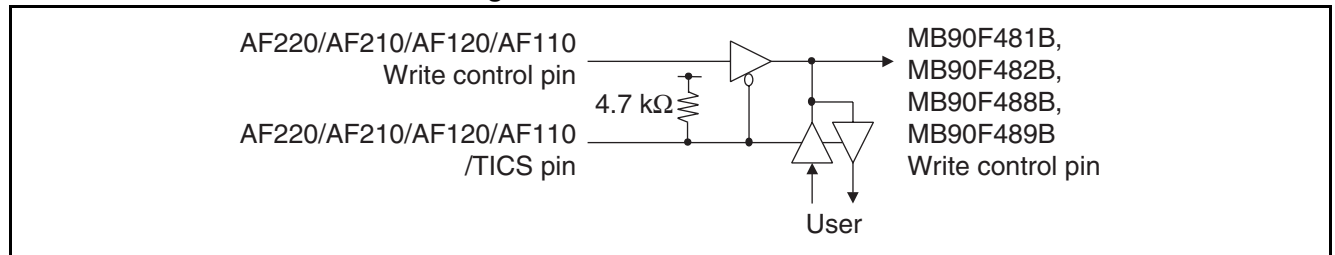


Connect to AF220/AF210/AF120/AF110 when the power supply of the user system is turned off.



Using the pins SIN0, SOT0, and SCK0 in the user system requires a control circuit as shown in Figure 24.3-2. The user circuit is disconnected in serial programming mode by the flash microcontroller programmer's "/TICS" signal for outputting "L".

**Figure 24.3-2 Pin Control Circuit**



Connect to AF220/AF210/AF120/AF110 when the power supply of the user system is turned off.





## CHAPTER 25 PWC TIMER (ONLY MB90485 SERIES)

---

**This chapter provides an overview of the PWC timer, explains the configuration, the configuration and functions of its registers interrupt, shows the precautions on use.**

---

- 25.1 Overview of PWC Timer
- 25.2 Configuration of PWC Timer
- 25.3 Configuration and Functions of PWC Timer Registers
- 25.4 Interrupt of PWC Timer
- 25.5 Operations of PWC Timer
- 25.6 Notes on PWC Timer Usage

## 25.1 Overview of PWC Timer

---

The PWC timer is a 16-bit multifunctional up-count timer used to measure the pulse width of input signals.

**PWC: Pulse Width Count (for pulse width measurement)**

---

### ■ PWC Timer Functions

On the hardware level, the PWC timer consists of one 16-bit up-count timer, one input pulse divider and divide ratio control register, one measurement input pin, and one 16-bit control register. When regarding each 16-bit control register as one channel, the PWC timer has a total of three channels. The PWC timer provides the following functions.

#### ○ Timer function

- Each time the timer function has been set, an interrupt request will be generated
- An internal clock used as a reference clock can be selected from three types (Divided-by 4/16/32 of machine clock).

#### ○ Pulse width measurement function

- Measures the time between any events input from the outside via the pulse input
- An internal clock used as a reference clock can be selected from three types (Divided-by 4/16/32 of machine clock)
- Various measurement modes
  - "H" pulse width (rising edge to falling edge)/"L" pulse width (rising edge to falling edge)
  - Rising interval (rising edge to rising edge)/falling interval (falling edge to falling edge)
  - Measurement between edges (rising edge or falling edge to falling edge or rising edge)
- An 8-bit input divider is used for divide measurement with "divide-by- $22 \times n$ " ( $n = 1, 2, 3, 4$ ) of the input pulse.
- At the end of measurement, an interrupt can be generated.
- Either one-time measurement or repeated measurement can be selected.

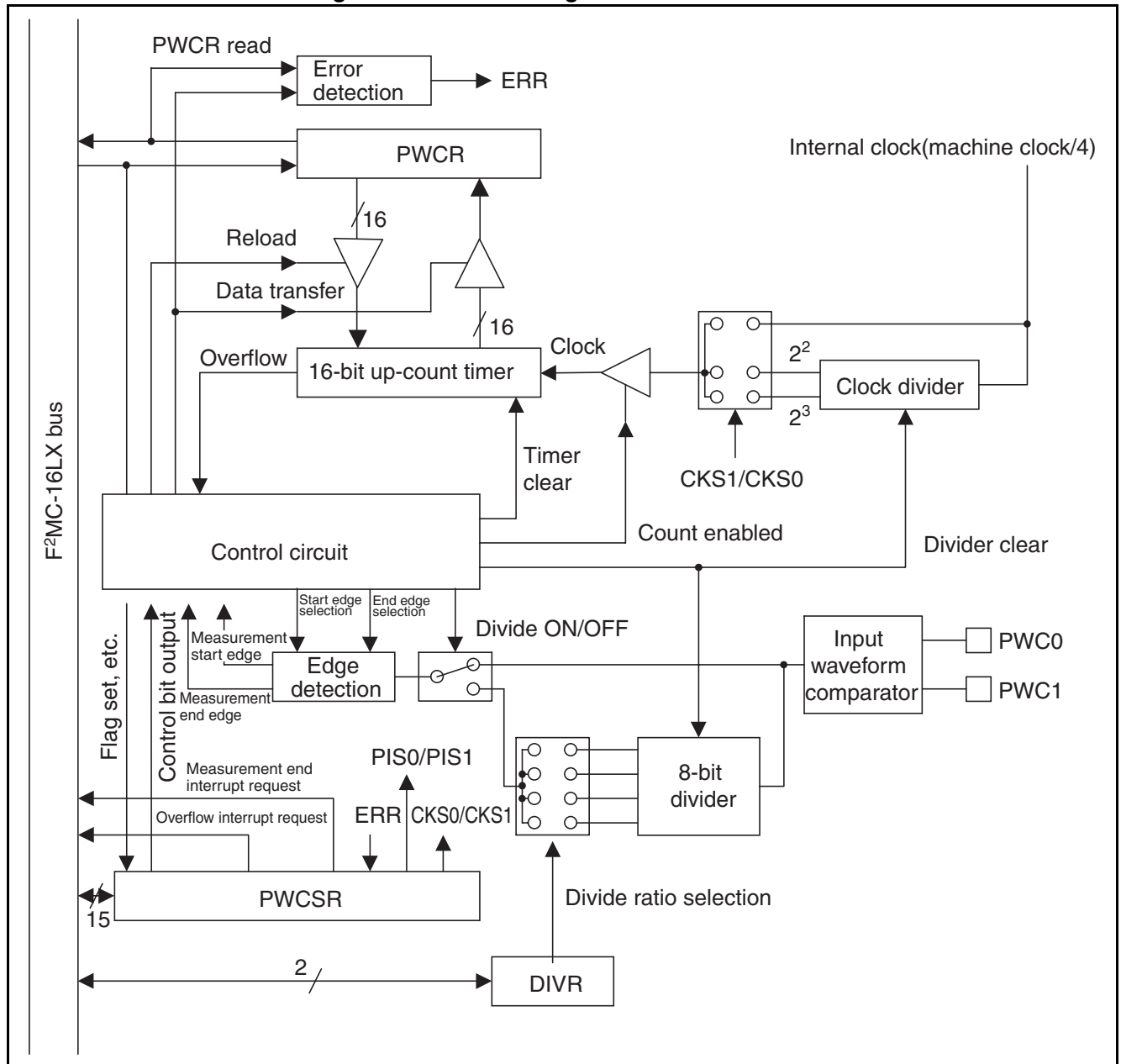
## 25.2 Configuration of PWC Timer

The PWC timer consists of the PWC control/statue register, PWC data buffer, and divide ratio control register.

### ■ Block Diagram of PWC Timer

Figure 25.2-1 shows a block diagram of the PWC timer.

Figure 25.2-1 Block Diagram of the PWC Timer



## ■ Pin Related to PWC Timer

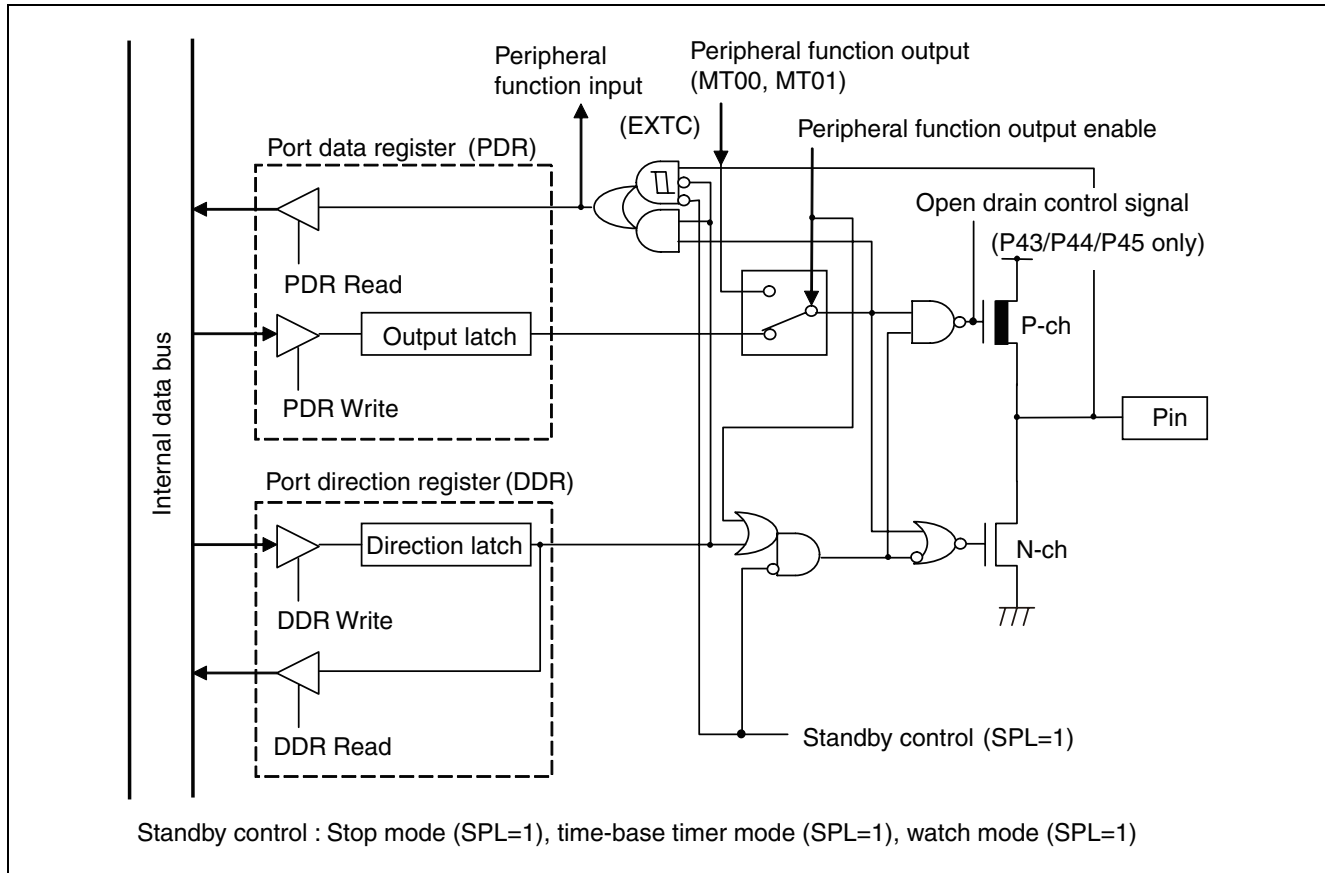
The pin related to the PWC timer has 3ch of PWC0/PWC1/PWC2 and functions as the input port when the PWC is used. The PWC0/PWC1/PWC2 pins function as the general-purpose I/O port (P36/PWC0, P37/PWC1, P75/PWC2) and input pin.

### ● Setting when using as PWC0/PWC1/PWC2 pins

When the PWC0/PWC1/PWC2 are used as input by the PWC timer, the P36/PWC0, P37/PWC1, P75/PWC2 pins should be set to the input port by the port direction register (DDR3→ bit14, 15→ "0" DDR7 bit13→ "0").

## ■ Block Diagram of Pin Related to PWC Timer

Figure 25.2-2 Block Diagram of Pin Related to PWC Timer



## 25.3 Configuration and Functions of PWC Timer Registers

This section describes the configuration and functions of the registers used in the PWC timer.

### ■ List of PWC Timer Registers

Figure 25.3-1 shows a list of the PWC timer registers.

**Figure 25.3-1 List of PWC Timer Registers**

|  |             |     |  |   |  |  |  |       |
|--|-------------|-----|--|---|--|--|--|-------|
|  | 15          | 8 7 |  | 0 |  |  |  |       |
|  | PWCSR0 to 2 |     |  |   |  |  |  | (R/W) |
|  | PWC0 to 2   |     |  |   |  |  |  | (R/W) |
|  | DIVR0 to 2  |     |  |   |  |  |  | (R/W) |

|                          |       |       |      |       |       |       |     |          |                                     |
|--------------------------|-------|-------|------|-------|-------|-------|-----|----------|-------------------------------------|
| ch.0 000077 <sub>H</sub> | 15    | 14    | 13   | 12    | 11    | 10    | 9   | 8        | PWCSR0 to PWCSR2                    |
| ch.1 00007B <sub>H</sub> | STRT  | STOP  | EDIR | EDIE  | OVIR  | OVIE  | ERR | Reserved | PWC control/status register         |
| ch.2 00007F <sub>H</sub> | (R/W) | (R/W) | (R)  | (R/W) | (R/W) | (R/W) | (R) | (-)      | Initial value 0000000X <sub>B</sub> |

|                          |       |       |       |       |       |       |       |       |                                     |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------------|
| ch.0 000076 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | PWCSR0 to PWCSR2                    |
| ch.1 00007A <sub>H</sub> | CKS1  | CKS0  | PIS1  | PIS0  | S/C   | MOD2  | MOD1  | MOD0  | PWC control/status register         |
| ch.2 00007E <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub> |

|                          |       |       |       |       |       |       |       |       |                                     |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------------|
| ch.0 000079 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | PWCR0 to PWCR2                      |
| ch.1 00007D <sub>H</sub> | D15   | D14   | D13   | D12   | D11   | D10   | D9    | D8    | PWC data buffer register            |
| ch.2 000081 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub> |

|                          |       |       |       |       |       |       |       |       |                                     |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------------|
| ch.0 000078 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | PWCR0 to PWCR2                      |
| ch.1 00007C <sub>H</sub> | D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    | PWC data buffer register            |
| ch.2 000080 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | Initial value 00000000 <sub>B</sub> |

|                          |     |     |     |     |     |     |       |       |  |
|--------------------------|-----|-----|-----|-----|-----|-----|-------|-------|--|
| ch.0 000082 <sub>H</sub> | 7   | 6   | 5   | 4   | 3   | 2   | 1     | 0     | DIVR0 to DIVR2                         |
| ch.1 000084 <sub>H</sub> | -   | -   | -   | -   | -   | -   | DIV1  | DIV0  | Divide ratio control register          |
| ch.2 000086 <sub>H</sub> | (-) | (-) | (-) | (-) | (-) | (-) | (R/W) | (R/W) | Initial value - - - - -00 <sub>B</sub> |

## 25.3.1 PWC Control/Status Register (PWCSR0 to PWCSR2)

This section describes the configuration and functions of the PWC control/status register (PWCSR0 to PWCSR2).

### ■ PWC Control/Status Register (PWCSR0 to PWCSR2)

Figure 25.3-2 shows the bit configuration of the PWC control/status register (PWCSR0 to PWCSR2).

**Figure 25.3-2 Bit Configuration of the PWC Control/Status Register (PWCSR0 to PWCSR2)**

|                          |       |       |       |       |       |       |       |          |                 |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|----------|-----------------|
| ch.0 000077 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8        | Bit No.         |
| ch.1 00007B <sub>H</sub> | STRT  | STOP  | EDIR  | EDIE  | OVIR  | OVIE  | ERR   | Reserved |                 |
| ch.2 00007F <sub>H</sub> | (R/W) | (R/W) | (R)   | (R/W) | (R/W) | (R/W) | (R)   | (-)      | ↔ Read/write    |
|                          | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (X)      | ↔ Initial value |
| ch.0 000076 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0        |                 |
| ch.1 00007A <sub>H</sub> | CKS1  | CKS0  | PIS1  | PIS0  | S/C   | MOD2  | MOD1  | MOD0     | PWCSR           |
| ch.2 00007E <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W)    | ↔ Read/write    |
|                          | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)      | ↔ Initial value |

The functions of bits in the PWC control/status register (PWCSR0 to PWCSR2) are described below.

#### [bit15, bit14] STRT, STOP (timer start bit, timer stop bit)

These bits are used to start/restart/stop the 16-bit up-count timer. The operation state of the timer is displayed in read operations.

The tables below show the functions of the STRT and STOP bits.

**Table 25.3-1 Functions Related to Write Operations (Operation Control of 16-bit Up-count Timer)**

| STRT | STOP | Operation control function                                   |
|------|------|--|
| 0    | 0    | No function/no effect on operation                           |
| 0    | 1    | Timer start/restart (when counting is allowed) *             |
| 1    | 0    | Timer operation forcible stop (when counting is prohibited)* |
| 1    | 1    | No function/no effect on operation                           |

\* : A clear bit-operation instruction is available

**Table 25.3-2 Functions Related to Read Operations (Indicating the Operation State of the 16-bit Up-count Timer)**

| STRT | STOP | Operation control function   |
|------|------|--|
| 0    | 0    | Timer stop mode (not started or end of measurement)<br>(initial value) |
| 1    | 1    | Timer count operation mode (measurement in progress)                   |

- At reset, initialized to "00<sub>B</sub>".
- Reading and writing are allowed. However, the meaning of the register contents is different for write and read operations, as indicated in Table 25.3-1 and Table 25.3-2.
- The reading value returned in read-modify-write instructions is fixed at "11<sub>B</sub>" regardless of all bit values.
- Writing the START/STOP bit to start/stop the timer can be performed for individual bits in order to execute bit operation instructions (bit clear). However, note that no bit operation instruction is available to read the operation state (the result of reading is always "operation in progress").

**[bit13] EDIR (measurement end interrupt request flag)**

This bit is a flag that indicates the end of measurement in pulse width measurement. By setting this bit when measurement end interrupt sources are enabled (bit12: EDIE = 1), a measurement end interrupt request is generated.

| EDIR              | Cause of setting or clearing  |
|-------------------|---|
| Cause for setting | Set when pulse width measurement ends (when the PWCR contains the measurement result) |
| Cause of clearing | Cleared when the PWCR (measurement result) is read                                    |

- Initialized to "0" at reset.
- Only reading is allowed.
- Bit values cannot be changed by writing

**[bit12] EDIE (measurement end interrupt enable)**

This bit is used for control of measurement end interrupt requests when pulse width measurement is performed as shown in the table below.

| EDIE | Operation   |
|------|---|
| 0    | Measurement end interrupt request output prohibited (no interrupts occur even if EDIR is set) [initial value] |
| 1    | Measurement end interrupt request output allowed (interrupt occurs if EDIR is set)                            |

- Initialized to "0" at reset
- Reading or writing is allowed.



**[bit11] OVIR (timer overflow interrupt request flag)**

This bit is a flag used to indicate an overflow of the 16-bit up-count timer to the area from FFFF<sub>H</sub> to 0000<sub>H</sub>. If this bit is set when timer overflow interrupt requests are enabled (bit10:OVIE = "1"), a timer overflow interrupt request is generated.

| OVIR              | Cause of setting or clearing  |
|-------------------|---|
| Cause for setting | Set if a timer overflow occurs (to the area from FFFF <sub>H</sub> to 0000 <sub>H</sub> ) |
| Cause of clearing | Writing "0" or clearing by $\mu$ DMAC   |

- Initialized to "0" at reset
- Reading and writing are allowed; however, only writing "0" is effective, while writing "1" causes no changes in the bit value.
- Read-modify-write type instructions read "1" irrespective of bit values.

**[bit10] OVIE (timer overflow interrupt request enable)**

This bit is used for control of measurement end interrupt requests during pulse width measurement, as shown in the following table.

| OVIE | Operation  |
|------|--|
| 0    | Overflow interrupt request output prohibited (No interrupts occur even if OVIR is set) [initial value] |
| 1    | Overflow interrupt request output allowed (An interrupt occurs if OVIR is set)                         |

- Initialized to "0" at reset.
- Reading and writing are allowed.

**[bit9] ERR (Error Flag)**

This bit is a flag that indicates that, in the repeated measurement mode of pulse width measurement, a measurement operation has completed before the previous measurement result was read out from the PWCR. In this case, the PWCR value will be updated to the new measurement result while the immediately previous measurement result will be lost. Measurement will continue irrespective of the value for this bit.

| ERR               | Cause of setting or clearing  |
|-------------------|---|
| Cause for setting | This bit is set if a measurement result that has not been read out is overwritten by the next result. |
| Cause of clearing | This bit is cleared whenever the PWCR (measurement result) is read                                    |

- Initialized to "0" at reset.
- Only reading is allowed. Write operations have no effect.

**[bit8] Reserved bit**

This bit is reserved.

**[bit7, bit6] CKS1, CKS0 (clock selection)**

These bits are used to select one internal count clock out of the three types listed in Table 25.3-3.

**Table 25.3-3 Count Clocks of the 16-bit Up-count Timer**

| CKS1 | CKS0 | Count clock selection   |
|------|------|---|
| 0    | 0    | Divide-by-4 clock of the machine clock (0.25 $\mu$ s for a machine clock of 16 MHz) [initial value] |
| 0    | 1    | Divide-by-16 clock of the machine clock (1.00 $\mu$ s for a machine clock of 16 MHz)                |
| 1    | 0    | Divide-by-32 clock of the machine clock (2.00 $\mu$ s for a machine clock of 16 MHz)                |
| 1    | 1    | Setting prohibited (result of setting is undefined)   |

- Initialized to "00<sub>B</sub>" at reset
- Reading and writing are allowed. However, setting to "11<sub>B</sub>" is prohibited.

---

Note:

Rewriting after timer start is prohibited. Write only before the timer is started or after the timer is stopped.

---

**[bit5, bit4] PIS1, PIS0 (Pulse width measurement input pin select)**

These bits are used to select the pulse width measurement input pin.

**Table 25.3-4 Selection of Pulse Width Measurement Input Pin**

| PIS1 | PIS0 | Input clock selection                                  |
|------|------|--|
| 0    | 0    | (Selecting PWC0 pin) [initial value]                   |
| 0    | 1    | Two inputs selected for compare (rising edge compare)  |
| 1    | 0    | Two inputs selected for compare (falling edge compare) |
| 1    | 1    | Setting prohibited (result of setting is undefined)    |

- Initialized to "00<sub>B</sub>" at reset.
- Reading and writing are allowed. However, setting to "11<sub>B</sub>" is prohibited.
- This bit is only valid in PWC0. (PWC0/PWC1 are used for input). For details, refer to Section "25.5.2 Operations of the Pulse Width Measurement Function".

---

Note:

Rewriting after timer start is prohibited. Write only before the timer is started or after the timer is stopped.

---

**[bit3] S/C (Selection of Measurement Mode (one-shot/repeated))**

This bit is used to select the measurement mode.

**Table 25.3-5 Selection of the Measurement Mode for the 16-bit Up-count Timer**

| S/C | Measurement mode selection                   | Timer mode                 | Pulse width                                   |
|-----|--|----------------------------|---|
| 0   | One-shot measurement mode<br>[initial value] | No reload (one-shot)       | Stopped after one-time measurement            |
| 1   | Repeated measurement mode                    | With reload (reload timer) | Repeated measurement: buffer register enabled |

- Initialized to "0" at reset.
- Reading and writing are allowed.

Note:

Rewriting after timer start is prohibited. Write always either before the timer is started or after the timer is stopped.

**[bit2, bit1, bit0] MOD2, MOD1, MOD0 (Selection of operation mode or measurement edge)**

These bits are used to select the operation mode and an edge at which width measurement is performed.

**Table 25.3-6 Selection of Operation Mode or Measurement Edge for the 16-bit Up-count Timer**

| MOD2 | MOD1 | MOD0 | Selection of operation mode/measurement edge  |
|------|------|------|---|
| 0    | 0    | 0    | Timer mode [initial value]  |
| 0    | 0    | 1    | Timer mode (reload mode only)   |
| 0    | 1    | 0    | Pulse width measurement mode between all edges (rising edge or falling edge to falling edge or rising edge) |
| 0    | 1    | 1    | Divide interval measurement mode (input divide enabled)   |
| 1    | 0    | 0    | Interval measurement mode between rising edges (rising edge to rising edge)                                 |
| 1    | 0    | 1    | "H" level pulse width measurement mode (rising edge to falling edge)  |
| 1    | 1    | 0    | "L" level pulse width measurement mode (falling edge to rising edge)  |
| 1    | 1    | 1    | Interval measurement mode between falling edges (falling edge to falling edge)                              |

- Initialized to "000<sub>B</sub>" at reset.
- Reading and writing are allowed.

Note:

Rewriting immediately after timer start is prohibited. Write always either before the timer is started or after the timer is stopped.

## 25.3.2 PWC Data Buffer Register (PWCR0 to PWCR2)

This section describes the configuration and functions of the PWC data buffer register (PWCR0 to PWCR2).

### ■ PWC Data Buffer Register (PWCR0 to PWCR2)

Figure 25.3-3 shows the bit configuration of the PWC data buffer register (PWCR0 to PWCR2).

**Figure 25.3-3 Bit Configuration of the PWC Data Buffer Register (PWCR0 to PWCR2)**

|                          |       |       |       |       |       |       |       |       |                          |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------------------|
| ch.0 000079 <sub>H</sub> | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | PWCR                     |
| ch.1 00007D <sub>H</sub> | D15   | D14   | D13   | D12   | D11   | D10   | D9    | D8    | PWC data buffer register |
| ch.2 000081 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | ↔ Read/write             |
|                          | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | ↔ Initial value          |
| ch.0 000078 <sub>H</sub> | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | PWCR                     |
| ch.1 00007C <sub>H</sub> | D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    | PWC data buffer register |
| ch.2 000080 <sub>H</sub> | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | ↔ Read/write             |
|                          | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | (0)   | ↔ Initial value          |

The function of the PWC data buffer register (PWCR0 to 2) depends on whether the timer mode is selected (by setting the PWCSR register bit2 to bit0 (MOD2 to 0)) or the pulse width measurement mode is selected.

#### ○ Timer mode (reading/writing allowed)

In a reload timer operation (PWCSR bit3:S/C = 1), the PWC data buffer register operates as a buffer register to store reload data. In this case, both reading and writing are allowed.

In one-shot timer operation mode (PWCSR bit3:S/C = 0), the PWC data buffer register is used to directly access the up-count timer. In this case, both reading and writing are allowed, though writing can only be performed when the timer is stopped. Reading is available at any time, enabling the timer value to be read during counting.

#### ○ Pulse width measurement mode (Only reading allowed)

In repeated measurement mode (PWCSR bit3: S/C = 1), the PWC data buffer register operates as a buffer register to store the results of previous measurement.

In this case, only reading is allowed. Write operations do not change the value of the register.

In one-shot mode (PWCSR bit3: S/C = 0), the PWC data buffer register is used to directly access the up-count timer.

In this case, only reading is allowed. Write operations do not change the value of the register. Reading is available at any time, enabling the timer value to be read during counting. After the measurement ends, the PWC data buffer register stores the measurement result.

#### Note:

This register can be accessed only via word transfer instructions. At reset, the register is initialized to "0000<sub>H</sub>".

### 25.3.3 Divide Ratio Control Register (DIVR0 to DIVR2)

This section describes the configuration and functions of the divide ratio control register (DIVR0 to 2).

■ Divide Ratio Control Register (DIVR0 to DIVR2)

Figure 25.3-4 shows the bit configuration of the divide ratio control register (DIVR0 to DIVR2).

Figure 25.3-4 Bit Configuration of the Divide Ratio Control Register (DIVR0 to DIVR2)

|                          |     |     |     |     |     |     |       |       |                               |
|--------------------------|-----|-----|-----|-----|-----|-----|-------|-------|-------------------------------|
| ch.0 000082 <sub>H</sub> | 7   | 6   | 5   | 4   | 3   | 2   | 1     | 0     | DIVR                          |
| ch.1 000084 <sub>H</sub> | -   | -   | -   | -   | -   | -   | DIV1  | DIV0  | Divide ratio control register |
| ch.2 000086 <sub>H</sub> | (-) | (-) | (-) | (-) | (-) | (-) | (R/W) | (R/W) | ↔ Read/write                  |
|                          | (-) | (-) | (-) | (-) | (-) | (-) | (0)   | (0)   | ↔ Initial value               |

This register is only used in divide interval measurement mode (PWCSR: bit2, bit1, bit0:MOD2, MOD1, MOD0 = 001<sub>B</sub>); it is not used in other modes.

In divide interval measurement mode, pulses input to the measurement pin are divided according to the divide ratio set in this register. This allows measuring one interval width.

Table 25.3-7 Selection of Divide Ratio

| DIV1 | DIV0 | Count clock selection       |
|------|------|-----------------------------|
| 0    | 0    | Divide-by-4 [initial value] |
| 0    | 1    | Divide-by-16                |
| 1    | 0    | Divide-by-64                |
| 1    | 1    | Divide-by-256               |

- Initialized to "00<sub>B</sub>" at reset
- Reading and writing are allowed.

Note:

Rewriting after timer start is prohibited. Write always either before the timer is started or after it is stopped.

## 25.4 Interrupt of PWC Timer

The interrupt of the PWC timer occurs when an overflow of the up counter in timer function and the pulse width measurement are terminated. The PWC0 is used for PWC interrupt that can activate the DMA transfer and extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Interrupt of PWC Timer

The interrupt control bit and interrupt source of the PWC timer is shown in the following table.

|                                     | Interrupt of termination for pulse width measurement                             | Overflow interrupt of timer at operation of timer mode                           |
|-------------------------------------|--|--|
| Interrupt request flag              | PWCSR0:EDIR (bit13) ch.0<br>PWCSR1:EDIR (bit13) ch.1<br>PWCSR2:EDIR (bit13) ch.2 | PWCSR0:OVIR (bit11) ch.0<br>PWCSR1:OVIR (bit11) ch.1<br>PWCSR2:OVIR (bit11) ch.2 |
| Interrupt request output enable bit | PWCSR0:EDIE (bit12) ch.0<br>PWCSR1:EDIE (bit12) ch.1<br>PWCSR2:EDIE (bit12) ch.2 | PWCSR0:OVIE (bit10) ch.0<br>PWCSR1:OVIE (bit10) ch.1<br>PWCSR2:OVIE (bit10) ch.2 |
| Interrupt generation source         | When pulse width measurement is terminated                                       | When 16-bit up count timer overflows from FFFF <sub>H</sub> → 0000 <sub>H</sub>  |

### ■ Interrupt Source Related to PWC Timer

The following two interrupt sources are available.

- In the timer function, overflow interrupt occurs if the OVIE (bit10) is "1" when the OVIR (bit11) flag in the PWC control/status register (PWCSR0 to PWCSR2) is set.
- In the pulse width measurement, the interrupt occurs if the EDIE (bit12) is "1" (termination of transmission) when the EDIR (bit13) flag in the PWC control/status register (PWCSR0 to PWCSR2) is set.

## ■ Interrupt of PWC Timer, DMA Transfer, and EI<sup>2</sup>OS

Table 25.4-1 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 25.4-1 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source             | EI <sup>2</sup> OS clear | $\mu$ DMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|------------------------------|--------------------------|---------------------------|------------------|---------------------|----------------------------|---------------------|
|                              |                          |                           | Number           | Address             | Number                     | Address             |
| PWC1 (only MB90485 series)   | ○                        | ×                         | #19              | FFFFB0 <sub>H</sub> | ICR04                      | 0000B4 <sub>H</sub> |
| PWC2 (only MB90485 series)   | ○                        | ×                         | #20              | FFFFAC <sub>H</sub> |                            |                     |
| PWC0 (only MB90485 series) * | ○                        | 1                         | #21              | FFFA8 <sub>H</sub>  | ICR05                      | 0000B5 <sub>H</sub> |

× : Interrupt request flag is not cleared.

○ : Interrupt request flag is cleared.

\* : This interrupt source shares the interrupt source and interrupt number of other peripheral function.

For details, see Table 3.2-2.

### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/ $\mu$ DMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing.

## ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The PWC timer does not correspond to the DMA transfer function, but the EI<sup>2</sup>OS function. When the EI<sup>2</sup>OS function is used, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

## 25.5 Operations of PWC Timer

---

This section describes the operations of the PWC timer.

---

### ■ Outline of PWC Timer Operations

The PWC timer is a multifunction timer based on an 16-bit up-count timer, which integrates measurement input pins with the 8-bit input divide circuit. The PWC timer has the two major functions listed below:

- Timer function
- Pulse width count function

For either function, a count clock can be selected among three types of clocks (divide-by 4/16/32 of machine clock). The basic performance and operation of each function are described below.



## 25.5.1 Operations of the Timer Function

This timer is an up-count (incrementing) timer providing both reload and one-shot operations.

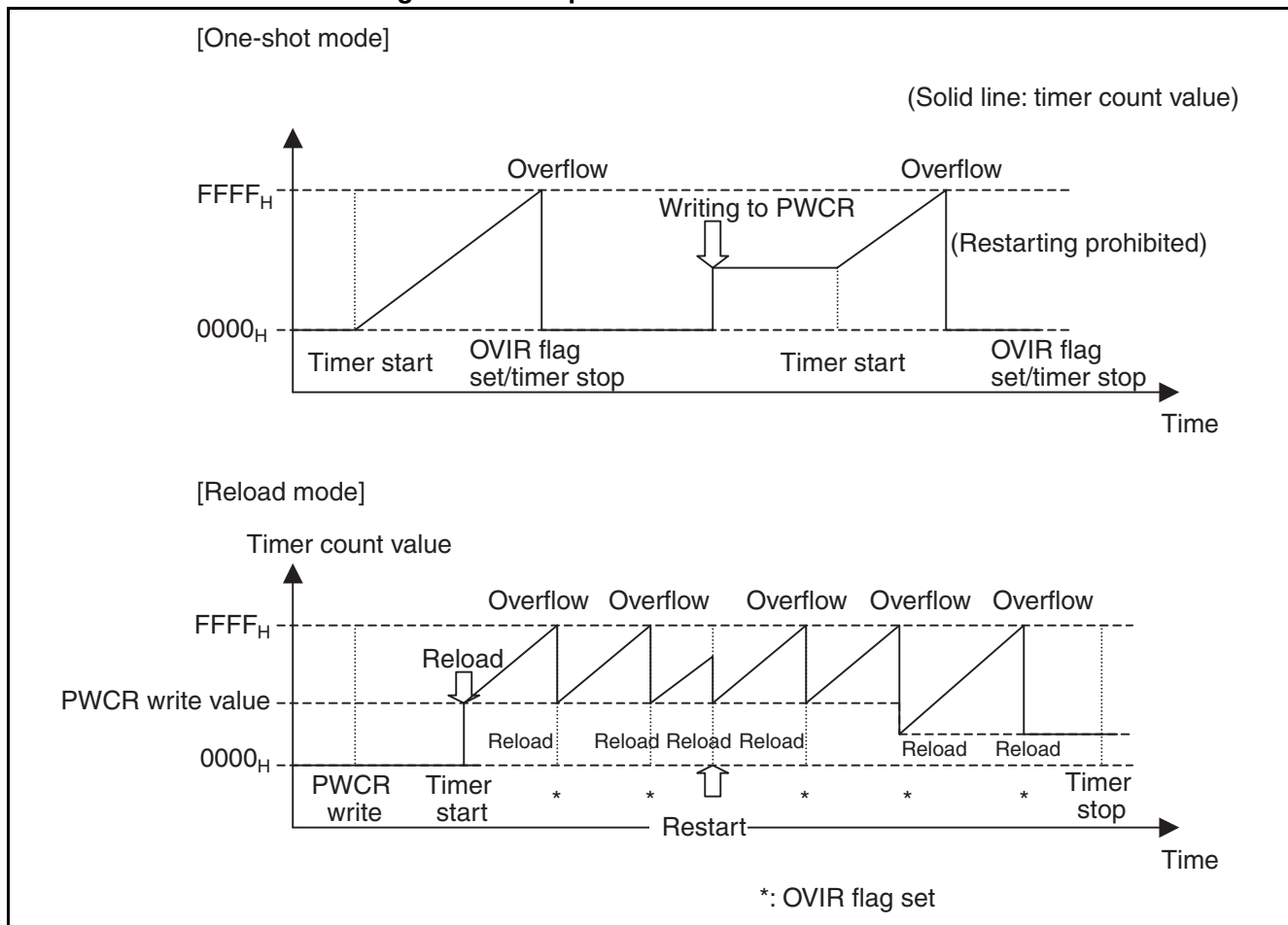
### ■ Operation of Timer Functions

After the timer starts, its value is incremented at each pulse of the count clock. If an overflow occurs in the range "FFFF<sub>H</sub>" --> "0000<sub>H</sub>", an interrupt request may be generated. If an overflow is generated, the following operations may be performed depending on the mode:

- In one-shot mode: Counting stops
- In reload mode: Reload register data is reloaded into the timer to restart counting.

Figure 25.5-1 shows the operations of the timer functions in one-shot mode and reload mode.

**Figure 25.5-1 Operations of Timer Functions**



## 25.5.2 Operations of the Pulse Width Measurement Function

With this function, the timer can be used to measure the time interval between any input pulse events.

### ■ Operations of the Pulse Width Measurement Function

After the start of the pulse width measurement function, counting does not start before the specified measurement start edge is input. The timer is cleared to "0000<sub>H</sub>" whenever a start edge is detected, and counting up starts. Counting stops when a stop edge is detected. The count value during this period is stored in the register as the pulse width. The end of measurement is detected by an interrupt.

After measurement ends, the following operations are performed depending on the measurement mode:

- In one-shot measurement mode: Operation is interrupted.
- In repeated measurement mode: The timer value is transferred to the buffer register and the measurement is suspended until input of the next start edge.

Figure 25.5-2 shows the operation in one-shot measurement mode. Figure 25.5-3 shows the operation in repeated measurement mode.

**Figure 25.5-2 Pulse Width Measurement Operation  
(One-shot Measurement Mode/"H" Level Pulse Width Measurement)**

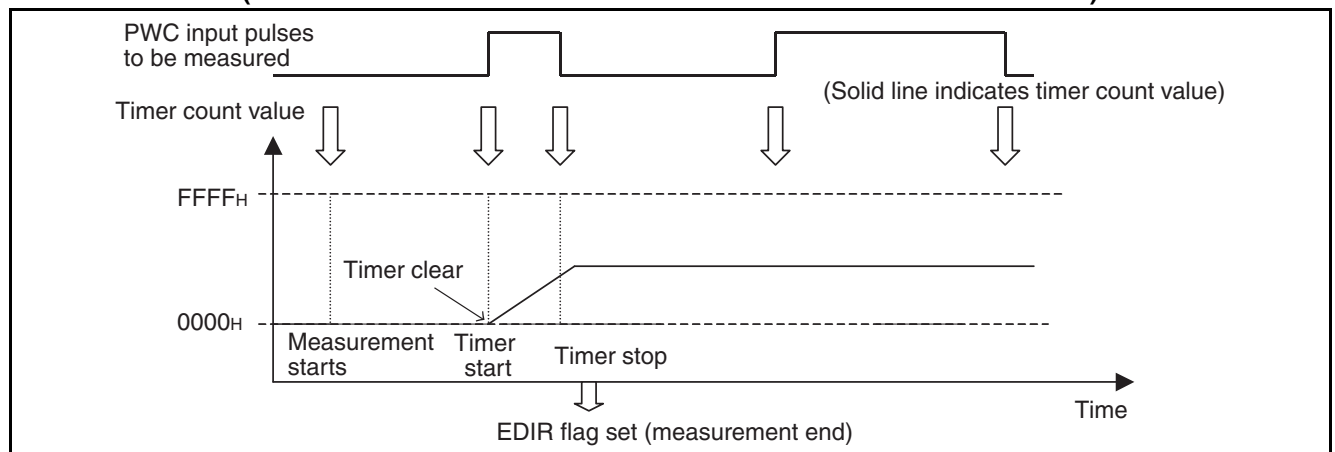
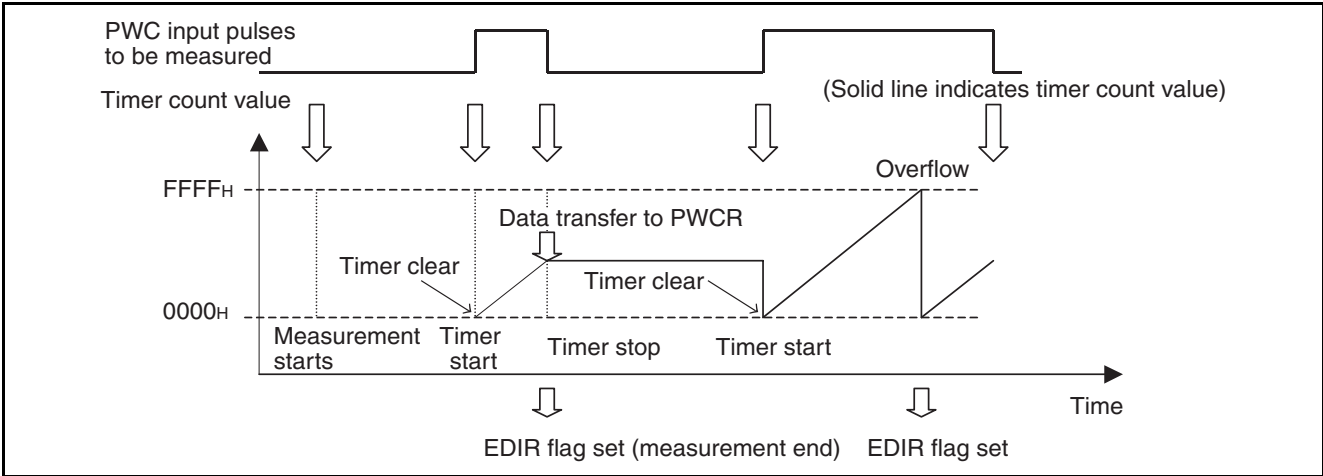


Figure 25.5-3 Pulse Width Measurement Operation  
(Repeated Measurement Mode/"H" Level Pulse Width Measurement)



## 25.5.3 Selection of Count Clock and Operation Mode

---

This section describes the selection of the count clock and the operation mode.

---

### ■ Count Clock Selection

A timer count clock can be selected from among three types of internal clocks by setting PWCSR: bit7 (CKS1) and bit6 (CKS0).

Table 25.5-1 shows how to select the count clock.

**Table 25.5-1 Count Clock Selection**

| PWCSR/bit7, bit6: CKS1, CKS0 | Internal count clock selected   |
|------------------------------|---|
| 00 <sub>B</sub>              | 1/4 of machine clock<br>(0.25 $\mu$ s for 16 MHz machine clock) [initial value] |
| 01 <sub>B</sub>              | 1/16 of machine clock<br>(1.0 $\mu$ s for 16 MHz machine clock)                 |
| 10 <sub>B</sub>              | 1/32 of machine clock<br>(2.0 $\mu$ s for 16 MHz machine clock)                 |

After reset, the divide-by-4 clock of the machine clock is initially selected.

---

Note:

Select the count clock before timer start.

---

### ■ Selection of Operation Mode

The operation mode or measurement mode is selected by setting the PWCSR bits.

- Selection of the operation mode: PWCSR: bit2, bit1, bit0 (MOD2, MOD1, MOD0 bits)  
Selecting the timer mode/pulse width measurement mode and specifying the measurement edge.
- Setting the measurement mode: PWCSR: bit3 (S/C bit)  
Selecting between one-shot measurement/repeated measurement or reload/one-shot

Table 25.5-2 shows the settings for selection of the operation mode/measurement mode.

**Table 25.5-2 Settings of Operation Mode/Measurement Mode**

| Operation mode          |   |                                       | S/C | MOD2 | MOD1 | MOD0 |
|-------------------------|---|---------------------------------------|-----|------|------|------|
| Timer                   | -   | One-shot timer                        | 0   | 0    | 0    | 0    |
|                         |   | Reload timer                          | 1   | 0    | 0    | 0    |
| Pulse width measurement | Rising edge or falling edge to rising edge or falling edge<br>Measurement between all edges | One-shot measurement: buffer disabled | 0   | 0    | 1    | 0    |
|                         |   | Repeated measurement: buffer enabled  | 1   | 0    | 1    | 0    |
|                         | Divide interval measurement<br>(Divide-by 1 to 256)   | One-shot measurement: buffer disabled | 0   | 0    | 1    | 1    |
|                         |   | Repeated measurement: buffer enabled  | 1   | 0    | 1    | 1    |
|                         | Rising edge to rising edge<br>Interval measurement between rising edges                     | One-shot measurement: buffer disabled | 0   | 1    | 0    | 0    |
|                         |   | Repeated measurement: buffer enabled  | 1   | 1    | 0    | 0    |
|                         | Rising edge to falling edge<br>"H" level pulse width measurement                            | One-shot measurement: buffer disabled | 0   | 1    | 0    | 1    |
|                         |   | Repeated measurement: buffer enabled  | 1   | 1    | 0    | 1    |
|                         | Falling edge to rising edge<br>L- pulse width measurement                                   | One-shot measurement: buffer disabled | 0   | 1    | 1    | 0    |
|                         |   | Repeated measurement: buffer enabled  | 1   | 1    | 1    | 0    |
|                         | Falling edge to falling edge<br>Interval measurement between falling edges                  | One-shot measurement: buffer disabled | 0   | 1    | 1    | 1    |
|                         |   | Repeated measurement: buffer enabled  | 1   | 1    | 1    | 1    |

The one-shot timer is selected initially after reset.

Note:

Select the operation mode always before the start of the timer.

## 25.5.4 Start and Stop of Timer/Pulse Width Measurement

Start/restart/stop/forcible stop of each operation are controlled by setting the PWCSR: bit15, bit14 (STRT, STOP bits).

### ■ Start and Stop of Timer/Pulse Width Measurement

Start/restart of the timer/pulse width measurement is initiated by setting the STRT bit to "0", while a forcible stop is initiated by setting the STOP bit to "0". However, this is only effective if the values are complementary to each other. When using instructions other than bit operation instructions (such as byte instructions), be sure to set only the bit combinations indicated in Table 25.5-3.

**Table 25.5-3 Function of STRT Bit and STOP Bit**

| STRT | STOP | Function                                       |
|------|------|--|
| 0    | 1    | Start/restart of timer/pulse width measurement |
| 1    | 0    | Forcible stop of timer/pulse width measurement |

When using bit operation instructions (the clear bit-operation instruction), special care is not required, since the hardware will ensure that only the combinations of values indicated in Table 25.5-3 are written.

### ■ Operation after Measurement Start

The operation of timer mode and pulse width measurement mode after measurement start are as follows.

#### ○ Timer mode

Count operation starts immediately.

#### ○ Pulse width measurement mode

Counting will not start until the measurement start edge is detected. After the measurement start edge is detected, the 16-bit up-count timer is cleared to "0000<sub>B</sub>" to start counting.

## ■ Restart

Restart is defined as a start operation (setting the STRT bit to "0") performed after entering timer/pulse width measurement mode.

Restart operates as follows depending on the mode:

### ○ One-shot timer mode

Restart has no effect.

### ○ Reload Timer Mode

Reload is performed and operation continues. If a restart occurs at the same time as an overflow, the overflow flag (OVIR) is set.

### ○ Pulse width measurement mode

In the measurement start edge wait state, restart has no effect on the operation. In measurement mode, counting stops and the measurement start edge wait state is entered. If, in this mode, measurement end edge detection and restart occur at the same time, the measurement end interrupt request flag (EDIR) is set. In repeated measurement mode, the result will be transferred to the PWCR.

## ■ Stop

In one-shot timer mode or one-shot measurement mode, no explicit stop operation needs to be performed because counting will automatically stop at a timer overflow or at measurement end. However, in other modes, the timer must be forcibly stopped. Moreover, providing an explicit stop operation allows the timer to stop before it would stop automatically.

### ○ Comparing and selection of two inputs

If a forcible stop is performed before the edge selected via PWC1 has been detected, the first measurement result after restart of measurement will contain an error. Be sure to perform a forcible stop only after the edge in PWC1 has been detected.

## ■ Confirmation of Operation State

The STRT/STOP bits, which have been explained above, operate as operation state indicator bits when read.

Table 25.5-4 shows the function of the operation state indicator bits.

**Table 25.5-4 Function of the Operation State Indicator Bits**

| STRT | STOP | Operation state   |
|------|------|---|
| 0    | 0    | Timer stop (except in measurement start edge wait state): Indicates that the timer has not started or that measurement has ended. |
| 1    | 1    | Timer counting is being performed or the system is in measurement start edge wait state   |

### Note:

The same value is read regardless of whether STRT or STOP bit is read. If these bits are read with read-modify-write instructions (such as bit operation instructions), "11<sub>B</sub>" is always returned. Do not use read-modify-write instructions for reading.

## 25.5.5 Timer Mode Operation

---

This section describes the device operation in timer mode.

---

### ■ Clearing the Timer

In the following cases, the 16-bit up-count timer is cleared to "0000<sub>H</sub>":

- At reset
- If, in pulse width measurement mode, a measurement start edge is detected and counting starts

### ■ One-shot Operation Mode

In one-shot operation mode, the timer count is incremented with every count clock after timer start. If an overflow occurs while incrementing from "FFFF<sub>H</sub>" to "0000<sub>H</sub>", the timer automatically stops. If the PWCR is set to any value before timer start, counting will start with that value. In this case, the value that was previously set will not be preserved, and the PWCR value will always indicate the current count value.

### ■ Reload Operation Mode

In reload operation mode, the timer is set to the reload value stored in PWCR, and the timer will be incremented with every count clock after the timer starts. If an overflow occurs while incrementing from "FFFF<sub>H</sub>" to "0000<sub>H</sub>", the reload value in PWCR will be loaded again into the timer (reload operation), and counting will be repeated from that value. The timer does not stop until it is forcibly stopped due to writing the PWCSR: STOP bit, or due to a reset. The value set in the PWCR before the timer starts is retained during counting as a reload value, and will be loaded into the timer if a start/restart or an overflow occurs. If the value that is set in the PWCR changes during counting, this new, changed reload value will be used at the next overflow or timer restart.

### ■ Timer Value and Reload Value

In one-shot operation mode, the PWCR is directly accessed by the up-count timer. Any value written to the PWCR will also be written to the timer as it is. When reading the PWCR while the timer is in progress, the value of the current timer count will be returned. If any value is set to the PWCR before the timer starts, counting will start from this value. In reload operation mode, accessing the count-up timer is prohibited. The PWCR operates as a reload register that stores a reload value. If a start/restart/overflow occurs, the timer is always set to the value stored in the PWCR. When reading the PWCR, the stored reload value will be read.

If the timer is placed in one-shot operation mode after the reload operation mode is forcibly canceled, the PWCR value and timer value are not specified. Be sure to always specify these values before using the timer.



## ■ Interrupt Generation Request

In timer mode, an interrupt request may be generated due to a timer overflow. If an overflow occurs due to incrementing the counter, the overflow flag is set and an overflow interrupt request is generated, if such requests are allowed.

## ■ Timer Interval

In one-shot operation mode, after PWCR is set to "0000<sub>H</sub>" to start the timer, an overflow is generated after the count "65536" is reached and counting will stop. The time that elapses from start to stop can be calculated with the following formula:

$$T_1 = (65536 - n_1) \times t$$

Where

$T_1$ : time from start to stop [ $\mu$ s]

$n_1$ : timer value stored in the PWCR at the start

$t$ : count clock interval [ $\mu$ s]

In reload operation mode, after PWCR is set to "0000<sub>H</sub>" to start the timer, an overflow will be generated every time the counter reaches "65536". The reload interval time can be calculated with the following formula:

$$T_R = (65536 - n_R) \times t$$

Where

$T_R$ : reload interval (overflow interval) [ $\mu$ s]

$n_R$ : reload value stored in PWCR

$t$ : count clock interval [ $\mu$ s]

## ■ Count Clock and Maximum Interval

In timer mode, the maximum interval is obtained when setting PWCR to the value "0000<sub>H</sub>".

Count clock interval and maximum timer interval for the case when the machine clock (represented as  $\phi$  hereafter) is 16 MHz are shown in Table 25.5-5.

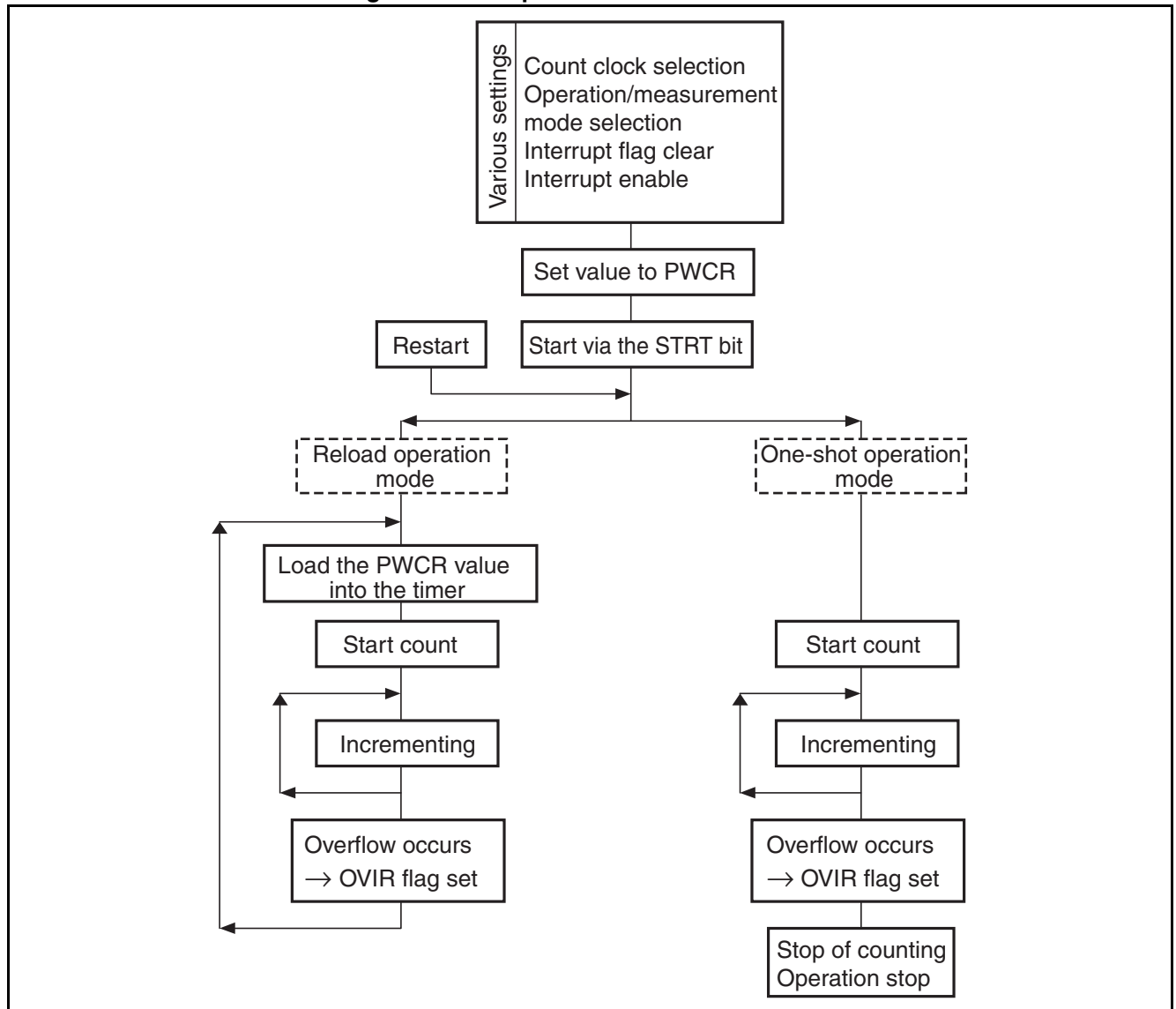
**Table 25.5-5 Count Clock and Interval**

| Count clock selection  | CSK1, 0 = 00: ( $\phi/4$ ) | CSK1, 0 = 00: ( $\phi/16$ ) | CSK1, 0 = 00: ( $\phi/32$ ) |
|------------------------|----------------------------|-----------------------------|-----------------------------|
| Count clock interval   | 0.25 $\mu$ s               | 1.0 $\mu$ s                 | 2.0 $\mu$ s                 |
| Maximum timer interval | 16.38 ms                   | 65.5 ms                     | 131.1 ms                    |

## ■ Timer Operation Flow

Figure 25.5-4 shows the operation flow of the timer.

**Figure 25.5-4 Operational Flow of the Timer**



## 25.5.6 Operation in Pulse Width Measurement Mode

---

This section describes operation in pulse width measurement mode.

---

### ■ One-shot Measurement and Repeated Measurement

There are two modes for pulse width measurement: a mode for one-time measurement and a mode for repeated measurement. The mode to use is selected via the PWCSR: S/C bit (Refer to Section "25.5.3 Selection of Count Clock and Operation Mode").

#### ○ One-Shot Measurement Mode

As soon as the first measurement end edge is detected, the timer counter will stop and the measurement end interrupt request flag (EDIR) in the PWCSR is set, causing a stop of measurement (however, if restart occurs at the same time, the device will wait for measurement to start again).

#### ○ Repeated measurement mode

If a measurement end edge is detected, the timer counter stops, the PWCSR's measurement end interrupt request flag (EDIR) is set, and counting stops until the next measurement start edge is detected. As soon as the next measurement start edge is detected, the timer is cleared to 0000<sub>H</sub>, and measurement starts again. At the end of measurement, the measurement result of the timer is transferred to the PWCR.

---

Note:

Be sure to change the measurement mode only while the timer is stopped.

---

### ■ Measurement Result Data

One-shot measurement mode and repeated measurement mode differ in handling of the measurement result, timer values, and PWCR functions. Measurement results in both modes are as follows:

#### ○ In one-shot measurement mode

- The timer value while measurement is in progress can be obtained by reading the PWCR.
- The end result data of measurement can be obtained by reading the PWCR after measurement has ended.

#### ○ In repeated measurement mode

- At the end of measurement, the timer measurement result is transferred to the PWCR.
- Reading the PWCR will return the immediately previously obtained measurement result, because the previous measurement result is kept during the measurement operation. The timer value while measurement is in progress cannot be read.

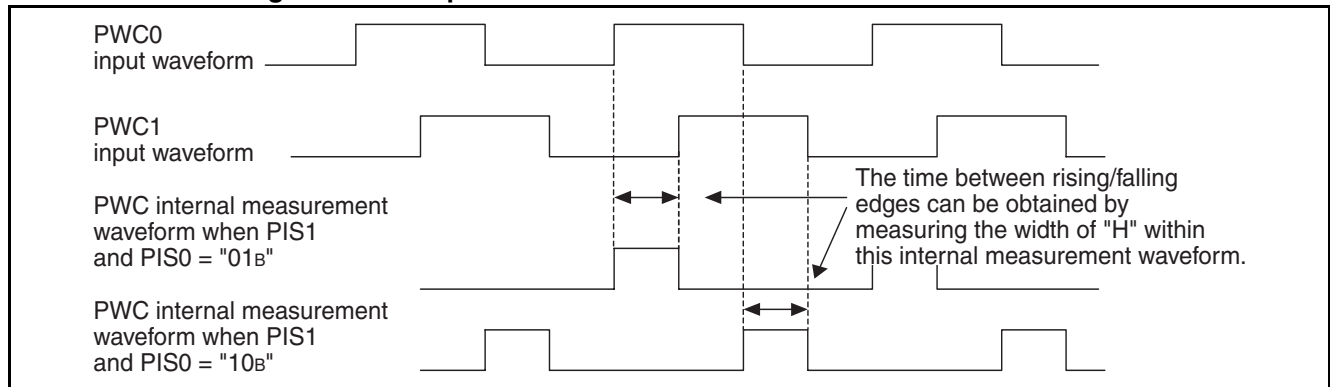
In this mode, if a measurement result is not read out before the next measurement operation ends, the measurement result will be overwritten with the next measurement result. In this case, an error flag (ERR) in the PWCSR is set. The error flag (ERR) is automatically cleared by reading the PWCR.

## ■ Selection of Input Pin

The PWC timer provides three channels, PWC0, PWC1, and PWC2, that are used as input signal pins for the pulse width counting. Each of these channels can therefore be used independently. Combining PWC0 and PWC1 with PIS1 and PIS0 in PWCSR0 enables the time between each input waveform's rising and falling edges to be measured. Note that the PWC register used in this case is PWC0.

Figure 25.5-5 shows the relationship between input waveform and internal measurement waveform.

**Figure 25.5-5 Input Waveform and Internal Measurement Waveform**



Notes:

- In comparing two inputs, start counting from PWC0 regardless of whether rising or falling edges are detected and stop with PWC1.
- If necessary, change the detection mode of rising and falling edges after measurement ends.

## ■ Measurement Mode and Counter Operation

The measurement mode is selected from among six types depending on which portions of the input pulse are to be measured. For a high precision measurement of a high frequency pulse width, a dedicated mode is provided to arbitrarily divide the input pulse for interval measurement. Table 25.5-6 shows a list of the measurement modes.

**Table 25.5-6 List of Measurement Modes (1/3)**

| Measurement mode            | MOD2 | MOD1 | MOD0 | Measurement items (W: pulse width to be measured)  |
|-----------------------------|------|------|------|--|
| "H" pulse width measurement | 1    | 0    | 1    | <p>Measures the width of the "H" pulse.<br/>           Start of counting (measurement): When rising edge is detected<br/>           End of counting (measurement): When falling edge is detected</p> |

Table 25.5-6 List of Measurement Modes (2/3)

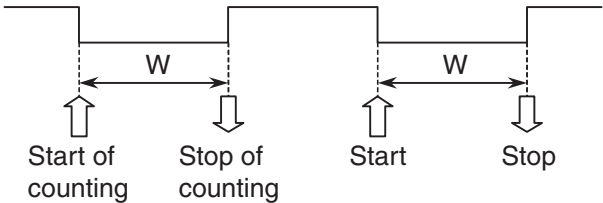
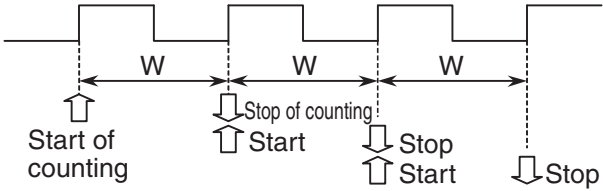
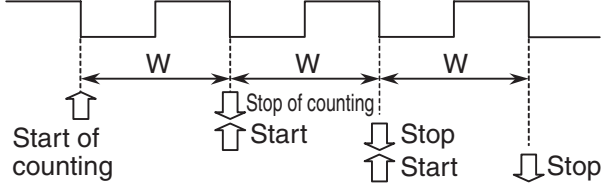
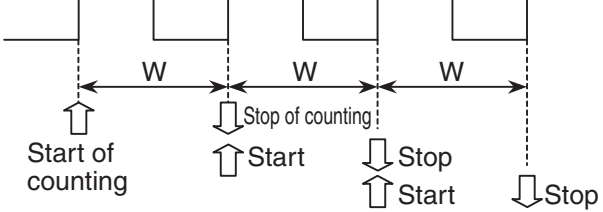
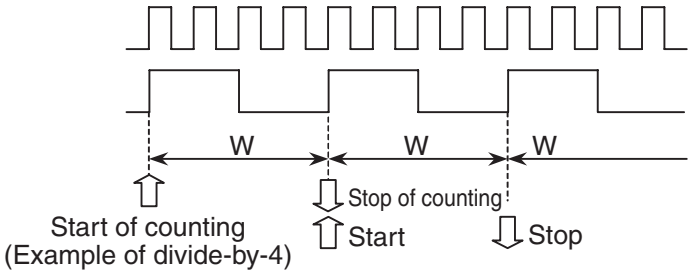
| Measurement mode                           | MOD2 | MOD1 | MOD0 | Measurement items (W: pulse width to be measured)   |
|--|------|------|------|---|
| "L" pulse width measurement                | 1    | 1    | 0    |  <p>Measures the width of the "L" pulse.<br/> Start of counting (measurement): When rising edge is detected<br/> End of counting (measurement): When falling edge is detected</p>             |
| Interval measurement between rising edges  | 1    | 0    | 0    |  <p>Measures the interval between rising edges.<br/> Start of counting (measurement): When rising edge is detected<br/> End of counting (measurement): When rising edge is detected</p>       |
| Interval measurement between falling edges | 1    | 1    | 1    |  <p>Measures the interval between falling edges.<br/> Start of counting (measurement): When falling edge is detected<br/> End of counting (measurement): When falling edge is detected</p>  |
| Pulse width measurement for all edges      | 0    | 1    | 0    |  <p>Measures the pulse width between repeatedly input edges<br/> Start of counting (measurement): When an edge is detected<br/> End of counting (measurement): When an edge is detected</p> |

Table 25.5-6 List of Measurement Modes (3/3)

| Measurement mode            | MOD2 | MOD1 | MOD0 | Measurement items (W: pulse width to be measured)   |
|-----------------------------|------|------|------|---|
| Divide interval measurement | 0    | 1    | 1    |  <p>Measures an interval by dividing the input pulse with a divide ratio selected in the divide ratio control register DIVR.<br/> Start of counting (measurement): When a rising edge is detected soon after the start<br/> End of counting (measurement): When an interval ends after dividing</p> |

In any mode, counting is not performed by the timer during the time from the start of counting to the input of the count start edge. When the count start edge is input, the timer is cleared to 0000<sub>H</sub>, and the timer counts every count clock until a count end edge is input. When a count end edge is entered, the following operations are executed:

- Set the PWCSR's measurement end interrupt request flag (EDIR)
- Stop the timer count operation (except in cases when the edge is input for a restart)
- In repeated measurement mode, the timer value (measurement result) is transferred to the PWCR, and the count operation is suspended until the next measurement start edge is entered.
- In one-shot measurement mode, the measurement will end (unless restart is entered at the same time)

If, in repeated measurement mode, pulse width measurement between all edges or divide measurement is performed, the end edge defines the next edge for measurement.

### ■ Minimum Input Pulse Width

The following restriction applies to pulses that are input to the pulse width measurement input pins (PWC2 to PWC0):

- The pulse width must be four machine cycles or more (0.25 μs for a 16 MHz machine clock).

### ■ Pulse Width/Interval Calculation Method

The pulse width/interval to be measured can be calculated with the following formula:

$$T_W = n \times t / D_{IV} [\mu s]$$

Where

$T_W$ : pulse width/interval to be measured [μs]

$n$ : measurement result data in PWCR

$t$ : count clock interval [μs]

$D_{IV}$ : divide ratio selected via the divide ratio control register DIVR

(use 1, except in divide frequency measurement mode)

## ■ Range for Counting the Pulse Width/Interval

Depending on the selected combination of count clock and divide ratio of the input divider, the allowed pulse width/interval range for measurement will vary.

Table 25.5-7 shows a list of measurement ranges when a machine clock ( $\phi$ , hereafter) of 16 MHz is applied.

**Table 25.5-7 List of Pulse Width Measurement Ranges**

| Divide ratio  | DIV1   | DIV0   | CKS1,0=00 ( $\phi/4$ )                        | CKS1,0=01 ( $\phi/16$ )                       | CKS1,0=10 ( $\phi/32$ )                  |
|---------------|--------|--------|---|---|--|
| No division   | $\phi$ | $\phi$ | 0.25 $\mu$ s to 16.38 ms<br>[0.25 $\mu$ s]    | 0.25 $\mu$ s to 65.5 ms<br>[1.6 $\mu$ s]      | 0.25 $\mu$ s to 131 ms<br>[3.2 $\mu$ s]  |
| Divide-by-4   | 0      | 0      | 0.25 $\mu$ s to 4.1 ms<br>[6.25 $\mu$ s]      | 0.25 $\mu$ s to 16.38 ms<br>[0.4 $\mu$ s]     | 0.25 $\mu$ s to 52.4 ms<br>[0.8 $\mu$ s] |
| Divide-by-16  | 0      | 1      | 0.25 $\mu$ s to 1.024 ms<br>[15.6 ns]         | 0.25 $\mu$ s to 4.1 ms<br>[1.6 $\mu$ s]       | 0.25 $\mu$ s to 13.1 ms<br>[0.2 $\mu$ s] |
| Divide-by-64  | 1      | 0      | 0.25 $\mu$ s to 256 $\mu$ s<br>[3.91 $\mu$ s] | 0.25 $\mu$ s to 1.024 ms<br>[25 $\mu$ s]      | 0.25 $\mu$ s to 3.27 ms<br>[50 ns]       |
| Divide-by-256 | 1      | 1      | 0.25 $\mu$ s to 64 $\mu$ s<br>[0.98 $\mu$ s]  | 0.25 $\mu$ s to 256 $\mu$ s<br>[6.25 $\mu$ s] | 0.25 $\mu$ s to 817 $\mu$ s<br>[12.5 ns] |

Note:

The value in [ ] indicates the resolution per bit.

## ■ Generation of Interrupt Requests

In pulse width measurement mode, two types of interrupt requests can be generated, as listed below:

### ○ Interrupt requests due to timer overflows

If an overflow occurs due to incrementing in measurement mode, an overflow flag is set. If overflow interrupt requests are enabled, an interrupt request is generated.

### ○ Interrupt requests due to the end of measurement

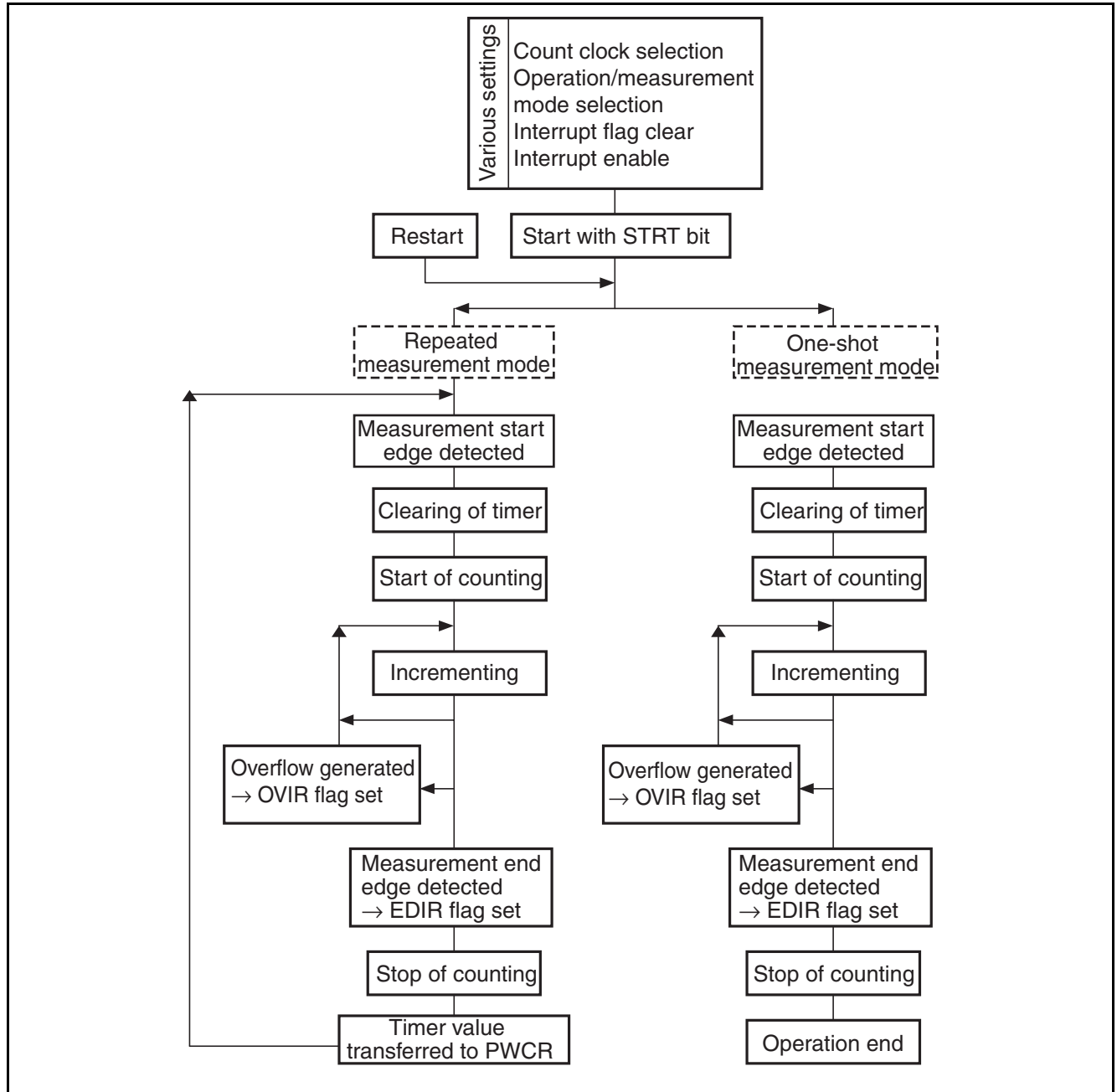
If measurement end interrupt requests are enabled, an interrupt request is generated when a measurement end edge is detected and the PWCSR's measurement end flag (EDIR) is set.

The measurement end flag EDIR is automatically cleared when the measurement result is read out from the PWCR.

## ■ Operational Flow of Pulse Width Measurement

Figure 25.5-6 shows the operational flow of pulse width measurement.

**Figure 25.5-6 Operational Flow of Pulse Width Measurement**





## 25.6 Notes on PWC Timer Usage

---

This section provides notes on using the PWC timer.

---

### ■ Notes on PWC Timer Usage

#### ○ Notes on rewriting the register

Rewriting the bits in the PWCSR is prohibited. Rewrite the register either before the timer is started or after the timer stops.

- [bit7, bit6] CKS1, CKS0 (clock selection)
- [bit5, bit4] PIS1, PIS0 (input signal selection)
- [bit3] CKS1, CKS0 (clock selection)
- [bit2, bit1, bit0] MOD2, MOD1, MOD0 (operation mode/measurement edge selection)
- Rewriting DIVR during timer operation is prohibited. Rewrite it either before start or after stop of timer operation.

#### ○ Handling the measurement end flag in timer mode

The value of the PWCSR's measurement end interrupt request flag (EDIR) is not significant in timer mode. Consequently, set the PWCSR's measurement end interrupt enable bit (EDIE) to "0" if it is to be used in timer mode.

#### ○ Handling of STRT and STOP bits in the PWCSR

Note that the meaning of these two bits is different depending on whether a write or read operation is performed (Refer to Section "25.3.1 PWC Control/Status Register (PWCSR0 to PWCSR2)"). The read value returned in read-modify-write instructions is always 11<sub>B</sub> irrespective of the value for these bits. Be sure, therefore, not to use bit operation instructions for reading the operation state (these bits will always return "operation in progress"). However, bit operation instructions (such as bit clear) can be used for writing the STRT and STOP bits to start or stop the timer.

#### ○ Timer Clear

In pulse width measurement mode, the timer is cleared at the measurement start edge. The timer data will therefore be invalid before the start of measurement.

#### ○ PWCR and timer values when the mode changes

- If the timer is forcibly stopped and set to one-shot timer mode from reload timer mode, the timer value and the value stored in the PWCR will become unspecified. Be sure to define these values before using the timer.
- If the timer is used in one-shot timer mode, the PWCR value is unspecified. Use the timer only after defining the value.
- To switch the mode from pulse width measurement mode to timer mode, specify the PWCR value again before starting the timer.

### ○ **Minimum pulse width**

The following restrictions apply to pulses that can be input to the pulse width measurement input pin.

- Minimum pulse width: machine clock divided-by-2 (0.25  $\mu$ s or more for 16 MHz machine clock)
- Minimum input frequency: machine clock divided-by-4 (4 MHz or less for 16 MHz machine clock)

If a pulse with lower width or higher frequency than specified above is input, correct operation cannot be assured. In the event that the input signal might contain noise, eliminate the noise using a filter externally from the chip before inputting the pulse.

### ○ **Divide frequency measurement mode**

In divide interval measurement mode, which is one of the pulse width measurement modes, the input pulse is divided. Note that the pulse width as calculated from the measurement results will therefore be an averaged value.

### ○ **Handling of clock selection bit**

Do not set [bit7, bit6] of the PWCR (clock selection bits CKS1 and CKS0) to "11<sub>B</sub>".

### ○ **Handling of reserved bit**

[bit8] in the PWCR is reserved. If necessary, clear this bit by writing "0".

### ○ **Restart during operation**

Restarting after the count operation has been started may cause the following events depending on the timing of restart:

- If, in reload timer mode, a restart occurs at the same time as an overflow:  
The restart is performed, but the overflow flag (OVIR) is set.
- If, in pulse width one-shot measurement mode, a restart occurs at the same time as detection of a measurement end edge:  
Restart is performed, and the measurement end interrupt request flag (EDIR) is set.
- If, in pulse width repeated measurement mode, a restart occurs at the same time as measurement end edge detection:  
Restart is performed, but the measurement end interrupt request flag (EDIR) is set and the measurement result is transferred to the PWCR.

### ○ **If the PWC timer is used in "H" pulse or "L" pulse width measurement mode as the repeated measurement mode**

If, after the end of pulse width measurement, the pulse width measurement start wait state is entered, the timer to be stopped might continue with the overflow flag (OVIR) being set before the pulse width measurement starts. In other words, even if no overflow occurs during the next pulse width measurement phase, the overflow flag may be set. Therefore, do not use the overflow flag when you use the PWC timer in "H" pulse or "L" pulse width measurement mode as repeated measurement mode.



## CHAPTER 26 $\mu$ PG TIMER (ONLY MB90485 SERIES)

---

**This chapter provides an overview, explains the configuration of the  $\mu$ PG timer and its timing chart, the configuration and functions of its registers.**

---

26.1 Overview and Configuration of  $\mu$ PG Timer

26.2 Configuration and Functions of  $\mu$ PG Timer Registers

26.3 Timing Chart of  $\mu$ PG Timer

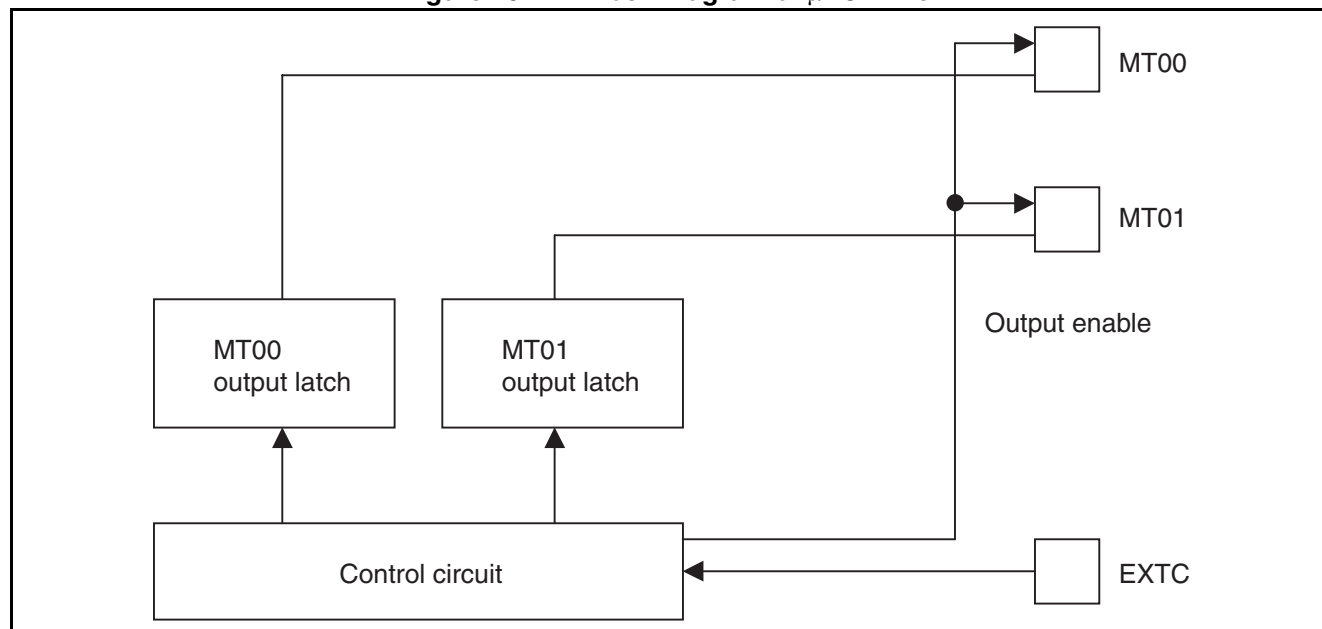
## 26.1 Overview and Configuration of $\mu$ PG Timer

The  $\mu$ PG timer is used to output pulses based on external input.

### ■ Block Diagram of $\mu$ PG Timer

Figure 26.1-1 shows a block diagram of the  $\mu$ PG timer.

Figure 26.1-1 Block Diagram of  $\mu$ PG Timer



### ■ Pin Related to $\mu$ PG Timer

The  $\mu$ PG timer functions as the input port when the EXTC pin is used and the output port when the MT00, MT01 pins are used. The EXTC pin functions as the general-purpose I/O port (P45/EXTC) and input pin of  $\mu$ PG timer. The MT00, MT01 pins function as the general-purpose I/O port (P43/MT00, P44/MT01) and output pin of  $\mu$ PG timer.

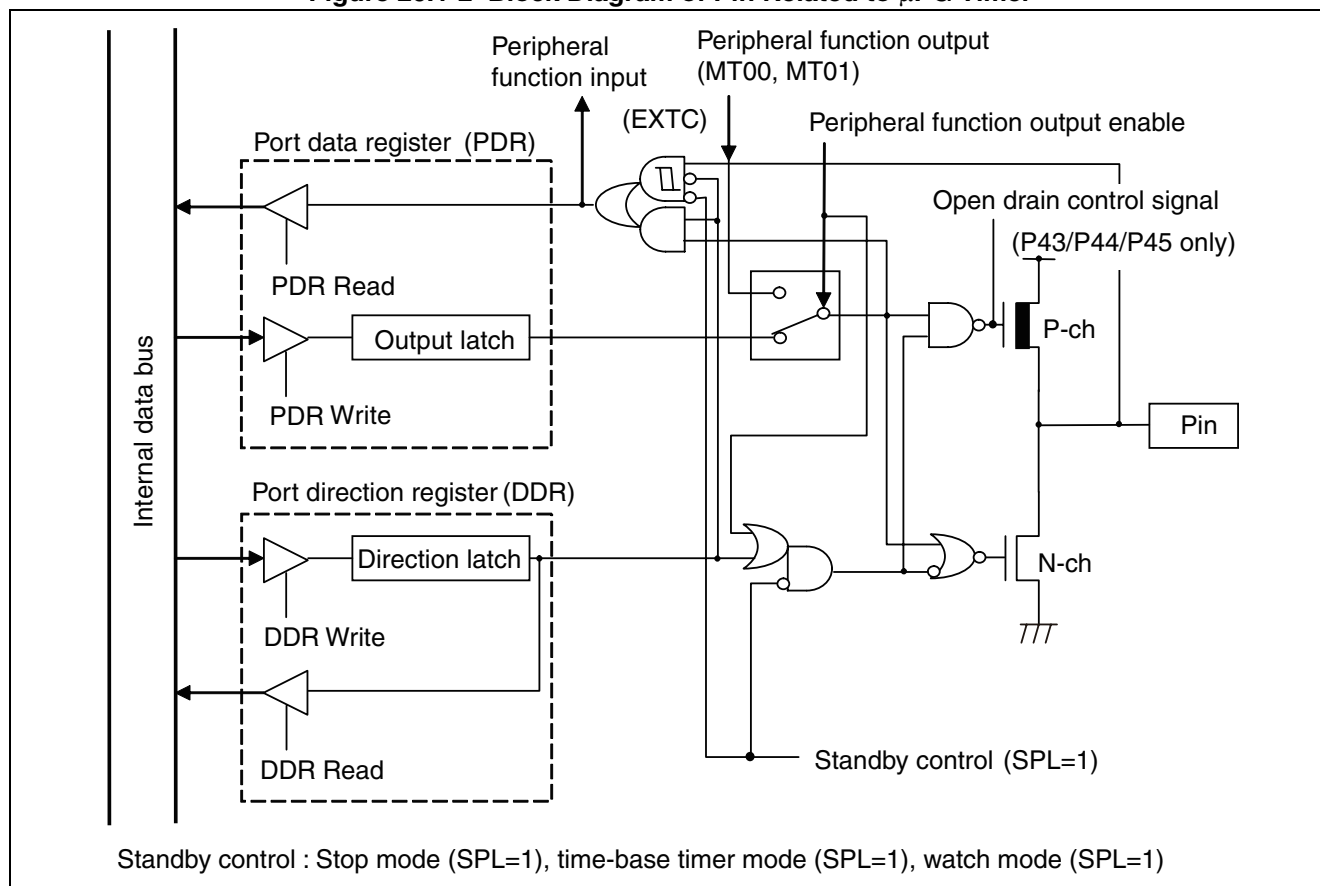
- Setting when using as EXTC pin

When using as the EXTC pin by the  $\mu$ PG timer, the P45/EXTC pin should be set to the input port by the port direction register (DDR4 bit5  $\rightarrow$  "0").

- Setting when using as MT00/MT01 pins

When MT00/MT01 are used as output by the  $\mu$ PG timer, be sure to set to the pulse output enable (PE0/PE1 bit5, 6  $\rightarrow$  "1") by the  $\mu$ PG control/status register (PGCSR).

## ■ Block Diagram of Pin Related to $\mu$ PG Timer

Figure 26.1-2 Block Diagram of Pin Related to  $\mu$ PG Timer

## 26.2 Configuration and Functions of $\mu$ PG Timer Registers

This section describes the configuration of the registers used in the  $\mu$ PG timer and their functions.

### ■ $\mu$ PG Control/Status Register (PGCSR)

The bit configuration of the  $\mu$ PG control/status register (PGCSR) is shown below.

|                     |       |       |       |       |       |     |     |     |                                  |
|---------------------|-------|-------|-------|-------|-------|-----|-----|-----|----------------------------------|
|                     | 7     | 6     | 5     | 4     | 3     | 2   | 1   | 0   | PGCSR                            |
| 00008E <sub>H</sub> | PEN0  | PE1   | PE0   | PMT1  | PMT0  | -   | -   | -   | $\mu$ PG control status register |
|                     | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (-) | (-) | (-) | Read/write                       |
|                     | (0)   | (0)   | (0)   | (0)   | (0)   | (-) | (-) | (-) | Initial value                    |

The functions of the bits in the  $\mu$ PG control/status register (PGCSR) are listed below.

#### [bit7] PEN0 (operation enable)

This bit is used to enable  $\mu$ PG timer operation.

| PEN0 | Function                                   |
|------|--|
| 0    | Stop (retaining "L" level) (initial value) |
| 1    | PG operation allowed                       |

This bit is initialized at reset.

#### [bit6, bit5] PE1, PE0 (output enable)

These bits are used to control the pulse output external pin.

| PE1 | PE0 | Operation control function   |
|-----|-----|--|
| 0   | 0   | General-purpose port pin (pulse output prohibited) (initial value) |
| 0   | 1   | MT00 pulse output pin only (output allowed)                        |
| 1   | 0   | MT01 pulse output pin only (output allowed)                        |
| 1   | 1   | MT00,MT01 pulse output pin (output allowed)                        |

These bits are initialized to "00<sub>B</sub>" at reset.

**[bit4, bit3] PMT1,PMT0 (invert output)**

These bits are used to invert the output of each pulse.

| PMT1 | PMT0 | Operation control function            |
|------|------|---------------------------------------|
| 0    | 0    | Waveform at the start (initial value) |
| 0    | 1    | Only MT00 inverted                    |
| 1    | 0    | Only MT01 inverted                    |
| 1    | 1    | MT00 and MT01 inverted                |

These bits are initialized to "00<sub>B</sub>" at reset.

**[bit2, bit1, bit0] Undefined bits**

These bits are undefined and not used. In ordinary cases, set them to "000<sub>B</sub>".



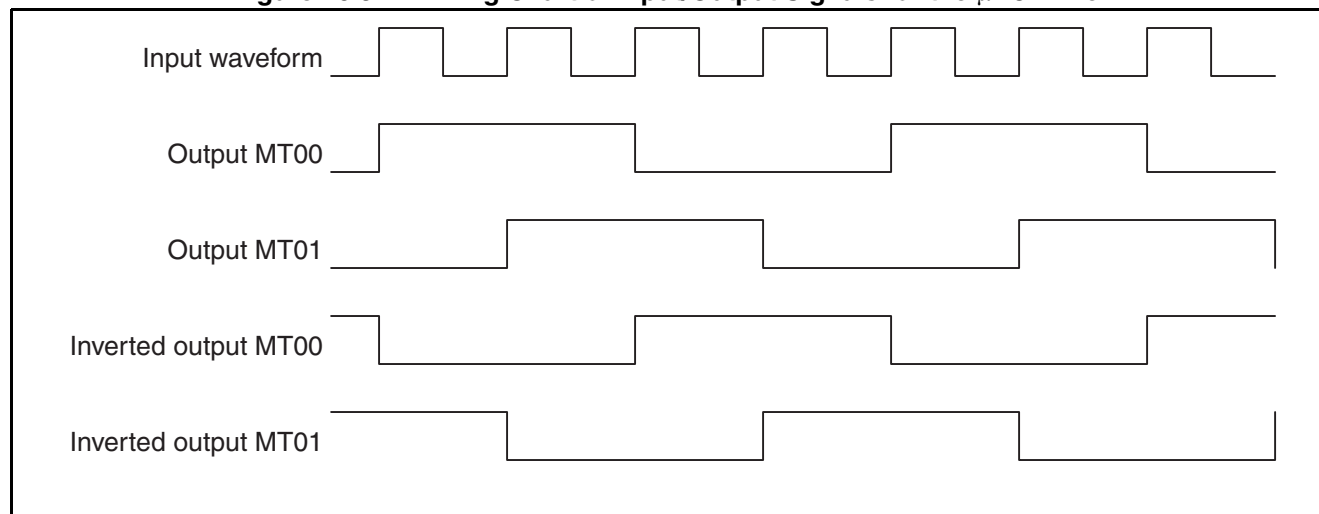
## 26.3 Timing Chart of $\mu$ PG Timer

This section shows a timing chart and timing for the  $\mu$ PG timer.

### ■ Timing Chart of $\mu$ PG Timer

Figure 26.3-1 shows the timing chart of input and output signals for the  $\mu$ PG timer.

**Figure 26.3-1 Timing Chart of Input/Output Signals for the  $\mu$ PG Timer**



### ■ Notes on Timing

- Figure 26.3-1 shows the output waveform against the input waveform. The duty ratio is fixed at 50%.
- In sync with the first rising pulse after start, the output starts from the 2nd rising pulse.
- Two outputs for one input waveform are only issued using inversion control via the program.
- For the input pulse (waveform at the EXTC pin), use an interval 10 times larger than one pulse of the internal clock (machine clock).
- To write "11<sub>B</sub>" for the PE0 and PE1 of the  $\mu$ PG control/status register (allow output for all pins), keep the value at the input pin (EXTC) constant. Otherwise, if a pulse is input to the input pin (EXTC) when pin output is allowed, the width of the first output pulse may become shorter than that of the input pulse.

## CHAPTER 27 I<sup>2</sup>C INTERFACE (ONLY MB90485 SERIES)

---

**This chapter provides an overview, explains configuration, interrupt, and operation of the I<sup>2</sup>C interface, the configuration and functions of its registers.**

---

27.1 Overview of I<sup>2</sup>C Interface

27.2 Configuration of I<sup>2</sup>C Interface

27.3 Configuration and Functions of I<sup>2</sup>C Interface Registers

27.4 Interrupt of I<sup>2</sup>C Interface

27.5 I<sup>2</sup>C Interface Operation

## 27.1 Overview of I<sup>2</sup>C Interface

---

The I<sup>2</sup>C interface is a serial I/O port supporting the Inter IC BUS, allowing master/slave devices to operate over the I<sup>2</sup>C bus.

---

### ■ I<sup>2</sup>C Interface Function

The I<sup>2</sup>C interface has the following functions.

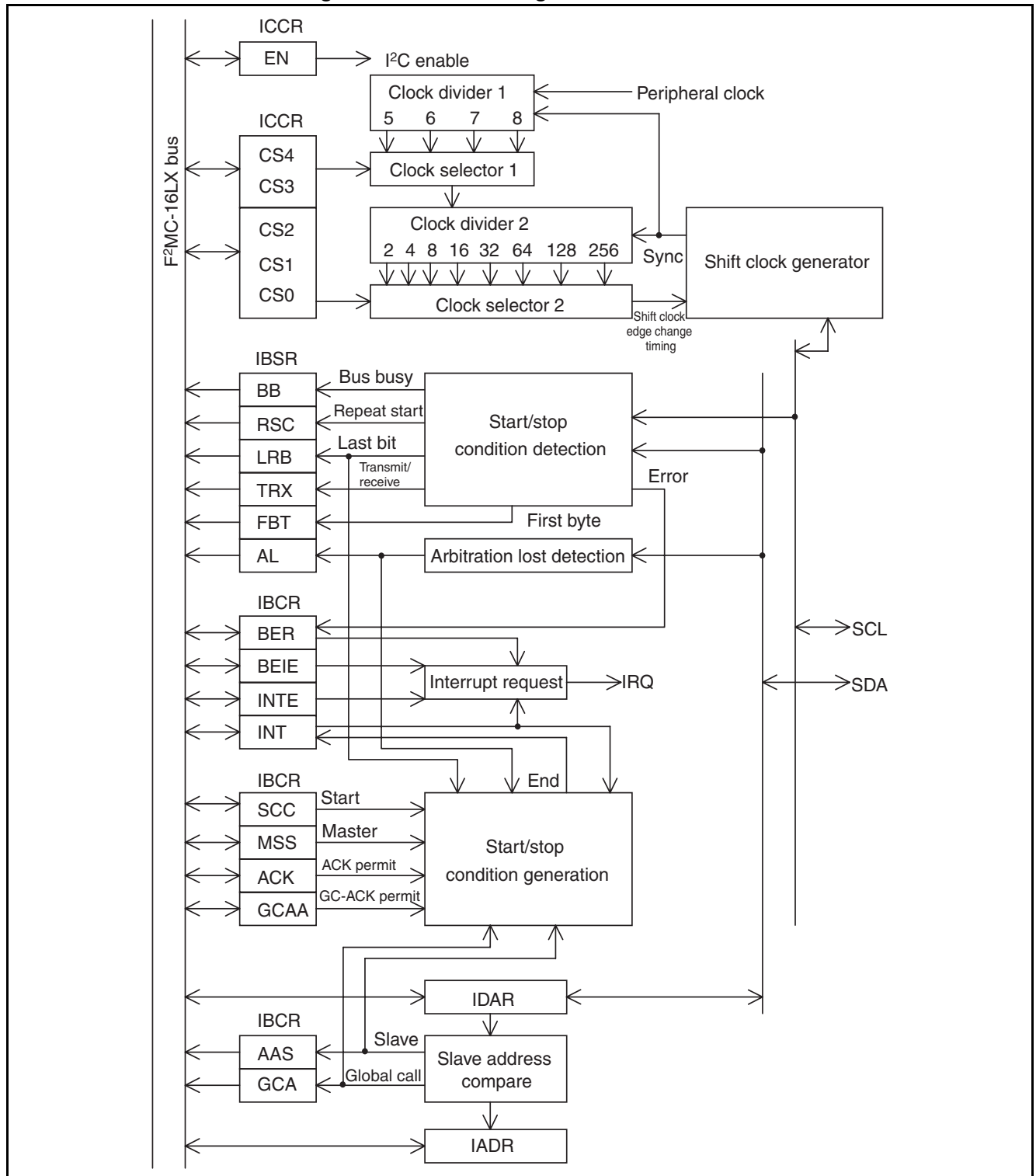
- Master/slave transmit/receive function
- Arbitration function
- Clock synchronization function
- Slave address/general call address detection function
- Transfer direction detection function
- Repeated issuance of start conditions and start condition detection function
- Bus error detection function

## 27.2 Configuration of I<sup>2</sup>C Interface

### ■ Block Diagram of the I<sup>2</sup>C Interface

Figure 27.2-1 shows a block diagram of the I<sup>2</sup>C interface.

**Figure 27.2-1 Block Diagram of I<sup>2</sup>C Interface**



## ■ Pin Related to I<sup>2</sup>C Interface

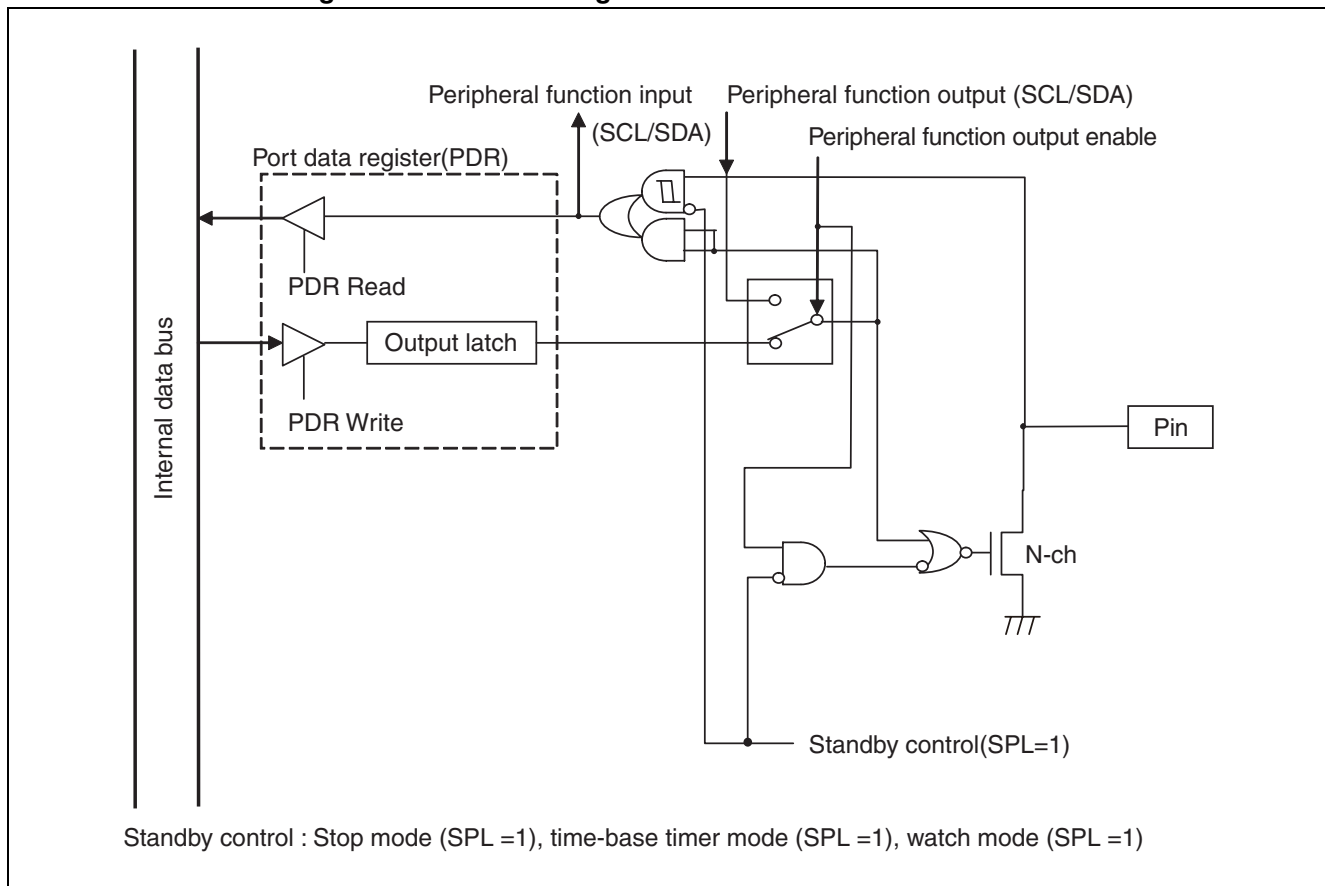
The pin related to the I<sup>2</sup>C interface has the SDA data input/output pin and SCL clock input/output pin. The SCL/SDA pins function as the general-purpose I/O port (P76/SCL, P77/SDA) and I<sup>2</sup>C interface. When using as the general-purpose I/O port, data is always enabled as port because P76, P77 of the port direction register (DDR) do not exist (only MB90485 series). In MB90485 series, the P76/SCL, P77/SDA pins are N-ch open-drain pin.

### ● Setting when using as SCL/SDA pin

When using as the I<sup>2</sup>C interface, the port data register should be set (PDR7 :bit14, 15 → "1"). Also, when using as the input port, the pull-up resistor must be added to the external pin.

## ■ Block Diagram of Pin Related to I<sup>2</sup>C Interface

Figure 27.2-2 Block Diagram of Pin Related to I<sup>2</sup>C Interface



## 27.3 Configuration and Functions of I<sup>2</sup>C Interface Registers

This section describes the configuration and functions of the I<sup>2</sup>C interface registers.

### ■ List of I<sup>2</sup>C Interface Registers

#### ○ Bus status register (IBSR)

|   |  |     |     |     |     |     |     |     |     |              |
|---|--|-----|-----|-----|-----|-----|-----|-----|-----|--------------|
| Bus status register<br>Address: 000088 <sub>H</sub> |  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | ↵ Bit number |
|   |  | BB  | RSC | AL  | LRB | TRX | AAS | GCA | FBT | IBSR         |
| Read/Write ↵  |  | (R) | (R) | (R) | (R) | (R) | (R) | (R) | (R) |              |
| Initial value ↵                                     |  | (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |              |

#### ○ Bus control register (IBCR)

|  |  |       |       |       |       |       |       |       |       |              |
|--|--|-------|-------|-------|-------|-------|-------|-------|-------|--------------|
| Bus control register<br>Address: 000089 <sub>H</sub> |  | 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     | ↵ Bit number |
|  |  | BER   | BEIE  | SCC   | MSS   | ACK   | GCAA  | INTE  | INT   | IBCR         |
| 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)   |              |

#### ○ Clock control register (ICCR)

|  |  |     |     |       |       |       |       |       |       |              |
|--|--|-----|-----|-------|-------|-------|-------|-------|-------|--------------|
| Address: 00008A <sub>H</sub><br>Clock control register |  | 7   | 6   | 5     | 4     | 3     | 2     | 1     | 0     | ↵ Bit number |
|  |  | -   | -   | EN    | CS4   | CS3   | CS2   | CS1   | CS0   | ICCR         |
| Read/Write ↵   |  | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) |              |
| Initial value ↵  |  | (-) | (-) | (0)   | (X)   | (X)   | (X)   | (X)   | (X)   |              |

#### ○ Address register (IADR)

|  |  |     |       |       |       |       |       |       |       |              |
|--|--|-----|-------|-------|-------|-------|-------|-------|-------|--------------|
| Address register<br>Address: 00008B <sub>H</sub> |  | 15  | 14    | 13    | 12    | 11    | 10    | 9     | 8     | ↵ Bit number |
|  |  | -   | A6    | A5    | A4    | A3    | A2    | A1    | A0    | IADR         |
| Read/Write ↵                                     |  | (-) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) |              |
| Initial value ↵                                  |  | (-) | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   |              |

#### ○ Data register (IDAR)

|   |  |       |       |       |       |       |       |       |       |              |
|---|--|-------|-------|-------|-------|-------|-------|-------|-------|--------------|
| Data register<br>Address: 00008C <sub>H</sub> |  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | ↵ Bit number |
|   |  | D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    | IDAR         |
| 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)   |              |

### 27.3.1 Bus Status Register (IBSR)

This section describes the configuration and functions of the bus status register (IBSR).

#### ■ Bus Status Register (IBSR)

The diagram below shows the bit configuration of the bus status register (IBSR).

|   |     |     |     |     |     |     |     |     |              |
|---|-----|-----|-----|-----|-----|-----|-----|-----|--------------|
| Bus status register<br>Address: 000088 <sub>H</sub> | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | ↩ Bit number |
|   | BB  | RSC | AL  | LRB | TRX | AAS | GCA | FBT | IBSR         |
| Read/Write ↗  | (R) | (R) | (R) | (R) | (R) | (R) | (R) | (R) |              |
| Initial value ↗                                     | (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |              |

The functions of bits in the bus status register (IBSR) are described below.

#### [bit7] BB: Bus Busy

This bit is used to indicate the I<sup>2</sup>C bus status.

|   |  |
|---|--|
| 0 | Stop condition is detected.                |
| 1 | Start condition is detected (bus is used). |

#### [bit6] RSC: Repeated Start Condition

This bit is used to detect a repeated start condition.

|   |   |
|---|---|
| 0 | Repeated start condition is not detected.           |
| 1 | Start condition is detected again when bus is used. |

This bit is cleared by setting the INT bit to "0" for addressing in a mode other than slave mode if a start condition is detected in bus idle state or if a stop condition is detected.

#### [bit5] AL: Arbitration Lost

This bit is used to detect the arbitration lost state.

|   |  |
|---|--|
| 0 | Arbitration lost is not detected.  |
| 1 | Arbitration lost is generated in master transfer mode, or the MSS bit is set to "1" while another system is using the bus. |

Cleared if the INT bit is set to "0".

#### [bit4] LRB: Last Received bit

This bit is an acknowledge storage bit used to store an acknowledgement from the reception side.

|   |                                |
|---|--------------------------------|
| 0 | Reception is acknowledged.     |
| 1 | Reception is not acknowledged. |

This bit is cleared if a start or stop condition is detected.

## [bit3] TRX: Transfer/Receive

This bit is used to indicate transmission or reception for data transfer.

|   |              |
|---|--------------|
| 0 | Reception    |
| 1 | Transmission |

## [bit2] AAS: Addressed As Slave

This bit is used to detect the addressing mode.

|   |   |
|---|---|
| 0 | Addressing was performed in a mode other than slave mode. |
| 1 | Addressing was performed in slave mode.                   |

This bit is cleared if a start or stop condition is detected.

## [bit1] GCA: General Call Address

This bit is used to detect the general call address (00<sub>H</sub>).

|   |  |
|---|--|
| 0 | No general call address is received in slave mode. |
| 1 | A general call address is received in slave mode.  |

This bit is cleared if a start or stop condition is detected.

## [bit0] FBT: First Byte Transfer

This bit is used to detect the first byte (address data).

|   |   |
|---|---|
| 0 | The data received is not first byte.            |
| 1 | The data received is first byte (address data). |

This bit is cleared if the INT bit is set to "0" or addressing was performed in a mode other than slave mode, even though the bit was set to "1" because of detection for a start condition.

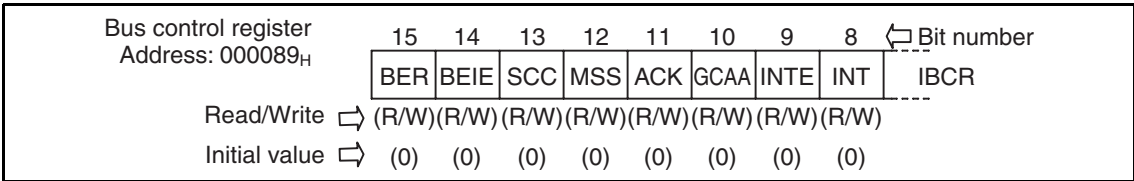


### 27.3.2 Bus Control Register (IBCR)

This section describes the configuration and functions of the bus control register (IBCR).

■ Bus Control Register (IBCR)

The diagram below shows the bit configuration of the bus control register (IBCR).



The functions of bits in the bus control register (IBCR) are as follows.

**[bit15] BER: Bus Error**

This bit is a bus error interrupt request flag. The function of this bit in write and read operations is different.

(During writing)

|   |  |
|---|--|
| 0 | Bus error interrupt request flag is cleared. |
| 1 | Not applicable.                              |

(During reading)

|   |  |
|---|--|
| 0 | No bus error was detected.   |
| 1 | An illegal start or stop condition was detected in data transfer mode. |

If this bit is set, the EN bit of the ICCR register is cleared and the I<sup>2</sup>C interface enters the stop state with the data transfer interrupted.

**[bit14] BEIE: Bus Error Interrupt Enable**

This bit is used to enable bus error interrupts.

|   |                                  |
|---|----------------------------------|
| 0 | Bus error interrupts prohibited. |
| 1 | Bus error interrupts allowed.    |

If, with this bit set to "1", the BER bit is set to "1", an interrupt is generated.

**[bit13] SCC: Start Condition Continue**

This bit is used to generate a start condition.

(During writing)

|   |   |
|---|---|
| 0 | Not applicable.   |
| 1 | Start condition is generated again in master transfer mode. |

Read operations always return "0" for this bit.

**[bit12] MSS: Master Slave Select**

This bit is used to select between master mode and slave mode.

|   |   |
|---|---|
| 0 | After the stop condition is generated and transferred, the device enters slave mode.  |
| 1 | The device enters master mode, the start condition is generated, and transfer starts. |

This bit is cleared if arbitration lost is detected in master transfer mode. The device then enters slave mode.

**Note:**

When using on the following condition, the transmission of the general call address is prohibited because it cannot receive it as a slave.

- When other LSI that becomes a mastering mode besides this LSI exists on the bus and this LSI transmits the general call address as a master and the arbitration lost is generated since the second byte.

**[bit11] ACK: ACKnowledge**

This bit is used to allow acknowledge generation when data is received.

|   |                           |
|---|---------------------------|
| 0 | No acknowledge generated. |
| 1 | Acknowledge generated.    |

This bit is invalid if address data is received in slave mode.

**[bit10] GCAA: General Call Address Acknowledge**

This bit is used to enable acknowledge generation when a general call address (00<sub>H</sub>) is received.

|   |                           |
|---|---------------------------|
| 0 | No acknowledge generated. |
| 1 | Acknowledge generated.    |

**[bit9] INTE: INTerrupt Enable**

This bit is used to enable interrupts.

|   |                        |
|---|------------------------|
| 0 | Interrupts prohibited. |
| 1 | Interrupts allowed.    |

If this bit is set to "1" when the INT bit is set to "1", an interrupt is generated.

**[bit8] INT: INTerrupt**

This bit is used as a transfer end interrupt request flag.

(During writing)

|   |   |
|---|---|
| 0 | Clears the transfer end interrupt request flag. |
| 1 | Not applicable.                                 |

CHAPTER 27 I<sup>2</sup>C INTERFACE (ONLY MB90485 SERIES)

(During reading)

|   |   |
|---|---|
| 0 | Transfer has not ended.   |
| 1 | <div>This bit is set if the following conditions are met when one byte including an acknowledge bit is transferred:<ul style="list-style-type: none"><li>• Byte transferred in bus master transfer</li><li>• Byte transferred in slave mode with addressing</li><li>• General call address is received</li><li>• Arbitration lost occurs</li><li>• Attempt to generate a start condition while other systems use the bus.</li></ul></div> |

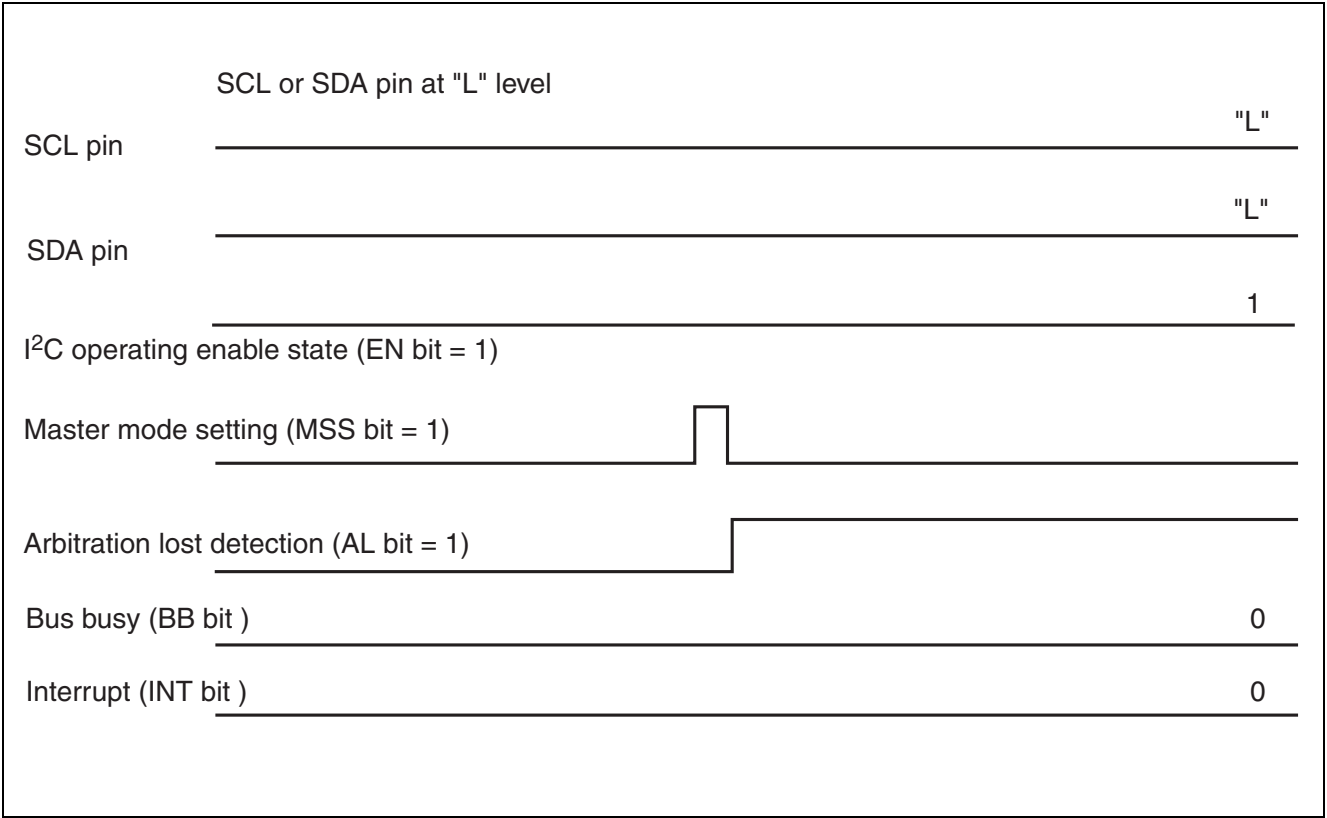
If this bit is "1", the SCL line is kept at the "L" level. This bit is cleared by writing "0" before the SCL line is opened for transfer of the next byte. Alternatively, this bit is reset to "0" if, in master mode, a start or stop condition is generated.

Note:

When an instruction which generates a start condition is executed (the MSS bit is set to 1) at the timing shown in Figure 27.3-1 and Figure 27.3-2, arbitration lost detection (AL bit = 1) prevents an interrupt (INT bit = 1) from being generated.

- Condition 1 in which an interrupt (INT bit = 1) upon detection of " AL bit = 1 " does not occurs  
When an instruction which generates a start condition is executed (setting the MSS bit in the IBCR register to 1) with no start condition detected (BB bit = 0) and with the SDA or SCL pin at the " L " level.

Figure 27.3-1 Diagram of Timing at which an Interrupt upon Detection of " AL Bit = 1 " does not Occur

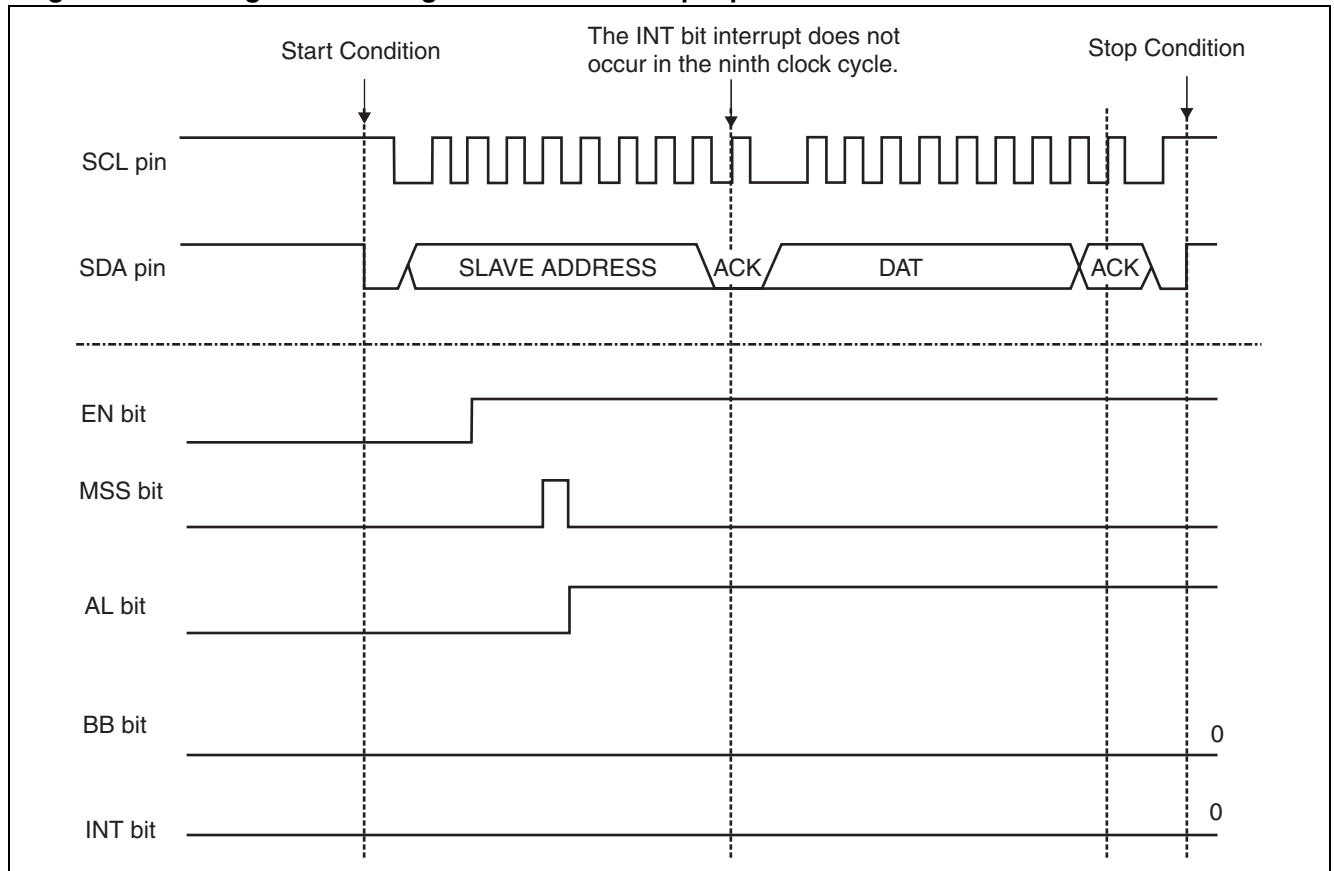


- Condition 2 in which an interrupt (INT bit = 1) upon detection of "AL bit = 1" does not occur

When an instruction which generates a start condition by enabling I<sup>2</sup>C operation (EN bit = 1) is executed (setting the MSS bit in the IBCR register to "1") with the I<sup>2</sup>C bus occupied by another master.

This is because, as shown in Figure 27.3-2, when the other master on the I<sup>2</sup>C bus starts communication with I<sup>2</sup>C disabled (EN bit = 0), the I<sup>2</sup>C bus enters the occupied state with no start condition detected (BB bit = 0).

**Figure 27.3-2 Diagram of Timing at which an Interrupt upon Detection of "AL Bit = 1" does not Occur**



If a symptom as described above can occur, follow the procedure below for software processing.

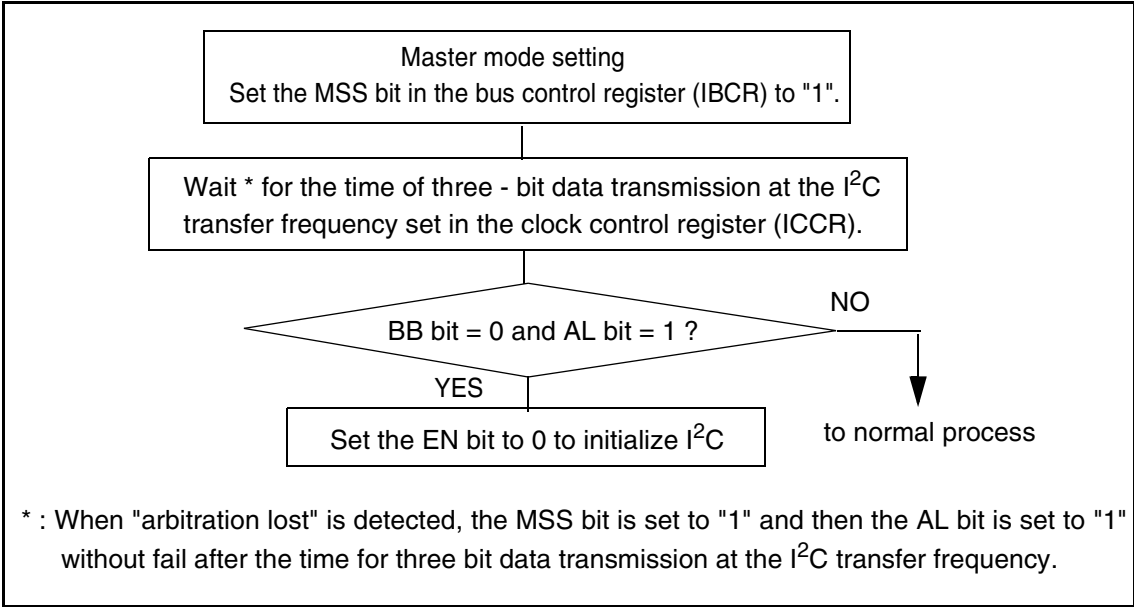
- 1) Execute the instruction that generates a start condition (set the MSS bit to "1").
- 2) Use, for example, the timer function to wait for the time \* for three - bit data transmission at the I<sup>2</sup>C transfer frequency set in the ICCR register.

Example: Time for three - bit data transmission at an I<sup>2</sup>C transfer frequency of 100 kHz

$$\{1/(100 \times 10^3)\} \times 3 = 30 \mu s$$

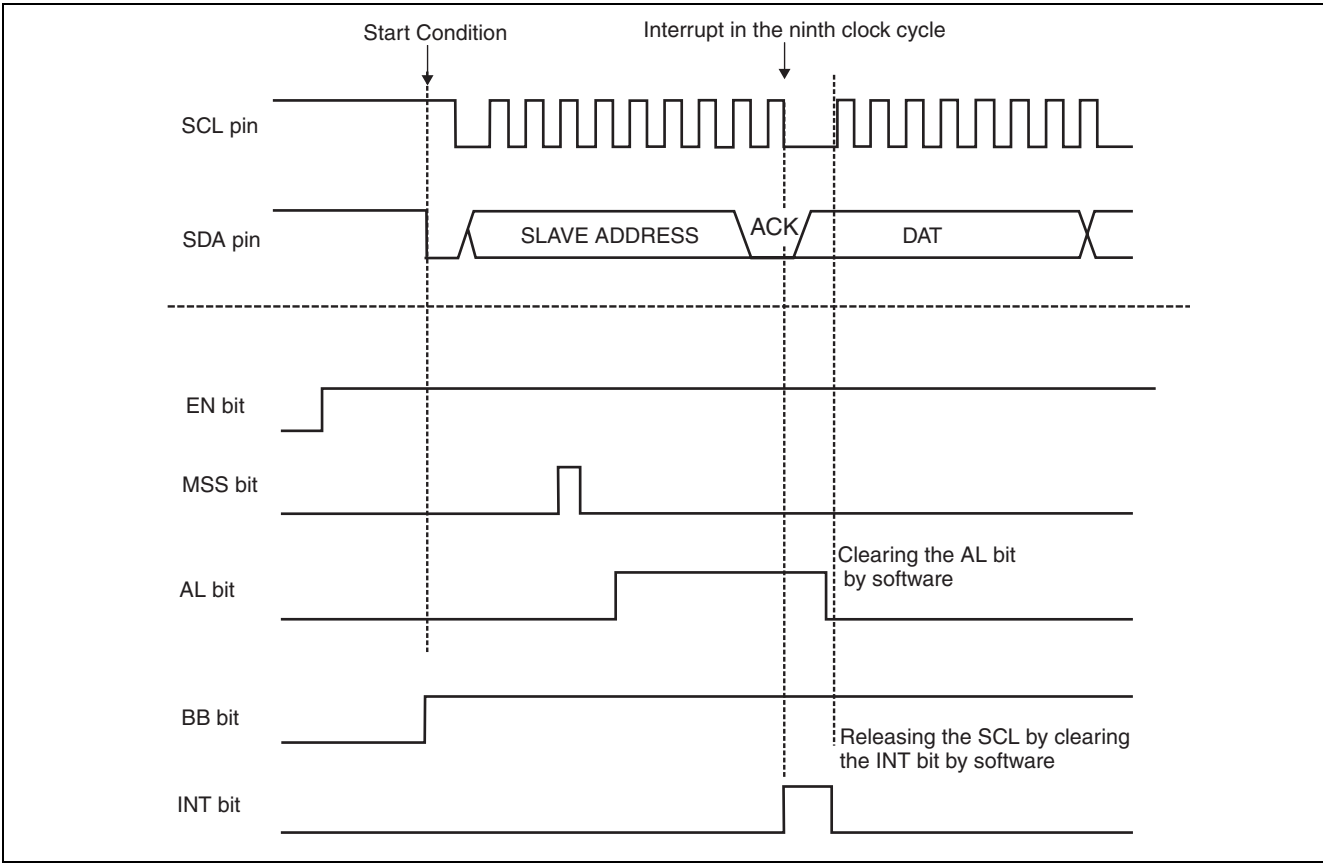
- 3) Check the AL and BB bits in the IBSR register and, if the AL and BB bits are 1 and 0, respectively, set the EN bit in the ICCR register to 0 to initialize I<sup>2</sup>C. When the AL and BB bits are not so, perform normal processing.

A sample flow is given below.



- Example of occurrence for an interrupt (INT bit = 1) upon detection of "AL bit = 1"
- When an instruction which generates a start condition is executed (setting the MSS bit to 1) with "bus busy" detected (BB bit = 1) and arbitration is lost, the INT bit interrupt occurs upon detection of "AL bit = 1".

**Figure 27.3-3 Diagram of Timing at which an Interrupt upon Detection of "AL Bit = 1" Occurs**



**■ Notes on Using the Bus Control Register (IBCR)**

Setting the SCC bit to "1" and the MSS bit to "0" at the same time is prohibited.

Writing to the SCC, MSS, and INT bits at the same time will cause a conflict between transfer of the next byte and generation of start or stop conditions. In this case, the priority is specified as follows.

☐ **Next byte transfer and stop condition generation**

If the INT and MSS bits are set to "0", setting of the MSS bit to "0" has priority and the stop condition is generated.

☐ **Next byte transfer and start condition generation**

If the INT bit is set to "0" and the SCC bit is set to "1", setting of the SCC bit to "1" has priority and the start condition is generated.

### 27.3.3 Clock Control Register (ICCR)

This section describes the configuration and functions of the clock control register (ICCR).

■ Clock Control Register (ICCR)

The diagram below shows the bit configuration of the clock control register (ICCR).

|                              |     |     |       |       |       |       |       |       |              |
|------------------------------|-----|-----|-------|-------|-------|-------|-------|-------|--------------|
| Address: 00008A <sub>H</sub> | 7   | 6   | 5     | 4     | 3     | 2     | 1     | 0     | ↵ Bit number |
| Clock control register       | -   | -   | EN    | CS4   | CS3   | CS2   | CS1   | CS0   | ICCR         |
| Read/Write ↵                 | (-) | (-) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) |              |
| Initial value ↵              | (-) | (-) | (0)   | (X)   | (X)   | (X)   | (X)   | (X)   |              |

The functions of the clock control register (ICCR) are as follows.

**[bit7, bit6] Unused**

These bits are unused.

**[bit5] EN: ENable**

This bit is used to enable I<sup>2</sup>C interface operation.

|   |                    |
|---|--------------------|
| 0 | Operation disabled |
| 1 | Operation enabled  |

- When this bit is set to "0", each bit of the IBSR register and the IBCR register (except for the BER and BEIE bits) is cleared.
- Setting the BER bit clears this bit.

**[bit4 to bit0] CS4 to CS0: Clock Period Select 4 to 0**

These bits are used to set the frequency of the serial clock. The shift clock frequency fsck is set in this register according to the following formula.

The values for m and n must be as shown in Table 27.3-1 for CS4 to CS0.

$$f_{sck} = \frac{\phi}{m \times n + 4}$$

ϕ: Machine clock

**Table 27.3-1 Serial Clock Frequency Settings**

| m | CS4 | CS3 |
|---|-----|-----|
| 5 | 0   | 0   |
| 6 | 0   | 1   |
| 7 | 1   | 0   |
| 8 | 1   | 1   |

| n   | CS2 | CS1 | CS0 |
|-----|-----|-----|-----|
| 4   | 0   | 0   | 0   |
| 8   | 0   | 0   | 1   |
| 16  | 0   | 1   | 0   |
| 32  | 0   | 1   | 1   |
| 64  | 1   | 0   | 0   |
| 128 | 1   | 0   | 1   |
| 256 | 1   | 1   | 0   |
| 512 | 1   | 1   | 1   |

---

**Note:**

The "+ 4" cycle in the formula reflects the minimum overhead for checking whether the output level of the SCL pin has changed. If the rising edge of the SCL pin is delayed or a slave device delays the clock, the overhead increases. Do not set the serial clock frequency to 100 kHz or more.

---



### 27.3.4 Address Register (IADR)

This section describes the configuration and functions of the address register (IADR).

■ Address Register (IADR)

The diagram below shows the bit configuration of the address register (IADR).

|                              |   |     |       |       |       |       |       |       |       |              |
|------------------------------|---|-----|-------|-------|-------|-------|-------|-------|-------|--------------|
| Address register             |   | 15  | 14    | 13    | 12    | 11    | 10    | 9     | 8     | ↩ Bit number |
| Address: 00008B <sub>H</sub> |   | -   | A6    | A5    | A4    | A3    | A2    | A1    | A0    | IADR         |
| Read/Write                   | ⇨ | (-) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) | (R/W) |              |
| Initial value                | ⇨ | (-) | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   | (X)   |              |

[bit15] Unused

This bit is unused.

[bit14 to bit8] A6 to A0

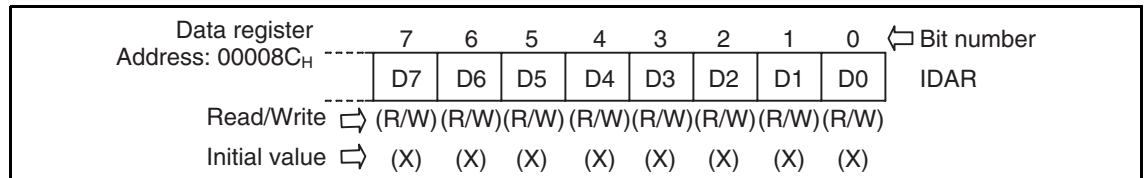
These bits are a slave address bit and used as a register to specify the slave address. In slave mode, received address data is compared with the data in the DAR register. If the data matches, the device transmits an acknowledge signal to the master.

## 27.3.5 Data Register (IDAR)

This section describes the configuration and functions of the data register (IDAR).

### ■ Data Register (IDAR)

The diagram below shows the bit configuration of the data register (IDAR).



#### [bit7 to bit0] D7 to D0

These bits are used as data bits.

These bits constitute a data register for serial transfer starting with the MSB. If data is received (TRX = 0), the data output value becomes "1".

With respect to writing, this register consists of a double buffer. If the bus is active (BB = 1), write data is loaded into the register for serial transfer. When the register is directly read for serial transfer, note that the receive data is only valid if the INT bit of the IBCR register is set.

## 27.4 Interrupt of I<sup>2</sup>C Interface

---

The interrupt of I<sup>2</sup>C interface occurs when the transfer of data is terminated.

---

### ■ Interrupt Control Bit and Interrupt Source of I<sup>2</sup>C Interface

The interrupt control bit and interrupt source of I<sup>2</sup>C interface is shown in the following table.

|                                     | I <sup>2</sup> C interrupt       |
|-------------------------------------|----------------------------------|
| Interrupt request flag              | IBCR:INT (bit8)                  |
| Interrupt request output enable bit | IBCR:INTE (bit9)                 |
| Interrupt generation source         | End of I <sup>2</sup> C transfer |

### ■ Interrupt Source of I<sup>2</sup>C Interface

Only one interrupt source is provided on the I<sup>2</sup>C bus. The interrupt occurs if the interrupt conditions are met when 1 byte transfer is terminated.

Each flag must be checked in the interrupt routine because multiple interrupt conditions are determined using one interrupt. The interrupt conditions at termination of 1 byte transfer are shown as follows.

- Byte transferred in bus master transfer
- Byte transferred in slave mode with addressing
- General call address is received
- Arbitration lost occurs

## ■ Interrupt of I<sup>2</sup>C Interface, DMA Transfer, and EI<sup>2</sup>OS

Table 27.4-1 shows the relationship between the interrupt source, interrupt vector, and interrupt control register other than software interrupt.

**Table 27.4-1 Interrupt Source, Interrupt Vector, and Interrupt Control Register**

| Interrupt source                                   | EI <sup>2</sup> OS clear | μDMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|--|--------------------------|----------------------|------------------|---------------------|----------------------------|---------------------|
|  |                          |                      | Number           | Address             | Number                     | Address             |
| I <sup>2</sup> C interface (only MB90485 series) * | ×                        | ×                    | #39              | FFFF60 <sub>H</sub> | ICR14                      | 0000BE <sub>H</sub> |

× : Interrupt request flag is not cleared.

\* : This interrupt source shares the interrupt source and interrupt number of other peripheral function.

For details, see Table 3.2-2.

### Note:

If there are two interrupt sources in the same interrupt number, resource clears both interrupt request flags. Therefore, when one of two sources uses the EI<sup>2</sup>OS/μDMAC function, the other interrupt function cannot use. The interrupt request enable bit of the relevant resource is set to 0 to execute the software polling processing.

## ■ Correspondence to DMA Transfer and EI<sup>2</sup>OS Function

The I<sup>2</sup>C interface does not correspond to the DMA transfer function and EI<sup>2</sup>OS function.

## 27.5 I<sup>2</sup>C Interface Operation

---

The I<sup>2</sup>C bus performs communication using two bidirectional bus lines that consist of one serial data line (SDA) and one serial clock line (SCL). The I<sup>2</sup>C interface has instead two open drain input/output pins (SDA, SCL) that allow hard-wired logic to be used.

---

### ■ Start Condition

If the bus is open (BB = 0, MSS = 0) and the MSS bit is set to "1", the I<sup>2</sup>C interface enters master mode and the start condition is generated as well. Even if the bus is active (BB = 1) in master mode, the start condition will be generated again if the SCC bit is set to "1". There are two ways to generate the start condition:

- In the state where the bus is not used (MSS = 0 & BB = 0 & INT = 0 & AL = 0), setting of the MSS bit to "1"
- In interrupt state and bus master mode (MSS = 1 & BB = 1 & INT = 1 & AL = 0), setting of the SCC bit to "1"

If the MSS bit is set to "1" when another system uses the bus (in idle state), the AL bit is set to "1". In states other than the above, setting the MSS bit and SCC bit to "1" is ignored.

### ■ Stop Condition

If the MSS bit is set to "0" in master mode (MSS = 1), a stop condition is generated and the devices enter slave mode. A stop condition is generated when the following conditions exist:

- If the MSS bit is set to "0" in bus master mode and in interrupt state (MSS = 1 & BB = 1 & INT = 1 & AL = 0).

In the other modes, setting the MSS bit to "0" is ignored.

#### Note:

It takes time from writing 0 to the MSS bit until the STOP condition is generated. Disabling the I<sup>2</sup>C interface (EN=0:ICCR) before the "STOP" condition occurs stops the operation immediately and generates an invalid clock on the SCL line. In this case, the I<sup>2</sup>C bus can be put in unfavorable situation.

Before disabling the I<sup>2</sup>C interface (EN=0:ICCR), check that the "STOP" condition has occurred (BB=0:IBSR).

---

## ■ Addressing

If, in master mode, a start condition is generated by setting BB = 1 and TRX = 1, the contents of the IDAR register are output starting with the MSB. When, after the address data has been transmitted, acknowledge is received from the slave, the TRX bit is set to the opposite value of bit0 for the transmitted data (IDAR register: bit0 after transmission).

In slave mode, after a start condition is generated by setting BB = 1 and TRX = 0, transmitted data from the master is received in the IDAR register. After the address data has been received, the IDAR register and IADR register are compared. If the contents of these registers match, AAS is set to "1", and acknowledge is transmitted to the master. Next, the TRX bit is set to the same value as bit0 of the received data (IDAR register: bit0 after reception).

## ■ Arbitration

If, when a master transmits, another master transmits data at the same time, arbitration occurs. If the signal of the locally transmitted data represents "1" and the data on the SDA line is represented by the "L" level, AL is set to "1" on the assumption that local arbitration is lost. As previously described, AL is set to "1" when a start condition occurs, even though the bus is active at the time. Setting AL to "1" results in MSS = 0 and TRX = 0, and the device enters slave reception mode.

### ■ Acknowledge

Acknowledge is transmitted from the receiving side to the transmitting side. The ACK bit is used to represent an acknowledge upon data reception. If data is transmitted, an acknowledge from the receive side is stored in the LRB bit.

If no acknowledge is received from the master side (receiving device) after reception from the slave (transmitting side), the TRX bit is set to "0" and the device enters slave reception state. The master device will in this case generate a stop condition when the slave opens the SCL line.

### ■ Bus Error

If the following conditions are satisfied, it can be assumed that a bus error occurred, and the I<sup>2</sup>C interface will enter stop mode.

- If an I<sup>2</sup>C bus basic standard violation is detected in data transfer mode (when an ACK bit is included).
- If a stop condition is detected in master mode
- If an I<sup>2</sup>C bus basic standard violation is detected in bus idle mode.

### ■ Other Considerations

#### ○ Processing after arbitration lost is detected

When arbitration lost is detected, the software has to determine whether local addressing was applied.

If arbitration lost occurs, the device enters slave mode on the hardware level, and after 1-byte transfer has been completed, both the CLK line and DATA line are set to "L" level. Consequently, without proper addressing, both the CLK line and DATA line will be immediately opened. With addressing, slave transmission or slave reception will have been set up before the CLK line and DATA line are opened (all of these preparations must be performed by software).

#### ○ Interrupt sources when arbitration lost is detected

If arbitration lost is detected, an interrupt source is not generated immediately, but only after the transfer of one byte is completed.

If arbitration lost is detected, the device enters slave mode on the hardware level. Even if this occurs in slave mode, a total of nine clocks will be output before the interrupt source is generated. Since interrupt sources are not generated immediately, no interrupt processing is performed after arbitration lost occurs.

### ○ **Interrupt conditions**

There is only one interrupt that can be generated related to the I<sup>2</sup>C bus. The interrupt source is generated either after the end of the transfer for one byte, or because another predefined interrupt condition was met.

Since there is only one interrupt, which of multiple interrupt conditions responsible for the interrupt must be identified by checking flags in the interrupt routine. Possible interrupt conditions after transfer of one byte has been performed are listed below.

- Interrupt in bus master mode
- Interrupt during slave mode with addressing
- Interrupt after a general call address is received
- "Arbitration lost" occurred

### ○ **Transfer speed**

Note that the maximum transfer speed of the I<sup>2</sup>C bus is 100 kHz (the frequency of the serial clock).





# APPENDIX

---

The appendix provides the memory map and lists the instructions used in the F<sup>2</sup>MC-16LX.

---

APPENDIX A Memory Map

APPENDIX B I/O Map

APPENDIX C Interrupt Source, Interrupt Vector, and Interrupt Control Register

APPENDIX D Instructions

# APPENDIX A    Memory Map

Memory space is divided according to three usage modes.

## ■ Memory Space

The memory space is divided according to three usage modes shown in Figure A-1.

Figure A-1    Memory Map

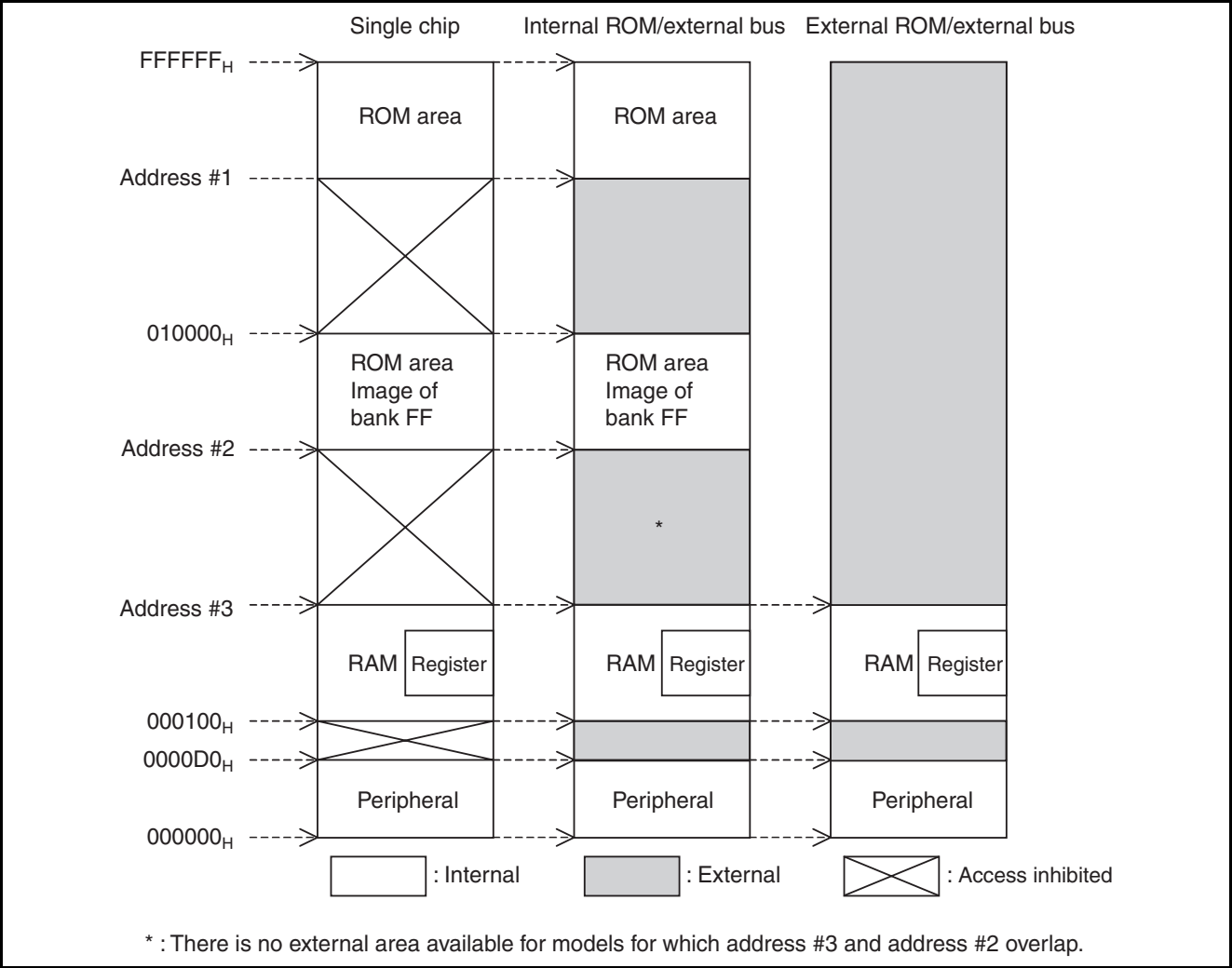


Table A-1 shows the relationship among address #1, address #2, and address #3 for each product type.

**Table A-1 Relationship Among Address #1, Address #2, and Address #3 by Product Type**

| Type      | Address #1                         | Address #2   | Address #3                         |
|-----------|------------------------------------|--|------------------------------------|
| MB90F481B | FC0000 <sub>H</sub>                | The MS bit of the ROMM register can be used to select 004000 <sub>H</sub> or 008000 <sub>H</sub> . | 001100 <sub>H</sub>                |
| MB90F482B | FC0000 <sub>H</sub>                |  | 001900 <sub>H</sub>                |
| MB90487B  | FD0000 <sub>H</sub>                |  | 002900 <sub>H</sub>                |
| MB90488B  | FC0000 <sub>H</sub>                |  | 002900 <sub>H</sub>                |
| MB90F488B | FC0000 <sub>H</sub>                |  | 002900 <sub>H</sub>                |
| MB90V480B | (FC0000 <sub>H</sub> )             |  | 004000 <sub>H</sub>                |
| MB90V485B | (FC0000 <sub>H</sub> )             |  | 004000 <sub>H</sub>                |
| MB90483C  | FB0000 <sub>H</sub> * <sup>1</sup> |  | 004000 <sub>H</sub>                |
| MB90F489B | F90000 <sub>H</sub> * <sup>2</sup> | 0080000 <sub>H</sub> fixed   | 006100 <sub>H</sub> * <sup>3</sup> |

\*1: In MB90F483C model, an access to F8 bank to FA bank and FC bank is not performed in the single-chip mode and internal ROM/external bus mode as shown in Figure A-3.

\*2: In MB90F489B model, an access to F8 bank and FC bank is not performed in the single-chip mode and internal ROM/external bus mode as shown in Figure A-2.

\*3: Because built-in RAM area in MB90F489B model is larger than that in MB90V485B model, the emulation area that exceeds 004000<sub>H</sub> must be set the emulation memory area at tool side.

---

**Note:**

In MB90F481B, the ROM contents of bank FF (between FC0000<sub>H</sub> and FC7FFF<sub>H</sub> or between FE0000<sub>H</sub> and FE7FFF<sub>H</sub>) can be viewed as an image in the upper part of bank 00, allowing the C compiler's small model to be more efficiently utilized. Because the lower 16 bits are identical, pointer declarations do not require the "far" specification to refer the table in ROM. For example, an access to 00C000<sub>H</sub> will actually be performed as access to the ROM contents at FFC000<sub>H</sub>. However, because the ROM area in bank FF exceeds 48 KB if the MS bit of the ROMM register is set to "0", not all of the areas can be viewed via their image in bank 00. For this reason, an image of the area from FF4000<sub>H</sub> to FFFFFFFF<sub>H</sub> can be seen in bank 00, while an image of the area from FF0000<sub>H</sub> to FF3FFF<sub>H</sub> can only be viewed via bank FF.

---

Figure A-2 shows the MB90F489B memory map.

Figure A-2 MB90F489B Memory Map

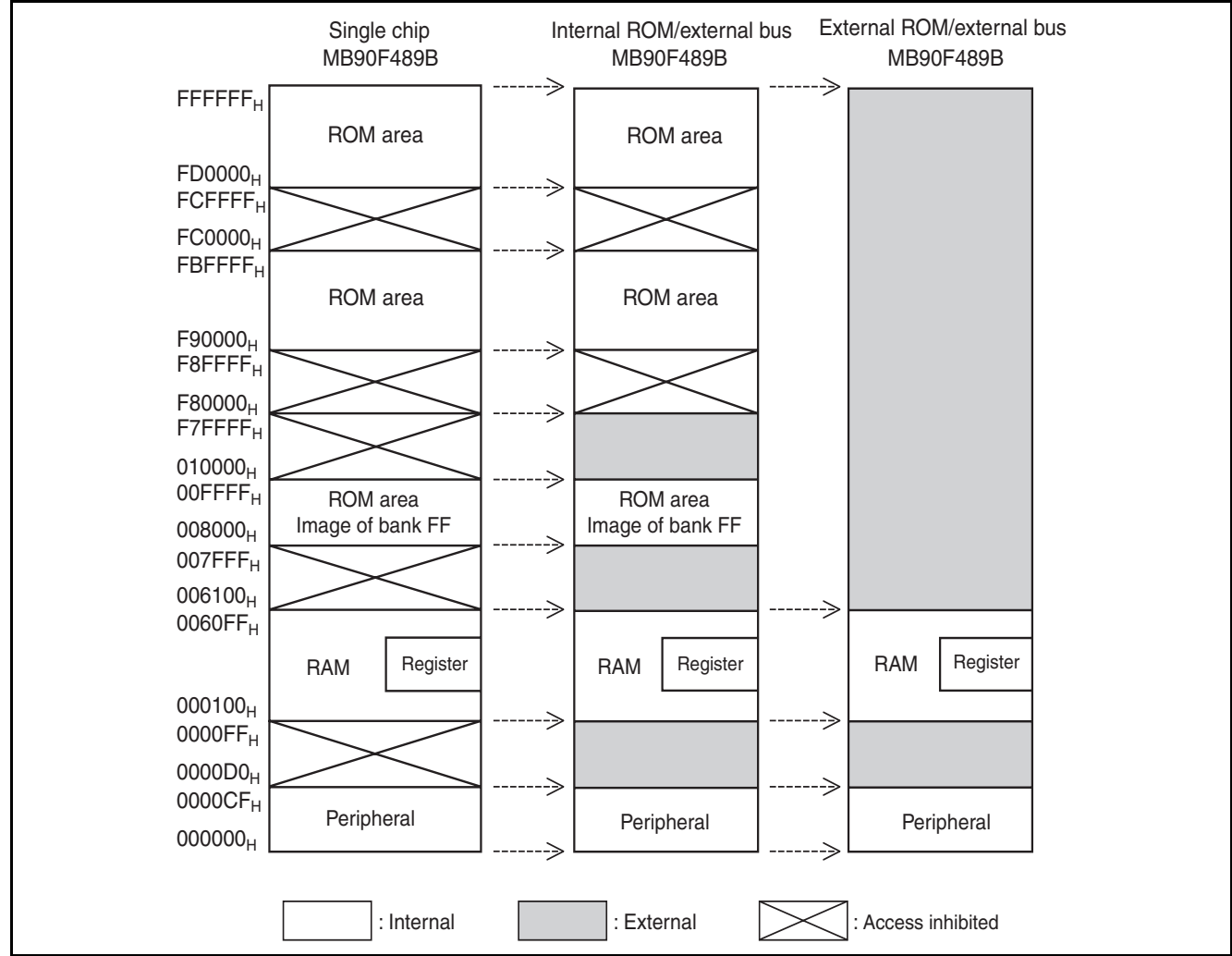
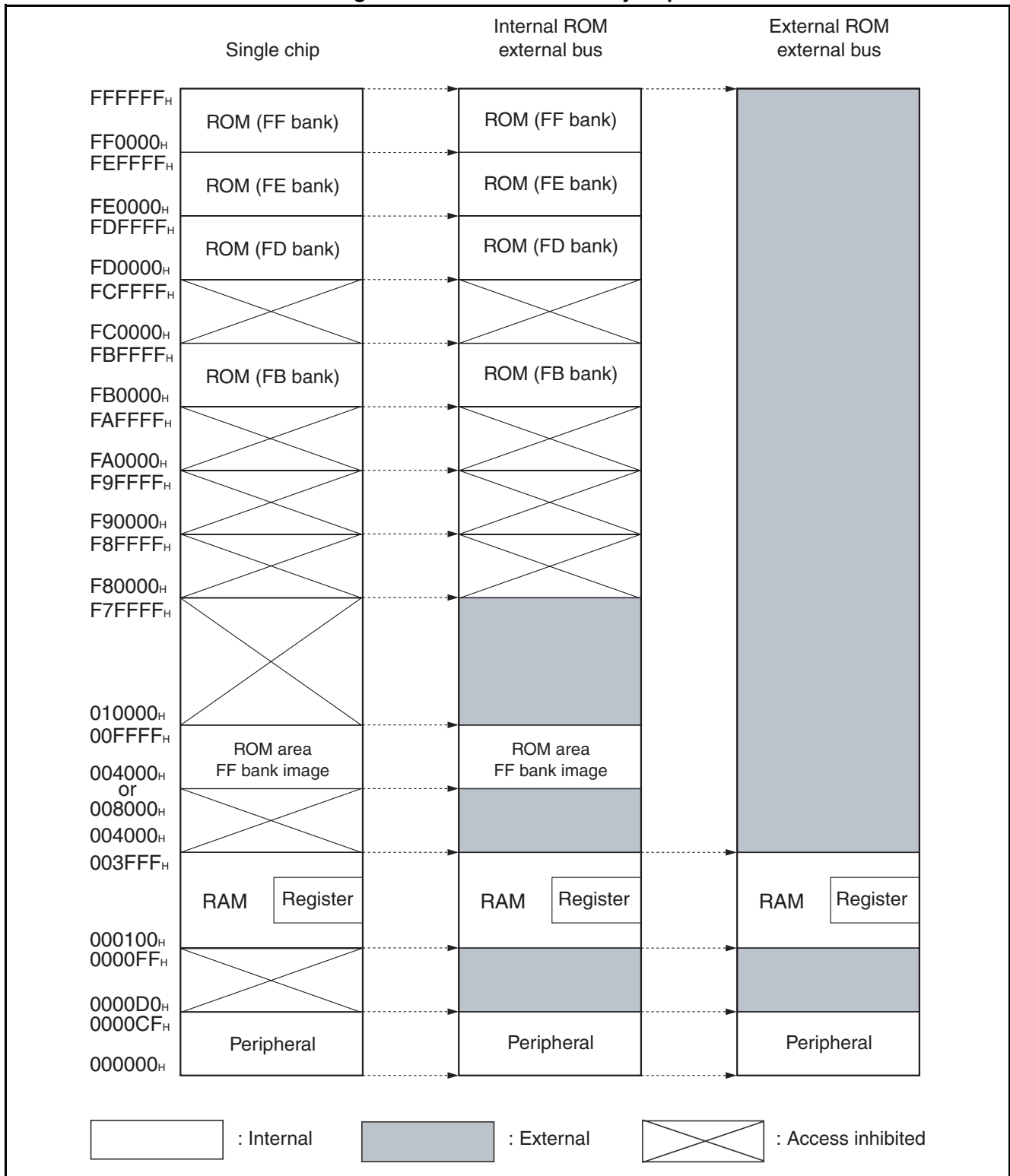


Figure A-3 shows the MB90483C memory map.

**Figure A-3 MB90483C Memory Map**



## APPENDIX B I/O Map

Table B-1 shows the addresses assigned to the registers for each peripheral function.

### ■ I/O Maps

Table B-1 shows the addresses assigned to the registers for each peripheral function.

**Table B-1 I/O Map (1/8)**

| Address         | Register                            | Abbreviation | Access | Resource                    | Initial value                               |
|-----------------|-------------------------------------|--------------|--------|-----------------------------|---|
| 00 <sub>H</sub> | Port 0 data register                | PDR0         | R/W    | Port 0                      | XXXXXXXX <sub>B</sub>                       |
| 01 <sub>H</sub> | Port 1 data register                | PDR1         | R/W    | Port 1                      | XXXXXXXX <sub>B</sub>                       |
| 02 <sub>H</sub> | Port 2 data register                | PDR2         | R/W    | Port 2                      | XXXXXXXX <sub>B</sub>                       |
| 03 <sub>H</sub> | Port 3 data register                | PDR3         | R/W    | Port 3                      | XXXXXXXX <sub>B</sub>                       |
| 04 <sub>H</sub> | Port 4 data register                | PDR4         | R/W    | Port 4                      | XXXXXXXX <sub>B</sub>                       |
| 05 <sub>H</sub> | Port 5 data register                | PDR5         | R/W    | Port 5                      | XXXXXXXX <sub>B</sub>                       |
| 06 <sub>H</sub> | Port 6 data register                | PDR6         | R/W    | Port 6                      | XXXXXXXX <sub>B</sub>                       |
| 07 <sub>H</sub> | Port 7 data register                | PDR7         | R/W    | Port 7                      | XXXXXXXX <sub>B</sub><br>(MB90480 series)   |
|                 |                                     |              |        |                             | 11XXXXXXXX <sub>B</sub><br>(MB90485 series) |
| 08 <sub>H</sub> | Port 8 data register                | PDR8         | R/W    | Port 8                      | XXXXXXXX <sub>B</sub>                       |
| 09 <sub>H</sub> | Port 9 data register                | PDR9         | R/W    | Port 9                      | XXXXXXXX <sub>B</sub>                       |
| 0A <sub>H</sub> | Port A data register                | PDRA         | R/W    | Port A                      | ----XXXX <sub>B</sub>                       |
| 0B <sub>H</sub> | Up/down timer input enable register | UDRE         | R/W    | Up/down timer input control | XX000000 <sub>B</sub>                       |
| 0C <sub>H</sub> | Interrupt/DTP enable register       | ENIR         | R/W    | DTP/external interrupt      | 00000000 <sub>B</sub>                       |
| 0D <sub>H</sub> | Interrupt/DTP enable register       | EIRR         | R/W    |                             | XXXXXXXX <sub>B</sub>                       |
| 0E <sub>H</sub> | Request level setting register      | ELVR         | R/W    |                             | 00000000 <sub>B</sub>                       |
| 0F <sub>H</sub> | Request level setting register      |              | R/W    |                             | 00000000 <sub>B</sub>                       |
| 10 <sub>H</sub> | Port 0 direction register           | DDR0         | R/W    | Port 0                      | 00000000 <sub>B</sub>                       |
| 11 <sub>H</sub> | Port 1 direction register           | DDR1         | R/W    | Port 1                      | 00000000 <sub>B</sub>                       |
| 12 <sub>H</sub> | Port 2 direction register           | DDR2         | R/W    | Port 2                      | 00000000 <sub>B</sub>                       |
| 13 <sub>H</sub> | Port 3 direction register           | DDR3         | R/W    | Port 3                      | 00000000 <sub>B</sub>                       |
| 14 <sub>H</sub> | Port 4 direction register           | DDR4         | R/W    | Port 4                      | 00000000 <sub>B</sub>                       |
| 15 <sub>H</sub> | Port 5 direction register           | DDR5         | R/W    | Port 5                      | 00000000 <sub>B</sub>                       |
| 16 <sub>H</sub> | Port 6 direction register           | DDR6         | R/W    | Port 6                      | 00000000 <sub>B</sub>                       |

Table B-1 I/O Map (2/8)

| Address         | Register                                     | Abbreviation  | Access | Resource                          | Initial value                             |
|-----------------|--|---------------|--------|-----------------------------------|---|
| 17 <sub>H</sub> | Port 7 direction register                    | DDR7          | R/W    | Port 7                            | 00000000 <sub>B</sub><br>(MB90480 series) |
|                 |  |               |        |                                   | XX000000 <sub>B</sub><br>(MB90485 series) |
| 18 <sub>H</sub> | Port 8 direction register                    | DDR8          | R/W    | Port 8                            | 00000000 <sub>B</sub>                     |
| 19 <sub>H</sub> | Port 9 direction register                    | DDR9          | R/W    | Port 9                            | 00000000 <sub>B</sub>                     |
| 1A <sub>H</sub> | Port A direction register                    | DDRA          | R/W    | Port A                            | ----0000 <sub>B</sub>                     |
| 1B <sub>H</sub> | Port 4 pin register                          | ODR4          | R/W    | Port 4 (Open drain control)       | 00000000 <sub>B</sub>                     |
| 1C <sub>H</sub> | Port 0 resistor register                     | RDR0          | R/W    | Port 0 (Pull-up resistor control) | 00000000 <sub>B</sub>                     |
| 1D <sub>H</sub> | Port 1 resistor register                     | RDR1          | R/W    | Port 1 (Pull-up resistor control) | 00000000 <sub>B</sub>                     |
| 1E <sub>H</sub> | Port 7 pin register                          | ODR7          | R/W    | Port 7 (Open drain control)       | 00000000 <sub>B</sub><br>(MB90480 series) |
|                 |  |               |        |                                   | XX000000 <sub>B</sub><br>(MB90485 series) |
| 1F <sub>H</sub> | Analog input enable register                 | ADER          | R/W    | Port 5, A/D                       | 11111111 <sub>B</sub>                     |
| 20 <sub>H</sub> | Serial mode register                         | SMR           | R/W    | UART0                             | 00000X00 <sub>B</sub>                     |
| 21 <sub>H</sub> | Serial control register                      | SCR           | R/W    |                                   | 00000100 <sub>B</sub>                     |
| 22 <sub>H</sub> | Serial input register/serial output register | SIDR/<br>SODR | R/W    |                                   | XXXXXXXX <sub>B</sub>                     |
| 23 <sub>H</sub> | Serial status register                       | SSR           | R/W    |                                   | 00001000 <sub>B</sub>                     |
| 24 <sub>H</sub> | Reserved area                                |               |        |                                   |   |
| 25 <sub>H</sub> | Clock division control register              | CDCR          | R/W    | Communication prescaler (UART)    | 00--0000 <sub>B</sub>                     |
| 26 <sub>H</sub> | Serial mode control status register 0        | SMCS0         | R, R/W | SC01 (ch.0)                       | ----0000 <sub>B</sub>                     |
| 27 <sub>H</sub> | Serial mode control status register 0        | SMCS0         | R, R/W |                                   | 00000010 <sub>B</sub>                     |
| 28 <sub>H</sub> | Serial data register 0                       | SDR0          | R/W    |                                   | XXXXXXXX <sub>B</sub>                     |
| 29 <sub>H</sub> | Clock division control register 0            | SDCR0         | R/W    | Communication prescaler (SC01)    | 0---0000 <sub>B</sub>                     |
| 2A <sub>H</sub> | Serial mode control status register 1        | SMCS1         | R, R/W | SCI2 (ch.1)                       | ----0000 <sub>B</sub>                     |
| 2B <sub>H</sub> | Serial mode control status register 1        | SMCS1         | R, R/W |                                   | 00000010 <sub>B</sub>                     |
| 2C <sub>H</sub> | Serial data register 1                       | SDR1          | R/W    |                                   | XXXXXXXX <sub>B</sub>                     |



**Table B-1 I/O Map (3/8)**

| Address         | Register                             | Abbreviation | Access | Resource                       | Initial value         |
|-----------------|--------------------------------------|--------------|--------|--------------------------------|-----------------------|
| 2D <sub>H</sub> | Clock division control register      | SDCR1        | R/W    | Communication prescaler (SCI1) | 0---0000 <sub>B</sub> |
| 2E <sub>H</sub> | PPG reload register L (ch.0)         | PRL0         | R/W    | 8/16-bit PPG<br>(ch.0 to ch.5) | XXXXXXXX <sub>B</sub> |
| 2F <sub>H</sub> | PPG reload register H (ch.0)         | PRLH0        | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 30 <sub>H</sub> | PPG reload register L (ch.1)         | PRL1         | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 31 <sub>H</sub> | PPG reload register H (ch.1)         | PRLH1        | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 32 <sub>H</sub> | PPG reload register L (ch.2)         | PRL2         | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 33 <sub>H</sub> | PPG reload register H (ch.2)         | PRLH2        | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 34 <sub>H</sub> | PPG reload register L (ch.3)         | PRL3         | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 35 <sub>H</sub> | PPG reload register H (ch.3)         | PRLH3        | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 36 <sub>H</sub> | PPG reload register L (ch.4)         | PRL4         | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 37 <sub>H</sub> | PPG reload register H (ch.4)         | PRLH4        | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 38 <sub>H</sub> | PPG reload register L (ch.5)         | PRL5         | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 39 <sub>H</sub> | PPG reload register H (ch.5)         | PRLH5        | R/W    |                                | XXXXXXXX <sub>B</sub> |
| 3A <sub>H</sub> | PPG0 operation mode control register | PPGC0        | R/W    |                                | 0X000XX1 <sub>B</sub> |
| 3B <sub>H</sub> | PPG1 operation mode control register | PPGC1        | R/W    |                                | 0X000001 <sub>B</sub> |
| 3C <sub>H</sub> | PPG2 operation mode control register | PPGC2        | R/W    |                                | 0X000XX1 <sub>B</sub> |
| 3D <sub>H</sub> | PPG3 operation mode control register | PPGC3        | R/W    |                                | 0X000001 <sub>B</sub> |
| 3E <sub>H</sub> | PPG4 operation mode control register | PPGC4        | R/W    |                                | 0X000XX1 <sub>B</sub> |
| 3F <sub>H</sub> | PPG5 operation mode control register | PPGC5        | R/W    |                                | 0X000001 <sub>B</sub> |
| 40 <sub>H</sub> | PPG0, 1 output control register      | PPG01        | R/W    | 8/16-bit PPG                   | 00000000 <sub>B</sub> |
| 41 <sub>H</sub> | Reserved area                        |              |        |                                |                       |
| 42 <sub>H</sub> | PPG2, 3 output control register      | PPG23        | R/W    | 8/16-bit PPG                   | 00000000 <sub>B</sub> |
| 43 <sub>H</sub> | Reserved area                        |              |        |                                |                       |
| 44 <sub>H</sub> | PPG4, 5 output control register      | PPG45        | R/W    | 8/16-bit PPG                   | 00000000 <sub>B</sub> |
| 45 <sub>H</sub> | Reserved area                        |              |        |                                |                       |
| 46 <sub>H</sub> | Control status register              | ADCS1        | R/W    | A/D converter                  | 00000000 <sub>B</sub> |
| 47 <sub>H</sub> |                                      | ADCS2        | R/W    |                                | 00000000 <sub>B</sub> |
| 48 <sub>H</sub> | Data register                        | ADCR1        | R      |                                | XXXXXXXX <sub>B</sub> |
| 49 <sub>H</sub> |                                      | ADCR2        | R,W    |                                | 00000XXX <sub>B</sub> |

Table B-1 I/O Map (4/8)

| Address         | Register                                 | Abbreviation | Access | Resource  | Initial value         |
|-----------------|--|--------------|--------|---|-----------------------|
| 4A <sub>H</sub> | Output compare register (ch.0) lower     | OCCP0        | R/W    | 16-bit output timer<br>output compare<br>(ch.0 to ch.5) | 00000000 <sub>B</sub> |
| 4B <sub>H</sub> | Output compare register (ch.0) upper     |              |        |   | 00000000 <sub>B</sub> |
| 4C <sub>H</sub> | Output compare register (ch.1) lower     | OCCP1        | R/W    |   | 00000000 <sub>B</sub> |
| 4D <sub>H</sub> | Output compare register (ch.1) upper     |              |        |   | 00000000 <sub>B</sub> |
| 4E <sub>H</sub> | Output compare register (ch.2) lower     | OCCP2        | R/W    |   | 00000000 <sub>B</sub> |
| 4F <sub>H</sub> | Output compare register (ch.2) upper     |              |        |   | 00000000 <sub>B</sub> |
| 50 <sub>H</sub> | Output compare register (ch.3) lower     | OCCP3        | R/W    |   | 00000000 <sub>B</sub> |
| 51 <sub>H</sub> | Output compare register (ch.3) upper     |              |        |   | 00000000 <sub>B</sub> |
| 52 <sub>H</sub> | Output compare register (ch.4) lower     | OCCP4        | R/W    |   | 00000000 <sub>B</sub> |
| 53 <sub>H</sub> | Output compare register (ch.4) upper     |              |        |   | 00000000 <sub>B</sub> |
| 54 <sub>H</sub> | Output compare register (ch.5) lower     | OCCP5        | R/W    |   | 00000000 <sub>B</sub> |
| 55 <sub>H</sub> | Output compare register (ch.5) upper     |              |        |   | 00000000 <sub>B</sub> |
| 56 <sub>H</sub> | Output compare control register (ch.0,1) | OCS01        | R/W    |   | 0000--00 <sub>B</sub> |
| 57 <sub>H</sub> | Output compare control register (ch.0,1) | OCS01        | R/W    |   | ---00000 <sub>B</sub> |
| 58 <sub>H</sub> | Output compare control register (ch.2,3) | OCS23        | R/W    |   | 0000--00 <sub>B</sub> |
| 59 <sub>H</sub> | Output compare control register (ch.2,3) | OCS23        | R/W    |   | ---00000 <sub>B</sub> |
| 5A <sub>H</sub> | Output compare control register (ch.4,5) | OCS45        | R/W    |   | 0000--00 <sub>B</sub> |
| 5B <sub>H</sub> | Output compare control register (ch.4,5) | OCS45        | R/W    |   | ---00000 <sub>B</sub> |
| 5C <sub>H</sub> | Input capture register (ch.0) lower      | IPCP0        | R      | 16-bit output timer<br>input capture<br>(ch.0, ch.1)    | XXXXXXXX <sub>B</sub> |
| 5D <sub>H</sub> | Input capture register (ch.0) upper      |              | R      |   | XXXXXXXX <sub>B</sub> |
| 5E <sub>H</sub> | Input capture register (ch.1) lower      | IPCP1        | R      |   | XXXXXXXX <sub>B</sub> |
| 5F <sub>H</sub> | Input capture register (ch.1) upper      |              | R      |   | XXXXXXXX <sub>B</sub> |
| 60 <sub>H</sub> | Input capture control register           | ICS01        | R/W    |   | 00000000 <sub>B</sub> |
| 61 <sub>H</sub> | Reserved area                            |              |        |   |                       |

Table B-1 I/O Map (5/8)

| Address                      | Register                              | Abbreviation | Access | Resource                              | Initial value         |
|------------------------------|---------------------------------------|--------------|--------|---------------------------------------|-----------------------|
| 62 <sub>H</sub>              | Timer data register lower             | TCDT         | R/W    | 16-bit output timer<br>free-run timer | 00000000 <sub>B</sub> |
| 63 <sub>H</sub>              | Timer data register upper             | TCDT         | R/W    |                                       | 00000000 <sub>B</sub> |
| 64 <sub>H</sub>              | Timer control status register         | TCCS         | R/W    |                                       | 00000000 <sub>B</sub> |
| 65 <sub>H</sub>              | Timer control status register         | TCCS         | R/W    |                                       | 0--00000 <sub>B</sub> |
| 66 <sub>H</sub>              | Compare clear register lower          | CPCLR        | R/W    |                                       | XXXXXXXX <sub>B</sub> |
| 67 <sub>H</sub>              | Compare clear register upper          |              |        |                                       | XXXXXXXX <sub>B</sub> |
| 68 <sub>H</sub>              | Up/down count register (ch.0)         | UDCR0        | R      | 8/16-bit up/down timer counter        | 00000000 <sub>B</sub> |
| 69 <sub>H</sub>              | Up/down count register (ch.1)         | UDCR1        | R      |                                       | 00000000 <sub>B</sub> |
| 6A <sub>H</sub>              | Reload compare register (ch.0)        | RCR0         | W      |                                       | 00000000 <sub>B</sub> |
| 6B <sub>H</sub>              | Reload compare register (ch.1)        | RCR1         | W      |                                       | 00000000 <sub>B</sub> |
| 6C <sub>H</sub>              | Counter control register (ch.0) lower | CCRL0        | W,R/W  |                                       | 0X00X000 <sub>B</sub> |
| 6D <sub>H</sub>              | Counter control register (ch.0) upper | CCRH0        | R/W    |                                       | 00000000 <sub>B</sub> |
| 6E <sub>H</sub>              | Reserved area                         |              |        |                                       |                       |
| 6F <sub>H</sub>              | ROM mirror function select register   | ROMM         | R/W    | ROM mirror function                   | -----+1 <sub>B</sub>  |
| 70 <sub>H</sub>              | Counter control register (ch.1) lower | CCRL1        | W,R/W  | 8/16-bit up/down timer counter        | 0X00X000 <sub>B</sub> |
| 71 <sub>H</sub>              | Counter control register (ch.1) upper | CCRH1        | R/W    |                                       | -0000000 <sub>B</sub> |
| 72 <sub>H</sub>              | Count status register (ch.0)          | CSR0         | R,R/W  |                                       | 00000000 <sub>B</sub> |
| 73 <sub>H</sub>              | Reserved area                         |              |        |                                       |                       |
| 74 <sub>H</sub>              | Count status register (ch.1)          | CSR1         | R,R/W  | 8/16-bit UDC                          | 00000000 <sub>B</sub> |
| 75 <sub>H</sub>              | Reserved area                         |              |        |                                       |                       |
| 76 <sub>H</sub> <sup>*</sup> | PWC control status register           | PWCSR0       | R,R/W  | PWC timer (ch.0)                      | 00000000 <sub>B</sub> |
| 77 <sub>H</sub> <sup>*</sup> |                                       |              |        |                                       | 0000000X <sub>B</sub> |
| 78 <sub>H</sub> <sup>*</sup> | PWC data buffer register              | PWCR0        | R/W    |                                       | 00000000 <sub>B</sub> |
| 79 <sub>H</sub> <sup>*</sup> |                                       |              |        |                                       | 00000000 <sub>B</sub> |
| 7A <sub>H</sub> <sup>*</sup> | PWC control status register           | PWCSR1       | R,R/W  | PWC timer (ch.1)                      | 00000000 <sub>B</sub> |
| 7B <sub>H</sub> <sup>*</sup> |                                       |              |        |                                       | 0000000X <sub>B</sub> |
| 7C <sub>H</sub> <sup>*</sup> | PWC data buffer register              | PWCR1        | R/W    |                                       | 00000000 <sub>B</sub> |
| 7D <sub>H</sub> <sup>*</sup> |                                       |              |        |                                       | 00000000 <sub>B</sub> |
| 7E <sub>H</sub> <sup>*</sup> | PWC control status register           | PWCSR2       | R,R/W  | PWC timer (ch.2)                      | 00000000 <sub>B</sub> |
| 7F <sub>H</sub> <sup>*</sup> |                                       |              |        |                                       | 0000000X <sub>B</sub> |
| 80 <sub>H</sub> <sup>*</sup> | PWC data buffer register              | PWCR2        | R/W    |                                       | 00000000 <sub>B</sub> |
| 81 <sub>H</sub> <sup>*</sup> |                                       |              |        |                                       | 00000000 <sub>B</sub> |
| 82 <sub>H</sub> <sup>*</sup> | divide ratio control register         | DIVR0        | R/W    | PWC (ch.0)                            | -----00 <sub>B</sub>  |

Table B-1 I/O Map (6/8)

| Address                            | Register  | Abbreviation | Access | Resource                         | Initial value          |
|------------------------------------|---|--------------|--------|----------------------------------|------------------------|
| 83 <sub>H</sub>                    | Reserved area                                     |              |        |                                  |                        |
| 84 <sub>H</sub> <sup>*</sup>       | divide ratio control register                     | DIVR1        | R/W    | PWC (ch.1)                       | -----00 <sub>B</sub>   |
| 85 <sub>H</sub>                    | Reserved area                                     |              |        |                                  |                        |
| 86 <sub>H</sub> <sup>*</sup>       | divide ratio control register                     | DIVR2        | R/W    | PWC (ch.2)                       | -----00 <sub>B</sub>   |
| 87 <sub>H</sub>                    | Reserved area                                     |              |        |                                  |                        |
| 88 <sub>H</sub> <sup>*</sup>       | bus status register                               | IBSR         | R      | I <sup>2</sup> C                 | 00000000 <sub>B</sub>  |
| 89 <sub>H</sub> <sup>*</sup>       | bus control register                              | IBCR         | R/W    |                                  | 00000000 <sub>B</sub>  |
| 8A <sub>H</sub> <sup>*</sup>       | bus clock select register                         | ICCR         | R/W    |                                  | --0XXXXX <sub>B</sub>  |
| 8B <sub>H</sub> <sup>*</sup>       | bus address register                              | IADR         | R/W    |                                  | -XXXXXXXX <sub>B</sub> |
| 8C <sub>H</sub> <sup>*</sup>       | bus data register                                 | IDAR         | R/W    |                                  | XXXXXXXXX <sub>B</sub> |
| 8D <sub>H</sub>                    | Reserved area                                     |              |        |                                  |                        |
| 8E <sub>H</sub> <sup>*</sup>       | μPG control status register                       | PGCSR        | R/W    | μPG                              | 00000--- <sub>B</sub>  |
| 8F <sub>H</sub> to 9B <sub>H</sub> | Use prohibited                                    |              |        |                                  |                        |
| 9C <sub>H</sub>                    | μDMAC status register                             | DSRL         | R/W    | μDMAC                            | 00000000 <sub>B</sub>  |
| 9D <sub>H</sub>                    | μDMAC status register                             | DSRH         | R/W    | μDMAC                            | 00000000 <sub>B</sub>  |
| 9E <sub>H</sub>                    | Program address detection control status register | PACSR        | R/W    | Address Match Detection Function | 00000000 <sub>B</sub>  |
| 9F <sub>H</sub>                    | Delay interrupt source generate/delete register   | DIRR         | R/W    | Delay interrupt generate module  | -----0 <sub>B</sub>    |
| A0 <sub>H</sub>                    | Low-power consumption mode register               | LPMCR        | W,R/W  | Low-power consumption power      | 00011000 <sub>B</sub>  |
| A1 <sub>H</sub>                    | Clock select register                             | CKSCR        | R,R/W  | Low-power consumption power      | 11111100 <sub>B</sub>  |
| A2 <sub>H</sub> , A3 <sub>H</sub>  | Reserved area                                     |              |        |                                  |                        |
| A4 <sub>H</sub>                    | μDMAC stop status register                        | DSSR         | R/W    | μDMAC                            | 00000000 <sub>B</sub>  |
| A5 <sub>H</sub>                    | Automatic ready function selection register       | ARSR         | W      | External pin                     | 0011--00 <sub>B</sub>  |
| A6 <sub>H</sub>                    | External address output control register          | HACR         | W      | External pin                     | ***** <sub>B</sub>     |
| A7 <sub>H</sub>                    | Bus control signal control register               | EPCR         | W      | External pin                     | 1000*10- <sub>B</sub>  |
| A8 <sub>H</sub>                    | Watchdog timer control register                   | WDTC         | R,R/W  | Watchdog timer                   | XXXXX111 <sub>B</sub>  |
| A9 <sub>H</sub>                    | Time-base timer control register                  | TBTC         | W,R/W  | Time-base timer                  | 1XX00100 <sub>B</sub>  |
| AA <sub>H</sub>                    | Watch timer control register                      | WTC          | R,R/W  | Watch timer                      | 10001000 <sub>B</sub>  |
| AB <sub>H</sub>                    | Reserved area                                     |              |        |                                  |                        |
| AC <sub>H</sub>                    | μDMAC control register                            | DERL         | R/W    | μDMAC                            | 00000000 <sub>B</sub>  |
| AD <sub>H</sub>                    | μDMAC control register                            | DERH         | R/W    | μDMAC                            | 00000000 <sub>B</sub>  |

Table B-1 I/O Map (7/8)

| Address                               | Register                                     | Abbreviation  | Access | Resource                | Initial value          |
|---------------------------------------|--|---------------|--------|-------------------------|------------------------|
| AE <sub>H</sub>                       | Flash memory control status register         | FMCR          | W,R/W  | Flash memory I/F        | 000X0000 <sub>B</sub>  |
| AF <sub>H</sub>                       | Use prohibited                               |               |        |                         |                        |
| B0 <sub>H</sub>                       | Interrupt control register 00                | ICR00         | W,R/W  | Interrupt controller    | 00000111 <sub>B</sub>  |
| B1 <sub>H</sub>                       | Interrupt control register 01                | ICR01         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B2 <sub>H</sub>                       | Interrupt control register 02                | ICR02         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B3 <sub>H</sub>                       | Interrupt control register 03                | ICR03         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B4 <sub>H</sub>                       | Interrupt control register 04                | ICR04         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B5 <sub>H</sub>                       | Interrupt control register 05                | ICR05         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B6 <sub>H</sub>                       | Interrupt control register 06                | ICR06         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B7 <sub>H</sub>                       | Interrupt control register 07                | ICR07         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B8 <sub>H</sub>                       | Interrupt control register 08                | ICR08         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| B9 <sub>H</sub>                       | Interrupt control register 09                | ICR09         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| BA <sub>H</sub>                       | Interrupt control register 10                | ICR10         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| BB <sub>H</sub>                       | Interrupt control register 11                | ICR11         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| BC <sub>H</sub>                       | Interrupt control register 12                | ICR12         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| BD <sub>H</sub>                       | Interrupt control register 13                | ICR13         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| BE <sub>H</sub>                       | Interrupt control register 14                | ICR14         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| BF <sub>H</sub>                       | Interrupt control register 15                | ICR15         | W,R/W  |                         | 00000111 <sub>B</sub>  |
| C0 <sub>H</sub>                       | Chip selection MASK register 0               | CMR0          | R/W    | Chip selection facility | 00001111 <sub>B</sub>  |
| C1 <sub>H</sub>                       | Chip selection area register 0               | CAR0          | R/W    |                         | 11111111 <sub>B</sub>  |
| C2 <sub>H</sub>                       | Chip selection MASK register 1               | CMR1          | R/W    |                         | 00001111 <sub>B</sub>  |
| C3 <sub>H</sub>                       | Chip selection area register 1               | CAR1          | R/W    |                         | 11111111 <sub>B</sub>  |
| C4 <sub>H</sub>                       | Chip selection MASK register 2               | CMR2          | R/W    |                         | 00001111 <sub>B</sub>  |
| C5 <sub>H</sub>                       | Chip selection area register 2               | CAR2          | R/W    |                         | 11111111 <sub>B</sub>  |
| C6 <sub>H</sub>                       | Chip selection MASK register 3               | CMR3          | R/W    |                         | 00001111 <sub>B</sub>  |
| C7 <sub>H</sub>                       | Chip selection area register 3               | CAR3          | R/W    |                         | 11111111 <sub>B</sub>  |
| C8 <sub>H</sub>                       | Chip selection control register              | CSCR          | R/W    |                         | ----000 <sub>B</sub> * |
| C9 <sub>H</sub>                       | Chip selection control active level register | CALR          | R/W    |                         | ----0000 <sub>B</sub>  |
| CA <sub>H</sub>                       | Control status register                      | TMCSR         | R/W    | 16-bit reload timer     | 00000000 <sub>B</sub>  |
| CB <sub>H</sub>                       |  |               |        |                         | ----0000 <sub>B</sub>  |
| CC <sub>H</sub>                       | 16-bit timer register/16-bit reload register | TMR/<br>TMRLR | R/W    |                         | XXXXXXXX <sub>B</sub>  |
| CD <sub>H</sub>                       |  |               |        |                         |                        |
| CE <sub>H</sub>                       | Reserved area                                |               |        |                         |                        |
| CF <sub>H</sub>                       | PLL output selection register                | PLLOS         | W      | Saving power            | -----00 <sub>B</sub>   |
| D0 <sub>H</sub> to<br>FF <sub>H</sub> | External area                                |               |        |                         |                        |

**Table B-1 I/O Map (8/8)**

| Address                            | Register                                      | Abbreviation | Access | Resource                         | Initial value         |
|------------------------------------|---|--------------|--------|----------------------------------|-----------------------|
| 100 <sub>H</sub> to # <sub>H</sub> | RAM area                                      |              |        |                                  |                       |
| 1FF0 <sub>H</sub>                  | Program address detection register 0 (lower)  | PADR0        | R/W    | Address match detection function | XXXXXXXX <sub>B</sub> |
| 1FF1 <sub>H</sub>                  | Program address detection register 0 (middle) |              | R/W    |                                  | XXXXXXXX <sub>B</sub> |
| 1FF2 <sub>H</sub>                  | Program address detection register 0 (upper)  |              | R/W    |                                  | XXXXXXXX <sub>B</sub> |
| 1FF3 <sub>H</sub>                  | Program address detection register 1 (lower)  | PADR1        | R/W    | Address match detection function | XXXXXXXX <sub>B</sub> |
| 1FF4 <sub>H</sub>                  | Program address detection register 1 (middle) |              | R/W    |                                  | XXXXXXXX <sub>B</sub> |
| 1FF5 <sub>H</sub>                  | Program address detection register 1 (upper)  |              | R/W    |                                  | XXXXXXXX <sub>B</sub> |

\* : These registers are only for MB90485 series.  
They are used as the reserved area on MB90480 series.

---

Note:

Descriptions for read/write  
R/W: Readable/Writable  
R: Read only  
W: Write only

Descriptions for initial value  
0: The initial value of this bit is "0".  
1: The initial value of this bit is "1".  
X: The initial value of this bit is indefinite.  
- : This bit is not used.  
\*: The initial value of this bit is "1" or "0".  
The value depends on the mode pin (MD2, MD1 and MD0) .  
+: The initial value of this bit is "1" or "0". The value depends on the RAM area of the device.

---

## APPENDIX C Interrupt Source, Interrupt Vector, and Interrupt Control Register

Table C-1 shows the relationship between interrupt sources and the interrupt vector/interrupt control registers.

### ■ Interrupt Sources, Interrupt Vectors, and Interrupt Control Registers

Table C-1 Relationship Between Interrupt Sources and Interrupt Vector/Interrupt Control Registers (1/2)

| Interrupt source   | EI <sup>2</sup> OS clear | $\mu$ DMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|--|--------------------------|---------------------------|------------------|---------------------|----------------------------|---------------------|
|  |                          |                           | Number           | Address             | Number                     | Address             |
| Reset  | ×                        | -                         | #08              | FFFFDC <sub>H</sub> | -                          | -                   |
| INT9 instruction   | ×                        | -                         | #09              | FFFFD8 <sub>H</sub> | -                          | -                   |
| Exception  | ×                        | -                         | #10              | FFFFD4 <sub>H</sub> | -                          | -                   |
| INT0 (IRQ0)  | ○                        | 0                         | #11              | FFFFD0 <sub>H</sub> | ICR00                      | 0000B0 <sub>H</sub> |
| INT1 (IRQ1)  | ○                        | ×                         | #12              | FFFFCC <sub>H</sub> |                            |                     |
| INT2 (IRQ2)  | ○                        | ×                         | #13              | FFFFC8 <sub>H</sub> | ICR01                      | 0000B1 <sub>H</sub> |
| INT3 (IRQ3)  | ○                        | ×                         | #14              | FFFFC4 <sub>H</sub> |                            |                     |
| INT4 (IRQ4)  | ○                        | ×                         | #15              | FFFFC0 <sub>H</sub> | ICR02                      | 0000B2 <sub>H</sub> |
| INT5 (IRQ5)  | ○                        | ×                         | #16              | FFFFBC <sub>H</sub> |                            |                     |
| INT6 (IRQ6)  | ○                        | ×                         | #17              | FFFFB8 <sub>H</sub> | ICR03                      | 0000B3 <sub>H</sub> |
| INT7 (IRQ7)  | ○                        | ×                         | #18              | FFFFB4 <sub>H</sub> |                            |                     |
| PWC1 (only MB90485 series)   | ○                        | ×                         | #19              | FFFFB0 <sub>H</sub> | ICR04                      | 0000B4 <sub>H</sub> |
| PWC2 (only MB90485 series)   | ○                        | ×                         | #20              | FFFFAC <sub>H</sub> |                            |                     |
| PWC0 (only MB90485 series)   | ○                        | ×                         | #21              | FFFFA8 <sub>H</sub> | ICR05                      | 0000B5 <sub>H</sub> |
| PPG0/PPG1 counter borrow   | ×                        | 2                         | #22              | FFFFA4 <sub>H</sub> |                            |                     |
| PPG2/PPG3 counter borrow   | ×                        | 3                         | #23              | FFFFA0 <sub>H</sub> | ICR06                      | 0000B6 <sub>H</sub> |
| PPG4/PPG5 counter borrow   | ×                        | 4                         | #24              | FFFF9C <sub>H</sub> |                            |                     |
| 8/16-bit up/down counter / timer (ch.0, ch.1) compare/underflow/overflow/up-down reverse | ○                        | ×                         | #25              | FFFF98 <sub>H</sub> | ICR07                      | 0000B7 <sub>H</sub> |
| Input capture (ch.0) load  | ○                        | 5                         | #26              | FFFF94 <sub>H</sub> |                            |                     |

**Table C-1 Relationship Between Interrupt Sources and Interrupt Vector/Interrupt Control Registers (2/2)**

| Interrupt source                                      | EI <sup>2</sup> OS clear | $\mu$ DMAC channel number | Interrupt vector |                     | Interrupt control register |                     |
|---|--------------------------|---------------------------|------------------|---------------------|----------------------------|---------------------|
|   |                          |                           | Number           | Address             | Number                     | Address             |
| Input capture (ch.1) load                             | ○                        | 6                         | #27              | FFFF90 <sub>H</sub> | ICR08                      | 0000B8 <sub>H</sub> |
| Output compare (ch.0) match                           | ○                        | 8                         | #28              | FFFF8C <sub>H</sub> |                            |                     |
| Output compare (ch.1) match                           | ○                        | 9                         | #29              | FFFF88 <sub>H</sub> | ICR09                      | 0000B9 <sub>H</sub> |
| Output compare (ch.2) match                           | ○                        | 10                        | #30              | FFFF84 <sub>H</sub> |                            |                     |
| Output compare (ch.3) match                           | ○                        | ×                         | #31              | FFFF80 <sub>H</sub> | ICR10                      | 0000BA <sub>H</sub> |
| Output compare (ch.4) match                           | ○                        | ×                         | #32              | FFFF7C <sub>H</sub> |                            |                     |
| Output compare (ch.5) match                           | ○                        | ×                         | #33              | FFFF78 <sub>H</sub> | ICR11                      | 0000BB <sub>H</sub> |
| UART transmit completed                               | ○                        | 11                        | #34              | FFFF74 <sub>H</sub> |                            |                     |
| 16-bit free-run timer/16-bit reload timer overflow *2 | ○                        | 12                        | #35              | FFFF70 <sub>H</sub> | ICR12                      | 0000BC <sub>H</sub> |
| UART receive completed                                | ◎                        | 7                         | #36              | FFFF6C <sub>H</sub> |                            |                     |
| SIO1 (ch.0)   | ○                        | 13                        | #37              | FFFF68 <sub>H</sub> | ICR13                      | 0000BD <sub>H</sub> |
| SIO2 (ch.1)   | ○                        | 14                        | #38              | FFFF64 <sub>H</sub> |                            |                     |
| I <sup>2</sup> C interface (only MB90485 series)      | ×                        | ×                         | #39              | FFFF60 <sub>H</sub> | ICR14                      | 0000BE <sub>H</sub> |
| A/D converter   | ○                        | 15                        | #40              | FFFF5C <sub>H</sub> |                            |                     |
| FLASH write/delete, time-base timer/watch timer *1    | ×                        | ×                         | #41              | FFFF58 <sub>H</sub> | ICR15                      | 0000BF <sub>H</sub> |
| Delay interrupt generate module                       | ×                        | ×                         | #42              | FFFF54 <sub>H</sub> |                            |                     |

×: The interrupt request can not be cleared by the interrupt clear signal.

○: The interrupt request can be cleared by the interrupt clear signal.

◎: The interrupt request can be cleared by the interrupt clear signal. With a stop request.

\*1: Make sure that flash write/delete operations are not used during operation of the time-base timer and watch timer.

\*2: When the reload timer underflow interrupt is changed from enabled (INTE bit of TMCSR register = 1) to disabled (INTE bit of TMCSR register = 0), "0" is written to the INTE bit after the IL2 to IL0 bits of the interrupt control register (ICR12) is set to "111<sub>B</sub>" to disable the interrupt.

#### Note:

If the same interrupt number is assigned to two interrupt sources, the resource is cleared only if both interrupt request flags are cleared. Therefore, if either of two interrupt functions uses the EI<sup>2</sup>OS function or  $\mu$ DMAC function, the other interrupt function cannot be used. Set the relevant resource's interrupt request enable bit to "0" and use software polling processing to handle this situation.



## APPENDIX D Instructions

---

**APPENDIX D describes the instructions used by the F<sup>2</sup>MC-16LX.**

---

- D.1 Instruction Types
- D.2 Addressing
- D.3 Direct Addressing
- D.4 Indirect Addressing
- D.5 Execution Cycle Count
- D.6 Effective address field
- D.7 How to Read the Instruction List
- D.8 F2MC-16LX Instruction List
- D.9 Instruction Map

## D.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

## D.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 D.2-1 lists the address formats specified by the effective address field.

**Table D.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          |

## D.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 D.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 D.3-1 lists the registers that can be specified. Figure D.3-2 shows an example of register direct addressing.

**Table D.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                       |

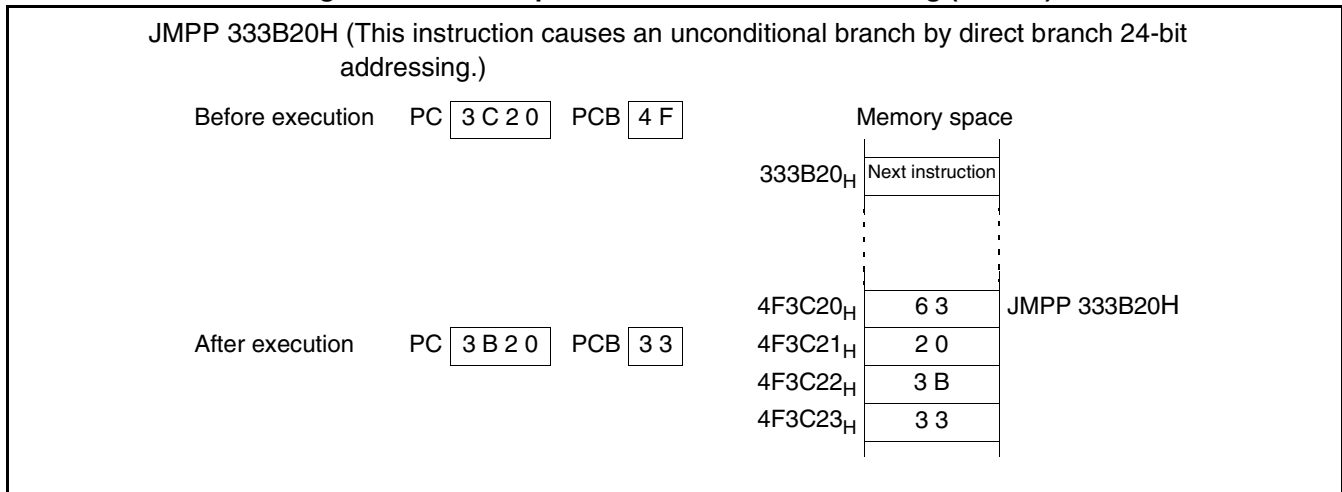
\*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.



- 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.

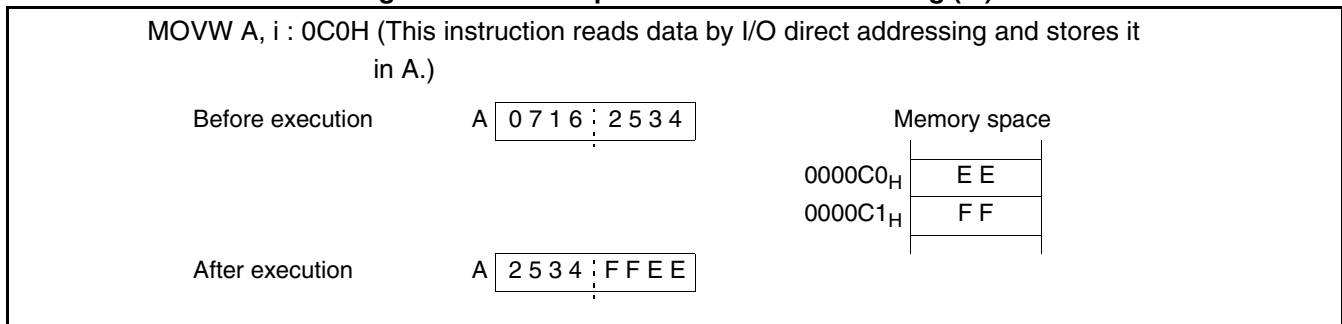
### Figure D.3-4 Example of Direct Branch Addressing (addr24)



- 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.

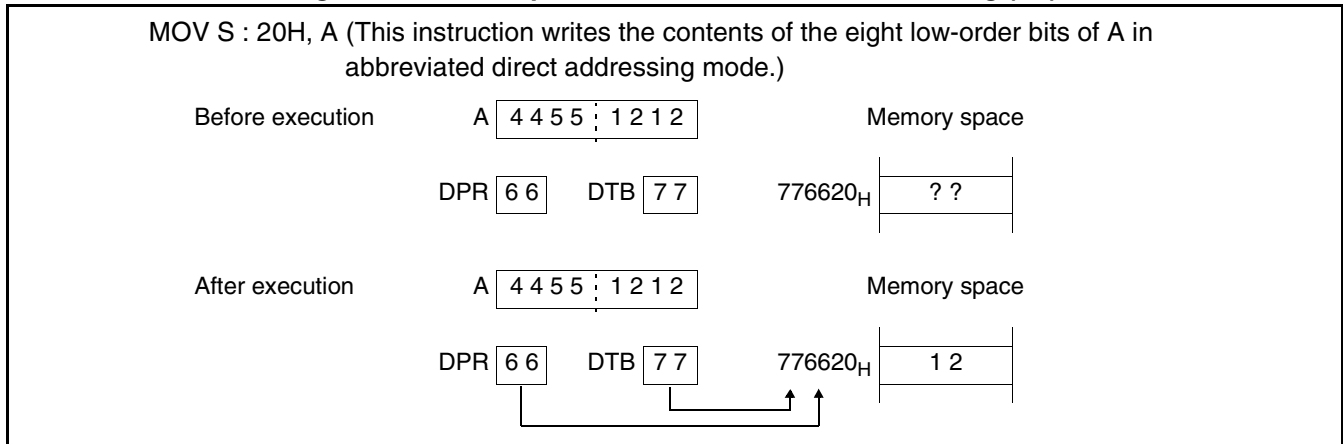
**Figure D.3-5 Example of I/O Direct Addressing (io)**



- 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).

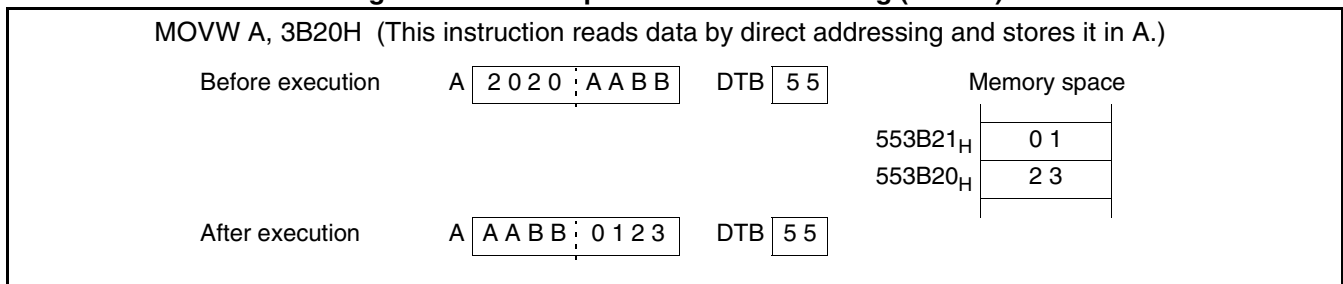
**Figure D.3-6 Example of Abbreviated Direct Addressing (dir)**



- 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 D.3-7 Example of Direct Addressing (addr16)**

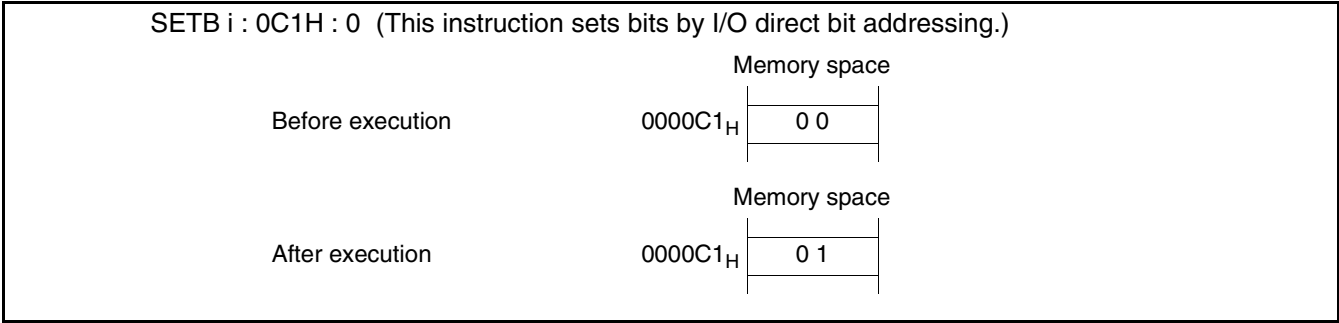




● I/O direct bit addressing (io:bp)

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).

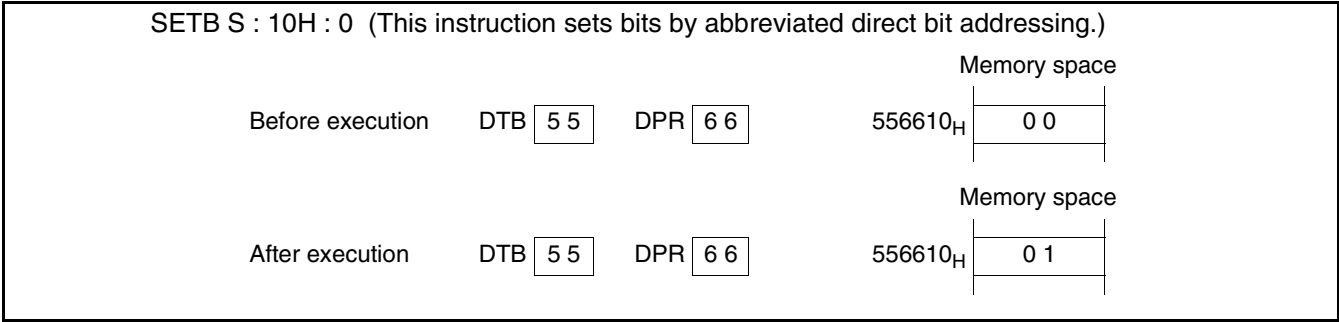
**Figure D.3-8 Example of I/O Direct Bit Addressing (io:bp)**



● 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).

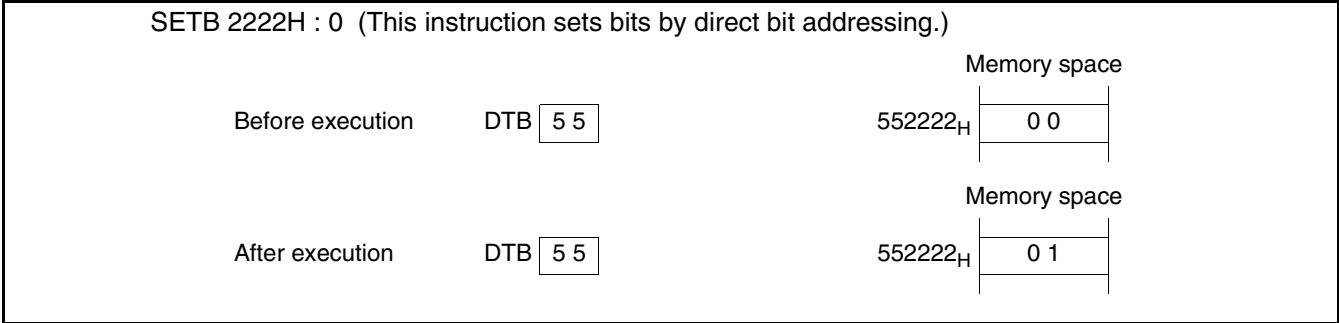
**Figure D.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)**



● 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).

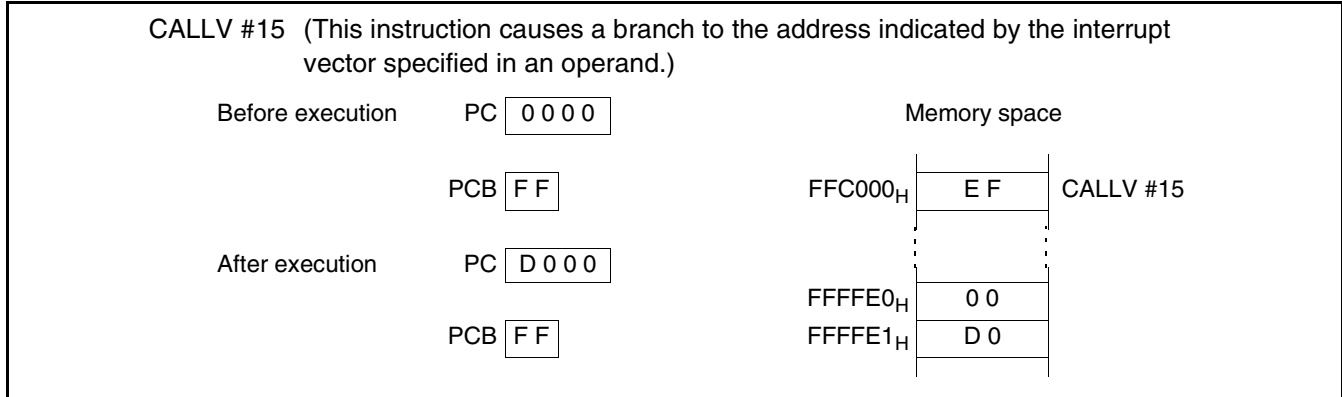
**Figure D.3-10 Example of Direct Bit Addressing (addr16:bp)**



● 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 D.3-11 Example of Vector Addressing (#vct)**



**Table D.3-2 CALLV Vector List**

| Instruction | Vector address L    | Vector address H    |
|-------------|---------------------|---------------------|
| CALLV #0    | XXFFFE <sub>H</sub> | XXFFFF <sub>H</sub> |
| CALLV #1    | XXFFFC <sub>H</sub> | XXFFFD <sub>H</sub> |
| CALLV #2    | XXFFFA <sub>H</sub> | XXFFFB <sub>H</sub> |
| CALLV #3    | XXFFF8 <sub>H</sub> | XXFFF9 <sub>H</sub> |
| CALLV #4    | XXFFF6 <sub>H</sub> | XXFFF7 <sub>H</sub> |
| CALLV #5    | XXFFF4 <sub>H</sub> | XXFFF5 <sub>H</sub> |
| CALLV #6    | XXFFF2 <sub>H</sub> | XXFFF3 <sub>H</sub> |
| CALLV #7    | XXFFF0 <sub>H</sub> | XXFFF1 <sub>H</sub> |
| CALLV #8    | XXFFEE <sub>H</sub> | XXFFEF <sub>H</sub> |
| CALLV #9    | XXFFEC <sub>H</sub> | XXFFED <sub>H</sub> |
| CALLV #10   | XXFFEA <sub>H</sub> | XXFFEB <sub>H</sub> |
| CALLV #11   | XXFFE8 <sub>H</sub> | XXFFE9 <sub>H</sub> |
| CALLV #12   | XXFFE6 <sub>H</sub> | XXFFE7 <sub>H</sub> |
| CALLV #13   | XXFFE4 <sub>H</sub> | XXFFE5 <sub>H</sub> |
| CALLV #14   | XXFFE2 <sub>H</sub> | XXFFE3 <sub>H</sub> |
| CALLV #15   | XXFFE0 <sub>H</sub> | XXFFE1 <sub>H</sub> |

Note: A PCB register value is set in XX.

Note:

When the program counter bank register (PCB) is FF<sub>H</sub>, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table D.3-2).

## D.4 Indirect Addressing

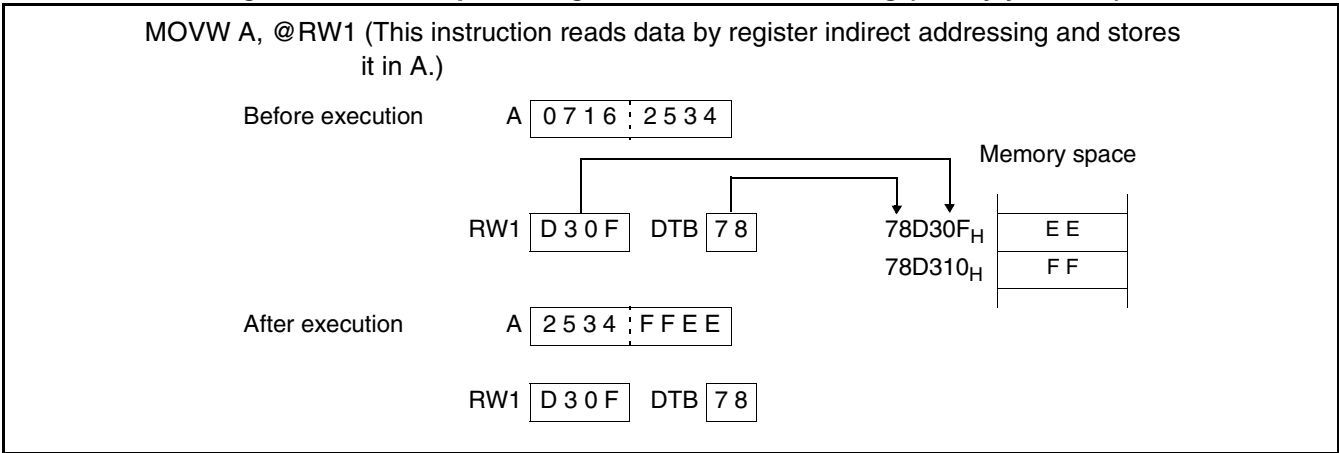
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

### ■ 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.

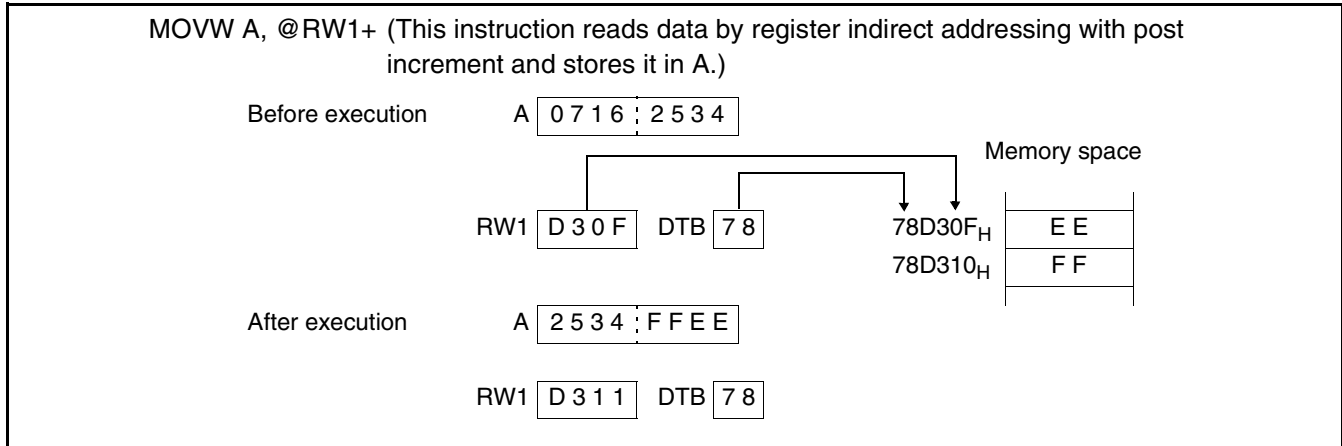
**Figure D.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)**



- Register indirect addressing with post increment (@RWj+ j = 0 to 3)

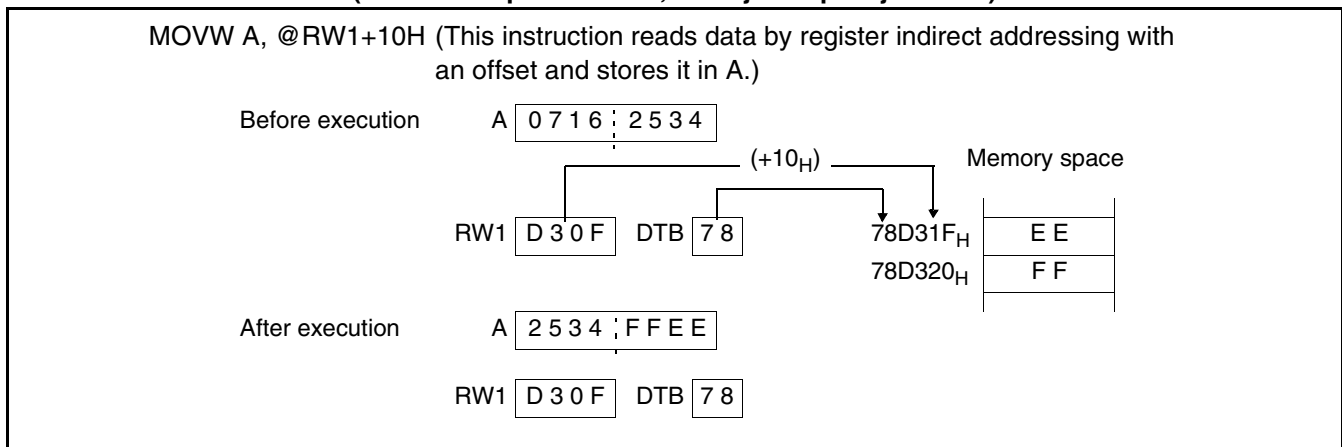
Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system 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 D.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)

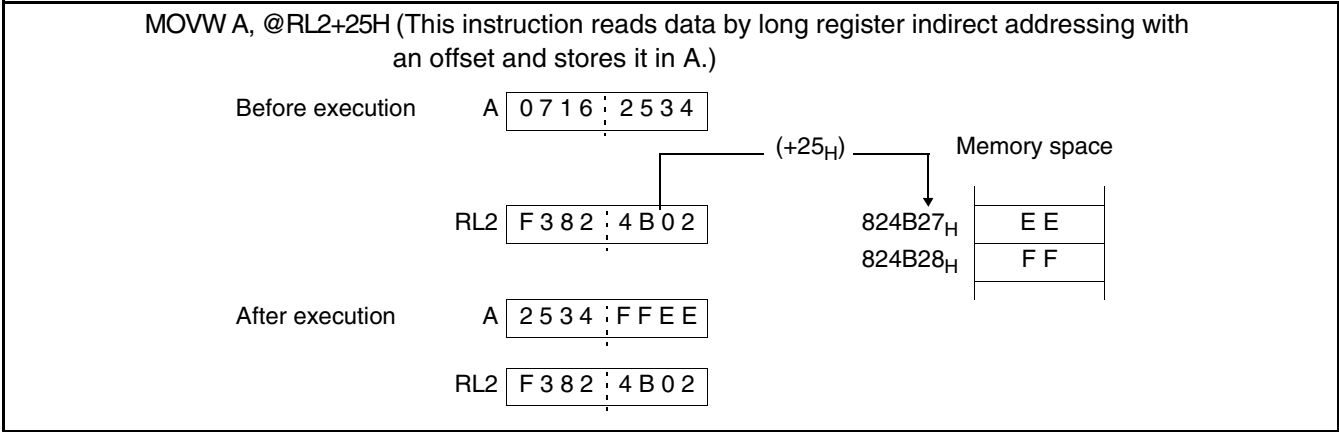
Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or 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 D.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 ( $@RLi + disp8 \ i = 0 \text{ to } 3$ )

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 D.4-4 Example of Long Register Indirect Addressing with Offset ( $@RLi + disp8 \ i = 0 \text{ to } 3$ )**

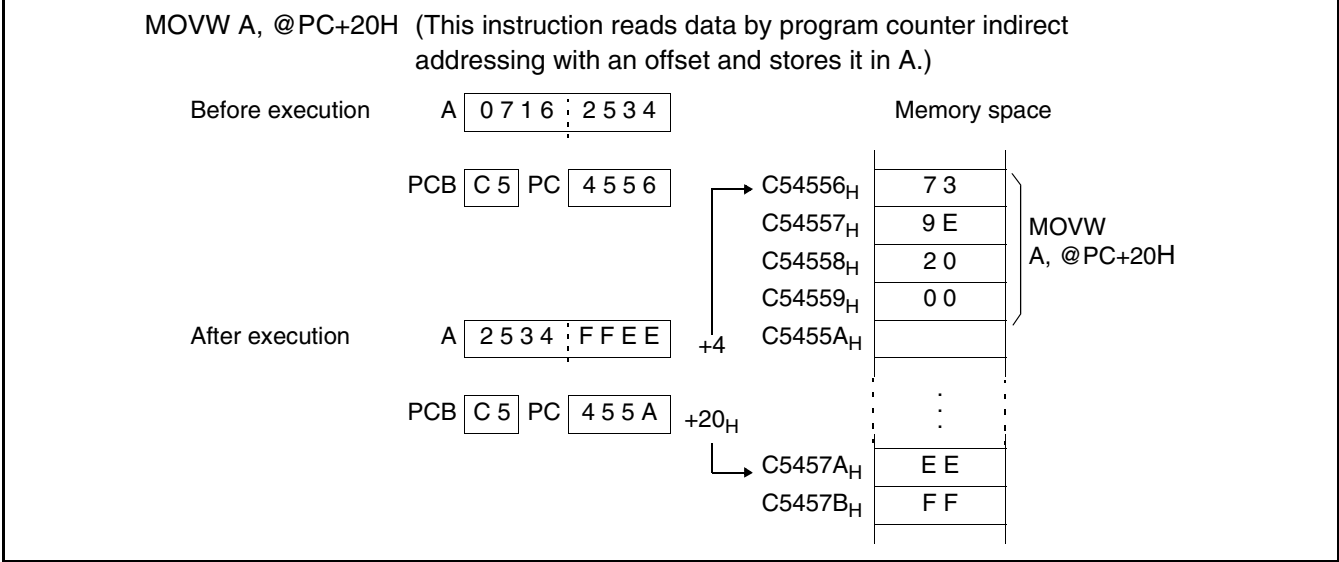


● Program counter indirect addressing with offset ( $@PC + disp16$ )

Memory is accessed using the address indicated by (instruction address + 4 + 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 + disp16):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8
- MOVW eam, #imm16

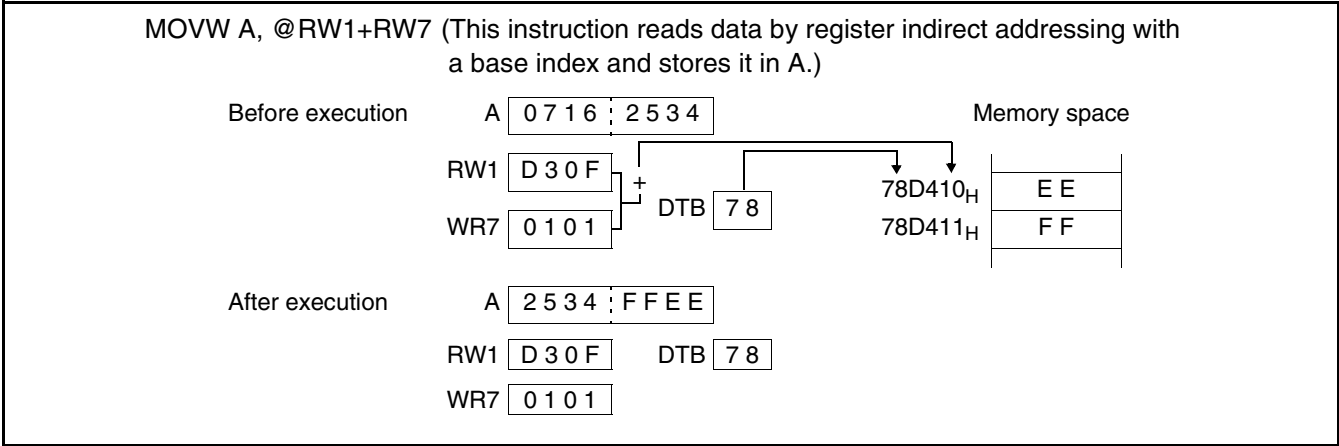
**Figure D.4-5 Example of Program Counter Indirect Addressing with Offset ( $@PC + disp16$ )**



● 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).

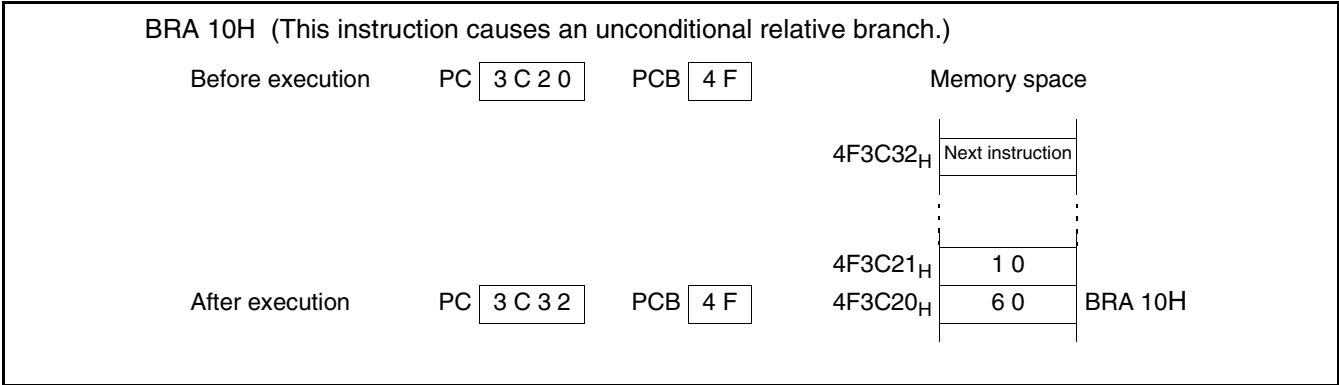
Figure D.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)



● 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 D.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 D.4-8 Configuration of the Register List**

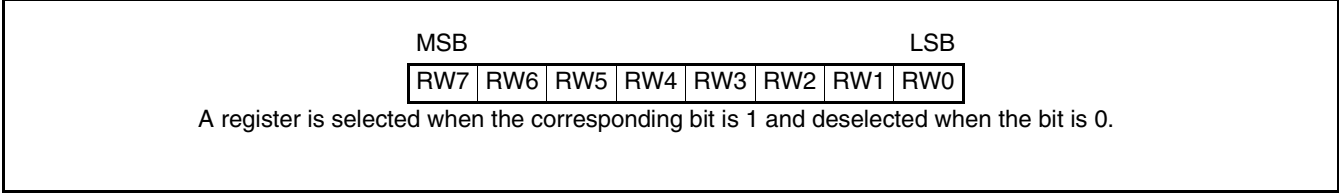
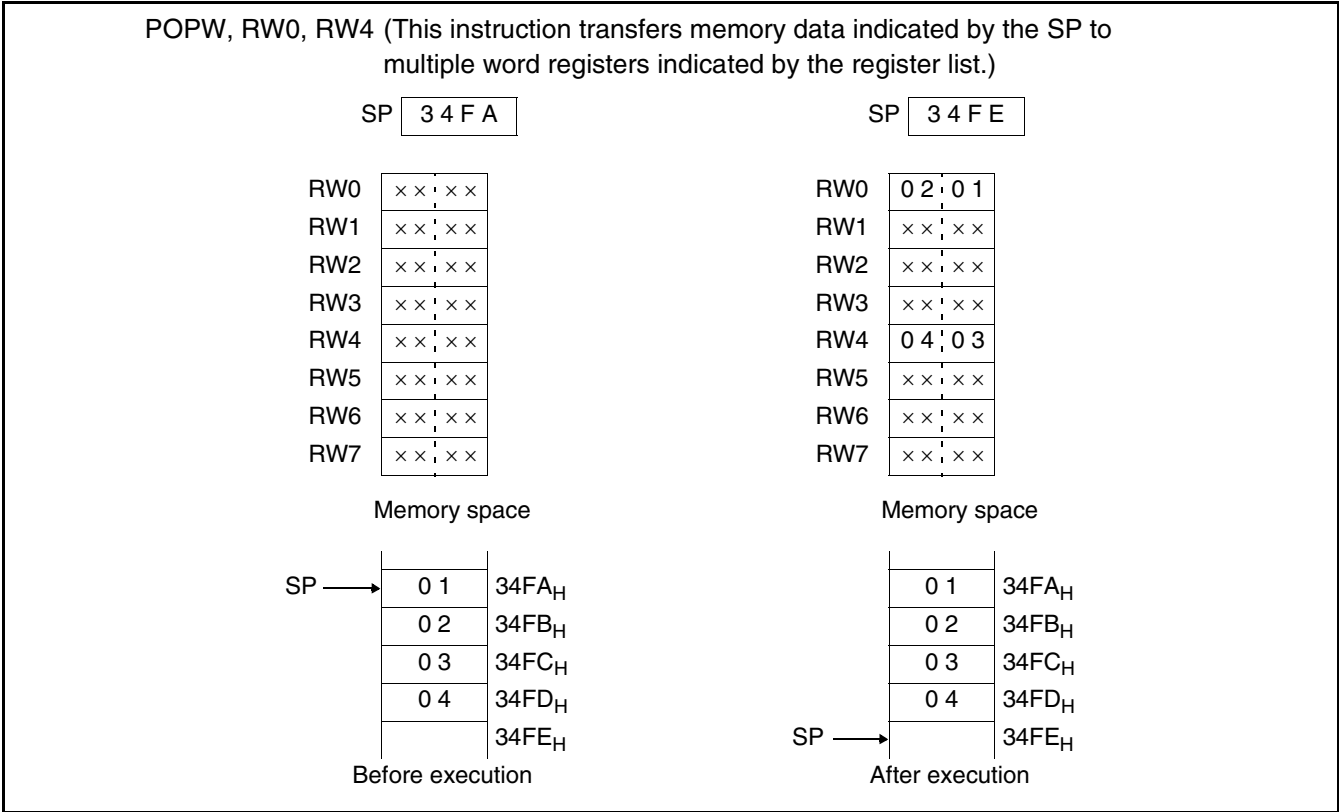


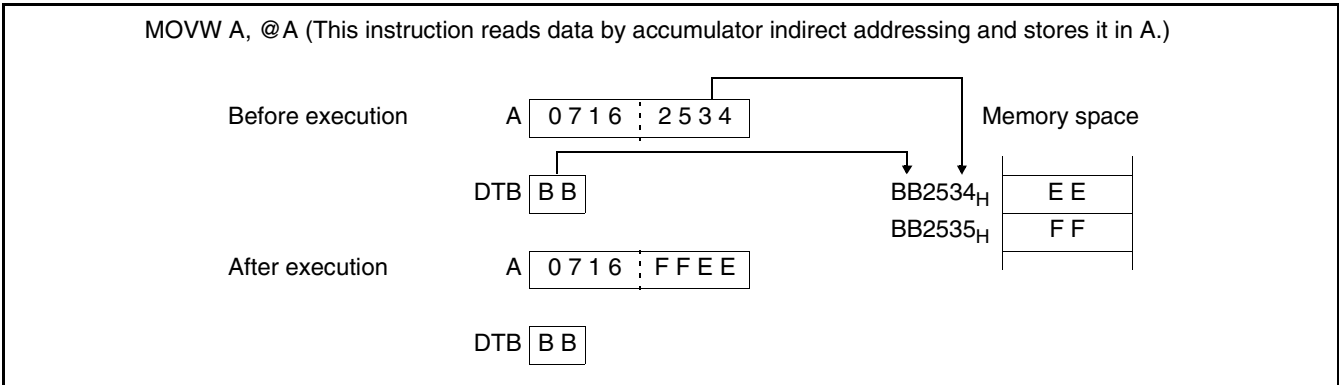
Figure D.4-9 Example of Register List (rlist)



● 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 D.4-10 Example of Accumulator Indirect Addressing (@A)



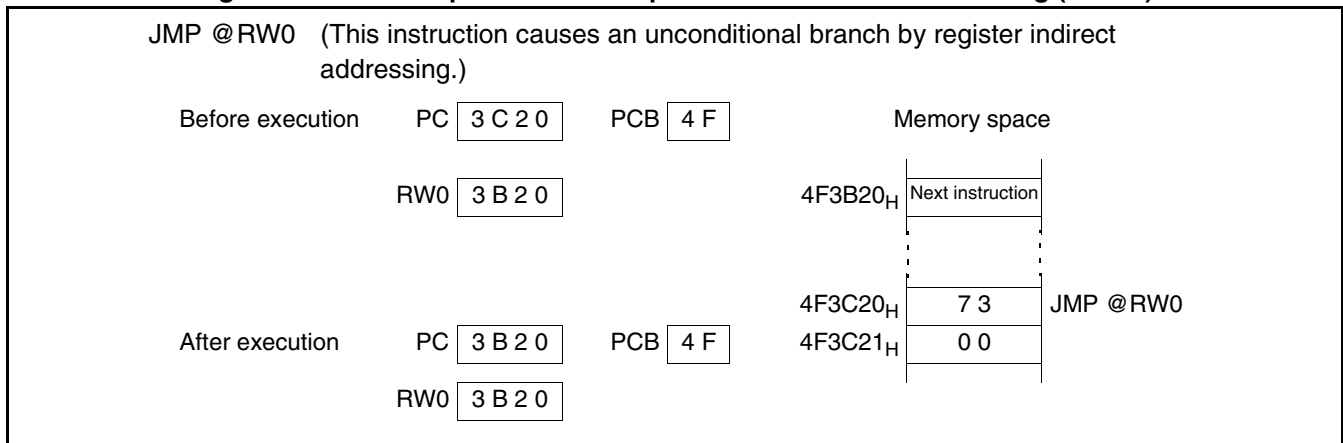




- Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by `eam`.

### Figure D.4-13 Example of Indirect Specification Branch Addressing (@eam)



## D.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 D.5-1 lists execution cycle counts and Table D.5-2 and Table D.5-3 summarize correction value data.

**Table D.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<br> <br>07        | Ri<br>Rwi<br>RLi                             | See the instruction list.                     | See the instruction list.                     |
| 08<br> <br>0B        | @RWj   | 2   | 1   |
| 0C<br> <br>0F        | @RWj+  | 4   | 2   |
| 10<br> <br>17        | @RWi+disp8                                   | 2   | 1   |
| 18<br> <br>1B        | @RWi+disp16                                  | 2   | 1   |
| 1C<br>1D<br>1E<br>1F | @RW0+RW7<br>@RW1+RW7<br>@PC+disp16<br>addr16 | 4<br>4<br>2<br>1                              | 2<br>2<br>0<br>0                              |

\*: (a) is used for ~ (cycle count) and B (correction value) in "D.8 F2MC-16LX Instruction List".

**Table D.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<br>Even address          | +0          | 1            | +0          | 1            | +0          | 2            |
| Internal memory<br>Odd address           | +0          | 1            | +2          | 2            | +4          | 4            |
| External data bus<br>16-bit even address | +1          | 1            | +1          | 1            | +2          | 2            |
| External data bus<br>16-bit odd address  | +1          | 1            | +4          | 2            | +8          | 4            |
| External data bus<br>8-bits              | +1          | 1            | +4          | 2            | +8          | 4            |

\*: (b), (c), and (d) are used for ~ (cycle count) and B (correction value) in "D.8 F2MC-16LX Instruction List".

**Note:**

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

**Table D.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.

## D.6 Effective address field

Table D.6-1 shows the effective address field.

### ■ Effective Address Field

**Table D.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 "D.8 F2MC-16LX Instruction List".

## D.7 How to Read the Instruction List

Table D.7-1 describes the items used in "D.8 F2MC-16LX Instruction List", and Table D.7-2 describes the symbols used in the same list.

### ■ Description of Instruction Presentation Items and Symbols

**Table D.7-1 Description of Items in the Instruction List (1/2)**

| Item      | Description   |
|-----------|---|
| Mnemonic  | Uppercase, symbol: Represented as is in the assembler.<br>Lowercase: Rewritten in the assembler.<br>Number of following lowercase: Indicates bit length in the instruction.   |
| #         | Indicates the number of bytes.  |
| ~         | Indicates the number of cycles.<br>See Table D.2-1 for the alphabetical letters in items.   |
| RG        | Indicates the number of times a register access is performed during instruction execution.<br>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.<br>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.<br>Z: Transfers 0.<br>X: Transfers after sign extension.<br>-: No transfer   |
| AH        | Indicates the special operation for the 16 high-order bits of the accumulator.<br>*: Transfers from AL to AH.<br>-: No transfer<br>Z: Transfers 00 to AH.<br>X: Transfers 00 <sub>H</sub> or FF <sub>H</sub> to AH after AL sign extension. |

**Table D.7-1 Description of Items in the Instruction List (1/2)**

| Item | Description  |
|------|--|
| I    | Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry).<br>*: Changes upon instruction execution.<br>-: No change<br>S: Set upon instruction execution.<br>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).<br>*: Read Modify Write instruction<br>-: Not Read Modify Write instruction<br><b>Note:</b><br>Cannot be used for an address that has different meanings between read and write operations. |

**Table D.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.<br>Byte: Low-order 8 bits of byte AL<br>Word: 16 bits of word AL<br>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 D.7-2 Explanation on Symbols in the Instruction List (1/2)**

| Symbol     | Explanation  |
|------------|--|
| Ri         | R0, R1, R2, R3, R4, R5, R6, R7                                 |
| RWi        | RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7                         |
| RWj        | RW0, RW1, RW2, RW3   |
| RLi        | RL0, RL1, RL2, RL3   |
| dir        | Abbreviated direct addressing                                  |
| addr16     | Direct addressing  |
| addr24     | Physical direct addressing                                     |
| ad24 0-15  | Bit0 to bit15 of addr24  |
| ad24 16-23 | Bit16 to bit23 of addr24                                       |
| io         | I/O area (000000 <sub>H</sub> to 0000FF <sub>H</sub> )         |
| #imm4      | 4-bit immediate data   |
| #imm8      | 8-bit immediate data   |
| #imm16     | 16-bit immediate data  |
| #imm32     | 32-bit immediate data  |
| ext (imm8) | 16-bit data obtained by sign extension of 8-bit immediate data |
| disp8      | 8-bit displacement   |
| disp16     | 16-bit displacement  |
| bp         | Bit offset   |
| vct4       | Vector number (0 to 15)  |
| vct8       | Vector number (0 to 255)                                       |
| ( ) b      | Bit address  |
| rel        | PC relative branch   |
| ear        | Effective addressing (code 00 to 07)                           |
| eam        | Effective addressing (code 08 to 1F)                           |
| rlst       | Register list  |

## D.8 F<sup>2</sup>MC-16LX Instruction List

Table D.8-1 to Table D.8-18 list the instructions used by the F<sup>2</sup>MC-16LX.

### ■ F<sup>2</sup>MC-16LX Instruction List

Table D.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 D.5-1 and Table D.5-2 for information on (a) and (b) in the table.

**Table D.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 SP,A         | 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 D.5-1 and Table D.5-2 for information on (a), (c), and (d) in the table.

**Table D.8-3 42 Addition/Subtraction Instructions (Byte, Word, Long Word)**

| Mnemonic      | #  | ~       | RG | B       | Operation                                 | LH | AH | I | S | T | N | Z | V | C | RMW |
|---------------|----|---------|----|---------|---|----|----|---|---|---|---|---|---|---|-----|
| ADD A,#imm8   | 2  | 2       | 0  | 0       | byte (A) ← (A) + imm8                     | Z  | -  | - | - | - | * | * | * | * | -   |
| ADD A,dir     | 2  | 5       | 0  | (b)     | byte (A) ← (A) + (dir)                    | Z  | -  | - | - | - | * | * | * | * | -   |
| ADD A,ear     | 2  | 3       | 1  | 0       | byte (A) ← (A) + (ear)                    | Z  | -  | - | - | - | * | * | * | * | -   |
| ADD A,eam     | 2+ | 4 + (a) | 0  | (b)     | byte (A) ← (A) + (eam)                    | Z  | -  | - | - | - | * | * | * | * | -   |
| ADD ear,A     | 2  | 3       | 2  | 0       | byte (ear) ← (ear) + (A)                  | -  | -  | - | - | - | * | * | * | * | -   |
| ADD eam,A     | 2+ | 5 + (a) | 0  | 2 × (b) | byte (eam) ← (eam) + (A)                  | Z  | -  | - | - | - | * | * | * | * | *   |
| ADDC A        | 1  | 2       | 0  | 0       | byte (A) ← (AH) + (AL) + (C)              | Z  | -  | - | - | - | * | * | * | * | -   |
| ADDC A,ear    | 2  | 3       | 1  | 0       | byte (A) ← (A) + (ear) + (C)              | Z  | -  | - | - | - | * | * | * | * | -   |
| ADDC A,eam    | 2+ | 4 + (a) | 0  | (b)     | byte (A) ← (A) + (eam) + (C)              | Z  | -  | - | - | - | * | * | * | * | -   |
| ADDC A        | 1  | 3       | 0  | 0       | byte (A) ← (AH) + (AL) + (C)<br>(decimal) | Z  | -  | - | - | - | * | * | * | * | -   |
| SUB A,#imm8   | 2  | 2       | 0  | 0       | byte (A) ← (A) - imm8                     | Z  | -  | - | - | - | * | * | * | * | -   |
| SUB A,dir     | 2  | 5       | 0  | (b)     | byte (A) ← (A) - (dir)                    | Z  | -  | - | - | - | * | * | * | * | -   |
| SUB A,ear     | 2  | 3       | 1  | 0       | byte (A) ← (A) - (ear)                    | Z  | -  | - | - | - | * | * | * | * | -   |
| SUB A,eam     | 2+ | 4 + (a) | 0  | (b)     | byte (A) ← (A) - (eam)                    | Z  | -  | - | - | - | * | * | * | * | -   |
| SUB ear,A     | 2  | 3       | 2  | 0       | byte (ear) ← (ear) - (A)                  | -  | -  | - | - | - | * | * | * | * | -   |
| SUB eam,A     | 2+ | 5 + (a) | 0  | 2 × (b) | byte (eam) ← (eam) - (A)                  | -  | -  | - | - | - | * | * | * | * | *   |
| SUBC A        | 1  | 2       | 0  | 0       | byte (A) ← (AH) - (AL) - (C)              | Z  | -  | - | - | - | * | * | * | * | -   |
| SUBC A,ear    | 2  | 3       | 1  | 0       | byte (A) ← (A) - (ear) - (C)              | Z  | -  | - | - | - | * | * | * | * | -   |
| SUBC A,eam    | 2+ | 4 + (a) | 0  | (b)     | byte (A) ← (A) - (eam) - (C)              | Z  | -  | - | - | - | * | * | * | * | -   |
| SUBC A        | 1  | 3       | 0  | 0       | byte (A) ← (AH) - (AL) - (C)<br>(decimal) | Z  | -  | - | - | - | * | * | * | * | -   |
| ADDW A        | 1  | 2       | 0  | 0       | word (A) ← (AH) + (AL)                    | -  | -  | - | - | - | * | * | * | * | -   |
| ADDW A,ear    | 2  | 3       | 1  | 0       | word (A) ← (A) + (ear)                    | -  | -  | - | - | - | * | * | * | * | -   |
| ADDW A,eam    | 2+ | 4+(a)   | 0  | (c)     | word (A) ← (A) + (eam)                    | -  | -  | - | - | - | * | * | * | * | -   |
| ADDW A,#imm16 | 3  | 2       | 0  | 0       | word (A) ← (A) + imm16                    | -  | -  | - | - | - | * | * | * | * | -   |
| ADDW ear,A    | 2  | 3       | 2  | 0       | word (ear) ← (ear) + (A)                  | -  | -  | - | - | - | * | * | * | * | -   |
| ADDW eam,A    | 2+ | 5+(a)   | 0  | 2 × (c) | word (eam) ← (eam) + (A)                  | -  | -  | - | - | - | * | * | * | * | *   |
| ADDCW A,ear   | 2  | 3       | 1  | 0       | word (A) ← (A) + (ear) + (C)              | -  | -  | - | - | - | * | * | * | * | -   |
| ADDCW A,eam   | 2+ | 4+(a)   | 0  | (c)     | word (A) ← (A) + (eam) + (C)              | -  | -  | - | - | - | * | * | * | * | -   |
| SUBW A        | 1  | 2       | 0  | 0       | word (A) ← (AH) - (AL)                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBW A,ear    | 2  | 3       | 1  | 0       | word (A) ← (A) - (ear)                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBW A,eam    | 2+ | 4+(a)   | 0  | (c)     | word (A) ← (A) - (eam)                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBW A,#imm16 | 3  | 2       | 0  | 0       | word (A) ← (A) - imm16                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBW ear,A    | 2  | 3       | 2  | 0       | word (ear) ← (ear) - (A)                  | -  | -  | - | - | - | * | * | * | * | -   |
| SUBW eam,A    | 2+ | 5+(a)   | 0  | 2 × (c) | word (eam) ← (eam) - (A)                  | -  | -  | - | - | - | * | * | * | * | *   |
| SUBCW A,ear   | 2  | 3       | 1  | 0       | word (A) ← (A) - (ear) - (C)              | -  | -  | - | - | - | * | * | * | * | -   |
| SUBCW A,eam   | 2+ | 4+(a)   | 0  | (c)     | word (A) ← (A) - (eam) - (C)              | -  | -  | - | - | - | * | * | * | * | -   |
| ADDL A,ear    | 2  | 6       | 2  | 0       | long (A) ← (A) + (ear)                    | -  | -  | - | - | - | * | * | * | * | -   |
| ADDL A,eam    | 2+ | 7+(a)   | 0  | (d)     | long (A) ← (A) + (eam)                    | -  | -  | - | - | - | * | * | * | * | -   |
| ADDL A,#imm32 | 5  | 4       | 0  | 0       | long (A) ← (A) + imm32                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBL A,ear    | 2  | 6       | 2  | 0       | long (A) ← (A) - (ear)                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBL A,eam    | 2+ | 7+(a)   | 0  | (d)     | long (A) ← (A) - (eam)                    | -  | -  | - | - | - | * | * | * | * | -   |
| SUBL A,#imm32 | 5  | 4       | 0  | 0       | long (A) ← (A) - imm32                    | -  | -  | - | - | - | * | * | * | * | -   |

Note:

See Table D.5-1 and Table D.5-2 for information on (a) to (d) in the table.

**Table D.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 D.5-1 and Table D.5-2 for information on (a) to (d) in the table.

**Table D.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 D.5-1 and Table D.5-2 for information on (a) to (d) in the table.

**Table D.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)<br>quotient → byte (AL) remainder → byte (AH) | -  | -  | - | - | - | - | - | * | * | -   |
| DIVU A,ear  | 2  | *2  | 1  | 0   | word (A) / byte (ear)<br>quotient → byte (A) remainder → byte (ear) | -  | -  | - | - | - | - | - | * | * | -   |
| DIVU A,eam  | 2+ | *3  | 0  | *6  | word (A) / byte (eam)<br>quotient → byte (A) remainder → byte (eam) | -  | -  | - | - | - | - | - | * | * | -   |
| DIVUW A,ear | 2  | *4  | 1  | 0   | long (A) / word (ear)<br>quotient → word (A) remainder → word (ear) | -  | -  | - | - | - | - | - | * | * | -   |
| DIVUW A,eam | 2+ | *5  | 0  | *7  | long (A) / word (eam)<br>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 D.5-1 and Table D.5-2 for information on (a) to (c) in the table.

**Table D.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)<br>quotient → byte (AL) remainder → byte (AH) | Z  | -  | - | - | - | - | - | * | * | -   |
| DIV A,ear  | 2  | *2  | 1  | 0   | word (A) / byte (ear)<br>quotient → byte (A) remainder → byte (ear) | Z  | -  | - | - | - | - | - | * | * | -   |
| DIV A,eam  | 2+ | *3  | 0  | *6  | word (A) / byte (eam)<br>quotient → byte (A) remainder → byte (eam) | Z  | -  | - | - | - | - | - | * | * | -   |
| DIVW A,ear | 2  | *4  | 1  | 0   | long (A) / word (ear)<br>quotient → word (A) remainder → word (ear) | -  | -  | - | - | - | - | - | * | * | -   |
| DIVW A,eam | 2+ | *5  | 0  | *7  | long (A) / word (eam)<br>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,  $2 \times (b)$ : Normal

\*7: (c): Division by 0 or overflow,  $2 \times (c)$ : 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 D.5-1 and Table D.5-2 for information on (a) to (c) in the table.

Table D.8-8 39 Logic 1 Instructions (Byte, Word)

| Mnemonic      | #  | ~     | RG | B       | Operation                  | LH | AH | I | S | T | N | Z | V | C | RMW |
|---------------|----|-------|----|---------|----------------------------|----|----|---|---|---|---|---|---|---|-----|
| AND A,#imm8   | 2  | 2     | 0  | 0       | byte (A) ← (A) and imm8    | -  | -  | - | - | - | * | * | R | - | -   |
| AND A,ear     | 2  | 3     | 1  | 0       | byte (A) ← (A) and (ear)   | -  | -  | - | - | - | * | * | R | - | -   |
| AND A,eam     | 2+ | 4+(a) | 0  | (b)     | byte (A) ← (A) and (eam)   | -  | -  | - | - | - | * | * | R | - | -   |
| AND ear,A     | 2  | 3     | 2  | 0       | byte (ear) ← (ear) and (A) | -  | -  | - | - | - | * | * | R | - | -   |
| AND eam,A     | 2+ | 5+(a) | 0  | 2 × (b) | byte (eam) ← (eam) and (A) | -  | -  | - | - | - | * | * | R | - | *   |
| OR A,#imm8    | 2  | 2     | 0  | 0       | byte (A) ← (A) or imm8     | -  | -  | - | - | - | * | * | R | - | -   |
| OR A,ear      | 2  | 3     | 1  | 0       | byte (A) ← (A) or (ear)    | -  | -  | - | - | - | * | * | R | - | -   |
| OR A,eam      | 2+ | 4+(a) | 0  | (b)     | byte (A) ← (A) or (eam)    | -  | -  | - | - | - | * | * | R | - | -   |
| OR ear,A      | 2  | 3     | 2  | 0       | byte (ear) ← (ear) or (A)  | -  | -  | - | - | - | * | * | R | - | -   |
| OR eam,A      | 2+ | 5+(a) | 0  | 2 × (b) | byte (eam) ← (eam) or (A)  | -  | -  | - | - | - | * | * | R | - | *   |
| XOR A,#imm8   | 2  | 2     | 0  | 0       | byte (A) ← (A) xor imm8    | -  | -  | - | - | - | * | * | R | - | -   |
| XOR A,ear     | 2  | 3     | 1  | 0       | byte (A) ← (A) xor (ear)   | -  | -  | - | - | - | * | * | R | - | -   |
| XOR A,eam     | 2+ | 4+(a) | 0  | (b)     | byte (A) ← (A) xor (eam)   | -  | -  | - | - | - | * | * | R | - | -   |
| XOR ear,A     | 2  | 3     | 2  | 0       | byte (ear) ← (ear) xor (A) | -  | -  | - | - | - | * | * | R | - | -   |
| XOR eam,A     | 2+ | 5+(a) | 0  | 2 × (b) | byte (eam) ← (eam) xor (A) | -  | -  | - | - | - | * | * | R | - | *   |
| NOT A         | 1  | 2     | 0  | 0       | byte (A) ← not (A)         | -  | -  | - | - | - | * | * | R | - | -   |
| NOT ear       | 2  | 3     | 2  | 0       | byte (ear) ← not (ear)     | -  | -  | - | - | - | * | * | R | - | -   |
| NOT eam       | 2+ | 5+(a) | 0  | 2 × (b) | byte (eam) ← not (eam)     | -  | -  | - | - | - | * | * | R | - | *   |
| ANDW A        | 1  | 2     | 0  | 0       | word (A) ← (AH) and (A)    | -  | -  | - | - | - | * | * | R | - | -   |
| ANDW A,#imm16 | 3  | 2     | 0  | 0       | word (A) ← (A) and imm16   | -  | -  | - | - | - | * | * | R | - | -   |
| ANDW A,ear    | 2  | 3     | 1  | 0       | word (A) ← (A) and (ear)   | -  | -  | - | - | - | * | * | R | - | -   |
| ANDW A,eam    | 2+ | 4+(a) | 0  | (c)     | word (A) ← (A) and (eam)   | -  | -  | - | - | - | * | * | R | - | -   |
| ANDW ear,A    | 2  | 3     | 2  | 0       | word (ear) ← (ear) and (A) | -  | -  | - | - | - | * | * | R | - | -   |
| ANDW eam,A    | 2+ | 5+(a) | 0  | 2 × (c) | word (eam) ← (eam) and (A) | -  | -  | - | - | - | * | * | R | - | *   |
| ORW A         | 1  | 2     | 0  | 0       | word (A) ← (AH) or (A)     | -  | -  | - | - | - | * | * | R | - | -   |
| ORW A,#imm16  | 3  | 2     | 0  | 0       | word (A) ← (A) or imm16    | -  | -  | - | - | - | * | * | R | - | -   |
| ORW A,ear     | 2  | 3     | 1  | 0       | word (A) ← (A) or (ear)    | -  | -  | - | - | - | * | * | R | - | -   |
| ORW A,eam     | 2+ | 4+(a) | 0  | (c)     | word (A) ← (A) or (eam)    | -  | -  | - | - | - | * | * | R | - | -   |
| ORW ear,A     | 2  | 3     | 2  | 0       | word (ear) ← (ear) or (A)  | -  | -  | - | - | - | * | * | R | - | -   |
| ORW eam,A     | 2+ | 5+(a) | 0  | 2 × (c) | word (eam) ← (eam) or (A)  | -  | -  | - | - | - | * | * | R | - | *   |
| XORW A        | 1  | 2     | 0  | 0       | word (A) ← (AH) xor (A)    | -  | -  | - | - | - | * | * | R | - | -   |
| XORW A,#imm16 | 3  | 2     | 0  | 0       | word (A) ← (A) xor imm16   | -  | -  | - | - | - | * | * | R | - | -   |
| XORW A,ear    | 2  | 3     | 1  | 0       | word (A) ← (A) xor (ear)   | -  | -  | - | - | - | * | * | R | - | -   |
| XORW A,eam    | 2+ | 4+(a) | 0  | (c)     | word (A) ← (A) xor (eam)   | -  | -  | - | - | - | * | * | R | - | -   |
| XORW ear,A    | 2  | 3     | 2  | 0       | word (ear) ← (ear) xor (A) | -  | -  | - | - | - | * | * | R | - | -   |
| XORW eam,A    | 2+ | 5+(a) | 0  | 2 × (c) | word (eam) ← (eam) xor (A) | -  | -  | - | - | - | * | * | R | - | *   |
| NOTW A        | 1  | 2     | 0  | 0       | word (A) ← not (A)         | -  | -  | - | - | - | * | * | R | - | -   |
| NOTW ear      | 2  | 3     | 2  | 0       | word (ear) ← not (ear)     | -  | -  | - | - | - | * | * | R | - | -   |
| NOTW eam      | 2+ | 5+(a) | 0  | 2 × (c) | word (eam) ← not (eam)     | -  | -  | - | - | - | * | * | R | - | *   |

Note:

See Table D.5-1 and Table D.5-2 for information on (a) to (c) in the table.



**Table D.8-9 6 Logic 2 Instructions (Long Word)**

| Mnemonic   | #  | ~     | RG | B   | Operation                | LH | AH | I | S | T | N | Z | V | C | RMW |
|------------|----|-------|----|-----|--------------------------|----|----|---|---|---|---|---|---|---|-----|
| ANDL A,ear | 2  | 6     | 2  | 0   | long (A) ← (A) and (ear) | -  | -  | - | - | - | * | * | R | - | -   |
| ANDL A,eam | 2+ | 7+(a) | 0  | (d) | long (A) ← (A) and (eam) | -  | -  | - | - | - | * | * | R | - | -   |
| ORL A,ear  | 2  | 6     | 2  | 0   | long (A) ← (A) or (ear)  | -  | -  | - | - | - | * | * | R | - | -   |
| ORL A,eam  | 2+ | 7+(a) | 0  | (d) | long (A) ← (A) or (eam)  | -  | -  | - | - | - | * | * | R | - | -   |
| XORL A,ear | 2  | 6     | 2  | 0   | long (A) ← (A) xor (ear) | -  | -  | - | - | - | * | * | R | - | -   |
| XORL A,eam | 2+ | 7+(a) | 0  | (d) | long (A) ← (A) xor (eam) | -  | -  | - | - | - | * | * | R | - | -   |

Note:

See Table D.5-1 and Table D.5-2 for information on (a) and (d) in the table.

**Table D.8-10 6 Sign Inversion Instructions (Byte, Word)**

| Mnemonic | #  | ~     | RG | B       | Operation              | LH | AH | I | S | T | N | Z | V | C | RMW |
|----------|----|-------|----|---------|------------------------|----|----|---|---|---|---|---|---|---|-----|
| NEG A    | 1  | 2     | 0  | 0       | byte (A) ← 0 - (A)     | X  | -  | - | - | - | * | * | * | * | -   |
| NEG ear  | 2  | 3     | 2  | 0       | byte (ear) ← 0 - (ear) | -  | -  | - | - | - | * | * | * | * | -   |
| NEG eam  | 2+ | 5+(a) | 0  | 2 × (b) | byte (eam) ← 0 - (eam) | -  | -  | - | - | - | * | * | * | * | *   |
| NEGW A   | 1  | 2     | 0  | 0       | word (A) ← 0 - (A)     | -  | -  | - | - | - | * | * | * | * | -   |
| NEGW ear | 2  | 3     | 2  | 0       | word (ear) ← 0 - (ear) | -  | -  | - | - | - | * | * | * | * | -   |
| NEGW eam | 2+ | 5+(a) | 0  | 2 × (c) | word (eam) ← 0 - (eam) | -  | -  | - | - | - | * | * | * | * | *   |

Note:

See Table D.5-1 and Table D.5-2 for information on (a) to (c) in the table.

**Table D.8-11 1 Normalization Instruction (Long Word)**

| Mnemonic  | # | ~  | RG | B | Operation  | LH | AH | I | S | T | N | Z | V | C | RMW |
|-----------|---|----|----|---|--|----|----|---|---|---|---|---|---|---|-----|
| NRML A,R0 | 2 | *1 | 1  | 0 | long (A) ← Shift left to the position where '1' is set for the first time.<br>byte (R0) ← Shift count at that time | -  | -  | - | - | - | - | * | - | - | -   |

\*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

**Table D.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 D.5-1 and Table D.5-2 for information on (a) and (b) in the table.

**Table D.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/BNE rel     | 2  | *1     | 0  | 0       | Branch on (Z) = 0                         | -  | -  | - | - | - | - | - | - | - | -   |
| BC/BLO rel      | 2  | *1     | 0  | 0       | Branch on (C) = 1                         | -  | -  | - | - | - | - | - | - | - | -   |
| BNC/BHS rel     | 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 × (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 D.5-1 and Table D.5-2 for information on (a) to (d) in the table.

**Table D.8-14 19 Branch 2 Instructions**

| Mnemonic               | #  | ~  | RG | B       | Operation   | LH | AH | I | S | T | N | Z | V | C | RMW |
|------------------------|----|----|----|---------|---|----|----|---|---|---|---|---|---|---|-----|
| CBNE A,#imm8,rel       | 3  | *1 | 0  | 0       | Branch on byte (A) not equal to imm8  | -  | -  | - | - | - | * | * | * | * | -   |
| CWBNE A,#imm16,rel     | 4  | *1 | 0  | 0       | Branch on word (A) not equal to imm16   | -  | -  | - | - | - | * | * | * | * | -   |
| CBNE ear,#imm8,rel     | 4  | *2 | 1  | 0       | Branch on byte (ear) not equal to imm8  | -  | -  | - | - | - | * | * | * | * | -   |
| CBNE eam,#imm8,rel *9  | 4+ | *3 | 0  | (b)     | Branch on byte (eam) not equal to imm8  | -  | -  | - | - | - | * | * | * | * | -   |
| CWBNE ear,#imm16,rel   | 5  | *4 | 1  | 0       | Branch on word (ear) not equal to imm16   | -  | -  | - | - | - | * | * | * | * | -   |
| CWBNE eam,#imm16,rel*9 | 5+ | *3 | 0  | (c)     | Branch on word (eam) not equal to imm16   | -  | -  | - | - | - | * | * | * | * | -   |
| DBNZ ear,rel           | 3  | *5 | 2  | 0       | byte (ear) ← (ear) - 1, Branch on (ear) not equal to 0  | -  | -  | - | - | - | * | * | * | - | -   |
| DBNZ eam,rel           | 3+ | *6 | 2  | 2 × (b) | byte (eam) ← (eam) - 1, Branch on (eam) not equal to 0  | -  | -  | - | - | - | * | * | * | - | *   |
| DWBNZ ear,rel          | 3  | *5 | 2  | 0       | word (ear) ← (ear) - 1, Branch on (ear) not equal to 0  | -  | -  | - | - | - | * | * | * | - | -   |
| DWBNZ eam,rel          | 3+ | *6 | 2  | 2 × (c) | word (eam) ← (eam) - 1, Branch on (eam) not equal to 0  | -  | -  | - | - | - | * | * | * | - | *   |
| INT #vct8              | 2  | 20 | 0  | 8 × (c) | Software interrupt  | -  | -  | R | S | - | - | - | - | - | -   |
| INT addr16             | 3  | 16 | 0  | 6 × (c) | Software interrupt  | -  | -  | R | S | - | - | - | - | - | -   |
| INTP addr24            | 4  | 17 | 0  | 6 × (c) | Software interrupt  | -  | -  | R | S | - | - | - | - | - | -   |
| INT9                   | 1  | 20 | 0  | 8 × (c) | Software interrupt  | -  | -  | R | S | - | - | - | - | - | -   |
| RETI                   | 1  | *8 | 0  | *7      | Return from interrupt   | -  | -  | * | * | * | * | * | * | * | -   |
| LINK #imm8             | 2  | 6  | 0  | (c)     | Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area. | -  | -  | - | - | - | - | - | - | - | -   |
| UNLINK                 | 1  | 5  | 0  | (c)     | Recovers the old frame pointer from the stack upon exiting the function.  | -  | -  | - | - | - | - | - | - | - | -   |
| RET *10                | 1  | 4  | 0  | (c)     | Return from subroutine  | -  | -  | - | - | - | - | - | - | - | -   |
| RETP *11               | 1  | 6  | 0  | (d)     | Return from subroutine  | -  | -  | - | - | - | - | - | - | - | -   |

\*1: 5 when a branch is made; otherwise, 4

\*2: 13 when a branch is made; otherwise, 12

\*3: 7+(a) when a branch is made; otherwise, 6+(a)

\*4: 8 when a branch is made; otherwise, 7

\*5: 7 when a branch is made; otherwise, 6

\*6: 8+(a) when a branch is made; otherwise, 7+(a)

\*7: 3 × (b) + 2 × (c) when jumping to the next interruption request; 6 × (c) when returning from the current interruption

\*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption

\*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.

\*10: Return from stack (word)

\*11: Return from stack (long word)

**Note:**

See Table D.5-1 and Table D.5-2 for information on (a) to (d) in the table.

**Table D.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 D.5-1 and Table D.5-2 for information on (a) and (c) in the table.

**Table D.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,<br>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 D.5-1 and Table D.5-2 for information on (b) in the table.

**Table D.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 D.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

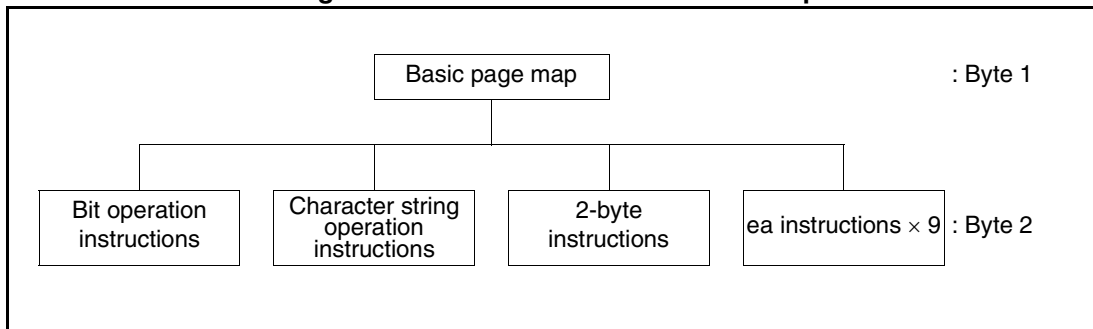
See Table D.5-1 and Table D.5-2 for information on (b) and (c) in the table.

## D.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 D.9-2 to Table D.9-21 summarize the F<sup>2</sup>MC-16LX instruction map.

### ■ Structure of Instruction Map

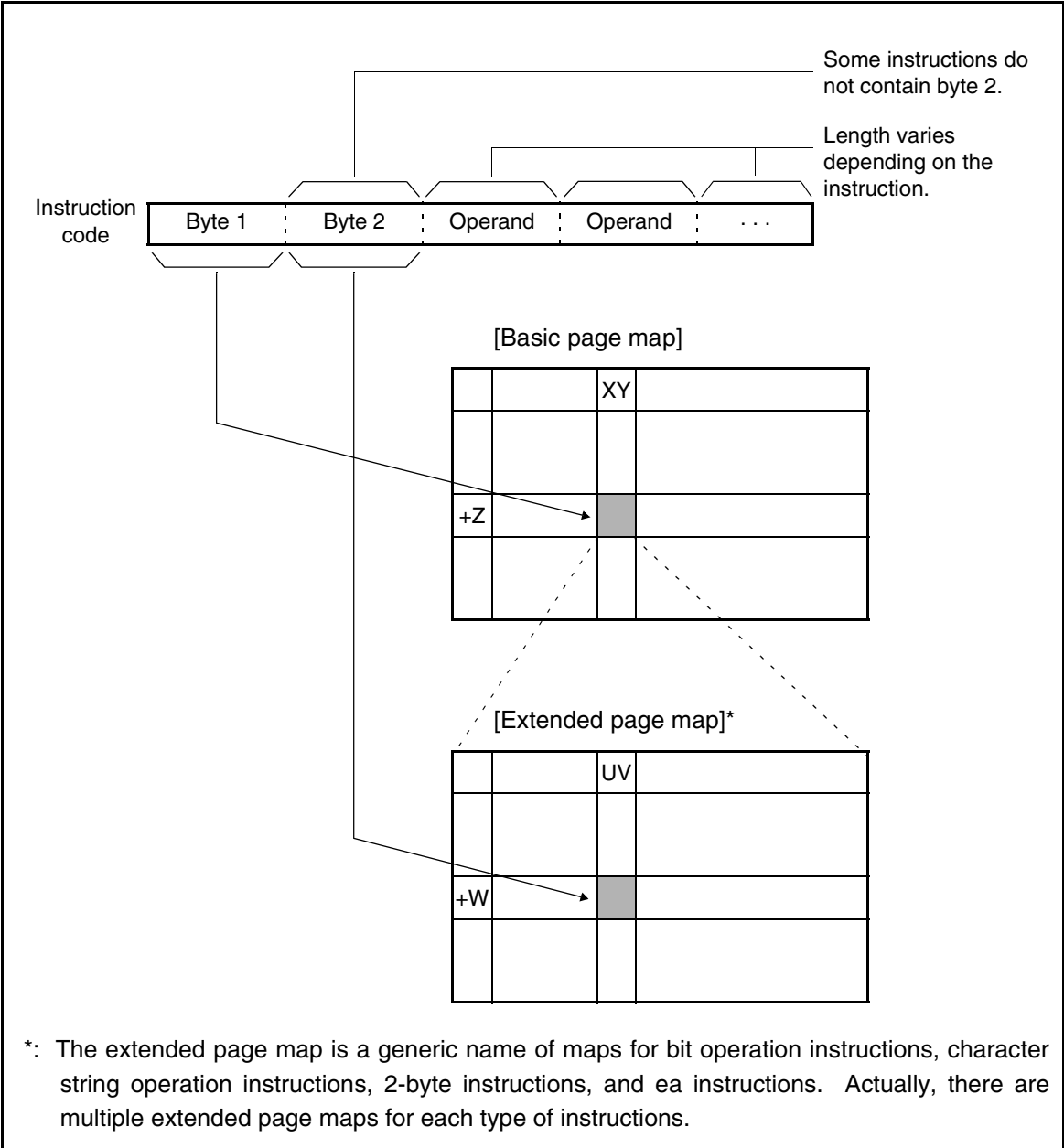
Figure D.9-1 Structure of Instruction Map



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 D.9-2 shows the correspondence between an actual instruction code and instruction map.



**Figure D.9-2 Correspondence between Actual Instruction Code and Instruction Map**



An example of an instruction code is shown in Table D.9-1.

**Table D.9-1 Example of an Instruction Code**

| Instruction      | Byte 1<br>(from basic page map) | Byte 2<br>(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 D.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 | BZBEQ 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 | CBNE 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 r1st  | POPW r1st      | 2-byte instruction                     | XCHW RWI, ea     |             |             |               |                 |                 |           |         | BHI rel     |

Table D.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<br>A, io:bp  |                     | MOVB<br>io:bp, A  |                     | CLRB<br>io:bp  |                  | SETB<br>io:bp  |                  | BBC<br>io:bp, rel  |                      | BBS<br>io:bp, rel  |                      | WBTS<br>io:bp |    | WBTC<br>io:bp |                  |
| +1 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +2 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +3 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +4 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +5 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +6 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +7 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +8 | MOVB<br>A, dir:bp | MOVB A,<br>addr16bp | MOVB<br>dir:bp, A | MOVB<br>addr16bp, A | CLRB<br>dir:bp | CLRB<br>addr16bp | SETB<br>dir:bp | SETB<br>addr16bp | BBC<br>dir:bp, rel | BBC<br>addr16bp, rel | BBS<br>dir:bp, rel | BBS<br>addr16bp, rel |               |    |               | SBBS<br>addr16bp |
| +9 |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +A |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +B |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +C |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +D |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +E |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |
| +F |                   |                     |                   |                     |                |                  |                |                  |                    |                      |                    |                      |               |    |               |                  |

Table D.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 |    |    |    |    | SCWEQI<br>PCB | SCEQD<br>PCB | SCWEQI<br>PCB | SCWEQD<br>PCB | FILSI<br>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 D.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<br>A, DTB | MOV<br>DTB, A   | MOVX<br>A, @RL0+d8 | MOV<br>@RL0+d8, A  | MOV<br>@RL0+d8, A  |    |           |    |    |    |    |    |    |    |    |    |
| +1 | MOV<br>A, ADB | MOV<br>ADB, A   |                    |                    |                    |    |           |    |    |    |    |    |    |    |    |    |
| +2 | MOV<br>A, SSB | MOV<br>SSB, A   | MOVX<br>A, @RL1+d8 | MOV<br>@RL1+d8, A  | MOV<br>@RL1+d8, A  |    |           |    |    |    |    |    |    |    |    |    |
| +3 | MOV<br>A, USB | MOV<br>USB, A   |                    |                    |                    |    |           |    |    |    |    |    |    |    |    |    |
| +4 | MOV<br>A, DPR | MOV<br>DPR, A   | MOVX<br>A, @RL2+d8 | MOV<br>@RL2+d8, A  | MOV<br>@RL2+d8, A  |    |           |    |    |    |    |    |    |    |    |    |
| +5 | MOV<br>A, @A  | MOV<br>@AL, AH  |                    |                    |                    |    |           |    |    |    |    |    |    |    |    |    |
| +6 | MOV<br>A, PCB | MOV<br>A, @A    | MOVX<br>A, @RL3+d8 | MOV<br>@RL3+d8, A  | MOV<br>@RL3+d8, A  |    |           |    |    |    |    |    |    |    |    |    |
| +7 | ROL<br>A      | ROL<br>A        |                    |                    |                    |    |           |    |    |    |    |    |    |    |    |    |
| +8 |               |                 |                    | MOVW<br>@RL0+d8, A | MOVW<br>@RL0+d8, A |    | MUL<br>A  |    |    |    |    |    |    |    |    |    |
| +9 |               |                 |                    |                    |                    |    | MULW<br>A |    |    |    |    |    |    |    |    |    |
| +A |               |                 |                    | MOVW<br>@RL1+d8, A | MOVW<br>@RL1+d8, A |    | DIVU<br>A |    |    |    |    |    |    |    |    |    |
| +B |               |                 |                    |                    |                    |    |           |    |    |    |    |    |    |    |    |    |
| +C | LSLW<br>A, R0 | LSLL<br>A, R0   | LSL<br>A, R0       | MOVW<br>@RL2+d8, A | MOVW<br>@RL2+d8, A |    |           |    |    |    |    |    |    |    |    |    |
| +D | MOVW<br>A, @A | MOVW<br>@AL, AH | NRML<br>A, R0      |                    |                    |    |           |    |    |    |    |    |    |    |    |    |
| +E | ASRW<br>A, R0 | ASRL<br>A, R0   | ASR<br>A, R0       | MOVW<br>@RL3+d8, A | MOVW<br>@RL3+d8, A |    |           |    |    |    |    |    |    |    |    |    |
| +F | LSRW<br>A, R0 | LSRL<br>A, R0   | LSR<br>A, R0       |                    |                    |    |           |    |    |    |    |    |    |    |    |    |

Table D.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 ↓                  |                  |
| +0 | ADDL A, RLO', @RW0+d8 | ADDL A, RLO', @RW0+d8 | SUBL A, RLO', @RW0+d8 | SUBL A, RLO', @RW0+d8 | RW0', @RW0+d8 #16, rel   | CMPL A, RLO', @RW0+d8 | CMPL A, RLO', @RW0+d8 | CMPL A, RLO', @RW0+d8 | ANDL A, RLO', @RW0+d8 | ANDL A, RLO', @RW0+d8 | ORL A, RLO', @RW0+d8 | ORL A, RLO', @RW0+d8 | XORL A, RLO', @RW0+d8 | XORL A, RLO', @RW0+d8 | R0', @RW0+d8 #8, rel    |                  |
| +1 | ADDL A, RLO', @RW1+d8 | ADDL A, RLO', @RW1+d8 | SUBL A, RLO', @RW1+d8 | SUBL A, RLO', @RW1+d8 | RW1', @RW1+d8 #16, rel   | CMPL A, RLO', @RW1+d8 | CMPL A, RLO', @RW1+d8 | CMPL A, RLO', @RW1+d8 | ANDL A, RLO', @RW1+d8 | ANDL A, RLO', @RW1+d8 | ORL A, RLO', @RW1+d8 | ORL A, RLO', @RW1+d8 | XORL A, RLO', @RW1+d8 | XORL A, RLO', @RW1+d8 | R1', @RW1+d8 #8, rel    |                  |
| +2 | ADDL A, RL1', @RW2+d8 | ADDL A, RL1', @RW2+d8 | SUBL A, RL1', @RW2+d8 | SUBL A, RL1', @RW2+d8 | RW2', @RW2+d8 #16, rel   | CMPL A, RL1', @RW2+d8 | CMPL A, RL1', @RW2+d8 | CMPL A, RL1', @RW2+d8 | ANDL A, RL1', @RW2+d8 | ANDL A, RL1', @RW2+d8 | ORL A, RL1', @RW2+d8 | ORL A, RL1', @RW2+d8 | XORL A, RL1', @RW2+d8 | XORL A, RL1', @RW2+d8 | R2', @RW2+d8 #8, rel    |                  |
| +3 | ADDL A, RL1', @RW3+d8 | ADDL A, RL1', @RW3+d8 | SUBL A, RL1', @RW3+d8 | SUBL A, RL1', @RW3+d8 | RW3', @RW3+d8 #16, rel   | CMPL A, RL1', @RW3+d8 | CMPL A, RL1', @RW3+d8 | CMPL A, RL1', @RW3+d8 | ANDL A, RL1', @RW3+d8 | ANDL A, RL1', @RW3+d8 | ORL A, RL1', @RW3+d8 | ORL A, RL1', @RW3+d8 | XORL A, RL1', @RW3+d8 | XORL A, RL1', @RW3+d8 | R3', @RW3+d8 #8, rel    |                  |
| +4 | ADDL A, RL2', @RW4+d8 | ADDL A, RL2', @RW4+d8 | SUBL A, RL2', @RW4+d8 | SUBL A, RL2', @RW4+d8 | RW4', @RW4+d8 #16, rel   | CMPL A, RL2', @RW4+d8 | CMPL A, RL2', @RW4+d8 | CMPL A, RL2', @RW4+d8 | ANDL A, RL2', @RW4+d8 | ANDL A, RL2', @RW4+d8 | ORL A, RL2', @RW4+d8 | ORL A, RL2', @RW4+d8 | XORL A, RL2', @RW4+d8 | XORL A, RL2', @RW4+d8 | R4', @RW4+d8 #8, rel    |                  |
| +5 | ADDL A, RL2', @RW5+d8 | ADDL A, RL2', @RW5+d8 | SUBL A, RL2', @RW5+d8 | SUBL A, RL2', @RW5+d8 | RW5', @RW5+d8 #16, rel   | CMPL A, RL2', @RW5+d8 | CMPL A, RL2', @RW5+d8 | CMPL A, RL2', @RW5+d8 | ANDL A, RL2', @RW5+d8 | ANDL A, RL2', @RW5+d8 | ORL A, RL2', @RW5+d8 | ORL A, RL2', @RW5+d8 | XORL A, RL2', @RW5+d8 | XORL A, RL2', @RW5+d8 | R5', @RW5+d8 #8, rel    |                  |
| +6 | ADDL A, RL3', @RW6+d8 | ADDL A, RL3', @RW6+d8 | SUBL A, RL3', @RW6+d8 | SUBL A, RL3', @RW6+d8 | RW6', @RW6+d8 #16, rel   | CMPL A, RL3', @RW6+d8 | CMPL A, RL3', @RW6+d8 | CMPL A, RL3', @RW6+d8 | ANDL A, RL3', @RW6+d8 | ANDL A, RL3', @RW6+d8 | ORL A, RL3', @RW6+d8 | ORL A, RL3', @RW6+d8 | XORL A, RL3', @RW6+d8 | XORL A, RL3', @RW6+d8 | R6', @RW6+d8 #8, rel    |                  |
| +7 | ADDL A, RL3', @RW7+d8 | ADDL A, RL3', @RW7+d8 | SUBL A, RL3', @RW7+d8 | SUBL A, RL3', @RW7+d8 | RW7', @RW7+d8 #16, rel   | CMPL A, RL3', @RW7+d8 | CMPL A, RL3', @RW7+d8 | CMPL A, RL3', @RW7+d8 | ANDL A, RL3', @RW7+d8 | ANDL A, RL3', @RW7+d8 | ORL A, RL3', @RW7+d8 | ORL A, RL3', @RW7+d8 | XORL A, RL3', @RW7+d8 | XORL A, RL3', @RW7+d8 | R7', @RW7+d8 #8, rel    |                  |
| +8 | ADDL A, @RW0          | ADDL A, @RW0+d16      | SUBL A, @RW0          | SUBL A, @RW0+d16      | @RW0', @RW0+d16 #16, rel | CMPL A, @RW0          | CMPL A, @RW0          | CMPL A, @RW0          | ANDL A, @RW0          | ANDL A, @RW0          | ORL A, @RW0          | ORL A, @RW0          | XORL A, @RW0          | XORL A, @RW0          | @RW0', @RW0+d16 #8, rel |                  |
| +9 | ADDL A, @RW1          | ADDL A, @RW1+d16      | SUBL A, @RW1          | SUBL A, @RW1+d16      | @RW1', @RW1+d16 #16, rel | CMPL A, @RW1          | CMPL A, @RW1          | CMPL A, @RW1          | ANDL A, @RW1          | ANDL A, @RW1          | ORL A, @RW1          | ORL A, @RW1          | XORL A, @RW1          | XORL A, @RW1          | @RW1', @RW1+d16 #8, rel |                  |
| +A | ADDL A, @RW2          | ADDL A, @RW2+d16      | SUBL A, @RW2          | SUBL A, @RW2+d16      | @RW2', @RW2+d16 #16, rel | CMPL A, @RW2          | CMPL A, @RW2          | CMPL A, @RW2          | ANDL A, @RW2          | ANDL A, @RW2          | ORL A, @RW2          | ORL A, @RW2          | XORL A, @RW2          | XORL A, @RW2          | @RW2', @RW2+d16 #8, rel |                  |
| +B | ADDL A, @RW3          | ADDL A, @RW3+d16      | SUBL A, @RW3          | SUBL A, @RW3+d16      | @RW3', @RW3+d16 #16, rel | CMPL A, @RW3          | CMPL A, @RW3          | CMPL A, @RW3          | ANDL A, @RW3          | ANDL A, @RW3          | ORL A, @RW3          | ORL A, @RW3          | XORL A, @RW3          | XORL A, @RW3          | @RW3', @RW3+d16 #8, rel |                  |
| +C | ADDL A, @RW0+         | ADDL A, @RW0+RW7      | SUBL A, @RW0+         | SUBL A, @RW0+RW7      | Use prohibited           | CMPL A, @RW0+         | CMPL A, @RW0+         | CMPL A, @RW0+         | ANDL A, @RW0+         | ANDL A, @RW0+         | ORL A, @RW0+         | ORL A, @RW0+         | XORL A, @RW0+         | XORL A, @RW0+         | Use prohibited          | @RW0+RW7 #8, rel |
| +D | ADDL A, @RW1+         | ADDL A, @RW1+RW7      | SUBL A, @RW1+         | SUBL A, @RW1+RW7      | Use prohibited           | CMPL A, @RW1+         | CMPL A, @RW1+         | CMPL A, @RW1+         | ANDL A, @RW1+         | ANDL A, @RW1+         | ORL A, @RW1+         | ORL A, @RW1+         | XORL A, @RW1+         | XORL A, @RW1+         | Use prohibited          | @RW1+RW7 #8, rel |
| +E | ADDL A, @PC+d16       | ADDL A, @PC+d16       | SUBL A, @PC+d16       | SUBL A, @PC+d16       | Use prohibited           | CMPL A, @PC+d16       | CMPL A, @PC+d16       | CMPL A, @PC+d16       | ANDL A, @PC+d16       | ANDL A, @PC+d16       | ORL A, @PC+d16       | ORL A, @PC+d16       | XORL A, @PC+d16       | XORL A, @PC+d16       | Use prohibited          | @PC+d16 #8, rel  |
| +F | ADDL A, @RW3+         | ADDL A, @RW3+addr16   | SUBL A, @RW3+         | SUBL A, @RW3+addr16   | Use prohibited           | CMPL A, @RW3+         | CMPL A, @RW3+         | CMPL A, @RW3+         | ANDL A, @RW3+         | ANDL A, @RW3+         | ORL A, @RW3+         | ORL A, @RW3+         | XORL A, @RW3+         | XORL A, @RW3+         | Use prohibited          | addr16 #8, rel   |

Table D.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<br>@ RLO, @ RW0+d8  | JMPP @<br>@ RLO, @ RW0+d8  | CALLP<br>@ RLO, @ RW0+d8  | CALLP @<br>@ RLO, @ RW0+d8  | INCL<br>RLO, @ RW0+d8  | INCL<br>RLO, @ RW0+d8  | DECL<br>RLO, @ RW0+d8  | DECL<br>RLO, @ RW0+d8  | MOVL<br>A, RLO, @ RW0+d8  | MOVL<br>A, RLO, @ RW0+d8  | MOVL<br>RLO, A, @ RW0+d8  | MOVL<br>RLO, A, @ RW0+d8  | MOV<br>R0, #8, @ RW0+d8  | MOV<br>R0, #8, @ RW0+d8  | MOVEA<br>A, RLO, @ RW0+d8  | MOVEA<br>A, RLO, @ RW0+d8  |
| +1 | JMPP<br>@ RLO, @ RW1+d8  | JMPP @<br>@ RLO, @ RW1+d8  | CALLP<br>@ RLO, @ RW1+d8  | CALLP @<br>@ RLO, @ RW1+d8  | INCL<br>RLO, @ RW1+d8  | INCL<br>RLO, @ RW1+d8  | DECL<br>RLO, @ RW1+d8  | DECL<br>RLO, @ RW1+d8  | MOVL<br>A, RLO, @ RW1+d8  | MOVL<br>A, RLO, @ RW1+d8  | MOVL<br>RLO, A, @ RW1+d8  | MOVL<br>RLO, A, @ RW1+d8  | MOV<br>R1, #8, @ RW1+d8  | MOV<br>R1, #8, @ RW1+d8  | MOVEA<br>A, RLO, @ RW1+d8  | MOVEA<br>A, RLO, @ RW1+d8  |
| +2 | JMPP<br>@ RLO, @ RW2+d8  | JMPP @<br>@ RLO, @ RW2+d8  | CALLP<br>@ RLO, @ RW2+d8  | CALLP @<br>@ RLO, @ RW2+d8  | INCL<br>RLO, @ RW2+d8  | INCL<br>RLO, @ RW2+d8  | DECL<br>RLO, @ RW2+d8  | DECL<br>RLO, @ RW2+d8  | MOVL<br>A, RLO, @ RW2+d8  | MOVL<br>A, RLO, @ RW2+d8  | MOVL<br>RLO, A, @ RW2+d8  | MOVL<br>RLO, A, @ RW2+d8  | MOV<br>R2, #8, @ RW2+d8  | MOV<br>R2, #8, @ RW2+d8  | MOVEA<br>A, RLO, @ RW2+d8  | MOVEA<br>A, RLO, @ RW2+d8  |
| +3 | JMPP<br>@ RLO, @ RW3+d8  | JMPP @<br>@ RLO, @ RW3+d8  | CALLP<br>@ RLO, @ RW3+d8  | CALLP @<br>@ RLO, @ RW3+d8  | INCL<br>RLO, @ RW3+d8  | INCL<br>RLO, @ RW3+d8  | DECL<br>RLO, @ RW3+d8  | DECL<br>RLO, @ RW3+d8  | MOVL<br>A, RLO, @ RW3+d8  | MOVL<br>A, RLO, @ RW3+d8  | MOVL<br>RLO, A, @ RW3+d8  | MOVL<br>RLO, A, @ RW3+d8  | MOV<br>R3, #8, @ RW3+d8  | MOV<br>R3, #8, @ RW3+d8  | MOVEA<br>A, RLO, @ RW3+d8  | MOVEA<br>A, RLO, @ RW3+d8  |
| +4 | JMPP<br>@ RLO, @ RW4+d8  | JMPP @<br>@ RLO, @ RW4+d8  | CALLP<br>@ RLO, @ RW4+d8  | CALLP @<br>@ RLO, @ RW4+d8  | INCL<br>RLO, @ RW4+d8  | INCL<br>RLO, @ RW4+d8  | DECL<br>RLO, @ RW4+d8  | DECL<br>RLO, @ RW4+d8  | MOVL<br>A, RLO, @ RW4+d8  | MOVL<br>A, RLO, @ RW4+d8  | MOVL<br>RLO, A, @ RW4+d8  | MOVL<br>RLO, A, @ RW4+d8  | MOV<br>R4, #8, @ RW4+d8  | MOV<br>R4, #8, @ RW4+d8  | MOVEA<br>A, RLO, @ RW4+d8  | MOVEA<br>A, RLO, @ RW4+d8  |
| +5 | JMPP<br>@ RLO, @ RW5+d8  | JMPP @<br>@ RLO, @ RW5+d8  | CALLP<br>@ RLO, @ RW5+d8  | CALLP @<br>@ RLO, @ RW5+d8  | INCL<br>RLO, @ RW5+d8  | INCL<br>RLO, @ RW5+d8  | DECL<br>RLO, @ RW5+d8  | DECL<br>RLO, @ RW5+d8  | MOVL<br>A, RLO, @ RW5+d8  | MOVL<br>A, RLO, @ RW5+d8  | MOVL<br>RLO, A, @ RW5+d8  | MOVL<br>RLO, A, @ RW5+d8  | MOV<br>R5, #8, @ RW5+d8  | MOV<br>R5, #8, @ RW5+d8  | MOVEA<br>A, RLO, @ RW5+d8  | MOVEA<br>A, RLO, @ RW5+d8  |
| +6 | JMPP<br>@ RLO, @ RW6+d8  | JMPP @<br>@ RLO, @ RW6+d8  | CALLP<br>@ RLO, @ RW6+d8  | CALLP @<br>@ RLO, @ RW6+d8  | INCL<br>RLO, @ RW6+d8  | INCL<br>RLO, @ RW6+d8  | DECL<br>RLO, @ RW6+d8  | DECL<br>RLO, @ RW6+d8  | MOVL<br>A, RLO, @ RW6+d8  | MOVL<br>A, RLO, @ RW6+d8  | MOVL<br>RLO, A, @ RW6+d8  | MOVL<br>RLO, A, @ RW6+d8  | MOV<br>R6, #8, @ RW6+d8  | MOV<br>R6, #8, @ RW6+d8  | MOVEA<br>A, RLO, @ RW6+d8  | MOVEA<br>A, RLO, @ RW6+d8  |
| +7 | JMPP<br>@ RLO, @ RW7+d8  | JMPP @<br>@ RLO, @ RW7+d8  | CALLP<br>@ RLO, @ RW7+d8  | CALLP @<br>@ RLO, @ RW7+d8  | INCL<br>RLO, @ RW7+d8  | INCL<br>RLO, @ RW7+d8  | DECL<br>RLO, @ RW7+d8  | DECL<br>RLO, @ RW7+d8  | MOVL<br>A, RLO, @ RW7+d8  | MOVL<br>A, RLO, @ RW7+d8  | MOVL<br>RLO, A, @ RW7+d8  | MOVL<br>RLO, A, @ RW7+d8  | MOV<br>R7, #8, @ RW7+d8  | MOV<br>R7, #8, @ RW7+d8  | MOVEA<br>A, RLO, @ RW7+d8  | MOVEA<br>A, RLO, @ RW7+d8  |
| +8 | JMPP<br>@ RLO, @ RW0+d16 | JMPP @<br>@ RLO, @ RW0+d16 | CALLP<br>@ RLO, @ RW0+d16 | CALLP @<br>@ RLO, @ RW0+d16 | INCL<br>RLO, @ RW0+d16 | INCL<br>RLO, @ RW0+d16 | DECL<br>RLO, @ RW0+d16 | DECL<br>RLO, @ RW0+d16 | MOVL<br>A, RLO, @ RW0+d16 | MOVL<br>A, RLO, @ RW0+d16 | MOVL<br>RLO, A, @ RW0+d16 | MOVL<br>RLO, A, @ RW0+d16 | MOV<br>R0, #8, @ RW0+d16 | MOV<br>R0, #8, @ RW0+d16 | MOVEA<br>A, RLO, @ RW0+d16 | MOVEA<br>A, RLO, @ RW0+d16 |
| +9 | JMPP<br>@ RLO, @ RW1+d16 | JMPP @<br>@ RLO, @ RW1+d16 | CALLP<br>@ RLO, @ RW1+d16 | CALLP @<br>@ RLO, @ RW1+d16 | INCL<br>RLO, @ RW1+d16 | INCL<br>RLO, @ RW1+d16 | DECL<br>RLO, @ RW1+d16 | DECL<br>RLO, @ RW1+d16 | MOVL<br>A, RLO, @ RW1+d16 | MOVL<br>A, RLO, @ RW1+d16 | MOVL<br>RLO, A, @ RW1+d16 | MOVL<br>RLO, A, @ RW1+d16 | MOV<br>R1, #8, @ RW1+d16 | MOV<br>R1, #8, @ RW1+d16 | MOVEA<br>A, RLO, @ RW1+d16 | MOVEA<br>A, RLO, @ RW1+d16 |
| +A | JMPP<br>@ RLO, @ RW2+d16 | JMPP @<br>@ RLO, @ RW2+d16 | CALLP<br>@ RLO, @ RW2+d16 | CALLP @<br>@ RLO, @ RW2+d16 | INCL<br>RLO, @ RW2+d16 | INCL<br>RLO, @ RW2+d16 | DECL<br>RLO, @ RW2+d16 | DECL<br>RLO, @ RW2+d16 | MOVL<br>A, RLO, @ RW2+d16 | MOVL<br>A, RLO, @ RW2+d16 | MOVL<br>RLO, A, @ RW2+d16 | MOVL<br>RLO, A, @ RW2+d16 | MOV<br>R2, #8, @ RW2+d16 | MOV<br>R2, #8, @ RW2+d16 | MOVEA<br>A, RLO, @ RW2+d16 | MOVEA<br>A, RLO, @ RW2+d16 |
| +B | JMPP<br>@ RLO, @ RW3+d16 | JMPP @<br>@ RLO, @ RW3+d16 | CALLP<br>@ RLO, @ RW3+d16 | CALLP @<br>@ RLO, @ RW3+d16 | INCL<br>RLO, @ RW3+d16 | INCL<br>RLO, @ RW3+d16 | DECL<br>RLO, @ RW3+d16 | DECL<br>RLO, @ RW3+d16 | MOVL<br>A, RLO, @ RW3+d16 | MOVL<br>A, RLO, @ RW3+d16 | MOVL<br>RLO, A, @ RW3+d16 | MOVL<br>RLO, A, @ RW3+d16 | MOV<br>R3, #8, @ RW3+d16 | MOV<br>R3, #8, @ RW3+d16 | MOVEA<br>A, RLO, @ RW3+d16 | MOVEA<br>A, RLO, @ RW3+d16 |
| +C | JMPP<br>@ RLO, @ RW0+RW7 | JMPP @<br>@ RLO, @ RW0+RW7 | CALLP<br>@ RLO, @ RW0+RW7 | CALLP @<br>@ RLO, @ RW0+RW7 | INCL<br>RLO, @ RW0+RW7 | INCL<br>RLO, @ RW0+RW7 | DECL<br>RLO, @ RW0+RW7 | DECL<br>RLO, @ RW0+RW7 | MOVL<br>A, RLO, @ RW0+RW7 | MOVL<br>A, RLO, @ RW0+RW7 | MOVL<br>RLO, A, @ RW0+RW7 | MOVL<br>RLO, A, @ RW0+RW7 | MOV<br>R0, #8, @ RW0+RW7 | MOV<br>R0, #8, @ RW0+RW7 | MOVEA<br>A, RLO, @ RW0+RW7 | MOVEA<br>A, RLO, @ RW0+RW7 |
| +D | JMPP<br>@ RLO, @ RW1+RW7 | JMPP @<br>@ RLO, @ RW1+RW7 | CALLP<br>@ RLO, @ RW1+RW7 | CALLP @<br>@ RLO, @ RW1+RW7 | INCL<br>RLO, @ RW1+RW7 | INCL<br>RLO, @ RW1+RW7 | DECL<br>RLO, @ RW1+RW7 | DECL<br>RLO, @ RW1+RW7 | MOVL<br>A, RLO, @ RW1+RW7 | MOVL<br>A, RLO, @ RW1+RW7 | MOVL<br>RLO, A, @ RW1+RW7 | MOVL<br>RLO, A, @ RW1+RW7 | MOV<br>R1, #8, @ RW1+RW7 | MOV<br>R1, #8, @ RW1+RW7 | MOVEA<br>A, RLO, @ RW1+RW7 | MOVEA<br>A, RLO, @ RW1+RW7 |
| +E | JMPP<br>@ RLO, @ PC+d16  | JMPP @<br>@ RLO, @ PC+d16  | CALLP<br>@ RLO, @ PC+d16  | CALLP @<br>@ RLO, @ PC+d16  | INCL<br>RLO, @ PC+d16  | INCL<br>RLO, @ PC+d16  | DECL<br>RLO, @ PC+d16  | DECL<br>RLO, @ PC+d16  | MOVL<br>A, RLO, @ PC+d16  | MOVL<br>A, RLO, @ PC+d16  | MOVL<br>RLO, A, @ PC+d16  | MOVL<br>RLO, A, @ PC+d16  | MOV<br>R2, #8, @ PC+d16  | MOV<br>R2, #8, @ PC+d16  | MOVEA<br>A, RLO, @ PC+d16  | MOVEA<br>A, RLO, @ PC+d16  |
| +F | JMPP<br>@ RLO, @ add16   | JMPP @<br>@ RLO, @ add16   | CALLP<br>@ RLO, @ add16   | CALLP @<br>@ RLO, @ add16   | INCL<br>RLO, @ add16   | INCL<br>RLO, @ add16   | DECL<br>RLO, @ add16   | DECL<br>RLO, @ add16   | MOVL<br>A, RLO, @ add16   | MOVL<br>A, RLO, @ add16   | MOVL<br>RLO, A, @ add16   | MOVL<br>RLO, A, @ add16   | MOV<br>R3, #8, @ add16   | MOV<br>R3, #8, @ add16   | MOVEA<br>A, RLO, @ add16   | MOVEA<br>A, RLO, @ add16   |

### Table D.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       | ROLc @RW0+d8  | RORc       | RORc R0' @RW0+d8 | INC       | INC R0' @RW0+d8 | DEC       | DEC R0' @RW0+d8 | MOV | MOV A, R0' @RW0+d8 | MOV | MOV R0, A' @RW0+d8,A     | MOVX | MOVX A, R0' @RW0+d8 | XCH | XCH A, R0' @RW0+d8 |
| +1 | ROLc       | ROLc @RW1+d8  | RORc       | RORc R1' @RW1+d8 | INC       | INC R1' @RW1+d8 | DEC       | DEC R1' @RW1+d8 | MOV | MOV A, R1' @RW1+d8 | MOV | MOV R1, A' @RW1+d8,A     | MOVX | MOVX A, R1' @RW1+d8 | XCH | XCH A, R1' @RW1+d8 |
| +2 | ROLc       | ROLc @RW2+d8  | RORc       | RORc R2' @RW2+d8 | INC       | INC R2' @RW2+d8 | DEC       | DEC R2' @RW2+d8 | MOV | MOV A, R2' @RW2+d8 | MOV | MOV R2, A' @RW2+d8,A     | MOVX | MOVX A, R2' @RW2+d8 | XCH | XCH A, R2' @RW2+d8 |
| +3 | ROLc       | ROLc @RW3+d8  | RORc       | RORc R3' @RW3+d8 | INC       | INC R3' @RW3+d8 | DEC       | DEC R3' @RW3+d8 | MOV | MOV A, R3' @RW3+d8 | MOV | MOV R3, A' @RW3+d8,A     | MOVX | MOVX A, R3' @RW3+d8 | XCH | XCH A, R3' @RW3+d8 |
| +4 | ROLc       | ROLc @RW4+d8  | RORc       | RORc R4' @RW4+d8 | INC       | INC R4' @RW4+d8 | DEC       | DEC R4' @RW4+d8 | MOV | MOV A, R4' @RW4+d8 | MOV | MOV R4, A' @RW4+d8,A     | MOVX | MOVX A, R4' @RW4+d8 | XCH | XCH A, R4' @RW4+d8 |
| +5 | ROLc       | ROLc @RW5+d8  | RORc       | RORc R5' @RW5+d8 | INC       | INC R5' @RW5+d8 | DEC       | DEC R5' @RW5+d8 | MOV | MOV A, R5' @RW5+d8 | MOV | MOV R5, A' @RW5+d8,A     | MOVX | MOVX A, R5' @RW5+d8 | XCH | XCH A, R5' @RW5+d8 |
| +6 | ROLc       | ROLc @RW6+d8  | RORc       | RORc R6' @RW6+d8 | INC       | INC R6' @RW6+d8 | DEC       | DEC R6' @RW6+d8 | MOV | MOV A, R6' @RW6+d8 | MOV | MOV R6, A' @RW6+d8,A     | MOVX | MOVX A, R6' @RW6+d8 | XCH | XCH A, R6' @RW6+d8 |
| +7 | ROLc       | ROLc @RW7+d8  | RORc       | RORc R7' @RW7+d8 | INC       | INC R7' @RW7+d8 | DEC       | DEC R7' @RW7+d8 | MOV | MOV A, R7' @RW7+d8 | MOV | MOV R7, A' @RW7+d8,A     | MOVX | MOVX A, R7' @RW7+d8 | XCH | XCH A, R7' @RW7+d8 |
| +8 | ROLc @RW0+ | ROLc @RW0+d16 | RORc @RW0+ | RORc @RW0+d16    | INC @RW0+ | INC @RW0+d16    | DEC @RW0+ | DEC @RW0+d16    | MOV | MOV A, @RW0+d16    | MOV | MOV @RW0, A' @RW0+d16,A  | MOVX | MOVX A, @RW0+d16    | XCH | XCH A, @RW0+d16    |
| +9 | ROLc @RW1+ | ROLc @RW1+d16 | RORc @RW1+ | RORc @RW1+d16    | INC @RW1+ | INC @RW1+d16    | DEC @RW1+ | DEC @RW1+d16    | MOV | MOV A, @RW1+d16    | MOV | MOV @RW1, A' @RW1+d16,A  | MOVX | MOVX A, @RW1+d16    | XCH | XCH A, @RW1+d16    |
| +A | ROLc @RW2+ | ROLc @RW2+d16 | RORc @RW2+ | RORc @RW2+d16    | INC @RW2+ | INC @RW2+d16    | DEC @RW2+ | DEC @RW2+d16    | MOV | MOV A, @RW2+d16    | MOV | MOV @RW2, A' @RW2+d16,A  | MOVX | MOVX A, @RW2+d16    | XCH | XCH A, @RW2+d16    |
| +B | ROLc @RW3+ | ROLc @RW3+d16 | RORc @RW3+ | RORc @RW3+d16    | INC @RW3+ | INC @RW3+d16    | DEC @RW3+ | DEC @RW3+d16    | MOV | MOV A, @RW3+d16    | MOV | MOV @RW3, A' @RW3+d16,A  | MOVX | MOVX A, @RW3+d16    | XCH | XCH A, @RW3+d16    |
| +C | ROLc @RW0+ | ROLc @RW0+RW7 | RORc @RW0+ | RORc @RW0+RW7    | INC @RW0+ | INC @RW0+RW7    | DEC @RW0+ | DEC @RW0+RW7    | MOV | MOV A, @RW0+RW7    | MOV | MOV @RW0+, A' @RW0+RW7,A | MOVX | MOVX A, @RW0+RW7    | XCH | XCH A, @RW0+RW7    |
| +D | ROLc @RW1+ | ROLc @RW1+RW7 | RORc @RW1+ | RORc @RW1+RW7    | INC @RW1+ | INC @RW1+RW7    | DEC @RW1+ | DEC @RW1+RW7    | MOV | MOV A, @RW1+RW7    | MOV | MOV @RW1+, A' @RW1+RW7,A | MOVX | MOVX A, @RW1+RW7    | XCH | XCH A, @RW1+RW7    |
| +E | ROLc @RW2+ | ROLc @PC+d16  | RORc @RW2+ | RORc @PC+d16     | INC @RW2+ | INC @PC+d16     | DEC @RW2+ | DEC @PC+d16     | MOV | MOV A, @RW2+RW7    | MOV | MOV @RW2+, A' @PC+d16,A  | MOVX | MOVX A, @PC+d16     | XCH | XCH A, @PC+d16     |
| +F | ROLc @RW3+ | ROLc addr16   | RORc @RW3+ | RORc addr16      | INC @RW3+ | INC addr16      | DEC @RW3+ | DEC addr16      | MOV | MOV A, @RW3+addr16 | MOV | MOV @RW3+, A' addr16,A   | MOVX | MOVX A, @RW3+addr16 | XCH | XCH A, @RW3+addr16 |



Table D.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<br>@ RW0, @ RW0+d8   | JMP<br>@ RW0, @ RW0+d8   | CALL<br>RW0, @ RW0+d8     | CALL<br>RW0, @ RW0+d8     | INCW<br>RW0, @ RW0+d8     | INCW<br>RW0, @ RW0+d8     | DECW<br>RW0, @ RW0+d8     | DECW<br>RW0, @ RW0+d8     | MOVW<br>A, RW0, @ RW0+d8     | MOVW<br>A, RW0, @ RW0+d8     | MOVW<br>RW0, #16, @ RW0+d8, #16     | MOVW<br>RW0, #16, @ RW0+d8, #16     | MOVW<br>RW0, #16, @ RW0+d8, #16     | MOVW<br>RW0, #16, @ RW0+d8, #16     | XCHW<br>A, RW0, @ RW0+d8     | XCHW<br>A, RW0, @ RW0+d8     |
| +1 | JMP<br>@ RW1, @ RW1+d8   | JMP<br>@ RW1, @ RW1+d8   | CALL<br>RW1, @ RW1+d8     | CALL<br>RW1, @ RW1+d8     | INCW<br>RW1, @ RW1+d8     | INCW<br>RW1, @ RW1+d8     | DECW<br>RW1, @ RW1+d8     | DECW<br>RW1, @ RW1+d8     | MOVW<br>A, RW1, @ RW1+d8     | MOVW<br>A, RW1, @ RW1+d8     | MOVW<br>RW1, #16, @ RW1+d8, #16     | MOVW<br>RW1, #16, @ RW1+d8, #16     | MOVW<br>RW1, #16, @ RW1+d8, #16     | MOVW<br>RW1, #16, @ RW1+d8, #16     | XCHW<br>A, RW1, @ RW1+d8     | XCHW<br>A, RW1, @ RW1+d8     |
| +2 | JMP<br>@ RW2, @ RW2+d8   | JMP<br>@ RW2, @ RW2+d8   | CALL<br>RW2, @ RW2+d8     | CALL<br>RW2, @ RW2+d8     | INCW<br>RW2, @ RW2+d8     | INCW<br>RW2, @ RW2+d8     | DECW<br>RW2, @ RW2+d8     | DECW<br>RW2, @ RW2+d8     | MOVW<br>A, RW2, @ RW2+d8     | MOVW<br>A, RW2, @ RW2+d8     | MOVW<br>RW2, #16, @ RW2+d8, #16     | MOVW<br>RW2, #16, @ RW2+d8, #16     | MOVW<br>RW2, #16, @ RW2+d8, #16     | MOVW<br>RW2, #16, @ RW2+d8, #16     | XCHW<br>A, RW2, @ RW2+d8     | XCHW<br>A, RW2, @ RW2+d8     |
| +3 | JMP<br>@ RW3, @ RW3+d8   | JMP<br>@ RW3, @ RW3+d8   | CALL<br>RW3, @ RW3+d8     | CALL<br>RW3, @ RW3+d8     | INCW<br>RW3, @ RW3+d8     | INCW<br>RW3, @ RW3+d8     | DECW<br>RW3, @ RW3+d8     | DECW<br>RW3, @ RW3+d8     | MOVW<br>A, RW3, @ RW3+d8     | MOVW<br>A, RW3, @ RW3+d8     | MOVW<br>RW3, #16, @ RW3+d8, #16     | MOVW<br>RW3, #16, @ RW3+d8, #16     | MOVW<br>RW3, #16, @ RW3+d8, #16     | MOVW<br>RW3, #16, @ RW3+d8, #16     | XCHW<br>A, RW3, @ RW3+d8     | XCHW<br>A, RW3, @ RW3+d8     |
| +4 | JMP<br>@ RW4, @ RW4+d8   | JMP<br>@ RW4, @ RW4+d8   | CALL<br>RW4, @ RW4+d8     | CALL<br>RW4, @ RW4+d8     | INCW<br>RW4, @ RW4+d8     | INCW<br>RW4, @ RW4+d8     | DECW<br>RW4, @ RW4+d8     | DECW<br>RW4, @ RW4+d8     | MOVW<br>A, RW4, @ RW4+d8     | MOVW<br>A, RW4, @ RW4+d8     | MOVW<br>RW4, #16, @ RW4+d8, #16     | MOVW<br>RW4, #16, @ RW4+d8, #16     | MOVW<br>RW4, #16, @ RW4+d8, #16     | MOVW<br>RW4, #16, @ RW4+d8, #16     | XCHW<br>A, RW4, @ RW4+d8     | XCHW<br>A, RW4, @ RW4+d8     |
| +5 | JMP<br>@ RW5, @ RW5+d8   | JMP<br>@ RW5, @ RW5+d8   | CALL<br>RW5, @ RW5+d8     | CALL<br>RW5, @ RW5+d8     | INCW<br>RW5, @ RW5+d8     | INCW<br>RW5, @ RW5+d8     | DECW<br>RW5, @ RW5+d8     | DECW<br>RW5, @ RW5+d8     | MOVW<br>A, RW5, @ RW5+d8     | MOVW<br>A, RW5, @ RW5+d8     | MOVW<br>RW5, #16, @ RW5+d8, #16     | MOVW<br>RW5, #16, @ RW5+d8, #16     | MOVW<br>RW5, #16, @ RW5+d8, #16     | MOVW<br>RW5, #16, @ RW5+d8, #16     | XCHW<br>A, RW5, @ RW5+d8     | XCHW<br>A, RW5, @ RW5+d8     |
| +6 | JMP<br>@ RW6, @ RW6+d8   | JMP<br>@ RW6, @ RW6+d8   | CALL<br>RW6, @ RW6+d8     | CALL<br>RW6, @ RW6+d8     | INCW<br>RW6, @ RW6+d8     | INCW<br>RW6, @ RW6+d8     | DECW<br>RW6, @ RW6+d8     | DECW<br>RW6, @ RW6+d8     | MOVW<br>A, RW6, @ RW6+d8     | MOVW<br>A, RW6, @ RW6+d8     | MOVW<br>RW6, #16, @ RW6+d8, #16     | MOVW<br>RW6, #16, @ RW6+d8, #16     | MOVW<br>RW6, #16, @ RW6+d8, #16     | MOVW<br>RW6, #16, @ RW6+d8, #16     | XCHW<br>A, RW6, @ RW6+d8     | XCHW<br>A, RW6, @ RW6+d8     |
| +7 | JMP<br>@ RW7, @ RW7+d8   | JMP<br>@ RW7, @ RW7+d8   | CALL<br>RW7, @ RW7+d8     | CALL<br>RW7, @ RW7+d8     | INCW<br>RW7, @ RW7+d8     | INCW<br>RW7, @ RW7+d8     | DECW<br>RW7, @ RW7+d8     | DECW<br>RW7, @ RW7+d8     | MOVW<br>A, RW7, @ RW7+d8     | MOVW<br>A, RW7, @ RW7+d8     | MOVW<br>RW7, #16, @ RW7+d8, #16     | MOVW<br>RW7, #16, @ RW7+d8, #16     | MOVW<br>RW7, #16, @ RW7+d8, #16     | MOVW<br>RW7, #16, @ RW7+d8, #16     | XCHW<br>A, RW7, @ RW7+d8     | XCHW<br>A, RW7, @ RW7+d8     |
| +8 | JMP<br>@ RW0, @ RW0+d16  | JMP<br>@ RW0, @ RW0+d16  | CALL<br>@ RW0, @ RW0+d16  | CALL<br>@ RW0, @ RW0+d16  | INCW<br>@ RW0, @ RW0+d16  | INCW<br>@ RW0, @ RW0+d16  | DECW<br>@ RW0, @ RW0+d16  | DECW<br>@ RW0, @ RW0+d16  | MOVW<br>A, @ RW0, @ RW0+d16  | MOVW<br>A, @ RW0, @ RW0+d16  | MOVW<br>@ RW0, #16, @ RW0+d16, #16  | MOVW<br>@ RW0, #16, @ RW0+d16, #16  | MOVW<br>@ RW0, #16, @ RW0+d16, #16  | MOVW<br>@ RW0, #16, @ RW0+d16, #16  | XCHW<br>A, @ RW0, @ RW0+d16  | XCHW<br>A, @ RW0, @ RW0+d16  |
| +9 | JMP<br>@ RW1, @ RW1+d16  | JMP<br>@ RW1, @ RW1+d16  | CALL<br>@ RW1, @ RW1+d16  | CALL<br>@ RW1, @ RW1+d16  | INCW<br>@ RW1, @ RW1+d16  | INCW<br>@ RW1, @ RW1+d16  | DECW<br>@ RW1, @ RW1+d16  | DECW<br>@ RW1, @ RW1+d16  | MOVW<br>A, @ RW1, @ RW1+d16  | MOVW<br>A, @ RW1, @ RW1+d16  | MOVW<br>@ RW1, #16, @ RW1+d16, #16  | MOVW<br>@ RW1, #16, @ RW1+d16, #16  | MOVW<br>@ RW1, #16, @ RW1+d16, #16  | MOVW<br>@ RW1, #16, @ RW1+d16, #16  | XCHW<br>A, @ RW1, @ RW1+d16  | XCHW<br>A, @ RW1, @ RW1+d16  |
| +A | JMP<br>@ RW2, @ RW2+d16  | JMP<br>@ RW2, @ RW2+d16  | CALL<br>@ RW2, @ RW2+d16  | CALL<br>@ RW2, @ RW2+d16  | INCW<br>@ RW2, @ RW2+d16  | INCW<br>@ RW2, @ RW2+d16  | DECW<br>@ RW2, @ RW2+d16  | DECW<br>@ RW2, @ RW2+d16  | MOVW<br>A, @ RW2, @ RW2+d16  | MOVW<br>A, @ RW2, @ RW2+d16  | MOVW<br>@ RW2, #16, @ RW2+d16, #16  | MOVW<br>@ RW2, #16, @ RW2+d16, #16  | MOVW<br>@ RW2, #16, @ RW2+d16, #16  | MOVW<br>@ RW2, #16, @ RW2+d16, #16  | XCHW<br>A, @ RW2, @ RW2+d16  | XCHW<br>A, @ RW2, @ RW2+d16  |
| +B | JMP<br>@ RW3, @ RW3+d16  | JMP<br>@ RW3, @ RW3+d16  | CALL<br>@ RW3, @ RW3+d16  | CALL<br>@ RW3, @ RW3+d16  | INCW<br>@ RW3, @ RW3+d16  | INCW<br>@ RW3, @ RW3+d16  | DECW<br>@ RW3, @ RW3+d16  | DECW<br>@ RW3, @ RW3+d16  | MOVW<br>A, @ RW3, @ RW3+d16  | MOVW<br>A, @ RW3, @ RW3+d16  | MOVW<br>@ RW3, #16, @ RW3+d16, #16  | MOVW<br>@ RW3, #16, @ RW3+d16, #16  | MOVW<br>@ RW3, #16, @ RW3+d16, #16  | MOVW<br>@ RW3, #16, @ RW3+d16, #16  | XCHW<br>A, @ RW3, @ RW3+d16  | XCHW<br>A, @ RW3, @ RW3+d16  |
| +C | JMP<br>@ RW0+, @ RW0+RW7 | JMP<br>@ RW0+, @ RW0+RW7 | CALL<br>@ RW0+, @ RW0+RW7 | CALL<br>@ RW0+, @ RW0+RW7 | INCW<br>@ RW0+, @ RW0+RW7 | INCW<br>@ RW0+, @ RW0+RW7 | DECW<br>@ RW0+, @ RW0+RW7 | DECW<br>@ RW0+, @ RW0+RW7 | MOVW<br>A, @ RW0+, @ RW0+RW7 | MOVW<br>A, @ RW0+, @ RW0+RW7 | MOVW<br>@ RW0+, #16, @ RW0+RW7, #16 | MOVW<br>@ RW0+, #16, @ RW0+RW7, #16 | MOVW<br>@ RW0+, #16, @ RW0+RW7, #16 | MOVW<br>@ RW0+, #16, @ RW0+RW7, #16 | XCHW<br>A, @ RW0+, @ RW0+RW7 | XCHW<br>A, @ RW0+, @ RW0+RW7 |
| +D | JMP<br>@ RW1+, @ RW1+RW7 | JMP<br>@ RW1+, @ RW1+RW7 | CALL<br>@ RW1+, @ RW1+RW7 | CALL<br>@ RW1+, @ RW1+RW7 | INCW<br>@ RW1+, @ RW1+RW7 | INCW<br>@ RW1+, @ RW1+RW7 | DECW<br>@ RW1+, @ RW1+RW7 | DECW<br>@ RW1+, @ RW1+RW7 | MOVW<br>A, @ RW1+, @ RW1+RW7 | MOVW<br>A, @ RW1+, @ RW1+RW7 | MOVW<br>@ RW1+, #16, @ RW1+RW7, #16 | MOVW<br>@ RW1+, #16, @ RW1+RW7, #16 | MOVW<br>@ RW1+, #16, @ RW1+RW7, #16 | MOVW<br>@ RW1+, #16, @ RW1+RW7, #16 | XCHW<br>A, @ RW1+, @ RW1+RW7 | XCHW<br>A, @ RW1+, @ RW1+RW7 |
| +E | JMP<br>@ PC-d16          | JMP<br>@ PC-d16          | CALL<br>@ PC-d16          | CALL<br>@ PC-d16          | INCW<br>@ PC-d16          | INCW<br>@ PC-d16          | DECW<br>@ PC-d16          | DECW<br>@ PC-d16          | MOVW<br>A, @ PC-d16          | MOVW<br>A, @ PC-d16          | MOVW<br>@ PC-d16, #16               | MOVW<br>@ PC-d16, #16               | MOVW<br>@ PC-d16, #16               | MOVW<br>@ PC-d16, #16               | XCHW<br>A, @ PC-d16          | XCHW<br>A, @ PC-d16          |
| +F | JMP<br>@ RW3+, @ addr16  | JMP<br>@ RW3+, @ addr16  | CALL<br>@ RW3+, @ addr16  | CALL<br>@ RW3+, @ addr16  | INCW<br>@ RW3+, @ addr16  | INCW<br>@ RW3+, @ addr16  | DECW<br>@ RW3+, @ addr16  | DECW<br>@ RW3+, @ addr16  | MOVW<br>A, @ RW3+, @ addr16  | MOVW<br>A, @ RW3+, @ addr16  | MOVW<br>@ RW3+, #16, @ addr16, #16  | MOVW<br>@ RW3+, #16, @ addr16, #16  | MOVW<br>@ RW3+, #16, @ addr16, #16  | MOVW<br>@ RW3+, #16, @ addr16, #16  | XCHW<br>A, @ RW3+, @ addr16  | XCHW<br>A, @ RW3+, @ addr16  |

Table D.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<br>A, R0', @RW0+d8     | SUB<br>A, R0', @RW0+d8     | SUB<br>A, R0', @RW0+d8     | SUB<br>A, R0', @RW0+d8     | ADDC<br>A, R0', @RW0+d8     | ADDC<br>A, R0', @RW0+d8     | CMP<br>A, R0', @RW0+d8     | CMP<br>A, R0', @RW0+d8     | AND<br>A, R0', @RW0+d8     | AND<br>A, R0', @RW0+d8     | OR<br>A, R0', @RW0+d8     | OR<br>A, R0', @RW0+d8     | XOR<br>A, R0', @RW0+d8     | XOR<br>A, R0', @RW0+d8     | DBNZ<br>R0, r'RW0+d8, r    | DBNZ<br>R0, r'RW0+d8, r    |
| +1 | ADD<br>A, R1', @RW1+d8     | SUB<br>A, R1', @RW1+d8     | SUB<br>A, R1', @RW1+d8     | SUB<br>A, R1', @RW1+d8     | ADDC<br>A, R1', @RW1+d8     | ADDC<br>A, R1', @RW1+d8     | CMP<br>A, R1', @RW1+d8     | CMP<br>A, R1', @RW1+d8     | AND<br>A, R1', @RW1+d8     | AND<br>A, R1', @RW1+d8     | OR<br>A, R1', @RW1+d8     | OR<br>A, R1', @RW1+d8     | XOR<br>A, R1', @RW1+d8     | XOR<br>A, R1', @RW1+d8     | DBNZ<br>R1, r'RW1+d8, r    | DBNZ<br>R1, r'RW1+d8, r    |
| +2 | ADD<br>A, R2', @RW2+d8     | SUB<br>A, R2', @RW2+d8     | SUB<br>A, R2', @RW2+d8     | SUB<br>A, R2', @RW2+d8     | ADDC<br>A, R2', @RW2+d8     | ADDC<br>A, R2', @RW2+d8     | CMP<br>A, R2', @RW2+d8     | CMP<br>A, R2', @RW2+d8     | AND<br>A, R2', @RW2+d8     | AND<br>A, R2', @RW2+d8     | OR<br>A, R2', @RW2+d8     | OR<br>A, R2', @RW2+d8     | XOR<br>A, R2', @RW2+d8     | XOR<br>A, R2', @RW2+d8     | DBNZ<br>R2, r'RW2+d8, r    | DBNZ<br>R2, r'RW2+d8, r    |
| +3 | ADD<br>A, R3', @RW3+d8     | SUB<br>A, R3', @RW3+d8     | SUB<br>A, R3', @RW3+d8     | SUB<br>A, R3', @RW3+d8     | ADDC<br>A, R3', @RW3+d8     | ADDC<br>A, R3', @RW3+d8     | CMP<br>A, R3', @RW3+d8     | CMP<br>A, R3', @RW3+d8     | AND<br>A, R3', @RW3+d8     | AND<br>A, R3', @RW3+d8     | OR<br>A, R3', @RW3+d8     | OR<br>A, R3', @RW3+d8     | XOR<br>A, R3', @RW3+d8     | XOR<br>A, R3', @RW3+d8     | DBNZ<br>R3, r'RW3+d8, r    | DBNZ<br>R3, r'RW3+d8, r    |
| +4 | ADD<br>A, R4', @RW4+d8     | SUB<br>A, R4', @RW4+d8     | SUB<br>A, R4', @RW4+d8     | SUB<br>A, R4', @RW4+d8     | ADDC<br>A, R4', @RW4+d8     | ADDC<br>A, R4', @RW4+d8     | CMP<br>A, R4', @RW4+d8     | CMP<br>A, R4', @RW4+d8     | AND<br>A, R4', @RW4+d8     | AND<br>A, R4', @RW4+d8     | OR<br>A, R4', @RW4+d8     | OR<br>A, R4', @RW4+d8     | XOR<br>A, R4', @RW4+d8     | XOR<br>A, R4', @RW4+d8     | DBNZ<br>R4, r'RW4+d8, r    | DBNZ<br>R4, r'RW4+d8, r    |
| +5 | ADD<br>A, R5', @RW5+d8     | SUB<br>A, R5', @RW5+d8     | SUB<br>A, R5', @RW5+d8     | SUB<br>A, R5', @RW5+d8     | ADDC<br>A, R5', @RW5+d8     | ADDC<br>A, R5', @RW5+d8     | CMP<br>A, R5', @RW5+d8     | CMP<br>A, R5', @RW5+d8     | AND<br>A, R5', @RW5+d8     | AND<br>A, R5', @RW5+d8     | OR<br>A, R5', @RW5+d8     | OR<br>A, R5', @RW5+d8     | XOR<br>A, R5', @RW5+d8     | XOR<br>A, R5', @RW5+d8     | DBNZ<br>R5, r'RW5+d8, r    | DBNZ<br>R5, r'RW5+d8, r    |
| +6 | ADD<br>A, R6', @RW6+d8     | SUB<br>A, R6', @RW6+d8     | SUB<br>A, R6', @RW6+d8     | SUB<br>A, R6', @RW6+d8     | ADDC<br>A, R6', @RW6+d8     | ADDC<br>A, R6', @RW6+d8     | CMP<br>A, R6', @RW6+d8     | CMP<br>A, R6', @RW6+d8     | AND<br>A, R6', @RW6+d8     | AND<br>A, R6', @RW6+d8     | OR<br>A, R6', @RW6+d8     | OR<br>A, R6', @RW6+d8     | XOR<br>A, R6', @RW6+d8     | XOR<br>A, R6', @RW6+d8     | DBNZ<br>R6, r'RW6+d8, r    | DBNZ<br>R6, r'RW6+d8, r    |
| +7 | ADD<br>A, R7', @RW7+d8     | SUB<br>A, R7', @RW7+d8     | SUB<br>A, R7', @RW7+d8     | SUB<br>A, R7', @RW7+d8     | ADDC<br>A, R7', @RW7+d8     | ADDC<br>A, R7', @RW7+d8     | CMP<br>A, R7', @RW7+d8     | CMP<br>A, R7', @RW7+d8     | AND<br>A, R7', @RW7+d8     | AND<br>A, R7', @RW7+d8     | OR<br>A, R7', @RW7+d8     | OR<br>A, R7', @RW7+d8     | XOR<br>A, R7', @RW7+d8     | XOR<br>A, R7', @RW7+d8     | DBNZ<br>R7, r'RW7+d8, r    | DBNZ<br>R7, r'RW7+d8, r    |
| +8 | ADD<br>A, @RW0, @RW0+d16   | SUB<br>A, @RW0, @RW0+d16   | SUB<br>A, @RW0, @RW0+d16   | SUB<br>A, @RW0, @RW0+d16   | ADDC<br>A, @RW0, @RW0+d16   | ADDC<br>A, @RW0, @RW0+d16   | CMP<br>A, @RW0, @RW0+d16   | CMP<br>A, @RW0, @RW0+d16   | AND<br>A, @RW0, @RW0+d16   | AND<br>A, @RW0, @RW0+d16   | OR<br>A, @RW0, @RW0+d16   | OR<br>A, @RW0, @RW0+d16   | XOR<br>A, @RW0, @RW0+d16   | XOR<br>A, @RW0, @RW0+d16   | DBNZ<br>@RW0, r'W0+d16, r  | DBNZ<br>@RW0, r'W0+d16, r  |
| +9 | ADD<br>A, @RW1, @RW1+d16   | SUB<br>A, @RW1, @RW1+d16   | SUB<br>A, @RW1, @RW1+d16   | SUB<br>A, @RW1, @RW1+d16   | ADDC<br>A, @RW1, @RW1+d16   | ADDC<br>A, @RW1, @RW1+d16   | CMP<br>A, @RW1, @RW1+d16   | CMP<br>A, @RW1, @RW1+d16   | AND<br>A, @RW1, @RW1+d16   | AND<br>A, @RW1, @RW1+d16   | OR<br>A, @RW1, @RW1+d16   | OR<br>A, @RW1, @RW1+d16   | XOR<br>A, @RW1, @RW1+d16   | XOR<br>A, @RW1, @RW1+d16   | DBNZ<br>@RW1, r'W1+d16, r  | DBNZ<br>@RW1, r'W1+d16, r  |
| +A | ADD<br>A, @RW2, @RW2+d16   | SUB<br>A, @RW2, @RW2+d16   | SUB<br>A, @RW2, @RW2+d16   | SUB<br>A, @RW2, @RW2+d16   | ADDC<br>A, @RW2, @RW2+d16   | ADDC<br>A, @RW2, @RW2+d16   | CMP<br>A, @RW2, @RW2+d16   | CMP<br>A, @RW2, @RW2+d16   | AND<br>A, @RW2, @RW2+d16   | AND<br>A, @RW2, @RW2+d16   | OR<br>A, @RW2, @RW2+d16   | OR<br>A, @RW2, @RW2+d16   | XOR<br>A, @RW2, @RW2+d16   | XOR<br>A, @RW2, @RW2+d16   | DBNZ<br>@RW2, r'W2+d16, r  | DBNZ<br>@RW2, r'W2+d16, r  |
| +B | ADD<br>A, @RW3, @RW3+d16   | SUB<br>A, @RW3, @RW3+d16   | SUB<br>A, @RW3, @RW3+d16   | SUB<br>A, @RW3, @RW3+d16   | ADDC<br>A, @RW3, @RW3+d16   | ADDC<br>A, @RW3, @RW3+d16   | CMP<br>A, @RW3, @RW3+d16   | CMP<br>A, @RW3, @RW3+d16   | AND<br>A, @RW3, @RW3+d16   | AND<br>A, @RW3, @RW3+d16   | OR<br>A, @RW3, @RW3+d16   | OR<br>A, @RW3, @RW3+d16   | XOR<br>A, @RW3, @RW3+d16   | XOR<br>A, @RW3, @RW3+d16   | DBNZ<br>@RW3, r'W3+d16, r  | DBNZ<br>@RW3, r'W3+d16, r  |
| +C | ADD<br>A, @RW0+, @RW0+RW7  | SUB<br>A, @RW0+, @RW0+RW7  | SUB<br>A, @RW0+, @RW0+RW7  | SUB<br>A, @RW0+, @RW0+RW7  | ADDC<br>A, @RW0+, @RW0+RW7  | ADDC<br>A, @RW0+, @RW0+RW7  | CMP<br>A, @RW0+, @RW0+RW7  | CMP<br>A, @RW0+, @RW0+RW7  | AND<br>A, @RW0+, @RW0+RW7  | AND<br>A, @RW0+, @RW0+RW7  | OR<br>A, @RW0+, @RW0+RW7  | OR<br>A, @RW0+, @RW0+RW7  | XOR<br>A, @RW0+, @RW0+RW7  | XOR<br>A, @RW0+, @RW0+RW7  | DBNZ<br>@RW0+, r'W0+RW7, r | DBNZ<br>@RW0+, r'W0+RW7, r |
| +D | ADD<br>A, @RW1+, @RW1+RW7  | SUB<br>A, @RW1+, @RW1+RW7  | SUB<br>A, @RW1+, @RW1+RW7  | SUB<br>A, @RW1+, @RW1+RW7  | ADDC<br>A, @RW1+, @RW1+RW7  | ADDC<br>A, @RW1+, @RW1+RW7  | CMP<br>A, @RW1+, @RW1+RW7  | CMP<br>A, @RW1+, @RW1+RW7  | AND<br>A, @RW1+, @RW1+RW7  | AND<br>A, @RW1+, @RW1+RW7  | OR<br>A, @RW1+, @RW1+RW7  | OR<br>A, @RW1+, @RW1+RW7  | XOR<br>A, @RW1+, @RW1+RW7  | XOR<br>A, @RW1+, @RW1+RW7  | DBNZ<br>@RW1+, r'W1+RW7, r | DBNZ<br>@RW1+, r'W1+RW7, r |
| +E | ADD<br>A, @RW2+, @PC+d16   | SUB<br>A, @RW2+, @PC+d16   | SUB<br>A, @RW2+, @PC+d16   | SUB<br>A, @RW2+, @PC+d16   | ADDC<br>A, @RW2+, @PC+d16   | ADDC<br>A, @RW2+, @PC+d16   | CMP<br>A, @RW2+, @PC+d16   | CMP<br>A, @RW2+, @PC+d16   | AND<br>A, @RW2+, @PC+d16   | AND<br>A, @RW2+, @PC+d16   | OR<br>A, @RW2+, @PC+d16   | OR<br>A, @RW2+, @PC+d16   | XOR<br>A, @RW2+, @PC+d16   | XOR<br>A, @RW2+, @PC+d16   | DBNZ<br>@RW2+, r'PC+d16, r | DBNZ<br>@RW2+, r'PC+d16, r |
| +F | ADD<br>A, @RW3+, A, addr16 | SUB<br>A, @RW3+, A, addr16 | SUB<br>A, @RW3+, A, addr16 | SUB<br>A, @RW3+, A, addr16 | ADDC<br>A, @RW3+, A, addr16 | ADDC<br>A, @RW3+, A, addr16 | CMP<br>A, @RW3+, A, addr16 | CMP<br>A, @RW3+, A, addr16 | AND<br>A, @RW3+, A, addr16 | AND<br>A, @RW3+, A, addr16 | OR<br>A, @RW3+, A, addr16 | OR<br>A, @RW3+, A, addr16 | XOR<br>A, @RW3+, A, addr16 | XOR<br>A, @RW3+, A, addr16 | DBNZ<br>@RW3+, r'addr16, r | DBNZ<br>@RW3+, r'addr16, r |

Table D.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<br>R0, A, @RW0+d8, A     | SUB<br>R0, A, @RW0+d8, A     | SUB<br>R0, A, @RW0+d8, A     | SUB<br>A, R0, @RW0+d8, A      | SUBC<br>A, R0, @RW0+d8, A      | SUBC<br>A, R0, @RW0+d8, A      | NEG<br>R0, @RW0+d8, A      | NEG A,<br>R0, @RW0+d8, A      | AND<br>R0, A, @RW0+d8, A     | AND<br>R0, A, @RW0+d8, A     | OR<br>R0, A, @RW0+d8, A     | OR<br>A, @RW0+d8, A  | XOR<br>R0, A, @RW0+d8, A     | XOR<br>A, @RW0+d8, A  | NOT<br>R0, @RW0+d8, A      | NOT<br>R0, @RW0+d8, A      |
| +1 | ADD<br>R1, A, @RW1+d8, A     | SUB<br>R1, A, @RW1+d8, A     | SUB<br>R1, A, @RW1+d8, A     | SUB<br>A, R1, @RW1+d8, A      | SUBC<br>A, R1, @RW1+d8, A      | SUBC<br>A, R1, @RW1+d8, A      | NEG<br>R1, @RW1+d8, A      | NEG A,<br>R1, @RW1+d8, A      | AND<br>R1, A, @RW1+d8, A     | AND<br>R1, A, @RW1+d8, A     | OR<br>R1, A, @RW1+d8, A     | OR<br>A, @RW1+d8, A  | XOR<br>R1, A, @RW1+d8, A     | XOR<br>A, @RW1+d8, A  | NOT<br>R1, @RW1+d8, A      | NOT<br>R1, @RW1+d8, A      |
| +2 | ADD<br>R2, A, @RW2+d8, A     | SUB<br>R2, A, @RW2+d8, A     | SUB<br>R2, A, @RW2+d8, A     | SUB<br>A, R2, @RW2+d8, A      | SUBC<br>A, R2, @RW2+d8, A      | SUBC<br>A, R2, @RW2+d8, A      | NEG<br>R2, @RW2+d8, A      | NEG A,<br>R2, @RW2+d8, A      | AND<br>R2, A, @RW2+d8, A     | AND<br>R2, A, @RW2+d8, A     | OR<br>R2, A, @RW2+d8, A     | OR<br>A, @RW2+d8, A  | XOR<br>R2, A, @RW2+d8, A     | XOR<br>A, @RW2+d8, A  | NOT<br>R2, @RW2+d8, A      | NOT<br>R2, @RW2+d8, A      |
| +3 | ADD<br>R3, A, @RW3+d8, A     | SUB<br>R3, A, @RW3+d8, A     | SUB<br>R3, A, @RW3+d8, A     | SUB<br>A, R3, @RW3+d8, A      | SUBC<br>A, R3, @RW3+d8, A      | SUBC<br>A, R3, @RW3+d8, A      | NEG<br>R3, @RW3+d8, A      | NEG A,<br>R3, @RW3+d8, A      | AND<br>R3, A, @RW3+d8, A     | AND<br>R3, A, @RW3+d8, A     | OR<br>R3, A, @RW3+d8, A     | OR<br>A, @RW3+d8, A  | XOR<br>R3, A, @RW3+d8, A     | XOR<br>A, @RW3+d8, A  | NOT<br>R3, @RW3+d8, A      | NOT<br>R3, @RW3+d8, A      |
| +4 | ADD<br>R4, A, @RW4+d8, A     | SUB<br>R4, A, @RW4+d8, A     | SUB<br>R4, A, @RW4+d8, A     | SUB<br>A, R4, @RW4+d8, A      | SUBC<br>A, R4, @RW4+d8, A      | SUBC<br>A, R4, @RW4+d8, A      | NEG<br>R4, @RW4+d8, A      | NEG A,<br>R4, @RW4+d8, A      | AND<br>R4, A, @RW4+d8, A     | AND<br>R4, A, @RW4+d8, A     | OR<br>R4, A, @RW4+d8, A     | OR<br>A, @RW4+d8, A  | XOR<br>R4, A, @RW4+d8, A     | XOR<br>A, @RW4+d8, A  | NOT<br>R4, @RW4+d8, A      | NOT<br>R4, @RW4+d8, A      |
| +5 | ADD<br>R5, A, @RW5+d8, A     | SUB<br>R5, A, @RW5+d8, A     | SUB<br>R5, A, @RW5+d8, A     | SUB<br>A, R5, @RW5+d8, A      | SUBC<br>A, R5, @RW5+d8, A      | SUBC<br>A, R5, @RW5+d8, A      | NEG<br>R5, @RW5+d8, A      | NEG A,<br>R5, @RW5+d8, A      | AND<br>R5, A, @RW5+d8, A     | AND<br>R5, A, @RW5+d8, A     | OR<br>R5, A, @RW5+d8, A     | OR<br>A, @RW5+d8, A  | XOR<br>R5, A, @RW5+d8, A     | XOR<br>A, @RW5+d8, A  | NOT<br>R5, @RW5+d8, A      | NOT<br>R5, @RW5+d8, A      |
| +6 | ADD<br>R6, A, @RW6+d8, A     | SUB<br>R6, A, @RW6+d8, A     | SUB<br>R6, A, @RW6+d8, A     | SUB<br>A, R6, @RW6+d8, A      | SUBC<br>A, R6, @RW6+d8, A      | SUBC<br>A, R6, @RW6+d8, A      | NEG<br>R6, @RW6+d8, A      | NEG A,<br>R6, @RW6+d8, A      | AND<br>R6, A, @RW6+d8, A     | AND<br>R6, A, @RW6+d8, A     | OR<br>R6, A, @RW6+d8, A     | OR<br>A, @RW6+d8, A  | XOR<br>R6, A, @RW6+d8, A     | XOR<br>A, @RW6+d8, A  | NOT<br>R6, @RW6+d8, A      | NOT<br>R6, @RW6+d8, A      |
| +7 | ADD<br>R7, A, @RW7+d8, A     | SUB<br>R7, A, @RW7+d8, A     | SUB<br>R7, A, @RW7+d8, A     | SUB<br>A, R7, @RW7+d8, A      | SUBC<br>A, R7, @RW7+d8, A      | SUBC<br>A, R7, @RW7+d8, A      | NEG<br>R7, @RW7+d8, A      | NEG A,<br>R7, @RW7+d8, A      | AND<br>R7, A, @RW7+d8, A     | AND<br>R7, A, @RW7+d8, A     | OR<br>R7, A, @RW7+d8, A     | OR<br>A, @RW7+d8, A  | XOR<br>R7, A, @RW7+d8, A     | XOR<br>A, @RW7+d8, A  | NOT<br>R7, @RW7+d8, A      | NOT<br>R7, @RW7+d8, A      |
| +8 | ADD<br>@RW0, A, @RW0+d16, A  | SUB<br>@RW0, A, @RW0+d16, A  | SUB<br>@RW0, A, @RW0+d16, A  | SUB<br>A, @RW0, @RW0+d16, A   | SUBC<br>A, @RW0, @RW0+d16, A   | SUBC<br>A, @RW0, @RW0+d16, A   | NEG<br>@RW0, @RW0+d16, A   | NEG A,<br>@RW0, @RW0+d16, A   | AND<br>@RW0, A, @RW0+d16, A  | AND<br>@RW0, A, @RW0+d16, A  | OR<br>@RW0, A, @RW0+d16, A  | OR<br>A, @RW0+d16, A | XOR<br>@RW0, A, @RW0+d16, A  | XOR<br>A, @RW0+d16, A | NOT<br>@RW0, @RW0+d16, A   | NOT<br>@RW0, @RW0+d16, A   |
| +9 | ADD<br>@RW1, A, @RW1+d16, A  | SUB<br>@RW1, A, @RW1+d16, A  | SUB<br>@RW1, A, @RW1+d16, A  | SUB<br>A, @RW1, @RW1+d16, A   | SUBC<br>A, @RW1, @RW1+d16, A   | SUBC<br>A, @RW1, @RW1+d16, A   | NEG<br>@RW1, @RW1+d16, A   | NEG A,<br>@RW1, @RW1+d16, A   | AND<br>@RW1, A, @RW1+d16, A  | AND<br>@RW1, A, @RW1+d16, A  | OR<br>@RW1, A, @RW1+d16, A  | OR<br>A, @RW1+d16, A | XOR<br>@RW1, A, @RW1+d16, A  | XOR<br>A, @RW1+d16, A | NOT<br>@RW1, @RW1+d16, A   | NOT<br>@RW1, @RW1+d16, A   |
| +A | ADD<br>@RW2, A, @RW2+d16, A  | SUB<br>@RW2, A, @RW2+d16, A  | SUB<br>@RW2, A, @RW2+d16, A  | SUB<br>A, @RW2, @RW2+d16, A   | SUBC<br>A, @RW2, @RW2+d16, A   | SUBC<br>A, @RW2, @RW2+d16, A   | NEG<br>@RW2, @RW2+d16, A   | NEG A,<br>@RW2, @RW2+d16, A   | AND<br>@RW2, A, @RW2+d16, A  | AND<br>@RW2, A, @RW2+d16, A  | OR<br>@RW2, A, @RW2+d16, A  | OR<br>A, @RW2+d16, A | XOR<br>@RW2, A, @RW2+d16, A  | XOR<br>A, @RW2+d16, A | NOT<br>@RW2, @RW2+d16, A   | NOT<br>@RW2, @RW2+d16, A   |
| +B | ADD<br>@RW3, A, @RW3+d16, A  | SUB<br>@RW3, A, @RW3+d16, A  | SUB<br>@RW3, A, @RW3+d16, A  | SUB<br>A, @RW3, @RW3+d16, A   | SUBC<br>A, @RW3, @RW3+d16, A   | SUBC<br>A, @RW3, @RW3+d16, A   | NEG<br>@RW3, @RW3+d16, A   | NEG A,<br>@RW3, @RW3+d16, A   | AND<br>@RW3, A, @RW3+d16, A  | AND<br>@RW3, A, @RW3+d16, A  | OR<br>@RW3, A, @RW3+d16, A  | OR<br>A, @RW3+d16, A | XOR<br>@RW3, A, @RW3+d16, A  | XOR<br>A, @RW3+d16, A | NOT<br>@RW3, @RW3+d16, A   | NOT<br>@RW3, @RW3+d16, A   |
| +C | ADD<br>@RW0+, A, @RW0+RW7, A | SUB<br>@RW0+, A, @RW0+RW7, A | SUB<br>@RW0+, A, @RW0+RW7, A | SUB<br>A, @RW0+, @RW0+RW7, A  | SUBC<br>A, @RW0+, @RW0+RW7, A  | SUBC<br>A, @RW0+, @RW0+RW7, A  | NEG<br>@RW0+, @RW0+RW7, A  | NEG A,<br>@RW0+, @RW0+RW7, A  | AND<br>@RW0+, A, @RW0+RW7, A | AND<br>@RW0+, A, @RW0+RW7, A | OR<br>@RW0+, A, @RW0+RW7, A | OR<br>A, @RW0+RW7, A | XOR<br>@RW0+, A, @RW0+RW7, A | XOR<br>A, @RW0+RW7, A | NOT<br>@RW0+, @RW0+RW7, A  | NOT<br>@RW0+, @RW0+RW7, A  |
| +D | ADD<br>@RW1+, A, @RW1+RW7, A | SUB<br>@RW1+, A, @RW1+RW7, A | SUB<br>@RW1+, A, @RW1+RW7, A | SUB<br>A, @RW1+, @RW1+RW7, A  | SUBC<br>A, @RW1+, @RW1+RW7, A  | SUBC<br>A, @RW1+, @RW1+RW7, A  | NEG<br>@RW1+, @RW1+RW7, A  | NEG A,<br>@RW1+, @RW1+RW7, A  | AND<br>@RW1+, A, @RW1+RW7, A | AND<br>@RW1+, A, @RW1+RW7, A | OR<br>@RW1+, A, @RW1+RW7, A | OR<br>A, @RW1+RW7, A | XOR<br>@RW1+, A, @RW1+RW7, A | XOR<br>A, @RW1+RW7, A | NOT<br>@RW1+, @RW1+RW7, A  | NOT<br>@RW1+, @RW1+RW7, A  |
| +E | ADD<br>@RW2+, A, @PC+d16, A  | SUB<br>@RW2+, A, @PC+d16, A  | SUB<br>@RW2+, A, @PC+d16, A  | SUB<br>A, @RW2+, @PC+d16, A   | SUBC<br>A, @RW2+, @PC+d16, A   | SUBC<br>A, @RW2+, @PC+d16, A   | NEG<br>@RW2+, @PC+d16, A   | NEG A,<br>@RW2+, @PC+d16, A   | AND<br>@RW2+, A, @PC+d16, A  | AND<br>@RW2+, A, @PC+d16, A  | OR<br>@RW2+, A, @PC+d16, A  | OR<br>A, @PC+d16, A  | XOR<br>@RW2+, A, @PC+d16, A  | XOR<br>A, @PC+d16, A  | NOT<br>@RW2+, @PC+d16, A   | NOT<br>@RW2+, @PC+d16, A   |
| +F | ADD<br>@RW3+, A, addr16, A   | SUB<br>@RW3+, A, addr16, A   | SUB<br>@RW3+, A, addr16, A   | SUB<br>A, @RW3+, A, addr16, A | SUBC<br>A, @RW3+, A, addr16, A | SUBC<br>A, @RW3+, A, addr16, A | NEG<br>@RW3+, A, addr16, A | NEG A,<br>@RW3+, A, addr16, A | AND<br>@RW3+, A, addr16, A   | AND<br>@RW3+, A, addr16, A   | OR<br>@RW3+, A, addr16, A   | OR<br>A, addr16, A   | XOR<br>@RW3+, A, addr16, A   | XOR<br>A, addr16, A   | NOT<br>@RW3+, A, addr16, A | NOT<br>@RW3+, A, addr16, A |

**Table D.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<br>A, RW0+<br>@ RW0+d8  | ADDW A,<br>RW0+<br>@ RW0+d8  | SUBW<br>A, RW0+<br>@ RW0+d8  | SUBW A,<br>@ RW0+d8  | ADDCW<br>A, RW0+<br>@ RW0+d8  | ADDCW A,<br>@ RW0+d8  | CMPW<br>A, RW0+<br>@ RW0+d8  | CMPW A,<br>@ RW0+d8  | ANDW<br>A, RW0+<br>@ RW0+d8  | ANDW A,<br>@ RW0+d8  | ORW<br>A, RW0+<br>@ RW0+d8  | ORW A,<br>@ RW0+d8  | XORW<br>A, RW0+<br>@ RW0+d8  | XORW A,<br>@ RW0+d8  | DWBZN<br>RW0, r1<br>@ RW0+d8,r   | DWBZN<br>@ RW0+d8,r            |
| +1 | ADDW<br>A, RW1+<br>@ RW1+d8  | ADDW A,<br>RW1+<br>@ RW1+d8  | SUBW<br>A, RW1+<br>@ RW1+d8  | SUBW A,<br>@ RW1+d8  | ADDCW<br>A, RW1+<br>@ RW1+d8  | ADDCW A,<br>@ RW1+d8  | CMPW<br>A, RW1+<br>@ RW1+d8  | CMPW A,<br>@ RW1+d8  | ANDW<br>A, RW1+<br>@ RW1+d8  | ANDW A,<br>@ RW1+d8  | ORW<br>A, RW1+<br>@ RW1+d8  | ORW A,<br>@ RW1+d8  | XORW<br>A, RW1+<br>@ RW1+d8  | XORW A,<br>@ RW1+d8  | DWBZN<br>RW1, r1<br>@ RW1+d8,r   | DWBZN<br>@ RW1+d8,r            |
| +2 | ADDW<br>A, RW2+<br>@ RW2+d8  | ADDW A,<br>RW2+<br>@ RW2+d8  | SUBW<br>A, RW2+<br>@ RW2+d8  | SUBW A,<br>@ RW2+d8  | ADDCW<br>A, RW2+<br>@ RW2+d8  | ADDCW A,<br>@ RW2+d8  | CMPW<br>A, RW2+<br>@ RW2+d8  | CMPW A,<br>@ RW2+d8  | ANDW<br>A, RW2+<br>@ RW2+d8  | ANDW A,<br>@ RW2+d8  | ORW<br>A, RW2+<br>@ RW2+d8  | ORW A,<br>@ RW2+d8  | XORW<br>A, RW2+<br>@ RW2+d8  | XORW A,<br>@ RW2+d8  | DWBZN<br>RW2, r1<br>@ RW2+d8,r   | DWBZN<br>@ RW2+d8,r            |
| +3 | ADDW<br>A, RW3+<br>@ RW3+d8  | ADDW A,<br>RW3+<br>@ RW3+d8  | SUBW<br>A, RW3+<br>@ RW3+d8  | SUBW A,<br>@ RW3+d8  | ADDCW<br>A, RW3+<br>@ RW3+d8  | ADDCW A,<br>@ RW3+d8  | CMPW<br>A, RW3+<br>@ RW3+d8  | CMPW A,<br>@ RW3+d8  | ANDW<br>A, RW3+<br>@ RW3+d8  | ANDW A,<br>@ RW3+d8  | ORW<br>A, RW3+<br>@ RW3+d8  | ORW A,<br>@ RW3+d8  | XORW<br>A, RW3+<br>@ RW3+d8  | XORW A,<br>@ RW3+d8  | DWBZN<br>RW3, r1<br>@ RW3+d8,r   | DWBZN<br>@ RW3+d8,r            |
| +4 | ADDW<br>A, RW4+<br>@ RW4+d8  | ADDW A,<br>RW4+<br>@ RW4+d8  | SUBW<br>A, RW4+<br>@ RW4+d8  | SUBW A,<br>@ RW4+d8  | ADDCW<br>A, RW4+<br>@ RW4+d8  | ADDCW A,<br>@ RW4+d8  | CMPW<br>A, RW4+<br>@ RW4+d8  | CMPW A,<br>@ RW4+d8  | ANDW<br>A, RW4+<br>@ RW4+d8  | ANDW A,<br>@ RW4+d8  | ORW<br>A, RW4+<br>@ RW4+d8  | ORW A,<br>@ RW4+d8  | XORW<br>A, RW4+<br>@ RW4+d8  | XORW A,<br>@ RW4+d8  | DWBZN<br>RW4, r1<br>@ RW4+d8,r   | DWBZN<br>@ RW4+d8,r            |
| +5 | ADDW<br>A, RW5+<br>@ RW5+d8  | ADDW A,<br>RW5+<br>@ RW5+d8  | SUBW<br>A, RW5+<br>@ RW5+d8  | SUBW A,<br>@ RW5+d8  | ADDCW<br>A, RW5+<br>@ RW5+d8  | ADDCW A,<br>@ RW5+d8  | CMPW<br>A, RW5+<br>@ RW5+d8  | CMPW A,<br>@ RW5+d8  | ANDW<br>A, RW5+<br>@ RW5+d8  | ANDW A,<br>@ RW5+d8  | ORW<br>A, RW5+<br>@ RW5+d8  | ORW A,<br>@ RW5+d8  | XORW<br>A, RW5+<br>@ RW5+d8  | XORW A,<br>@ RW5+d8  | DWBZN<br>RW5, r1<br>@ RW5+d8,r   | DWBZN<br>@ RW5+d8,r            |
| +6 | ADDW<br>A, RW6+<br>@ RW6+d8  | ADDW A,<br>RW6+<br>@ RW6+d8  | SUBW<br>A, RW6+<br>@ RW6+d8  | SUBW A,<br>@ RW6+d8  | ADDCW<br>A, RW6+<br>@ RW6+d8  | ADDCW A,<br>@ RW6+d8  | CMPW<br>A, RW6+<br>@ RW6+d8  | CMPW A,<br>@ RW6+d8  | ANDW<br>A, RW6+<br>@ RW6+d8  | ANDW A,<br>@ RW6+d8  | ORW<br>A, RW6+<br>@ RW6+d8  | ORW A,<br>@ RW6+d8  | XORW<br>A, RW6+<br>@ RW6+d8  | XORW A,<br>@ RW6+d8  | DWBZN<br>RW6, r1<br>@ RW6+d8,r   | DWBZN<br>@ RW6+d8,r            |
| +7 | ADDW<br>A, RW7+<br>@ RW7+d8  | ADDW A,<br>RW7+<br>@ RW7+d8  | SUBW<br>A, RW7+<br>@ RW7+d8  | SUBW A,<br>@ RW7+d8  | ADDCW<br>A, RW7+<br>@ RW7+d8  | ADDCW A,<br>@ RW7+d8  | CMPW<br>A, RW7+<br>@ RW7+d8  | CMPW A,<br>@ RW7+d8  | ANDW<br>A, RW7+<br>@ RW7+d8  | ANDW A,<br>@ RW7+d8  | ORW<br>A, RW7+<br>@ RW7+d8  | ORW A,<br>@ RW7+d8  | XORW<br>A, RW7+<br>@ RW7+d8  | XORW A,<br>@ RW7+d8  | DWBZN<br>RW7, r1<br>@ RW7+d8,r   | DWBZN<br>@ RW7+d8,r            |
| +8 | ADDW<br>A, RW0+<br>@ RW0+d16 | ADDW A,<br>RW0+<br>@ RW0+d16 | SUBW<br>A, RW0+<br>@ RW0+d16 | SUBW A,<br>@ RW0+d16 | ADDCW<br>A, RW0+<br>@ RW0+d16 | ADDCW A,<br>@ RW0+d16 | CMPW<br>A, RW0+<br>@ RW0+d16 | CMPW A,<br>@ RW0+d16 | ANDW<br>A, RW0+<br>@ RW0+d16 | ANDW A,<br>@ RW0+d16 | ORW<br>A, RW0+<br>@ RW0+d16 | ORW A,<br>@ RW0+d16 | XORW<br>A, RW0+<br>@ RW0+d16 | XORW A,<br>@ RW0+d16 | DWBZN<br>RW0, r1<br>@ RW0+d16,r  | DWBZN<br>@ RW0+d16,r           |
| +9 | ADDW<br>A, RW1+<br>@ RW1+d16 | ADDW A,<br>RW1+<br>@ RW1+d16 | SUBW<br>A, RW1+<br>@ RW1+d16 | SUBW A,<br>@ RW1+d16 | ADDCW<br>A, RW1+<br>@ RW1+d16 | ADDCW A,<br>@ RW1+d16 | CMPW<br>A, RW1+<br>@ RW1+d16 | CMPW A,<br>@ RW1+d16 | ANDW<br>A, RW1+<br>@ RW1+d16 | ANDW A,<br>@ RW1+d16 | ORW<br>A, RW1+<br>@ RW1+d16 | ORW A,<br>@ RW1+d16 | XORW<br>A, RW1+<br>@ RW1+d16 | XORW A,<br>@ RW1+d16 | DWBZN<br>RW1, r1<br>@ RW1+d16,r  | DWBZN<br>@ RW1+d16,r           |
| +A | ADDW<br>A, RW2+<br>@ RW2+d16 | ADDW A,<br>RW2+<br>@ RW2+d16 | SUBW<br>A, RW2+<br>@ RW2+d16 | SUBW A,<br>@ RW2+d16 | ADDCW<br>A, RW2+<br>@ RW2+d16 | ADDCW A,<br>@ RW2+d16 | CMPW<br>A, RW2+<br>@ RW2+d16 | CMPW A,<br>@ RW2+d16 | ANDW<br>A, RW2+<br>@ RW2+d16 | ANDW A,<br>@ RW2+d16 | ORW<br>A, RW2+<br>@ RW2+d16 | ORW A,<br>@ RW2+d16 | XORW<br>A, RW2+<br>@ RW2+d16 | XORW A,<br>@ RW2+d16 | DWBZN<br>RW2, r1<br>@ RW2+d16,r  | DWBZN<br>@ RW2+d16,r           |
| +B | ADDW<br>A, RW3+<br>@ RW3+d16 | ADDW A,<br>RW3+<br>@ RW3+d16 | SUBW<br>A, RW3+<br>@ RW3+d16 | SUBW A,<br>@ RW3+d16 | ADDCW<br>A, RW3+<br>@ RW3+d16 | ADDCW A,<br>@ RW3+d16 | CMPW<br>A, RW3+<br>@ RW3+d16 | CMPW A,<br>@ RW3+d16 | ANDW<br>A, RW3+<br>@ RW3+d16 | ANDW A,<br>@ RW3+d16 | ORW<br>A, RW3+<br>@ RW3+d16 | ORW A,<br>@ RW3+d16 | XORW<br>A, RW3+<br>@ RW3+d16 | XORW A,<br>@ RW3+d16 | DWBZN<br>RW3, r1<br>@ RW3+d16,r  | DWBZN<br>@ RW3+d16,r           |
| +C | ADDW<br>A, RW0+<br>@ RW0+RW7 | ADDW A,<br>RW0+<br>@ RW0+RW7 | SUBW<br>A, RW0+<br>@ RW0+RW7 | SUBW A,<br>@ RW0+RW7 | ADDCW<br>A, RW0+<br>@ RW0+RW7 | ADDCW A,<br>@ RW0+RW7 | CMPW<br>A, RW0+<br>@ RW0+RW7 | CMPW A,<br>@ RW0+RW7 | ANDW<br>A, RW0+<br>@ RW0+RW7 | ANDW A,<br>@ RW0+RW7 | ORW<br>A, RW0+<br>@ RW0+RW7 | ORW A,<br>@ RW0+RW7 | XORW<br>A, RW0+<br>@ RW0+RW7 | XORW A,<br>@ RW0+RW7 | DWBZN<br>RW0+, r1<br>@ RW0+RW7,r | DWBZN<br>@ RW0+RW7,r           |
| +D | ADDW<br>A, RW1+<br>@ RW1+RW7 | ADDW A,<br>RW1+<br>@ RW1+RW7 | SUBW<br>A, RW1+<br>@ RW1+RW7 | SUBW A,<br>@ RW1+RW7 | ADDCW<br>A, RW1+<br>@ RW1+RW7 | ADDCW A,<br>@ RW1+RW7 | CMPW<br>A, RW1+<br>@ RW1+RW7 | CMPW A,<br>@ RW1+RW7 | ANDW<br>A, RW1+<br>@ RW1+RW7 | ANDW A,<br>@ RW1+RW7 | ORW<br>A, RW1+<br>@ RW1+RW7 | ORW A,<br>@ RW1+RW7 | XORW<br>A, RW1+<br>@ RW1+RW7 | XORW A,<br>@ RW1+RW7 | DWBZN<br>RW1+, r1<br>@ RW1+RW7,r | DWBZN<br>@ RW1+RW7,r           |
| +E | ADDW<br>A, RW2+<br>@ PC+d16  | ADDW A,<br>PC+d16            | SUBW<br>A, RW2+<br>@ PC+d16  | SUBW A,<br>@ PC+d16  | ADDCW<br>A, RW2+<br>@ PC+d16  | ADDCW A,<br>@ PC+d16  | CMPW<br>A, RW2+<br>@ PC+d16  | CMPW A,<br>@ PC+d16  | ANDW<br>A, RW2+<br>@ PC+d16  | ANDW A,<br>@ PC+d16  | ORW<br>A, RW2+<br>@ PC+d16  | ORW A,<br>@ PC+d16  | XORW<br>A, RW2+<br>@ PC+d16  | XORW A,<br>@ PC+d16  | DWBZN<br>RW2+, r1<br>@ PC+d16,r  | DWBZN<br>@ PC+d16,r            |
| +F | ADDW<br>A, RW3+<br>addr 16   | ADDW A,<br>addr 16           | SUBW<br>A, RW3+<br>addr 16   | SUBW A,<br>addr 16   | ADDCW<br>A, RW3+<br>addr 16   | ADDCW A,<br>addr 16   | CMPW<br>A, RW3+<br>addr 16   | CMPW A,<br>addr 16   | ANDW<br>A, RW3+<br>addr 16   | ANDW A,<br>addr 16   | ORW<br>A, RW3+<br>addr 16   | ORW A,<br>addr 16   | XORW<br>A, RW3+<br>addr 16   | XORW A,<br>addr 16   | DWBZN<br>RW3+, r1<br>addr 16     | DWBZN<br>@ RW3+, r1<br>addr 16 |

Table D.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<br>RW0, A, @RW0-d8, A    | SUBW<br>RW0, A, @RW0-d8, A    | SUBW<br>A, RW0, @RW0-d8     | NEGW<br>RW0, @RW0-d8     | ANDW<br>RW0, A, @RW0-d8, A    | ORW<br>RW0, A, @RW0-d8, A    | ANDW<br>RW0, A, @RW0-d8, A    | XORW<br>RW0, A, @RW0-d8, A    | NOTW<br>RW0, @RW0-d8     |    |    |    |    |    |    |    |
| +1 | ADDW<br>RW1, A, @RW1-d8, A    | SUBW<br>RW1, A, @RW1-d8, A    | SUBW<br>A, RW1, @RW1-d8     | NEGW<br>RW1, @RW1-d8     | ANDW<br>RW1, A, @RW1-d8, A    | ORW<br>RW1, A, @RW1-d8, A    | ANDW<br>RW1, A, @RW1-d8, A    | XORW<br>RW1, A, @RW1-d8, A    | NOTW<br>RW1, @RW1-d8     |    |    |    |    |    |    |    |
| +2 | ADDW<br>RW2, A, @RW2-d8, A    | SUBW<br>RW2, A, @RW2-d8, A    | SUBW<br>A, RW2, @RW2-d8     | NEGW<br>RW2, @RW2-d8     | ANDW<br>RW2, A, @RW2-d8, A    | ORW<br>RW2, A, @RW2-d8, A    | ANDW<br>RW2, A, @RW2-d8, A    | XORW<br>RW2, A, @RW2-d8, A    | NOTW<br>RW2, @RW2-d8     |    |    |    |    |    |    |    |
| +3 | ADDW<br>RW3, A, @RW3-d8, A    | SUBW<br>RW3, A, @RW3-d8, A    | SUBW<br>A, RW3, @RW3-d8     | NEGW<br>RW3, @RW3-d8     | ANDW<br>RW3, A, @RW3-d8, A    | ORW<br>RW3, A, @RW3-d8, A    | ANDW<br>RW3, A, @RW3-d8, A    | XORW<br>RW3, A, @RW3-d8, A    | NOTW<br>RW3, @RW3-d8     |    |    |    |    |    |    |    |
| +4 | ADDW<br>RW4, A, @RW4-d8, A    | SUBW<br>RW4, A, @RW4-d8, A    | SUBW<br>A, RW4, @RW4-d8     | NEGW<br>RW4, @RW4-d8     | ANDW<br>RW4, A, @RW4-d8, A    | ORW<br>RW4, A, @RW4-d8, A    | ANDW<br>RW4, A, @RW4-d8, A    | XORW<br>RW4, A, @RW4-d8, A    | NOTW<br>RW4, @RW4-d8     |    |    |    |    |    |    |    |
| +5 | ADDW<br>RW5, A, @RW5-d8, A    | SUBW<br>RW5, A, @RW5-d8, A    | SUBW<br>A, RW5, @RW5-d8     | NEGW<br>RW5, @RW5-d8     | ANDW<br>RW5, A, @RW5-d8, A    | ORW<br>RW5, A, @RW5-d8, A    | ANDW<br>RW5, A, @RW5-d8, A    | XORW<br>RW5, A, @RW5-d8, A    | NOTW<br>RW5, @RW5-d8     |    |    |    |    |    |    |    |
| +6 | ADDW<br>RW6, A, @RW6-d8, A    | SUBW<br>RW6, A, @RW6-d8, A    | SUBW<br>A, RW6, @RW6-d8     | NEGW<br>RW6, @RW6-d8     | ANDW<br>RW6, A, @RW6-d8, A    | ORW<br>RW6, A, @RW6-d8, A    | ANDW<br>RW6, A, @RW6-d8, A    | XORW<br>RW6, A, @RW6-d8, A    | NOTW<br>RW6, @RW6-d8     |    |    |    |    |    |    |    |
| +7 | ADDW<br>RW7, A, @RW7-d8, A    | SUBW<br>RW7, A, @RW7-d8, A    | SUBW<br>A, RW7, @RW7-d8     | NEGW<br>RW7, @RW7-d8     | ANDW<br>RW7, A, @RW7-d8, A    | ORW<br>RW7, A, @RW7-d8, A    | ANDW<br>RW7, A, @RW7-d8, A    | XORW<br>RW7, A, @RW7-d8, A    | NOTW<br>RW7, @RW7-d8     |    |    |    |    |    |    |    |
| +8 | ADDW<br>@RW0, A, @RW0-d16, A  | SUBW<br>@RW0, A, @RW0-d16, A  | SUBW<br>A, @RW0, @RW0-d16   | NEGW<br>@RW0, @RW0-d16   | ANDW<br>@RW0, A, @RW0-d16, A  | ORW<br>@RW0, A, @RW0-d16, A  | ANDW<br>@RW0, A, @RW0-d16, A  | XORW<br>@RW0, A, @RW0-d16, A  | NOTW<br>@RW0, @RW0-d16   |    |    |    |    |    |    |    |
| +9 | ADDW<br>@RW1, A, @RW1-d16, A  | SUBW<br>@RW1, A, @RW1-d16, A  | SUBW<br>A, @RW1, @RW1-d16   | NEGW<br>@RW1, @RW1-d16   | ANDW<br>@RW1, A, @RW1-d16, A  | ORW<br>@RW1, A, @RW1-d16, A  | ANDW<br>@RW1, A, @RW1-d16, A  | XORW<br>@RW1, A, @RW1-d16, A  | NOTW<br>@RW1, @RW1-d16   |    |    |    |    |    |    |    |
| +A | ADDW<br>@RW2, A, @RW2-d16, A  | SUBW<br>@RW2, A, @RW2-d16, A  | SUBW<br>A, @RW2, @RW2-d16   | NEGW<br>@RW2, @RW2-d16   | ANDW<br>@RW2, A, @RW2-d16, A  | ORW<br>@RW2, A, @RW2-d16, A  | ANDW<br>@RW2, A, @RW2-d16, A  | XORW<br>@RW2, A, @RW2-d16, A  | NOTW<br>@RW2, @RW2-d16   |    |    |    |    |    |    |    |
| +B | ADDW<br>@RW3, A, @RW3-d16, A  | SUBW<br>@RW3, A, @RW3-d16, A  | SUBW<br>A, @RW3, @RW3-d16   | NEGW<br>@RW3, @RW3-d16   | ANDW<br>@RW3, A, @RW3-d16, A  | ORW<br>@RW3, A, @RW3-d16, A  | ANDW<br>@RW3, A, @RW3-d16, A  | XORW<br>@RW3, A, @RW3-d16, A  | NOTW<br>@RW3, @RW3-d16   |    |    |    |    |    |    |    |
| +C | ADDW<br>@RW0+, A, @RW0+RW7, A | SUBW<br>@RW0+, A, @RW0+RW7, A | SUBW<br>A, @RW0+, @RW0+RW7  | NEGW<br>@RW0+, @RW0+RW7  | ANDW<br>@RW0+, A, @RW0+RW7, A | ORW<br>@RW0+, A, @RW0+RW7, A | ANDW<br>@RW0+, A, @RW0+RW7, A | XORW<br>@RW0+, A, @RW0+RW7, A | NOTW<br>@RW0+, @RW0+RW7  |    |    |    |    |    |    |    |
| +D | ADDW<br>@RW1+, A, @RW1+RW7, A | SUBW<br>@RW1+, A, @RW1+RW7, A | SUBW<br>A, @RW1+, @RW1+RW7  | NEGW<br>@RW1+, @RW1+RW7  | ANDW<br>@RW1+, A, @RW1+RW7, A | ORW<br>@RW1+, A, @RW1+RW7, A | ANDW<br>@RW1+, A, @RW1+RW7, A | XORW<br>@RW1+, A, @RW1+RW7, A | NOTW<br>@RW1+, @RW1+RW7  |    |    |    |    |    |    |    |
| +E | ADDW<br>@RW2+, A, @PC-d16, A  | SUBW<br>@RW2+, A, @PC-d16, A  | SUBW<br>A, @RW2+, @PC-d16   | NEGW<br>@RW2+, @PC-d16   | ANDW<br>@RW2+, A, @PC-d16, A  | ORW<br>@RW2+, A, @PC-d16, A  | ANDW<br>@RW2+, A, @PC-d16, A  | XORW<br>@RW2+, A, @PC-d16, A  | NOTW<br>@RW2+, @PC-d16   |    |    |    |    |    |    |    |
| +F | ADDW<br>@RW3+, A, addr16, A   | SUBW<br>@RW3+, A, addr16, A   | SUBW<br>A, @RW3+, addr16, A | NEGW<br>@RW3+, addr16, A | ANDW<br>@RW3+, A, addr16, A   | ORW<br>@RW3+, A, addr16, A   | ANDW<br>@RW3+, A, addr16, A   | XORW<br>@RW3+, A, addr16, A   | NOTW<br>@RW3+, addr16, A |    |    |    |    |    |    |    |

**Table D.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<br>A, R0' @RW0+d8      | MULU<br>A, R0' @RW0+d8      | MULUW<br>A, RW0' @RW0+d8    | MULUW<br>A, RW0' @RW0+d8    | MUL<br>A, R0' @RW0+d8     | MUL<br>A, R0' @RW0+d8     | MULW<br>A, RW0' @RW0+d8    | MULW<br>A, RW0' @RW0+d8    | DIVU<br>A, R0' @RW0+d8      | DIVU<br>A, R0' @RW0+d8      | DIVUW<br>A, RW0' @RW0+d8    | DIVUW<br>A, RW0' @RW0+d8    | DIV<br>A, R0' @RW0+d8      | DIV<br>A, R0' @RW0+d8      | DIVW<br>A, RW0' @RW0+d8     | DIVW<br>A, RW0' @RW0+d8     |
| +1 | MULU<br>A, R1' @RW1+d8      | MULU<br>A, R1' @RW1+d8      | MULUW<br>A, RW1' @RW1+d8    | MULUW<br>A, RW1' @RW1+d8    | MUL<br>A, R1' @RW1+d8     | MUL<br>A, R1' @RW1+d8     | MULW<br>A, RW1' @RW1+d8    | MULW<br>A, RW1' @RW1+d8    | DIVU<br>A, R1' @RW1+d8      | DIVU<br>A, R1' @RW1+d8      | DIVUW<br>A, RW1' @RW1+d8    | DIVUW<br>A, RW1' @RW1+d8    | DIV<br>A, R1' @RW1+d8      | DIV<br>A, R1' @RW1+d8      | DIVW<br>A, RW1' @RW1+d8     | DIVW<br>A, RW1' @RW1+d8     |
| +2 | MULU<br>A, R2' @RW2+d8      | MULU<br>A, R2' @RW2+d8      | MULUW<br>A, RW2' @RW2+d8    | MULUW<br>A, RW2' @RW2+d8    | MUL<br>A, R2' @RW2+d8     | MUL<br>A, R2' @RW2+d8     | MULW<br>A, RW2' @RW2+d8    | MULW<br>A, RW2' @RW2+d8    | DIVU<br>A, R2' @RW2+d8      | DIVU<br>A, R2' @RW2+d8      | DIVUW<br>A, RW2' @RW2+d8    | DIVUW<br>A, RW2' @RW2+d8    | DIV<br>A, R2' @RW2+d8      | DIV<br>A, R2' @RW2+d8      | DIVW<br>A, RW2' @RW2+d8     | DIVW<br>A, RW2' @RW2+d8     |
| +3 | MULU<br>A, R3' @RW3+d8      | MULU<br>A, R3' @RW3+d8      | MULUW<br>A, RW3' @RW3+d8    | MULUW<br>A, RW3' @RW3+d8    | MUL<br>A, R3' @RW3+d8     | MUL<br>A, R3' @RW3+d8     | MULW<br>A, RW3' @RW3+d8    | MULW<br>A, RW3' @RW3+d8    | DIVU<br>A, R3' @RW3+d8      | DIVU<br>A, R3' @RW3+d8      | DIVUW<br>A, RW3' @RW3+d8    | DIVUW<br>A, RW3' @RW3+d8    | DIV<br>A, R3' @RW3+d8      | DIV<br>A, R3' @RW3+d8      | DIVW<br>A, RW3' @RW3+d8     | DIVW<br>A, RW3' @RW3+d8     |
| +4 | MULU<br>A, R4' @RW4+d8      | MULU<br>A, R4' @RW4+d8      | MULUW<br>A, RW4' @RW4+d8    | MULUW<br>A, RW4' @RW4+d8    | MUL<br>A, R4' @RW4+d8     | MUL<br>A, R4' @RW4+d8     | MULW<br>A, RW4' @RW4+d8    | MULW<br>A, RW4' @RW4+d8    | DIVU<br>A, R4' @RW4+d8      | DIVU<br>A, R4' @RW4+d8      | DIVUW<br>A, RW4' @RW4+d8    | DIVUW<br>A, RW4' @RW4+d8    | DIV<br>A, R4' @RW4+d8      | DIV<br>A, R4' @RW4+d8      | DIVW<br>A, RW4' @RW4+d8     | DIVW<br>A, RW4' @RW4+d8     |
| +5 | MULU<br>A, R5' @RW5+d8      | MULU<br>A, R5' @RW5+d8      | MULUW<br>A, RW5' @RW5+d8    | MULUW<br>A, RW5' @RW5+d8    | MUL<br>A, R5' @RW5+d8     | MUL<br>A, R5' @RW5+d8     | MULW<br>A, RW5' @RW5+d8    | MULW<br>A, RW5' @RW5+d8    | DIVU<br>A, R5' @RW5+d8      | DIVU<br>A, R5' @RW5+d8      | DIVUW<br>A, RW5' @RW5+d8    | DIVUW<br>A, RW5' @RW5+d8    | DIV<br>A, R5' @RW5+d8      | DIV<br>A, R5' @RW5+d8      | DIVW<br>A, RW5' @RW5+d8     | DIVW<br>A, RW5' @RW5+d8     |
| +6 | MULU<br>A, R6' @RW6+d8      | MULU<br>A, R6' @RW6+d8      | MULUW<br>A, RW6' @RW6+d8    | MULUW<br>A, RW6' @RW6+d8    | MUL<br>A, R6' @RW6+d8     | MUL<br>A, R6' @RW6+d8     | MULW<br>A, RW6' @RW6+d8    | MULW<br>A, RW6' @RW6+d8    | DIVU<br>A, R6' @RW6+d8      | DIVU<br>A, R6' @RW6+d8      | DIVUW<br>A, RW6' @RW6+d8    | DIVUW<br>A, RW6' @RW6+d8    | DIV<br>A, R6' @RW6+d8      | DIV<br>A, R6' @RW6+d8      | DIVW<br>A, RW6' @RW6+d8     | DIVW<br>A, RW6' @RW6+d8     |
| +7 | MULU<br>A, R7' @RW7+d8      | MULU<br>A, R7' @RW7+d8      | MULUW<br>A, RW7' @RW7+d8    | MULUW<br>A, RW7' @RW7+d8    | MUL<br>A, R7' @RW7+d8     | MUL<br>A, R7' @RW7+d8     | MULW<br>A, RW7' @RW7+d8    | MULW<br>A, RW7' @RW7+d8    | DIVU<br>A, R7' @RW7+d8      | DIVU<br>A, R7' @RW7+d8      | DIVUW<br>A, RW7' @RW7+d8    | DIVUW<br>A, RW7' @RW7+d8    | DIV<br>A, R7' @RW7+d8      | DIV<br>A, R7' @RW7+d8      | DIVW<br>A, RW7' @RW7+d8     | DIVW<br>A, RW7' @RW7+d8     |
| +8 | MULU<br>A, @RW0' @RW0+d16   | MULU<br>A, @RW0' @RW0+d16   | MULUW<br>A, RW0' @RW0+d16   | MULUW<br>A, RW0' @RW0+d16   | MUL<br>A, @RW0' @RW0+d16  | MUL<br>A, @RW0' @RW0+d16  | MULW<br>A, RW0' @RW0+d16   | MULW<br>A, RW0' @RW0+d16   | DIVU<br>A, @RW0' @RW0+d16   | DIVU<br>A, @RW0' @RW0+d16   | DIVUW<br>A, RW0' @RW0+d16   | DIVUW<br>A, RW0' @RW0+d16   | DIV<br>A, @RW0' @RW0+d16   | DIV<br>A, @RW0' @RW0+d16   | DIVW<br>A, RW0' @RW0+d16    | DIVW<br>A, RW0' @RW0+d16    |
| +9 | MULU<br>A, @RW1' @RW1+d16   | MULU<br>A, @RW1' @RW1+d16   | MULUW<br>A, RW1' @RW1+d16   | MULUW<br>A, RW1' @RW1+d16   | MUL<br>A, @RW1' @RW1+d16  | MUL<br>A, @RW1' @RW1+d16  | MULW<br>A, RW1' @RW1+d16   | MULW<br>A, RW1' @RW1+d16   | DIVU<br>A, @RW1' @RW1+d16   | DIVU<br>A, @RW1' @RW1+d16   | DIVUW<br>A, RW1' @RW1+d16   | DIVUW<br>A, RW1' @RW1+d16   | DIV<br>A, @RW1' @RW1+d16   | DIV<br>A, @RW1' @RW1+d16   | DIVW<br>A, RW1' @RW1+d16    | DIVW<br>A, RW1' @RW1+d16    |
| +A | MULU<br>A, @RW2' @RW2+d16   | MULU<br>A, @RW2' @RW2+d16   | MULUW<br>A, RW2' @RW2+d16   | MULUW<br>A, RW2' @RW2+d16   | MUL<br>A, @RW2' @RW2+d16  | MUL<br>A, @RW2' @RW2+d16  | MULW<br>A, RW2' @RW2+d16   | MULW<br>A, RW2' @RW2+d16   | DIVU<br>A, @RW2' @RW2+d16   | DIVU<br>A, @RW2' @RW2+d16   | DIVUW<br>A, RW2' @RW2+d16   | DIVUW<br>A, RW2' @RW2+d16   | DIV<br>A, @RW2' @RW2+d16   | DIV<br>A, @RW2' @RW2+d16   | DIVW<br>A, RW2' @RW2+d16    | DIVW<br>A, RW2' @RW2+d16    |
| +B | MULU<br>A, @RW3' @RW3+d16   | MULU<br>A, @RW3' @RW3+d16   | MULUW<br>A, RW3' @RW3+d16   | MULUW<br>A, RW3' @RW3+d16   | MUL<br>A, @RW3' @RW3+d16  | MUL<br>A, @RW3' @RW3+d16  | MULW<br>A, RW3' @RW3+d16   | MULW<br>A, RW3' @RW3+d16   | DIVU<br>A, @RW3' @RW3+d16   | DIVU<br>A, @RW3' @RW3+d16   | DIVUW<br>A, RW3' @RW3+d16   | DIVUW<br>A, RW3' @RW3+d16   | DIV<br>A, @RW3' @RW3+d16   | DIV<br>A, @RW3' @RW3+d16   | DIVW<br>A, RW3' @RW3+d16    | DIVW<br>A, RW3' @RW3+d16    |
| +C | MULU<br>A, @RW0+1' @RW0+RW7 | MULU<br>A, @RW0+1' @RW0+RW7 | MULUW<br>A, RW0+1' @RW0+RW7 | MULUW<br>A, RW0+1' @RW0+RW7 | MUL<br>A, @RW0+1' @RW0    | MUL<br>A, @RW0+1' @RW0    | MULW<br>A, RW0+1' @RW0+RW7 | MULW<br>A, RW0+1' @RW0+RW7 | DIVU<br>A, @RW0+1' @RW0+RW7 | DIVU<br>A, @RW0+1' @RW0+RW7 | DIVUW<br>A, RW0+1' @RW0+RW7 | DIVUW<br>A, RW0+1' @RW0+RW7 | DIV<br>A, @RW0+1' @RW0+RW7 | DIV<br>A, @RW0+1' @RW0+RW7 | DIVW<br>A, RW0+1' @RW0+RW7  | DIVW<br>A, RW0+1' @RW0+RW7  |
| +D | MULU<br>A, @RW1+1' @RW1+RW7 | MULU<br>A, @RW1+1' @RW1+RW7 | MULUW<br>A, RW1+1' @RW1+RW7 | MULUW<br>A, RW1+1' @RW1+RW7 | MUL<br>A, @RW1+1' @RW1    | MUL<br>A, @RW1+1' @RW1    | MULW<br>A, RW1+1' @RW1+RW7 | MULW<br>A, RW1+1' @RW1+RW7 | DIVU<br>A, @RW1+1' @RW1+RW7 | DIVU<br>A, @RW1+1' @RW1+RW7 | DIVUW<br>A, RW1+1' @RW1+RW7 | DIVUW<br>A, RW1+1' @RW1+RW7 | DIV<br>A, @RW1+1' @RW1+RW7 | DIV<br>A, @RW1+1' @RW1+RW7 | DIVW<br>A, @RW1+1' @RW1+RW7 | DIVW<br>A, @RW1+1' @RW1+RW7 |
| +E | MULU<br>A, @RW2+1' @PC+d16  | MULU<br>A, @RW2+1' @PC+d16  | MULUW<br>A, RW2+1' @PC+d16  | MULUW<br>A, RW2+1' @PC+d16  | MUL<br>A, @RW2+1' @PC+d16 | MUL<br>A, @RW2+1' @PC+d16 | MULW<br>A, RW2+1' @PC+d16  | MULW<br>A, RW2+1' @PC+d16  | DIVU<br>A, @RW2+1' @PC+d16  | DIVU<br>A, @RW2+1' @PC+d16  | DIVUW<br>A, RW2+1' @PC+d16  | DIVUW<br>A, RW2+1' @PC+d16  | DIV<br>A, @RW2+1' @PC+d16  | DIV<br>A, @RW2+1' @PC+d16  | DIVW<br>A, @RW2+1' @PC+d16  | DIVW<br>A, @RW2+1' @PC+d16  |
| +F | MULU<br>A, @RW3+1' addr16   | MULU<br>A, @RW3+1' addr16   | MULUW<br>A, RW3+1' addr16   | MULUW<br>A, RW3+1' addr16   | MUL<br>A, @RW3+1' addr16  | MUL<br>A, @RW3+1' addr16  | MULW<br>A, RW3+1' addr16   | MULW<br>A, RW3+1' addr16   | DIVU<br>A, @RW3+1' addr16   | DIVU<br>A, @RW3+1' addr16   | DIVUW<br>A, RW3+1' addr16   | DIVUW<br>A, RW3+1' addr16   | DIV<br>A, @RW3+1' addr16   | DIV<br>A, @RW3+1' addr16   | DIVW<br>A, @RW3+1' addr16   | DIVW<br>A, @RW3+1' addr16   |

**Table D.9-15 MOVEA RWi, ea Instruction (First Byte = 79<sub>H</sub>)**

[illegible]

**Table D.9-16 MOV Ri, ea Instruction (First Byte = 7A<sub>H</sub>)**

[illegible]





Table D.9-18 MOV ea, Ri Instruction (First Byte = 7C<sub>H</sub>)

|    | 00                             | 10                             | 20                             | 30                             | 40                             | 50                             | 60                             | 70                             | 80                             | 90                             | A0                               | B0                               | C0                               | D0                               | E0                               | F0                               |
|----|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| +0 | MOV R0, R0, @RW0-d8, R0        | MOV R0, R1, @RW0-d8, R1        | MOV R0, R2, @RW0-d8, R2        | MOV R0, R3, @RW0-d8, R3        | MOV R0, R4, @RW0-d8, R4        | MOV R0, R5, @RW0-d8, R5        | MOV R0, R6, @RW0-d8, R6        | MOV R0, R7, @RW0-d8, R7        | MOV R0, R8, @RW0-d8, R8        | MOV R0, R9, @RW0-d8, R9        | MOV R0, R10, @RW0-d8, R10        | MOV R0, R11, @RW0-d8, R11        | MOV R0, R12, @RW0-d8, R12        | MOV R0, R13, @RW0-d8, R13        | MOV R0, R14, @RW0-d8, R14        | MOV R0, R15, @RW0-d8, R15        |
| +1 | MOV R1, R0, @RW1-d8, R0        | MOV R1, R1, @RW1-d8, R1        | MOV R1, R2, @RW1-d8, R2        | MOV R1, R3, @RW1-d8, R3        | MOV R1, R4, @RW1-d8, R4        | MOV R1, R5, @RW1-d8, R5        | MOV R1, R6, @RW1-d8, R6        | MOV R1, R7, @RW1-d8, R7        | MOV R1, R8, @RW1-d8, R8        | MOV R1, R9, @RW1-d8, R9        | MOV R1, R10, @RW1-d8, R10        | MOV R1, R11, @RW1-d8, R11        | MOV R1, R12, @RW1-d8, R12        | MOV R1, R13, @RW1-d8, R13        | MOV R1, R14, @RW1-d8, R14        | MOV R1, R15, @RW1-d8, R15        |
| +2 | MOV R2, R0, @RW2-d8, R0        | MOV R2, R1, @RW2-d8, R1        | MOV R2, R2, @RW2-d8, R2        | MOV R2, R3, @RW2-d8, R3        | MOV R2, R4, @RW2-d8, R4        | MOV R2, R5, @RW2-d8, R5        | MOV R2, R6, @RW2-d8, R6        | MOV R2, R7, @RW2-d8, R7        | MOV R2, R8, @RW2-d8, R8        | MOV R2, R9, @RW2-d8, R9        | MOV R2, R10, @RW2-d8, R10        | MOV R2, R11, @RW2-d8, R11        | MOV R2, R12, @RW2-d8, R12        | MOV R2, R13, @RW2-d8, R13        | MOV R2, R14, @RW2-d8, R14        | MOV R2, R15, @RW2-d8, R15        |
| +3 | MOV R3, R0, @RW3-d8, R0        | MOV R3, R1, @RW3-d8, R1        | MOV R3, R2, @RW3-d8, R2        | MOV R3, R3, @RW3-d8, R3        | MOV R3, R4, @RW3-d8, R4        | MOV R3, R5, @RW3-d8, R5        | MOV R3, R6, @RW3-d8, R6        | MOV R3, R7, @RW3-d8, R7        | MOV R3, R8, @RW3-d8, R8        | MOV R3, R9, @RW3-d8, R9        | MOV R3, R10, @RW3-d8, R10        | MOV R3, R11, @RW3-d8, R11        | MOV R3, R12, @RW3-d8, R12        | MOV R3, R13, @RW3-d8, R13        | MOV R3, R14, @RW3-d8, R14        | MOV R3, R15, @RW3-d8, R15        |
| +4 | MOV R4, R0, @RW4-d8, R0        | MOV R4, R1, @RW4-d8, R1        | MOV R4, R2, @RW4-d8, R2        | MOV R4, R3, @RW4-d8, R3        | MOV R4, R4, @RW4-d8, R4        | MOV R4, R5, @RW4-d8, R5        | MOV R4, R6, @RW4-d8, R6        | MOV R4, R7, @RW4-d8, R7        | MOV R4, R8, @RW4-d8, R8        | MOV R4, R9, @RW4-d8, R9        | MOV R4, R10, @RW4-d8, R10        | MOV R4, R11, @RW4-d8, R11        | MOV R4, R12, @RW4-d8, R12        | MOV R4, R13, @RW4-d8, R13        | MOV R4, R14, @RW4-d8, R14        | MOV R4, R15, @RW4-d8, R15        |
| +5 | MOV R5, R0, @RW5-d8, R0        | MOV R5, R1, @RW5-d8, R1        | MOV R5, R2, @RW5-d8, R2        | MOV R5, R3, @RW5-d8, R3        | MOV R5, R4, @RW5-d8, R4        | MOV R5, R5, @RW5-d8, R5        | MOV R5, R6, @RW5-d8, R6        | MOV R5, R7, @RW5-d8, R7        | MOV R5, R8, @RW5-d8, R8        | MOV R5, R9, @RW5-d8, R9        | MOV R5, R10, @RW5-d8, R10        | MOV R5, R11, @RW5-d8, R11        | MOV R5, R12, @RW5-d8, R12        | MOV R5, R13, @RW5-d8, R13        | MOV R5, R14, @RW5-d8, R14        | MOV R5, R15, @RW5-d8, R15        |
| +6 | MOV R6, R0, @RW6-d8, R0        | MOV R6, R1, @RW6-d8, R1        | MOV R6, R2, @RW6-d8, R2        | MOV R6, R3, @RW6-d8, R3        | MOV R6, R4, @RW6-d8, R4        | MOV R6, R5, @RW6-d8, R5        | MOV R6, R6, @RW6-d8, R6        | MOV R6, R7, @RW6-d8, R7        | MOV R6, R8, @RW6-d8, R8        | MOV R6, R9, @RW6-d8, R9        | MOV R6, R10, @RW6-d8, R10        | MOV R6, R11, @RW6-d8, R11        | MOV R6, R12, @RW6-d8, R12        | MOV R6, R13, @RW6-d8, R13        | MOV R6, R14, @RW6-d8, R14        | MOV R6, R15, @RW6-d8, R15        |
| +7 | MOV R7, R0, @RW7-d8, R0        | MOV R7, R1, @RW7-d8, R1        | MOV R7, R2, @RW7-d8, R2        | MOV R7, R3, @RW7-d8, R3        | MOV R7, R4, @RW7-d8, R4        | MOV R7, R5, @RW7-d8, R5        | MOV R7, R6, @RW7-d8, R6        | MOV R7, R7, @RW7-d8, R7        | MOV R7, R8, @RW7-d8, R8        | MOV R7, R9, @RW7-d8, R9        | MOV R7, R10, @RW7-d8, R10        | MOV R7, R11, @RW7-d8, R11        | MOV R7, R12, @RW7-d8, R12        | MOV R7, R13, @RW7-d8, R13        | MOV R7, R14, @RW7-d8, R14        | MOV R7, R15, @RW7-d8, R15        |
| +8 | MOV @RW0, R0, @RW0-d16, R0     | MOV @RW0, R1, @RW0-d16, R1     | MOV @RW0, R2, @RW0-d16, R2     | MOV @RW0, R3, @RW0-d16, R3     | MOV @RW0, R4, @RW0-d16, R4     | MOV @RW0, R5, @RW0-d16, R5     | MOV @RW0, R6, @RW0-d16, R6     | MOV @RW0, R7, @RW0-d16, R7     | MOV @RW0, R8, @RW0-d16, R8     | MOV @RW0, R9, @RW0-d16, R9     | MOV @RW0, R10, @RW0-d16, R10     | MOV @RW0, R11, @RW0-d16, R11     | MOV @RW0, R12, @RW0-d16, R12     | MOV @RW0, R13, @RW0-d16, R13     | MOV @RW0, R14, @RW0-d16, R14     | MOV @RW0, R15, @RW0-d16, R15     |
| +9 | MOV @RW1, R0, @RW1-d16, R0     | MOV @RW1, R1, @RW1-d16, R1     | MOV @RW1, R2, @RW1-d16, R2     | MOV @RW1, R3, @RW1-d16, R3     | MOV @RW1, R4, @RW1-d16, R4     | MOV @RW1, R5, @RW1-d16, R5     | MOV @RW1, R6, @RW1-d16, R6     | MOV @RW1, R7, @RW1-d16, R7     | MOV @RW1, R8, @RW1-d16, R8     | MOV @RW1, R9, @RW1-d16, R9     | MOV @RW1, R10, @RW1-d16, R10     | MOV @RW1, R11, @RW1-d16, R11     | MOV @RW1, R12, @RW1-d16, R12     | MOV @RW1, R13, @RW1-d16, R13     | MOV @RW1, R14, @RW1-d16, R14     | MOV @RW1, R15, @RW1-d16, R15     |
| +A | MOV @RW2, R0, @RW2-d16, R0     | MOV @RW2, R1, @RW2-d16, R1     | MOV @RW2, R2, @RW2-d16, R2     | MOV @RW2, R3, @RW2-d16, R3     | MOV @RW2, R4, @RW2-d16, R4     | MOV @RW2, R5, @RW2-d16, R5     | MOV @RW2, R6, @RW2-d16, R6     | MOV @RW2, R7, @RW2-d16, R7     | MOV @RW2, R8, @RW2-d16, R8     | MOV @RW2, R9, @RW2-d16, R9     | MOV @RW2, R10, @RW2-d16, R10     | MOV @RW2, R11, @RW2-d16, R11     | MOV @RW2, R12, @RW2-d16, R12     | MOV @RW2, R13, @RW2-d16, R13     | MOV @RW2, R14, @RW2-d16, R14     | MOV @RW2, R15, @RW2-d16, R15     |
| +B | MOV @RW3, R0, @RW3-d16, R0     | MOV @RW3, R1, @RW3-d16, R1     | MOV @RW3, R2, @RW3-d16, R2     | MOV @RW3, R3, @RW3-d16, R3     | MOV @RW3, R4, @RW3-d16, R4     | MOV @RW3, R5, @RW3-d16, R5     | MOV @RW3, R6, @RW3-d16, R6     | MOV @RW3, R7, @RW3-d16, R7     | MOV @RW3, R8, @RW3-d16, R8     | MOV @RW3, R9, @RW3-d16, R9     | MOV @RW3, R10, @RW3-d16, R10     | MOV @RW3, R11, @RW3-d16, R11     | MOV @RW3, R12, @RW3-d16, R12     | MOV @RW3, R13, @RW3-d16, R13     | MOV @RW3, R14, @RW3-d16, R14     | MOV @RW3, R15, @RW3-d16, R15     |
| +C | MOV @RW0+, R0, @RW0+RW7, R0    | MOV @RW0+, R1, @RW0+RW7, R1    | MOV @RW0+, R2, @RW0+RW7, R2    | MOV @RW0+, R3, @RW0+RW7, R3    | MOV @RW0+, R4, @RW0+RW7, R4    | MOV @RW0+, R5, @RW0+RW7, R5    | MOV @RW0+, R6, @RW0+RW7, R6    | MOV @RW0+, R7, @RW0+RW7, R7    | MOV @RW0+, R8, @RW0+RW7, R8    | MOV @RW0+, R9, @RW0+RW7, R9    | MOV @RW0+, R10, @RW0+RW7, R10    | MOV @RW0+, R11, @RW0+RW7, R11    | MOV @RW0+, R12, @RW0+RW7, R12    | MOV @RW0+, R13, @RW0+RW7, R13    | MOV @RW0+, R14, @RW0+RW7, R14    | MOV @RW0+, R15, @RW0+RW7, R15    |
| +D | MOV @RW1+, R0, @RW1+RW7, R0    | MOV @RW1+, R1, @RW1+RW7, R1    | MOV @RW1+, R2, @RW1+RW7, R2    | MOV @RW1+, R3, @RW1+RW7, R3    | MOV @RW1+, R4, @RW1+RW7, R4    | MOV @RW1+, R5, @RW1+RW7, R5    | MOV @RW1+, R6, @RW1+RW7, R6    | MOV @RW1+, R7, @RW1+RW7, R7    | MOV @RW1+, R8, @RW1+RW7, R8    | MOV @RW1+, R9, @RW1+RW7, R9    | MOV @RW1+, R10, @RW1+RW7, R10    | MOV @RW1+, R11, @RW1+RW7, R11    | MOV @RW1+, R12, @RW1+RW7, R12    | MOV @RW1+, R13, @RW1+RW7, R13    | MOV @RW1+, R14, @RW1+RW7, R14    | MOV @RW1+, R15, @RW1+RW7, R15    |
| +E | MOV @RW2+, R0, @RW2+PC-d16, R0 | MOV @RW2+, R1, @RW2+PC-d16, R1 | MOV @RW2+, R2, @RW2+PC-d16, R2 | MOV @RW2+, R3, @RW2+PC-d16, R3 | MOV @RW2+, R4, @RW2+PC-d16, R4 | MOV @RW2+, R5, @RW2+PC-d16, R5 | MOV @RW2+, R6, @RW2+PC-d16, R6 | MOV @RW2+, R7, @RW2+PC-d16, R7 | MOV @RW2+, R8, @RW2+PC-d16, R8 | MOV @RW2+, R9, @RW2+PC-d16, R9 | MOV @RW2+, R10, @RW2+PC-d16, R10 | MOV @RW2+, R11, @RW2+PC-d16, R11 | MOV @RW2+, R12, @RW2+PC-d16, R12 | MOV @RW2+, R13, @RW2+PC-d16, R13 | MOV @RW2+, R14, @RW2+PC-d16, R14 | MOV @RW2+, R15, @RW2+PC-d16, R15 |
| +F | MOV @RW3+, R0, addr16, R0      | MOV @RW3+, R1, addr16, R1      | MOV @RW3+, R2, addr16, R2      | MOV @RW3+, R3, addr16, R3      | MOV @RW3+, R4, addr16, R4      | MOV @RW3+, R5, addr16, R5      | MOV @RW3+, R6, addr16, R6      | MOV @RW3+, R7, addr16, R7      | MOV @RW3+, R8, addr16, R8      | MOV @RW3+, R9, addr16, R9      | MOV @RW3+, R10, addr16, R10      | MOV @RW3+, R11, addr16, R11      | MOV @RW3+, R12, addr16, R12      | MOV @RW3+, R13, addr16, R13      | MOV @RW3+, R14, addr16, R14      | MOV @RW3+, R15, addr16, R15      |

**Table D.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<br>RW0, RW0,@RW0+d8,RW0  | MOVW<br>RW0, RW1,@RW0+d8,RW1  | MOVW<br>RW0, RW1,@RW0+d8,RW1  | MOVW<br>RW0, RW2,@RW0+d8,RW2  | MOVW<br>RW0, RW2,@RW0+d8,RW2  | MOVW<br>RW0, RW3,@RW0+d8,RW3  | MOVW<br>RW0, RW3,@RW0+d8,RW3  | MOVW<br>RW0, RW3,@RW0+d8,RW3  | MOVW<br>RW0, RW4,@RW0+d8,RW4  | MOVW<br>RW0, RW4,@RW0+d8,RW4  | MOVW<br>RW0, RW5,@RW0+d8,RW5  | MOVW<br>RW0, RW5,@RW0+d8,RW5  | MOVW<br>RW0, RW6,@RW0+d8,RW6  | MOVW<br>RW0, RW7,@RW0+d8,RW7  | MOVW<br>RW0, RW7,@RW0+d8,RW7  | MOVW<br>RW0, RW7,@RW0+d8,RW7  |
| +1 | MOVW<br>RW1, RW0,@RW1+d8,RW0  | MOVW<br>RW1, RW1,@RW1+d8,RW1  | MOVW<br>RW1, RW1,@RW1+d8,RW1  | MOVW<br>RW1, RW2,@RW1+d8,RW2  | MOVW<br>RW1, RW2,@RW1+d8,RW2  | MOVW<br>RW1, RW3,@RW1+d8,RW3  | MOVW<br>RW1, RW3,@RW1+d8,RW3  | MOVW<br>RW1, RW3,@RW1+d8,RW3  | MOVW<br>RW1, RW4,@RW1+d8,RW4  | MOVW<br>RW1, RW4,@RW1+d8,RW4  | MOVW<br>RW1, RW5,@RW1+d8,RW5  | MOVW<br>RW1, RW5,@RW1+d8,RW5  | MOVW<br>RW1, RW6,@RW1+d8,RW6  | MOVW<br>RW1, RW7,@RW1+d8,RW7  | MOVW<br>RW1, RW7,@RW1+d8,RW7  | MOVW<br>RW1, RW7,@RW1+d8,RW7  |
| +2 | MOVW<br>RW2, RW0,@RW2+d8,RW0  | MOVW<br>RW2, RW1,@RW2+d8,RW1  | MOVW<br>RW2, RW1,@RW2+d8,RW1  | MOVW<br>RW2, RW2,@RW2+d8,RW2  | MOVW<br>RW2, RW2,@RW2+d8,RW2  | MOVW<br>RW2, RW3,@RW2+d8,RW3  | MOVW<br>RW2, RW3,@RW2+d8,RW3  | MOVW<br>RW2, RW3,@RW2+d8,RW3  | MOVW<br>RW2, RW4,@RW2+d8,RW4  | MOVW<br>RW2, RW4,@RW2+d8,RW4  | MOVW<br>RW2, RW5,@RW2+d8,RW5  | MOVW<br>RW2, RW5,@RW2+d8,RW5  | MOVW<br>RW2, RW6,@RW2+d8,RW6  | MOVW<br>RW2, RW7,@RW2+d8,RW7  | MOVW<br>RW2, RW7,@RW2+d8,RW7  | MOVW<br>RW2, RW7,@RW2+d8,RW7  |
| +3 | MOVW<br>RW3, RW0,@RW3+d8,RW0  | MOVW<br>RW3, RW1,@RW3+d8,RW1  | MOVW<br>RW3, RW1,@RW3+d8,RW1  | MOVW<br>RW3, RW2,@RW3+d8,RW2  | MOVW<br>RW3, RW2,@RW3+d8,RW2  | MOVW<br>RW3, RW3,@RW3+d8,RW3  | MOVW<br>RW3, RW3,@RW3+d8,RW3  | MOVW<br>RW3, RW3,@RW3+d8,RW3  | MOVW<br>RW3, RW4,@RW3+d8,RW4  | MOVW<br>RW3, RW4,@RW3+d8,RW4  | MOVW<br>RW3, RW5,@RW3+d8,RW5  | MOVW<br>RW3, RW5,@RW3+d8,RW5  | MOVW<br>RW3, RW6,@RW3+d8,RW6  | MOVW<br>RW3, RW7,@RW3+d8,RW7  | MOVW<br>RW3, RW7,@RW3+d8,RW7  | MOVW<br>RW3, RW7,@RW3+d8,RW7  |
| +4 | MOVW<br>RW4, RW0,@RW4+d8,RW0  | MOVW<br>RW4, RW1,@RW4+d8,RW1  | MOVW<br>RW4, RW1,@RW4+d8,RW1  | MOVW<br>RW4, RW2,@RW4+d8,RW2  | MOVW<br>RW4, RW2,@RW4+d8,RW2  | MOVW<br>RW4, RW3,@RW4+d8,RW3  | MOVW<br>RW4, RW3,@RW4+d8,RW3  | MOVW<br>RW4, RW3,@RW4+d8,RW3  | MOVW<br>RW4, RW4,@RW4+d8,RW4  | MOVW<br>RW4, RW4,@RW4+d8,RW4  | MOVW<br>RW4, RW5,@RW4+d8,RW5  | MOVW<br>RW4, RW5,@RW4+d8,RW5  | MOVW<br>RW4, RW6,@RW4+d8,RW6  | MOVW<br>RW4, RW7,@RW4+d8,RW7  | MOVW<br>RW4, RW7,@RW4+d8,RW7  | MOVW<br>RW4, RW7,@RW4+d8,RW7  |
| +5 | MOVW<br>RW5, RW0,@RW5+d8,RW0  | MOVW<br>RW5, RW1,@RW5+d8,RW1  | MOVW<br>RW5, RW1,@RW5+d8,RW1  | MOVW<br>RW5, RW2,@RW5+d8,RW2  | MOVW<br>RW5, RW2,@RW5+d8,RW2  | MOVW<br>RW5, RW3,@RW5+d8,RW3  | MOVW<br>RW5, RW3,@RW5+d8,RW3  | MOVW<br>RW5, RW3,@RW5+d8,RW3  | MOVW<br>RW5, RW4,@RW5+d8,RW4  | MOVW<br>RW5, RW4,@RW5+d8,RW4  | MOVW<br>RW5, RW5,@RW5+d8,RW5  | MOVW<br>RW5, RW5,@RW5+d8,RW5  | MOVW<br>RW5, RW6,@RW5+d8,RW6  | MOVW<br>RW5, RW7,@RW5+d8,RW7  | MOVW<br>RW5, RW7,@RW5+d8,RW7  | MOVW<br>RW5, RW7,@RW5+d8,RW7  |
| +6 | MOVW<br>RW6, RW0,@RW6+d8,RW0  | MOVW<br>RW6, RW1,@RW6+d8,RW1  | MOVW<br>RW6, RW1,@RW6+d8,RW1  | MOVW<br>RW6, RW2,@RW6+d8,RW2  | MOVW<br>RW6, RW2,@RW6+d8,RW2  | MOVW<br>RW6, RW3,@RW6+d8,RW3  | MOVW<br>RW6, RW3,@RW6+d8,RW3  | MOVW<br>RW6, RW3,@RW6+d8,RW3  | MOVW<br>RW6, RW4,@RW6+d8,RW4  | MOVW<br>RW6, RW4,@RW6+d8,RW4  | MOVW<br>RW6, RW5,@RW6+d8,RW5  | MOVW<br>RW6, RW5,@RW6+d8,RW5  | MOVW<br>RW6, RW6,@RW6+d8,RW6  | MOVW<br>RW6, RW7,@RW6+d8,RW7  | MOVW<br>RW6, RW7,@RW6+d8,RW7  | MOVW<br>RW6, RW7,@RW6+d8,RW7  |
| +7 | MOVW<br>RW7, RW0,@RW7+d8,RW0  | MOVW<br>RW7, RW1,@RW7+d8,RW1  | MOVW<br>RW7, RW1,@RW7+d8,RW1  | MOVW<br>RW7, RW2,@RW7+d8,RW2  | MOVW<br>RW7, RW2,@RW7+d8,RW2  | MOVW<br>RW7, RW3,@RW7+d8,RW3  | MOVW<br>RW7, RW3,@RW7+d8,RW3  | MOVW<br>RW7, RW3,@RW7+d8,RW3  | MOVW<br>RW7, RW4,@RW7+d8,RW4  | MOVW<br>RW7, RW4,@RW7+d8,RW4  | MOVW<br>RW7, RW5,@RW7+d8,RW5  | MOVW<br>RW7, RW5,@RW7+d8,RW5  | MOVW<br>RW7, RW6,@RW7+d8,RW6  | MOVW<br>RW7, RW7,@RW7+d8,RW7  | MOVW<br>RW7, RW7,@RW7+d8,RW7  | MOVW<br>RW7, RW7,@RW7+d8,RW7  |
| +8 | MOVW<br>@RW0, RW0,-d16,RW0    | MOVW<br>@RW0, RW1,-d16,RW1    | MOVW<br>@RW0, RW1,-d16,RW1    | MOVW<br>@RW0, RW2,-d16,RW2    | MOVW<br>@RW0, RW2,-d16,RW2    | MOVW<br>@RW0, RW3,-d16,RW3    | MOVW<br>@RW0, RW3,-d16,RW3    | MOVW<br>@RW0, RW3,-d16,RW3    | MOVW<br>@RW0, RW4,-d16,RW4    | MOVW<br>@RW0, RW4,-d16,RW4    | MOVW<br>@RW0, RW5,-d16,RW5    | MOVW<br>@RW0, RW5,-d16,RW5    | MOVW<br>@RW0, RW6,-d16,RW6    | MOVW<br>@RW0, RW7,-d16,RW7    | MOVW<br>@RW0, RW7,-d16,RW7    | MOVW<br>@RW0, RW7,-d16,RW7    |
| +9 | MOVW<br>@RW1, RW0,-d16,RW0    | MOVW<br>@RW1, RW1,-d16,RW1    | MOVW<br>@RW1, RW1,-d16,RW1    | MOVW<br>@RW1, RW2,-d16,RW2    | MOVW<br>@RW1, RW2,-d16,RW2    | MOVW<br>@RW1, RW3,-d16,RW3    | MOVW<br>@RW1, RW3,-d16,RW3    | MOVW<br>@RW1, RW3,-d16,RW3    | MOVW<br>@RW1, RW4,-d16,RW4    | MOVW<br>@RW1, RW4,-d16,RW4    | MOVW<br>@RW1, RW5,-d16,RW5    | MOVW<br>@RW1, RW5,-d16,RW5    | MOVW<br>@RW1, RW6,-d16,RW6    | MOVW<br>@RW1, RW7,-d16,RW7    | MOVW<br>@RW1, RW7,-d16,RW7    | MOVW<br>@RW1, RW7,-d16,RW7    |
| +A | MOVW<br>@RW2, RW0,-d16,RW0    | MOVW<br>@RW2, RW1,-d16,RW1    | MOVW<br>@RW2, RW1,-d16,RW1    | MOVW<br>@RW2, RW2,-d16,RW2    | MOVW<br>@RW2, RW2,-d16,RW2    | MOVW<br>@RW2, RW3,-d16,RW3    | MOVW<br>@RW2, RW3,-d16,RW3    | MOVW<br>@RW2, RW3,-d16,RW3    | MOVW<br>@RW2, RW4,-d16,RW4    | MOVW<br>@RW2, RW4,-d16,RW4    | MOVW<br>@RW2, RW5,-d16,RW5    | MOVW<br>@RW2, RW5,-d16,RW5    | MOVW<br>@RW2, RW6,-d16,RW6    | MOVW<br>@RW2, RW7,-d16,RW7    | MOVW<br>@RW2, RW7,-d16,RW7    | MOVW<br>@RW2, RW7,-d16,RW7    |
| +B | MOVW<br>@RW3, RW0,-d16,RW0    | MOVW<br>@RW3, RW1,-d16,RW1    | MOVW<br>@RW3, RW1,-d16,RW1    | MOVW<br>@RW3, RW2,-d16,RW2    | MOVW<br>@RW3, RW2,-d16,RW2    | MOVW<br>@RW3, RW3,-d16,RW3    | MOVW<br>@RW3, RW3,-d16,RW3    | MOVW<br>@RW3, RW3,-d16,RW3    | MOVW<br>@RW3, RW4,-d16,RW4    | MOVW<br>@RW3, RW4,-d16,RW4    | MOVW<br>@RW3, RW5,-d16,RW5    | MOVW<br>@RW3, RW5,-d16,RW5    | MOVW<br>@RW3, RW6,-d16,RW6    | MOVW<br>@RW3, RW7,-d16,RW7    | MOVW<br>@RW3, RW7,-d16,RW7    | MOVW<br>@RW3, RW7,-d16,RW7    |
| +C | MOVW<br>@RW0,-RW0,+RW7,RW0    | MOVW<br>@RW0,-RW1,+RW7,RW1    | MOVW<br>@RW0,-RW1,+RW7,RW1    | MOVW<br>@RW0,-RW2,+RW7,RW2    | MOVW<br>@RW0,-RW2,+RW7,RW2    | MOVW<br>@RW0,-RW3,+RW7,RW3    | MOVW<br>@RW0,-RW3,+RW7,RW3    | MOVW<br>@RW0,-RW3,+RW7,RW3    | MOVW<br>@RW0,-RW4,+RW7,RW4    | MOVW<br>@RW0,-RW4,+RW7,RW4    | MOVW<br>@RW0,-RW5,+RW7,RW5    | MOVW<br>@RW0,-RW5,+RW7,RW5    | MOVW<br>@RW0,-RW6,+RW7,RW6    | MOVW<br>@RW0,-RW7,+RW7,RW7    | MOVW<br>@RW0,-RW7,+RW7,RW7    | MOVW<br>@RW0,-RW7,+RW7,RW7    |
| +D | MOVW<br>@RW1,-RW0,+RW7,RW0    | MOVW<br>@RW1,-RW1,+RW7,RW1    | MOVW<br>@RW1,-RW1,+RW7,RW1    | MOVW<br>@RW1,-RW2,+RW7,RW2    | MOVW<br>@RW1,-RW2,+RW7,RW2    | MOVW<br>@RW1,-RW3,+RW7,RW3    | MOVW<br>@RW1,-RW3,+RW7,RW3    | MOVW<br>@RW1,-RW3,+RW7,RW3    | MOVW<br>@RW1,-RW4,+RW7,RW4    | MOVW<br>@RW1,-RW4,+RW7,RW4    | MOVW<br>@RW1,-RW5,+RW7,RW5    | MOVW<br>@RW1,-RW5,+RW7,RW5    | MOVW<br>@RW1,-RW6,+RW7,RW6    | MOVW<br>@RW1,-RW7,+RW7,RW7    | MOVW<br>@RW1,-RW7,+RW7,RW7    | MOVW<br>@RW1,-RW7,+RW7,RW7    |
| +E | MOVW<br>@RW2,-RW0,+d16,RW0    | MOVW<br>@RW2,-RW1,+d16,RW1    | MOVW<br>@RW2,-RW1,+d16,RW1    | MOVW<br>@RW2,-RW2,+d16,RW2    | MOVW<br>@RW2,-RW2,+d16,RW2    | MOVW<br>@RW2,-RW3,+d16,RW3    | MOVW<br>@RW2,-RW3,+d16,RW3    | MOVW<br>@RW2,-RW3,+d16,RW3    | MOVW<br>@RW2,-RW4,+d16,RW4    | MOVW<br>@RW2,-RW4,+d16,RW4    | MOVW<br>@RW2,-RW5,+d16,RW5    | MOVW<br>@RW2,-RW5,+d16,RW5    | MOVW<br>@RW2,-RW6,+d16,RW6    | MOVW<br>@RW2,-RW7,+d16,RW7    | MOVW<br>@RW2,-RW7,+d16,RW7    | MOVW<br>@RW2,-RW7,+d16,RW7    |
| +F | MOVW<br>@RW3,-RW0,+addr16,RW0 | MOVW<br>@RW3,-RW1,+addr16,RW1 | MOVW<br>@RW3,-RW1,+addr16,RW1 | MOVW<br>@RW3,-RW2,+addr16,RW2 | MOVW<br>@RW3,-RW2,+addr16,RW2 | MOVW<br>@RW3,-RW3,+addr16,RW3 | MOVW<br>@RW3,-RW3,+addr16,RW3 | MOVW<br>@RW3,-RW3,+addr16,RW3 | MOVW<br>@RW3,-RW4,+addr16,RW4 | MOVW<br>@RW3,-RW4,+addr16,RW4 | MOVW<br>@RW3,-RW5,+addr16,RW5 | MOVW<br>@RW3,-RW5,+addr16,RW5 | MOVW<br>@RW3,-RW6,+addr16,RW6 | MOVW<br>@RW3,-RW7,+addr16,RW7 | MOVW<br>@RW3,-RW7,+addr16,RW7 | MOVW<br>@RW3,-RW7,+addr16,RW7 |

**Table D.9-20 XCH Ri, ea Instruction (First Byte = 7E<sub>H</sub>)**

[illegible]

**Table D.9-21 XCHW RWi, ea Instruction (First Byte = 7F<sub>H</sub>)**

[illegible]

# Index

## Numerics

### 16-bit

|  |     |
|--|-----|
| 16-bit Reload Register (TMRLR) .....   | 302 |
| 16-bit Timer Register (TMR).....   | 301 |
| 16-bit Timer Register<br>(TMR)/16-bit Reload Register (TMRLR)<br>.....                                 | 301 |
| Block Diagram of Pin Related to 16-bit Input/Output Timer<br>.....                                     | 222 |
| Block Diagram of Pin Related to 16-bit Reload Timer<br>.....   | 295 |
| Block Diagram of the 16-bit Reload Timer .....   | 294 |
| Functions of 16-bit Input/Output Timer .....   | 218 |
| Interrupt of 16-bit Input/Output Timer .....   | 235 |
| Interrupt of 16-bit Input/Output Timer,DMA Transfer,<br>and EI <sup>2</sup> OS.....                    | 236 |
| Interrupt of 16-bit Reload Timer .....   | 303 |
| Interrupt of 16-bit Reload Timer,DMA Transfer,and<br>EI <sup>2</sup> OS .....                          | 303 |
| Operation and Timing of 16-bit Input/Output Timer<br>.....   | 237 |
| Operation Modes of the 16-bit Reload Timer .....   | 292 |
| Pin Related to 16-bit Input/Output Timer .....   | 221 |
| Pin Related to 16-bit Reload Timer .....   | 294 |
| Pin States in External Bus 16-bit Data Bus Mode and<br>Multiplex 16-bit External Bus Mode .....        | 144 |
| Pin States in External Bus 16-bit Data Bus Mode and<br>Non-multiplex 16-bit External Bus Mode<br>..... | 146 |
| Program Example of 16-bit Reload Timer .....   | 312 |
| Settings of the 16-bit Reload Timer.....   | 304 |

### 8/10-bit

|  |     |
|--|-----|
| Block Diagram of 8/10-bit A/D Converter .....                                    | 357 |
| Block Diagram of Pin Related to 8/10-bit A/D Converter<br>.....                  | 358 |
| Features of the 8/10-bit A/D Converter .....                                     | 356 |
| Interrupt of 8/10-bit A/D Converter.....   | 367 |
| Interrupt of 8/10-bit A/D Converter,DMA Transfer,and<br>EI <sup>2</sup> OS ..... | 367 |
| List of Registers for 8/10-bit A/D Converter .....                               | 359 |
| Pin Related to 8/10-bit A/D Converter .....                                      | 358 |
| Program Example of 8/10-bit A/D Converter .....                                  | 379 |

### 8/16-bit

|   |     |
|---|-----|
| Block Diagram of 8/16-bit Up/Down Counter/Timer<br>.....                      | 261 |
| Block Diagram of Pin Related to 8/16-bit PPG Timer<br>.....                   | 321 |
| Block Diagram of Pin Related to 8/16-bit Up/Down<br>Counter/Timer .....       | 263 |
| Block Diagram of the 8/16-bit PPG Timer .....                                 | 319 |
| Functions of 8/16-bit PPG Timer.....  | 318 |
| Interrupt of 8/16-bit PPG Timer .....   | 331 |
| Interrupt of 8/16-bit PPG Timer,DMA Transfer, and EI <sup>2</sup> OS<br>..... | 332 |
| Interrupt of 8/16-bit Up/Down Counter/Timer .....                             | 275 |

|  |     |
|--|-----|
| Interrupt of 8/16-bit Up/Down Counter/Timer,DMA<br>Transfer,and EI <sup>2</sup> OS ..... | 276 |
| Interrupts of the 8/16-bit PPG Timer.....  | 336 |
| List of 8/16-bit PPG Timer Registers .....   | 322 |
| List of 8/16-bit Up/Down Counter/Timer Registers<br>.....                                | 264 |
| Major Functions of 8/16-bit Up/Down Counter/Timer<br>.....                               | 260 |
| Outline of 8/16-bit PPG Timer Operation.....   | 333 |
| Pin Related to 8/16-bit PPG Timer .....  | 320 |
| Pin Related to 8/16-bit Up/Down Counter/Timer .....                                      | 262 |
| Program Example of 8/16-bit PPG Timer.....   | 338 |
| Program Example of 8/16-bit Up/Down Counter/Timer<br>.....                               | 285 |

### 8-bit

|  |     |
|--|-----|
| Pin States in External Bus 8-bit Data Bus Mode and<br>Multiplex 8-bit External Bus Mode.....         | 145 |
| Pin States in External Bus 8-bit Data Bus Mode and<br>Non-multiplex 8-bit External Bus Mode<br>..... | 147 |

## Index

|   |          |
|---|----------|
| <b>A</b>  |          |
| <b>A</b>  |          |
| Accumulator (A).....  | 31       |
| <b>A/D Converter</b>  |          |
| Block Diagram of 8/10-bit A/D Converter .....   | 357      |
| Block Diagram of Pin Related to 8/10-bit A/D Converter.....   | 358      |
| Features of the 8/10-bit A/D Converter .....  | 356      |
| Interrupt of 8/10-bit A/D Converter .....   | 367      |
| Interrupt of 8/10-bit A/D Converter,DMA Transfer,and EI <sup>2</sup> OS .....                       | 367      |
| Interrupt of A/D Converter .....  | 367      |
| List of Registers for 8/10-bit A/D Converter.....   | 359      |
| Pin Related to 8/10-bit A/D Converter .....   | 358      |
| Program Example of 8/10-bit A/D Converter.....  | 379      |
| <b>Access Mode</b>  |          |
| Access Mode .....   | 154      |
| <b>Accumulator</b>  |          |
| Accumulator (A).....  | 31       |
| <b>Acknowledge</b>  |          |
| Acknowledge .....   | 576      |
| <b>ADB</b>  |          |
| Bank Select Prefix (PCB,DTB,ADB,SPB) .....  | 40       |
| <b>ADCR</b>   |          |
| Data Registers (ADCR2 and ADCR1) .....  | 366      |
| <b>ADCS</b>   |          |
| Control Status Register 1 (ADCS1) .....   | 360      |
| Control Status Register 2 (ADCS2) .....   | 363      |
| <b>Address</b>  |          |
| Address Generation Type.....  | 25       |
| Address Register (IADR).....  | 570      |
| Setting Detection Address .....   | 465      |
| <b>Address Match</b>  |          |
| Block Diagram of Address Match Detection Function .....   | 459      |
| List of Registers and Initial Values of Address Match Detection Function .....                      | 460      |
| Operation of Address Match Detection Function .....   | 465      |
| Operation of Address Match Detection Function at Storing Patch Program in E <sup>2</sup> PROM ..... | 469      |
| Overview of Address Match Detection Function .....  | 458      |
| Program Example for Address Match Detection Function .....  | 471      |
| <b>Address Pointer</b>  |          |
| I/O Register Address Pointer (IOA).....   | 71       |
| <b>Addressing</b>   |          |
| Addressing .....  | 575, 596 |
| Addressing Type by Bank .....   | 26       |
| Direct Addressing .....   | 598      |
| Indirect Addressing .....   | 604      |
| <b>ADER</b>   |          |
| Analog Input Enable Register (ADER) .....   | 184      |
| <b>Analog</b>   |          |
| Handling of Analog Input Pins .....   | 378      |
| <b>Analog Input</b>   |          |
| Analog Input Enable Register (ADER) .....   | 184      |
| <b>Arbitration</b>  |          |
| Arbitration .....   | 575      |
| <b>ARSR</b>   |          |
| Automatic Ready Function Selection Register (ARSR) .....  | 162      |
| <b>Asynchronous Mode</b>  |          |
| Operation in Asynchronous Mode (Operation Modes 0 and 1) .....                                      | 427      |
| <b>Automatic</b>  |          |
| Automatic Ready Function Selection Register (ARSR) .....  | 162      |
| <b>Automatic Algorithm</b>  |          |
| End Timing of the Automatic Algorithm.....  | 482      |
| <b>B</b>  |          |
| <b>Bank</b>   |          |
| Addressing Type by Bank.....  | 26       |
| <b>Bank Select</b>  |          |
| Bank Select Prefix (PCB,DTB,ADB,SPB).....   | 40       |
| <b>BAP</b>  |          |
| Buffer Address Pointer (BAP) .....  | 73, 83   |
| <b>Block Diagram</b>  |          |
| Block Diagram.....  | 160, 219 |
| Block Diagram of 8/10-bit A/D Converter.....  | 357      |
| Block Diagram of 8/16-bit Up/Down Counter/Timer .....   | 261      |
| Block Diagram of Address Match Detection Function .....   | 459      |
| Block Diagram of Clock Generator.....   | 110      |
| Block Diagram of Delay Interrupt Generation Module .....  | 92       |
| Block Diagram of DTP/external Interrupt Unit .....  | 342      |
| Block Diagram of Expanded I/O Serial Interface.....   | 387      |
| Block Diagram of External-Reset Pin .....   | 100      |
| Block Diagram of Free-run Timer .....   | 220      |
| Block Diagram of Input Capture .....  | 221      |
| Block Diagram of Low-power Consumption Control Circuit .....  | 126      |
| Block Diagram of MB90480/485 Series.....  | 6        |
| Block Diagram of $\mu$ PG Timer.....  | 550      |
| Block Diagram of Output Compare.....  | 220      |
| Block Diagram of Pin Related to 16-bit Input/Output Timer .....                                     | 222      |
| Block Diagram of Pin Related to 16-bit Reload Timer .....   | 295      |
| Block Diagram of Pin Related to 8/10-bit A/D Converter .....  | 358      |
| Block Diagram of Pin Related to 8/16-bit PPG Timer .....  | 321      |
| Block Diagram of Pin Related to 8/16-bit Up/Down Counter/Timer .....                                | 263      |
| Block Diagram of Pin Related to Chip Select Facility .....  | 448      |
| Block Diagram of Pin Related to DTP/external Interrupt .....  | 343      |
| Block Diagram of Pin Related to Expanded I/O Serial Interface.....                                  | 388      |
| Block Diagram of Pin Related to I <sup>2</sup> C Interface .....                                    | 558      |
| Block Diagram of Pin Related to $\mu$ PG Timer .....  | 551      |

|  |          |
|--|----------|
| Block Diagram of Pin Related to PWC Timer .....                      | 518      |
| Block Diagram of Pin Related to UART .....                           | 410      |
| Block Diagram of PWC Timer.....                                      | 517      |
| Block Diagram of the 16-bit Reload Timer .....                       | 294      |
| Block Diagram of the 8/16-bit PPG Timer.....                         | 319      |
| Block Diagram of the Chip Selection Facility .....                   | 447      |
| Block Diagram of the I <sup>2</sup> C Interface.....                 | 557      |
| Block Diagram of the ROM Mirror Function Selection<br>Module .....   | 474      |
| Block Diagram of Time-base Timer .....                               | 188      |
| Block Diagram of Watch Timer .....                                   | 211      |
| Block Diagram of Watchdog Timer .....                                | 203      |
| UART Block Diagram .....   | 409      |
| <b>Buffer</b>  |          |
| Buffer Address Pointer (BAP) .....                                   | 73       |
| <b>Buffer Address</b>  |          |
| Buffer Address Pointer (BAP) .....                                   | 83       |
| <b>Bus</b>   |          |
| Bus Error .....  | 576      |
| Bus Status Register (IBSR) .....                                     | 560      |
| Notes on Using the Bus Control Register (IBCR) .....                 | 567      |
| <b>Bus Control</b>   |          |
| Bus Control Register (IBCR).....                                     | 562      |
| Bus Control Signal Selection Register (EPCR) .....                   | 165      |
| <b>Bus Mode</b>  |          |
| Bus Mode Setting Bits (M1,M0) .....                                  | 157      |
| Bus Modes .....  | 154      |
| <b>C</b>   |          |
| <b>Calculating</b>   |          |
| Calculating the Execution Cycle Count .....                          | 613      |
| <b>CALR</b>  |          |
| Chip Selection Active Level Register (CALR).....                     | 453      |
| <b>CARx</b>  |          |
| Chip Selection Area Register (CARx) .....                            | 451      |
| <b>CCR</b>   |          |
| Condition Code Register (CCR) .....                                  | 33       |
| <b>CCRH</b>  |          |
| Counter Control Register (ch.0) Upper (CCRH0) .....                  | 265      |
| Counter Control Register (ch.1) Upper (CCRH1) .....                  | 267      |
| <b>CCRL</b>  |          |
| Counter Control Register (ch.0/ch.1) Lower<br>(CCRL0/CCRL1) .....    | 269      |
| <b>CDCR</b>  |          |
| Communication Prescaler Control Register (CDCR)<br>.....             | 419      |
| <b>Chip</b>  |          |
| Chip/Sector Erase Operation .....                                    | 489      |
| Write/Chip Sector Erase Operation .....                              | 492      |
| Write/Chip Sector Erase Operations .....                             | 491      |
| <b>Chip Erase</b>  |          |
| Erasing All Data in the Flash Memory (Chip Erase)<br>.....           | 498      |
| <b>Chip Select</b>   |          |
| Block Diagram of Pin Related to Chip Select Facility<br>.....        | 448      |
| <b>Chip Selection</b>  |          |
| Block Diagram of the Chip Selection Facility .....                   | 447      |
| Chip Selection Active Level Register (CALR) .....                    | 453      |
| Chip Selection Area MASK Register (CMRx) .....                       | 450      |
| Chip Selection Area Register (CARx) .....                            | 451      |
| Chip Selection Control Register (CSCR) .....                         | 452      |
| Example of Using the Chip Selection Facility .....                   | 454      |
| List of Registers Used for the Chip Selection Facility<br>.....      | 449      |
| Notes on Using the Chip Selection Facility.....                      | 455      |
| Overview of the Chip Selection Facility .....                        | 446      |
| Pin Related to Chip Selection Facility .....                         | 447      |
| <b>Circuit Type</b>  |          |
| I/O Circuit Type.....  | 18       |
| <b>CKSCR</b>   |          |
| Configuration of Clock Selection Register (CKSCR)<br>.....           | 112      |
| <b>Clear</b>   |          |
| Count Clear/Gate Function .....                                      | 283      |
| <b>Clearing</b>  |          |
| Clearing the Counter .....   | 283      |
| Clearing the Timer.....  | 537      |
| <b>CLK</b>   |          |
| Operation in CLK Synchronous Mode (Operation Mode 2)<br>.....        | 430      |
| <b>Clock</b>   |          |
| Block Diagram of Clock Generator.....                                | 110      |
| Clock Control Register (ICCR).....                                   | 568      |
| Clock Source for Watchdog Timer Specifying Function<br>.....         | 215      |
| Clock Supply Map.....  | 109      |
| Clock Supplying Function .....                                       | 187, 194 |
| Configuration of Clock Selection Register (CKSCR)<br>.....           | 112      |
| Count Clock Selection.....   | 533      |
| Internal Clock Mode .....  | 292      |
| Oscillation Clock Frequency and Serial Clock Input<br>Frequency..... | 508      |
| UART Clock Selection.....  | 424      |
| <b>Clock Mode</b>  |          |
| Change of Clock Mode.....  | 117      |
| Switching the Clock Mode .....                                       | 149      |
| Clock Modes.....   | 125      |
| <b>Clock</b>   |          |
| Overview of Clocks .....   | 108      |
| <b>CMR</b>   |          |
| Common Register Bank Prefix (CMR) .....                              | 41       |
| <b>CMRx</b>  |          |
| Chip Selection Area MASK Register (CMRx) .....                       | 450      |
| <b>Code</b>  |          |
| Continuous Prefix Codes.....   | 42       |
| <b>Command Sequence</b>  |          |
| Command Sequence Table .....   | 486      |
| <b>Common</b>  |          |
| Common Register Bank Prefix (CMR) .....                              | 41       |
| <b>Communication Prescaler</b>                                       |          |
| Communication Prescaler Control Register (CDCR)<br>.....             | 419      |



## Index

|  |          |
|--|----------|
| Communication Prescaler Control Register0/1<br>(SDCR0/SDCR1).....                                      | 395      |
| <b>Compare</b>   |          |
| Compare Clear Register (CPCLR) .....   | 224      |
| Compare Function.....  | 281      |
| Reload/Compare Register (ch.0/ch.1) (RCR0/RCR1)<br>.....   | 274      |
| Selection of Reload and Compare Functions.....   | 280      |
| Up/Down Count at any Width in Reload/Compare Function<br>.....   | 281      |
| <b>Condition Code Register</b>   |          |
| Condition Code Register (CCR) .....  | 33       |
| <b>Continuous Mode</b>   |          |
| Example of $\mu$ DMAC Start in Continuous Mode .....   | 372      |
| <b>Control</b>   |          |
| Control Status Register 1 (ADCS1) .....  | 360      |
| Control Status Register 2 (ADCS2) .....  | 363      |
| <b>Conversion Data</b>   |          |
| Caution When Using the Conversion Data Protection<br>Function .....                                    | 376      |
| Conversion Data Protection Function.....   | 376      |
| Operation Flow of Conversion Data Protection Function<br>(when $\mu$ DMAC is Used) .....               | 377      |
| <b>Correspondence</b>  |          |
| Correspondence to DMA Transfer and EI <sup>2</sup> OS Function<br>.....                                | 396      |
| <b>Count</b>   |          |
| Count Clear/Gate Function .....  | 283      |
| Count Direction Flag, Count Direction Reversal Flag<br>.....   | 284      |
| State Transitions During Count Operation.....  | 305      |
| <b>Count Clock</b>   |          |
| Count Clock and Maximum Interval.....  | 538      |
| Count Clock Selection.....   | 533      |
| Selection of Count Clock .....   | 335      |
| <b>Count Mode</b>  |          |
| Selection of Count Mode .....  | 277      |
| <b>Counter</b>   |          |
| Block Diagram of 8/16-bit Up/Down Counter/Timer<br>.....   | 261      |
| Block Diagram of Pin Related to 8/16-bit Up/Down<br>Counter/Timer.....                                 | 263      |
| Clearing the Counter.....  | 283      |
| Counter Control Register (ch.0) Upper (CCR0) .....   | 265      |
| Counter Control Register (ch.0/ch.1) Lower<br>(CCRL0/CCRL1).....                                       | 269      |
| Counter Control Register (ch.1) Upper (CCR1) .....   | 267      |
| Counter Operation Modes .....  | 293      |
| Counter Status Register 0/1 (CSR0/CSR1).....   | 271      |
| Data Counter (DCT) .....   | 71       |
| Interrupt of 8/16-bit Up/Down Counter/Timer .....  | 275      |
| Interrupt of PPG Counter Underflow .....   | 331      |
| List of 8/16-bit Up/Down Counter/Timer Registers<br>.....  | 264      |
| Major Functions of 8/16-bit Up/Down Counter/Timer<br>.....   | 260      |
| Pin Related to 8/16-bit Up/Down Counter/Timer .....  | 262      |
| Program Example of 8/16-bit Up/Down Counter/Timer<br>.....   | 285      |
| Interrupt of 8/16-bit Up/Down Counter/Timer,DMA<br>Transfer, and EI <sup>2</sup> OS .....              | 276      |
| <b>Counter Operation</b>   |          |
| Measurement Mode and Counter Operation .....   | 541      |
| <b>CPU</b>   |          |
| Connection Between CPUs in Master/Slave Communication<br>.....   | 434      |
| CPU Intermittent Operation Mode .....  | 125, 131 |
| CPU Operation Mode and Current Consumption.....  | 124      |
| Overview of the CPU Specifications .....   | 24       |
| <b>CPUs</b>  |          |
| Connection Between CPUs in Two-way communication<br>.....  | 432      |
| <b>CSCR</b>  |          |
| Chip Selection Control Register (CSCR).....  | 452      |
| <b>CSR</b>   |          |
| Counter Status Register 0/1 (CSR0/CSR1) .....  | 271      |
| <b>D</b>   |          |
| <b>Data</b>  |          |
| Data Counter (DCT).....  | 71       |
| Data Register (IDAR).....  | 571      |
| Data Registers (ADCR2 and ADCR1) .....   | 366      |
| <b>Data Bus</b>  |          |
| Pin States in External Bus 16-bit Data Bus Mode and<br>Multiplex 16-bit External Bus Mode .....        | 144      |
| Pin States in External Bus 16-bit Data Bus Mode and<br>Non-multiplex 16-bit External Bus Mode<br>..... | 146      |
| Pin States in External Bus 8-bit Data Bus Mode and<br>Non-multiplex 8-bit External Bus Mode<br>.....   | 147      |
| <b>Data Counter</b>  |          |
| Data Counter (DCT).....  | 81       |
| <b>Data Polling</b>  |          |
| State Transitions of the Data Polling Flag (DQ7) .....   | 489      |
| <b>DCT</b>   |          |
| Data Counter (DCT).....  | 71, 81   |
| <b>DDR</b>   |          |
| Port Direction Registers (DDR0 to DDRA) .....  | 181      |
| <b>Delay</b>   |          |
| Block Diagram of Delay Interrupt Generation Module<br>.....  | 92       |
| List of Registers in Delay Interrupt Generation Module<br>.....  | 92       |
| Notes on Using Delay Interrupt Generation Module<br>(Delay Interrupt Request Latch) .....              | 93       |
| Operation of Delay Interrupt Generation Module .....   | 93       |
| <b>Description</b>   |          |
| Description of Instruction Presentation Items and Symbols<br>.....                                     | 616      |
| <b>Detection</b>   |          |
| Setting Detection Address .....  | 465      |
| <b>Different Mode</b>  |          |
| Setting Bits of Different Modes (S1,S0).....   | 156      |
| <b>Direct Addressing</b>   |          |
| Direct Addressing.....   | 598      |

|  |  |
|--|--|
| <b>Direct Page</b>   |  |
| Direct Page Register (DPR)<Initial Value: 01 <sub>H</sub> >.....                                 | 38   |
| <b>Divide Ratio</b>  |  |
| Divide Ratio Control Register (DIVR0 to DIVR2)   |  |
| .....  | 526  |
| <b>DIVR</b>  |  |
| Divide Ratio Control Register (DIVR0 to DIVR2)   |  |
| .....  | 526  |
| <b>DMA</b>   |  |
| Correspondence to DMA Transfer and EI <sup>2</sup> OS Function                                   |  |
| .....  | 236, 276, 303, 332, 347, 367, 396, 422, 528, 573 |
| DMA Control Status Register (DMACS).....   | 72   |
| DMA Descriptor Configuration.....  | 69   |
| DTP/external Interrupt,DMA Transfer,and EI <sup>2</sup> OS                                       |  |
| .....  | 347  |
| Interrupt of 16-bit Input/Output Timer,DMA Transfer, and EI <sup>2</sup> OS                      |  |
| .....  | 236  |
| Interrupt of 16-bit Reload Timer,DMA Transfer,and EI <sup>2</sup> OS                             |  |
| .....  | 303  |
| Interrupt of 8/10-bit A/D Converter,DMA Transfer,and EI <sup>2</sup> OS                          |  |
| .....  | 367  |
| Interrupt of 8/16-bit PPG Timer,DMA Transfer, and EI <sup>2</sup> OS                             |  |
| .....  | 332  |
| Interrupt of 8/16-bit Up/Down Counter/Timer,DMA Transfer,and EI <sup>2</sup> OS                  |  |
| .....  | 276  |
| Interrupt of Expanded I/O Serial Interface,DMA Transfer, and EI <sup>2</sup> OS.....             | 396  |
| Interrupt of I <sup>2</sup> C Interface,DMA Transfer, and EI <sup>2</sup> OS                     |  |
| .....  | 573  |
| Interrupt of PWC Timer, DMA Transfer, and EI <sup>2</sup> OS                                     |  |
| .....  | 528  |
| Interrupt of UART,DMA Transfer,and EI <sup>2</sup> OS  |  |
| .....  | 422  |
| <b>DMACS</b>   |  |
| DMA Control Status Register (DMACS).....   | 72   |
| <b>DPR</b>   |  |
| Direct Page Register (DPR)<Initial Value: 01 <sub>H</sub> >.....                                 | 38   |
| <b>DQ</b>  |  |
| State Transitions of Sector Erase Timer Flag (DQ3)   |  |
| .....  | 493  |
| State Transitions of the Data Polling Flag (DQ7).....  | 489  |
| State Transitions of the Timing Limit Excess Flag (DQ5)  |  |
| .....  | 492  |
| State Transitions of the Toggle Bit Flag (DQ6).....  | 491  |
| <b>DTB</b>   |  |
| Bank Select Prefix (PCB,DTB,ADB,SPB).....  | 40   |
| <b>DTP</b>   |  |
| Block Diagram of DTP/external Interrupt Unit   |  |
| .....  | 342  |
| Block Diagram of Pin Related to DTP/external Interrupt   |  |
| .....  | 343  |
| DTP Operation.....   | 349  |
| DTP/External Interrupt   |  |
| .....  | 346  |
| DTP/external Interrupt,DMA Transfer,and EI <sup>2</sup> OS                                       |  |
| .....  | 347  |
| Interrupt/DTP Enable Register (ENIR: Enable Interrupt Request Register).....                     | 344  |
| Interrupt/DTP Source Register (EIRR: External Interrupt Request Register)                        |  |
| .....  | 345  |
| List of Registes for DTP/external Interrupt Unit.....  | 344  |
| Overview of DTP/external Interrupt Unit  |  |
| .....  | 342  |
| Pin Related to DTP/external Interrupt  |  |
| .....  | 342  |
| Procedures for DTP/external Interrupt Unit Operation   |  |
| .....  | 350  |
| Program Example of DTP/External Interrupt  |  |
| .....  | 352  |
| <b>E</b>   |  |
| <b>E<sup>2</sup>PROM</b>   |  |
| E <sup>2</sup> PROM Memory Map   |  |
| .....  | 467  |
| Operation of Address Match Detection Function at Storing Patch Program in E <sup>2</sup> PROM    |  |
| .....  | 469  |
| System Configuration and E <sup>2</sup> PROM Memory Map  |  |
| .....  | 466  |
| <b>Effective Address Field</b>   |  |
| Effective Address Field  |  |
| .....  | 597, 615   |
| <b>EI<sup>2</sup>OS</b>  |  |
| Configuration of EI <sup>2</sup> OS Descriptor (ISD).....  | 79   |
| Correspondence to DMA Transfer and EI <sup>2</sup> OS Function                                   |  |
| .....  | 236, 276, 303, 332, 347, 367, 396, 422, 528, 573 |
| DTP/external Interrupt,DMA Transfer,and EI <sup>2</sup> OS                                       |  |
| .....  | 347  |
| EI <sup>2</sup> OS Status Register (ISCS).....   | 82   |
| Extended Intelligent I/O Service (EI <sup>2</sup> OS).....                                       | 77   |
| Flowchart of Operation of EI <sup>2</sup> OS   |  |
| .....  | 84   |
| Interrupt of 16-bit Input/Output Timer,DMA Transfer, and EI <sup>2</sup> OS.....                 | 236  |
| Interrupt of 16-bit Reload Timer,DMA Transfer, and EI <sup>2</sup> OS                            |  |
| .....  | 303  |
| Interrupt of 8/10-bit A/D Converter,DMA Transfer, and EI <sup>2</sup> OS.....                    | 367  |
| Interrupt of 8/16-bit PPG Timer,DMA Transfer, and EI <sup>2</sup> OS                             |  |
| .....  | 332  |
| Interrupt of 8/16-bit Up/Down Counter/Timer,DMA Transfer, and EI <sup>2</sup> OS.....            | 276  |
| Interrupt of Expanded I/O Serial Interface,DMA Transfer, and EI <sup>2</sup> OS                  |  |
| .....  | 396  |
| Interrupt of I <sup>2</sup> C Interface,DMA Transfer, and EI <sup>2</sup> OS                     |  |
| .....  | 573  |
| Interrupt of PWC Timer, DMA Transfer, and EI <sup>2</sup> OS                                     |  |
| .....  | 528  |
| Interrupt of UART,DMA Transfer,and EI <sup>2</sup> OS  |  |
| .....  | 422  |
| Operation of EI <sup>2</sup> OS  |  |
| .....  | 78   |
| Procedure for Use of EI <sup>2</sup> OS  |  |
| .....  | 85   |
| Processing Time (One Transfer Time) of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) |  |
| .....  | 86   |
| <b>EIRR</b>  |  |
| Interrupt/DTP Source Register (EIRR: External Interrupt Request Register)                        |  |
| .....  | 345  |
| <b>ELVR</b>  |  |
| Request Level Setting Register (ELVR: External Level Register)                                   |  |
| .....  | 345  |
| <b>ENIR</b>  |  |
| Interrupt/DTP Enable Register (ENIR: Enable Interrupt Request Register)                          |  |
| .....  | 344  |
| <b>EPCR</b>  |  |
| Bus Control Signal Selection Register (EPCR).....  | 165  |
| <b>Erase</b>   |  |
| Flash Memory Write/Erase   |  |
| .....  | 494  |

## Index

### Erasing

|   |     |
|---|-----|
| Erasing All Data in the Flash Memory (Chip Erase)         | 498 |
| Erasing Arbitrary Data in the Flash Memory (Sector Erase) | 499 |
| Methods for Writing/Erasing Flash Memory                  | 478 |

### Error

|           |     |
|-----------|-----|
| Bus Error | 576 |
|-----------|-----|

### Event

|  |     |
|--|-----|
| Event Count Mode (External Clock Mode) | 292 |
| Operation in Event Count Mode          | 310 |

### Execution Cycle Count

|                                       |     |
|---------------------------------------|-----|
| Calculating the Execution Cycle Count | 613 |
| Execution Cycle Count                 | 612 |

### Extended Intelligent I/O Service

|  |    |
|--|----|
| Extended Intelligent I/O Service (EI <sup>2</sup> OS)  | 77 |
| Processing Time (One Transfer Time) of the Extended Intelligent I/O Service (EI <sup>2</sup> OS) | 86 |

### External Address

|   |     |
|---|-----|
| External Address Output Control Register (HACR) | 164 |
|---|-----|

### External Bus

|  |     |
|--|-----|
| Pin States in External Bus 16-bit Data Bus Mode and Multiplex 16-bit External Bus Mode     | 144 |
| Pin States in External Bus 16-bit Data Bus Mode and Non-multiplex 16-bit External Bus Mode | 146 |
| Pin States in External Bus 8-bit Data Bus Mode and Multiplex 8-bit External Bus Mode       | 145 |
| Pin States in External Bus 8-bit Data Bus Mode and Non-multiplex 8-bit External Bus Mode   | 147 |

### External Clock

|   |     |
|---|-----|
| Connection of Oscillator and External Clock | 122 |
|---|-----|

### External Connection

|  |     |
|--|-----|
| Conditions for External Connection of Peripheral Devices | 350 |
|--|-----|

### External Interrupt

|  |     |
|--|-----|
| Block Diagram of DTP/external Interrupt Unit           | 342 |
| Block Diagram of Pin Related to DTP/external Interrupt | 343 |
| External Interrupt Request Level                       | 351 |
| List of Registers for DTP/external Interrupt Unit      | 344 |
| Operation of External Interrupt Unit                   | 348 |
| Overview of DTP/external Interrupt Unit                | 342 |
| Pin Related to DTP/external Interrupt                  | 342 |
| Procedures for DTP/external Interrupt Unit Operation   | 350 |
| DTP/External Interrupt                                 | 346 |
| Program Example of DTP/External Interrupt              | 352 |

### External interrupt

|  |     |
|--|-----|
| DTP/external Interrupt,DMA Transfer,and EI <sup>2</sup> OS | 347 |
|--|-----|

### External Memory

|  |     |
|--|-----|
| External Memory access Control Signal      | 168 |
| I/O Signal Pins for External Memory Access | 160 |

### External Pin

|   |     |
|---|-----|
| Operation of External Pins in Each Mode | 159 |
|---|-----|

### External Trigger

|  |     |
|--|-----|
| Precautions When Starting by External Trigger/<br>Internal Trigger | 378 |
|--|-----|

## F

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

|   |     |
|---|-----|
| F <sup>2</sup> MC-16LX Instruction List | 619 |
|---|-----|

### Features

|  |     |
|--|-----|
| Features of the 8/10-bit A/D Converter | 356 |
|--|-----|

### Flag

|   |     |
|---|-----|
| Count Direction Flag,Count Direction Reversal Flag      | 284 |
| Flag Change Suppress Prefix (NCC)                       | 41  |
| Hardware Sequence Flags                                 | 487 |
| State Transitions of Sector Erase Timer Flag (DQ3)      | 493 |
| State Transitions of the Data Polling Flag (DQ7)        | 489 |
| State Transitions of the Timing Limit Excess Flag (DQ5) | 492 |
| State Transitions of the Toggle Bit Flag (DQ6)          | 491 |

### Flash Memory

|   |          |
|---|----------|
| Erasing All Data in the Flash Memory (Chip Erase)         | 498      |
| Erasing Arbitrary Data in the Flash Memory (Sector Erase) | 499      |
| Features of the 2M/3M Bit Flash Memory                    | 478      |
| Flash Memory Control Status Register (FMCS)               | 478, 480 |
| Flash Memory Write/Erase                                  | 494      |
| Methods for Writing/Erasing Flash Memory                  | 478      |
| Operation for Writing to Flash Memory                     | 497      |
| Resuming the Sector Erasure of Flash Memory               | 502      |
| Setting the Flash Memory to the Read/Reset State          | 495      |
| Suspending Sector Erasure for the Flash Memory            | 501      |
| Writing Data to Flash Memory                              | 496      |

### Flash Microcontroller Programmer

|  |     |
|--|-----|
| System Configuration of Flash Microcontroller Programmer (Yokogawa Digital Computer Corporation)       | 508 |
| Example of Minimum Connection with Flash Microcontroller Programmer (When Using the User Power Supply) | 512 |

### FMCS

|   |          |
|---|----------|
| Flash Memory Control Status Register (FMCS) | 478, 480 |
|---|----------|

### Free-run Timer

|  |     |
|--|-----|
| Block Diagram of Free-run Timer                                | 220 |
| Clear Timing of Free-run Timer (Match with Compare Register 0) | 243 |
| Count Timing of Free-run Timer                                 | 243 |
| List of Free-run Timer Registers                               | 224 |
| Operation of Free-run Timer                                    | 238 |
| Program Example of Free-run Timer                              | 246 |

|   |        |
|---|--------|
| <b>Frequency</b>  |        |
| Oscillation Clock Frequency and Serial Clock Input Frequency .....                    | 508    |
| <b>Function</b>   |        |
| Function Selection .....  | 435    |
| Functions of 8/16-bit PPG Timer .....   | 318    |
| <b>G</b>  |        |
| <b>Gate</b>   |        |
| Count Clear/Gate Function .....   | 283    |
| <b>General-purpose</b>  |        |
| General-purpose Register .....  | 30     |
| General-purpose Register (Register Bank) .....  | 39     |
| <b>H</b>  |        |
| <b>HACR</b>   |        |
| External Address Output Control Register (HACR) .....                                 | 164    |
| <b>Handling</b>   |        |
| Handling of Analog Input Pins .....   | 378    |
| Notes on Handling the Device .....  | 21     |
| Notes on Handling the Power Supply .....  | 22     |
| <b>Hardware</b>   |        |
| Configuration of Hardware Interrupt .....   | 54     |
| Hardware Interrupt Function .....   | 53     |
| Hardware Interrupt Operation .....  | 57     |
| Hardware Interrupt Operation Flow .....   | 58     |
| Hardware Interrupt Processing Time .....  | 62     |
| Initial Value of Hardware Components .....  | 337    |
| Procedure for Using Hardware Interrupt .....  | 59     |
| Return from Hardware Interrupt .....  | 56     |
| Starting Hardware Interrupt .....   | 56     |
| Suppressing Hardware Interrupt .....  | 54     |
| <b>Hardware Sequence</b>  |        |
| Hardware Sequence Flags .....   | 487    |
| <b>Hold</b>   |        |
| Operation of Hold Function .....  | 174    |
| <b>I</b>  |        |
| <b>I/O</b>  |        |
| Block Diagram of Expanded I/O Serial Interface .....                                  | 387    |
| Block Diagram of Pin Related to Expanded I/O Serial Interface .....                   | 388    |
| I/O Circuit Type .....  | 18     |
| I/O Register Address Pointer (IOA) .....  | 71, 81 |
| Interrupt Function of Expanded I/O Serial Interface .....                             | 403    |
| Interrupt of Expanded I/O Serial Interface .....                                      | 396    |
| Interrupt of Expanded I/O Serial Interface,DMA Transfer, and EI <sup>2</sup> OS ..... | 396    |
| Interrupt Source Related to Expanded I/O Serial Interface .....                       | 396    |
| List of Registers for Expanded I/O Serial Interface .....                             | 389    |
| Overview of Expanded I/O Serial Interface .....                                       | 386    |
| Overview of Operation for Expanded I/O Serial Interface .....                         | 397    |
| Pin Related to Expanded I/O Serial Interface .....                                    | 387    |
| Program Example of Expanded I/O Serial Interface .....                                | 404    |
| <b>I/O Map</b>  |        |
| I/O Maps .....  | 584    |
| <b>I/O Port</b>   |        |
| Functions of I/O Port .....   | 178    |
| Registers for I/O Port .....  | 179    |
| <b>I<sup>2</sup>C</b>   |        |
| Block Diagram of Pin Related to I <sup>2</sup> C Interface .....                      | 558    |
| Block Diagram of the I <sup>2</sup> C Interface .....                                 | 557    |
| I <sup>2</sup> C Interface Function .....   | 556    |
| Interrupt Control Bit and Interrupt Source of I <sup>2</sup> C Interface .....        | 572    |
| Interrupt of I <sup>2</sup> C Interface,DMA Transfer, and EI <sup>2</sup> OS .....    | 573    |
| Interrupt Source of I <sup>2</sup> C Interface .....                                  | 572    |
| List of I <sup>2</sup> C Interface Registers .....                                    | 559    |
| Pin Related to I <sup>2</sup> C Interface .....                                       | 558    |
| <b>IADR</b>   |        |
| Address Register (IADR) .....   | 570    |
| <b>IBCR</b>   |        |
| Bus Control Register (IBCR) .....   | 562    |
| Notes on Using the Bus Control Register (IBCR) .....                                  | 567    |
| <b>IBSR</b>   |        |
| Bus Status Register (IBSR) .....  | 560    |
| <b>ICCR</b>   |        |
| Clock Control Register (ICCR) .....   | 568    |
| <b>ICR</b>  |        |
| Configuration of Interrupt Control Register (ICR00 to ICR15) .....                    | 50     |
| Function of Each Bit in Interrupt Control Register (ICR00 to ICR15) .....             | 51     |
| Interrupt Control Register (ICR00 to ICR15) Function .....                            | 50     |
| <b>ICS</b>  |        |
| Input Capture Control Status Register (ICS01) .....                                   | 234    |
| <b>IDAR</b>   |        |
| Data Register (IDAR) .....  | 571    |
| <b>ILM</b>  |        |
| Interrupt Level Mask Register (ILM) .....   | 34     |
| <b>Indirect Addressing</b>  |        |
| Indirect Addressing .....   | 604    |
| <b>Initial Value</b>  |        |
| Initial Value of Hardware Components .....  | 337    |
| <b>Input Capture</b>  |        |
| Block Diagram of Input Capture .....  | 221    |
| Example of Input Capture Timing .....   | 242    |
| Input Capture Control Status Register (ICS01) .....                                   | 234    |
| Input Capture Data Register (IPCP0,IPCP1) .....                                       | 233    |
| List of Input Capture Registers .....   | 233    |
| Program Example of Input Capture .....  | 255    |
| <b>Input Pin</b>  |        |
| Selection of Input Pin .....  | 541    |
| <b>Instruction</b>  |        |
| Description of Instruction Presentation Items and Symbols .....                       | 616    |

## Index

|  |     |
|--|-----|
| Exception Processing Due to Execution of Undefined Instruction .....                   | 88  |
| F <sup>2</sup> MC-16LX Instruction List .....  | 619 |
| Instruction Types .....  | 595 |
| Interrupt Suppress Instruction .....   | 42  |
| Restrictions on Interrupt Suppress Instruction and Prefix Instruction .....            | 42  |
| Structure of Instruction Map .....   | 633 |
| <b>Instruction Presentation Items and Symbols</b>                                      |     |
| Description of Instruction Presentation Items and Symbols .....                        | 616 |
| <b>Interface</b>   |     |
| Block Diagram of Expanded I/O Serial Interface .....                                   | 387 |
| Block Diagram of Pin Related to Expanded I/O Serial Interface .....                    | 388 |
| Interrupt Function of Expanded I/O Serial Interface .....                              | 403 |
| Interrupt of Expanded I/O Serial Interface .....                                       | 396 |
| Interrupt Source Related to Expanded I/O Serial Interface .....                        | 396 |
| List of Registers for Expanded I/O Serial Interface .....                              | 389 |
| Overview of Expanded I/O Serial Interface .....  | 386 |
| Overview of Operation for Expanded I/O Serial Interface .....                          | 397 |
| Pin Related to Expanded I/O Serial Interface .....                                     | 387 |
| Program Example of Expanded I/O Serial Interface .....                                 | 404 |
| Interrupt of Expanded I/O Serial Interface,DMA Transfer, and EI <sup>2</sup> OS .....  | 396 |
| <b>Internal Clock</b>  |     |
| Internal Clock Mode .....  | 292 |
| Operations of Internal Clock Mode (Reload Mode) .....                                  | 306 |
| Operation in Internal Clock Mode (One-Shot Mode) .....                                 | 308 |
| <b>Internal Trigger</b>  |     |
| Precautions When Starting by External Trigger/Internal Trigger .....                   | 378 |
| <b>Interrupt</b>   |     |
| Block Diagram of Delay Interrupt Generation Module .....                               | 92  |
| Cancellation of Standby Mode by Interrupt .....  | 148 |
| Change to Standby Mode and Interrupts .....  | 148 |
| Configuration of Hardware Interrupt .....  | 54  |
| Example of Multiple Interrupts .....   | 60  |
| Generation of Interrupt Requests .....   | 544 |
| Hardware Interrupt Function .....  | 53  |
| Hardware Interrupt Operation .....   | 57  |
| Hardware Interrupt Operation Flow .....  | 58  |
| Hardware Interrupt Processing Time .....   | 62  |
| Interrupt Control Bit and Interrupt Source of I <sup>2</sup> C Interface .....         | 572 |
| Interrupt Control Register (ICR00 to ICR15) Function .....                             | 50  |
| Interrupt Factors,Interrupt Vector,and Interrupt Control Register .....                | 47  |
| Interrupt Generation Request .....   | 538 |
| Interrupt Level Mask Register (ILM) .....  | 34  |
| Interrupt of 16-bit Input/Output Timer .....   | 235 |
| Interrupt of 16-bit Reload Timer .....   | 303 |
| Interrupt of 16-bit Reload Timer,DMA Transfer, and EI <sup>2</sup> OS .....            | 303 |
| Interrupt of 8/10-bit A/D Converter .....  | 367 |
| Interrupt of 8/10-bit A/D Converter,DMA Transfer, and EI <sup>2</sup> OS .....         | 367 |
| Interrupt of 8/16-bit PPG Timer .....  | 331 |
| Interrupt of 8/16-bit Up/Down Counter/Timer .....                                      | 275 |
| Interrupt of A/D Converter .....   | 367 |
| Interrupt of Expanded I/O Serial Interface .....                                       | 396 |
| Interrupt of PPG Counter Underflow .....   | 331 |
| Interrupt of PWC Timer .....   | 527 |
| Interrupt of PWC Timer, DMA Transfer, and EI <sup>2</sup> OS .....                     | 528 |
| Interrupt Source of I <sup>2</sup> C Interface .....                                   | 572 |
| Interrupt Source Related to Expanded I/O Serial Interface .....                        | 396 |
| Interrupt Source Related to PWC Timer .....  | 527 |
| Interrupt Sources,Interrupt Vectors,and Interrupt Control Registers .....              | 592 |
| Interrupt Suppress Instruction .....   | 42  |
| Interrupt Timing .....   | 244 |
| Interrupt Vector .....   | 46  |
| Interrupt/DTP Enable Register (ENIR: Enable Interrupt Request Register) .....          | 344 |
| Interrupt/DTP Source Register (EIRR: External Interrupt Request Register) .....        | 345 |
| Interrupts of the 8/16-bit PPG Timer .....   | 336 |
| Interval Interrupt Function of Watch Timer .....                                       | 214 |
| List of Interrupt Control Registers .....  | 49  |
| List of Registers in Delay Interrupt Generation Module .....                           | 92  |
| Multiple Interrupt Operations .....  | 60  |
| Notes on Software Interrupts .....   | 65  |
| Notes on Using Delay Interrupt Generation Module (Delay Interrupt Request Latch) ..... | 93  |
| Operation of Delay Interrupt Generation Module .....                                   | 93  |
| Overall Flow of Interrupt Operation .....  | 45  |
| Procedure for Using Hardware Interrupt .....   | 59  |
| Restrictions on Interrupt Suppress Instruction and Prefix Instruction .....            | 42  |
| Return from Hardware Interrupt .....   | 56  |
| Return from Software Interrupt .....   | 64  |
| Sample Program for Interrupt Processing .....  | 91  |
| Software Interrupt Operation .....   | 65  |
| Stack Operation During Return from Interrupt Processing .....                          | 89  |
| Stack Operation When Interrupt Processing Starts .....                                 | 89  |
| Start of Software Interrupt .....  | 64  |
| Starting Hardware Interrupt .....  | 56  |
| Suppressing Hardware Interrupt .....   | 54  |
| Time-base Timer Interrupt and $\mu$ DMAC .....   | 192 |
| Types and Functions of Interrupts .....  | 44  |
| Configuration of Interrupt Control Register (ICR00 to ICR15) .....                     | 50  |
| Function of Each Bit in Interrupt Control Register (ICR00 to ICR15) .....              | 51  |
| Interrupt of 16-bit Input/Output Timer,DMA Transfer, and EI <sup>2</sup> OS .....      | 236 |
| Interrupt of 8/16-bit PPG Timer,DMA Transfer, and EI <sup>2</sup> OS .....             | 332 |
| Interrupt of 8/16-bit Up/Down Counter/Timer, DMA Transfer,and EI <sup>2</sup> OS ..... | 276 |

|  |        |
|--|--------|
| Interrupt of Expanded I/O Serial Interface,DMA Transfer,<br>and EI <sup>2</sup> OS.....                          | 396    |
| Interrupt of I <sup>2</sup> C Interface,DMA Transfer, and EI <sup>2</sup> OS<br>.....                            | 573    |
| Interrupt of UART .....  | 421    |
| Interrupt of UART,DMA Transfer,and EI <sup>2</sup> OS .....  | 422    |
| Interrupt Source Related to UART.....  | 421    |
| Time-base Timer Interrupt .....  | 192    |
| <b>Interval</b>  |        |
| Count Clock and Maximum Interval .....   | 538    |
| Pulse Width/Interval Calculation Method .....  | 543    |
| Range for Counting the Pulse Width/Interval .....  | 544    |
| Timer Interval.....  | 538    |
| <b>Interval Timer</b>  |        |
| Interval Timer Function .....  | 186    |
| Operation of Interval Timer Function (Time-base Timer)<br>.....  | 193    |
| <b>IOA</b>   |        |
| I/O Register Address Pointer (IOA) .....   | 71, 81 |
| <b>IPCP</b>  |        |
| Input Capture Data Register (IPCP0,IPCP1) .....  | 233    |
| <b>ISCS</b>  |        |
| EI <sup>2</sup> OS Status Register (ISCS) .....  | 82     |
| <b>ISD</b>   |        |
| Configuration of EI <sup>2</sup> OS Descriptor (ISD).....  | 79     |
| <b>L</b>   |        |
| <b>Limit</b>   |        |
| State Transitions of the Timing Limit Excess Flag (DQ5)<br>.....   | 492    |
| <b>List</b>  |        |
| List of Registers for 8/10-bit A/D Converter .....   | 359    |
| <b>Low-power</b>   |        |
| Accessing Low-power Consumption Mode Control Register<br>.....   | 130    |
| Block Diagram of Low-power Consumption Control<br>Circuit.....   | 126    |
| Low-power Consumption Mode Control Register (LPMCR)<br>.....   | 128    |
| Operational State in Low-power Consumption Mode<br>.....   | 142    |
| Notes on Accessing the Low-Power Consumption Mode<br>Control Register (LPMCR) to Enter the Standby<br>Mode ..... | 150    |
| <b>LPMCR</b>   |        |
| Low-power Consumption Mode Control Register<br>(LPMCR) .....   | 128    |
| Notes on Accessing the Low-Power Consumption Mode<br>Control Register (LPMCR) to Enter the Standby<br>Mode ..... | 150    |
| <b>LQFP-100</b>  |        |
| Package Dimensions (LQFP-100).....   | 7      |
| Pin Assignment Diagram (LQFP-100) .....  | 10     |
| <b>M</b>   |        |
| <b>M</b>   |        |
| Bus Mode Setting Bits (M1,M0) .....  | 157    |
| <b>Machine Clock</b>   |        |
| Machine Clock .....  | 118    |
| <b>Main Clock</b>  |        |
| Main Clock Mode,PLL Clock Mode,and Sub Clock Mode<br>.....   | 117    |
| <b>Map</b>   |        |
| Clock Supply Map.....  | 109    |
| <b>MASK</b>  |        |
| Chip Selection Area MASK Register (CMRx) .....   | 450    |
| <b>Master</b>  |        |
| Communication Procedure of Master/Slave Communication<br>Function.....   | 435    |
| Connection Between CPUs in Master/Slave Communication<br>.....   | 434    |
| Register Settings in Master/Slave Communication<br>.....   | 434    |
| <b>MB90480/485</b>   |        |
| MB90480/485 Series Features .....  | 2      |
| <b>MB90F481B</b>   |        |
| Basic Configuration of Serial Programming Connection with<br>MB90F481B/MB90F482B/MB90F488B/<br>MB90F489B .....   | 506    |
| <b>MB90F482B</b>   |        |
| Basic Configuration of Serial Programming Connection with<br>MB90F481B/MB90F482B/MB90F488B/<br>MB90F489B .....   | 506    |
| <b>MB90F488B</b>   |        |
| Basic Configuration of Serial Programming Connection with<br>MB90F481B/MB90F482B/MB90F488B/<br>MB90F489B .....   | 506    |
| <b>MB90F489B</b>   |        |
| Basic Configuration of Serial Programming Connection with<br>MB90F481B/MB90F482B/MB90F488B/<br>MB90F489B .....   | 506    |
| <b>MD</b>  |        |
| Settings of Mode Pins (MD2 to MD0).....  | 155    |
| <b>Measurement</b>   |        |
| Measurement Result Data.....   | 540    |
| One-shot Measurement and Repeated Measurement<br>.....   | 540    |
| Operation after Measurement Start .....  | 535    |
| <b>Measurement Mode</b>  |        |
| Measurement Mode and Counter Operation.....  | 541    |
| <b>Memory Map</b>  |        |
| E <sup>2</sup> PROM Memory Map .....   | 467    |
| Memory Map .....   | 25     |
| System Configuration and E <sup>2</sup> PROM Memory Map<br>.....   | 466    |
| <b>Memory Space</b>  |        |
| Allocation for Data of Multi-byte Length in Memory Space<br>.....  | 28     |
| Memory Space .....   | 580    |
| <b>Mirror</b>  |        |
| Block Diagram of the ROM Mirror Function Selection<br>Module .....   | 474    |
| Registers of the ROM Mirror Function Selection Module<br>.....   | 474    |

## Index

|   |     |
|---|-----|
| ROMM (ROM Mirror Function Selection Register)   | 475 |
| Mode  |     |
| Types of Mode   | 167 |
| Mode Data   |     |
| Mode Data   | 156 |
| Pin States after Mode Data Is Read  | 105 |
| Relationship Between Mode Pins and Mode Data<br>(an Example Showing Recommended Relationship) | 158 |
| Mode Fetch  |     |
| Mode Fetch  | 101 |
| Mode Pin  |     |
| Mode Pins   | 101 |
| Relationship Between Mode Pins and Mode Data<br>(an Example Showing Recommended Relationship) | 158 |
| Settings of Mode Pins (MD2 to MD0)  | 155 |
| Mode Setting  |     |
| Mode Setting  | 154 |
| Multi-byte Length   |     |
| Access to Data of Multi-byte Length   | 28  |
| Allocation for Data of Multi-byte Length in Memory Space                                      | 28  |
| Multiple  |     |
| Example of Multiple Interrupt   | 60  |
| Multiple Interrupt Operations   | 60  |
| Multiplex   |     |
| Multiplex Mode  | 175 |
| Pin States in External Bus 16-bit Data Bus Mode and<br>Multiplex 16-bit External Bus Mode     | 144 |
| Pin States in External Bus 8-bit Data Bus Mode and<br>Multiplex 8-bit External Bus Mode       | 145 |
| <b>N</b>  |     |
| NCC   |     |
| Flag Change Suppress Prefix (NCC)   | 41  |
| Non-Multiplex   |     |
| Non-Multiplex Mode  | 175 |
| Pin States in External Bus 16-bit Data Bus Mode and<br>Non-multiplex 16-bit External Bus Mode | 146 |
| Pin States in External Bus 8-bit Data Bus Mode and<br>Non-multiplex 8-bit External Bus Mode   | 147 |
| <b>O</b>  |     |
| OCCP  |     |
| Output Compare Register (OCCP0 to OCCP5)  | 230 |
| OCS   |     |
| Output Compare Control Register (OCS01/OCS23/OCS45)   | 230 |
| ODR   |     |
| Port Output Pin Registers (ODR7,ODR4)   | 183 |
| On-board  |     |
| Pins Used for Fujitsu Standard Serial On-board Writing  | 507 |

|   |          |
|---|----------|
| One-shot  |          |
| One-shot Measurement and Repeated Measurement                         | 540      |
| One-shot Operation Mode   | 537      |
| Operation in Internal Clock Mode (One-Shot Mode)                      | 308      |
| Operating   |          |
| Setting and Operating State   | 468      |
| Operation   |          |
| Operation Mode  | 333      |
| Operation Modes   | 368      |
| Operation Mode  |          |
| Operation Modes   | 154, 423 |
| Selection of Operation Mode   | 533      |
| Operation State   |          |
| Confirmation of Operation State                                       | 536      |
| Oscillation   |          |
| Oscillation Clock Frequency and Serial Clock Input<br>Frequency       | 508      |
| Reset State Waiting for Stable Oscillation                            | 99       |
| Oscillation Stabilization   |          |
| Oscillation Stabilization Wait Time                                   | 121, 149 |
| Reset Factors and Oscillation Stabilization Wait Time                 | 98       |
| Sub Clock Oscillation Stabilization Wait Time Function                | 215      |
| Timer Function for Oscillation Stabilization Wait Time                | 193      |
| Oscillator  |          |
| Connection of Oscillator and External Clock                           | 122      |
| Other   |          |
| Other Considerations  | 576      |
| the Others  | 503      |
| Outline   |          |
| Outline of Operations   | 454      |
| Output Compare  |          |
| Block Diagram of Output Compare                                       | 220      |
| List of Output Compare Registers                                      | 229      |
| Output Compare Control Register (OCS01/OCS23/OCS45)                   | 230      |
| Output Compare Register (OCCP0 to OCCP5)                              | 230      |
| Program Example of Output Compare                                     | 249      |
| Overview  |          |
| Overview  | 503      |
| <b>P</b>  |          |
| Package   |          |
| Package Dimensions (LQFP-100)   | 7        |
| Package Dimensions (QFP-100)  | 8        |
| Package of Corresponding Products                                     | 5        |
| PACSR   |          |
| Program Address Detection Control Status Register<br>(PACSR)          | 461      |
| PADR  |          |
| Functions of Program Address Detection Registers<br>(PADR0 and PADR1) | 464      |

|   |        |   |   |
|---|--------|---|---|
| Program Address Detection Registers (PADR0,PADR1)   | 463    | Interrupt of 8/16-bit PPG Timer,DMA Transfer, and EI <sup>2</sup> OS                              | 332   |
| Patch   |        | Interrupt of PPG Counter Underflow  | 331   |
| Flow of Patch Processing  | 470    | Interrupts of the 8/16-bit PPG Timer  | 336   |
| Operation of Address Match Detection Function at Storing Patch Program in E <sup>2</sup> PROM | 469    | List of 8/16-bit PPG Timer Registers  | 322   |
| PC  |        | Outline of 8/16-bit PPG Timer Operation   | 333   |
| Program Counter (PC)  | 36     | Pin Related to 8/16-bit PPG Timer   | 320   |
| PCB   |        | PPG Output Operation  | 334   |
| Bank Select Prefix (PCB,DTB,ADB,SPB)  | 40     | PPG0 to PPG5 Output Control Registers (PPG01,PPG23,PPG45)   | 328   |
| Program Counter Bank Register (PCB)   |        | PPG0/PPG2/PPG4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4)                                | 323   |
| <Initial Value: Value in Reset Vector>  | 37     | PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5)                                | 325   |
| PDR   |        | Program Example of 8/16-bit PPG Timer   | 338   |
| Port Registers (PDR0 to PDRA)   | 180    | PPGC  |   |
| Peripheral Devices  |        | PPG0/PPG2/PPG4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4)                                | 323   |
| Conditions for External Connection of Peripheral Devices                                      | 350    | PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5)                                | 325   |
| PGCSR   |        |   |   |
| μPG Control/Status Register (PGCSR)   | 552    | Prefix  |   |
| Pin   |        | Continuous Prefix Codes   | 42  |
| Pin Related to μPG Timer  | 550    | PRLH  |   |
| Pin Assignment  |        | Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5)   | 330   |
| Pin Assignment Diagram (LQFP-100)   | 10     |   |   |
| Pin Assignment Diagram (QFP-100)  | 9      | PRL   |   |
| Pin Functions   |        | Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5)   | 330   |
| Pin Functions   | 11     | Processor Status  |   |
| Pin State   |        | Processor Status (PS)   | 33  |
| Pin State in Single Chip Mode   | 143    | Product   |   |
| Pin States after Mode Data Is Read  | 105    | Product Configuration   | 4   |
| Pin States in External Bus 8-bit Data Bus Mode and Multiplex 8-bit External Bus Mode          | 145    | Program   |   |
| PLL   |        | Sample Program for Interrupt Processing   | 91  |
| Configuration of PLL Output Selection Register (PLLOS)  | 115    | Setting Method Other than Program Example   | 247, 250, 286, 313, 339, 353, 380, 405, 439 |
| PLL Clock   |        | Setting Methods Other than Program Example  | 256   |
| Main Clock Mode,PLL Clock Mode,and Sub Clock Mode   | 117    | Operation of Address Match Detection Function at Storing Patch Program in E <sup>2</sup> PROM     | 469   |
| Selection of PLL Clock Multiplication Rate  | 118    | Program Execution   | 465   |
| PLLOS   |        | Program Address   |   |
| Configuration of PLL Output Selection Register (PLLOS)  | 115    | Program Address Detection Control Status Register (PACSR)   | 461   |
| Pointer   |        | Program Address Detection Registers (PADR0,PADR1)   | 463   |
| Buffer Address Pointer (BAP)  | 73, 83 | Program Address Detection   |   |
| I/O Register Address Pointer (IOA)  | 81     | Functions of Program Address Detection Registers (PADR0 and PADR1)                                | 464   |
| Port  |        | Program Counter   |   |
| Port Direction Registers (DDR0 to DDRA)   | 181    | Program Counter (PC)  | 36  |
| Port Input Resistor Registers (RDR0,RDR1)   | 183    | Program Counter Bank Register (PCB)   |   |
| Port Output Pin Registers (ODR7,ODR4)   | 183    | <Initial Value: Value in Reset Vector>  | 37  |
| Port Registers (PDR0 to PDRA)   | 180    | Programming   |   |
| Power Supply  |        | Basic Configuration of Serial Programming Connection with MB90F481B/MB90F482B/MB90F488B/MB90F489B | 506   |
| Notes on Handling the Power Supply  | 22     |   |   |
| PPG   |        |   |   |
| Block Diagram of Pin Related to 8/16-bit PPG Timer  | 321    |   |   |
| Block Diagram of the 8/16-bit PPG Timer   | 319    |   |   |
| Functions of 8/16-bit PPG Timer   | 318    |   |   |
| Interrupt of 8/16-bit PPG Timer   | 331    |   |   |



## Index

### Protection

|  |     |
|--|-----|
| Caution When Using the Conversion Data Protection Function ..... | 376 |
| Conversion Data Protection Function.....                         | 376 |

### PS

|                             |    |
|-----------------------------|----|
| Processor Status (PS) ..... | 33 |
|-----------------------------|----|

### Pulse

|   |     |
|---|-----|
| Minimum Input Pulse Width.....                          | 543 |
| Operational Flow of Pulse Width Measurement .....       | 545 |
| Pin Output Control of Pulses .....                      | 335 |
| Pulse Width/Interval Calculation Method .....           | 543 |
| Range for Counting the Pulse Width/Interval.....        | 544 |
| Relationship Between Reload Value and Pulse Width ..... | 334 |
| Start and Stop of Timer/Pulse Width Measurement .....   | 535 |

### Pulse Width

|  |     |
|--|-----|
| Operations of the Pulse Width Measurement Function ..... | 531 |
|--|-----|

### PWC

|  |     |
|--|-----|
| Block Diagram of Pin Related to PWC Timer .....                    | 518 |
| Block Diagram of PWC Timer .....                                   | 517 |
| Interrupt of PWC Timer .....                                       | 527 |
| Interrupt of PWC Timer, DMA Transfer, and EI <sup>2</sup> OS ..... | 528 |
| Interrupt Source Related to PWC Timer .....                        | 527 |
| List of PWC Timer Registers .....                                  | 519 |
| Notes on PWC Timer Usage .....                                     | 546 |
| Outline of PWC Timer Operations .....                              | 529 |
| Pin Related to PWC Timer .....                                     | 518 |
| PWC Control/Status Register (PWCSR0 to PWCSR2) .....               | 520 |
| PWC Data Buffer Register (PWCR0 to PWCR 2) .....                   | 525 |
| PWC Timer Functions.....   | 516 |

### PWCR

|  |     |
|--|-----|
| PWC Data Buffer Register (PWCR0 to PWCR 2) ..... | 525 |
|--|-----|

### PWCSR

|  |     |
|--|-----|
| PWC Control/Status Register (PWCSR0 to PWCSR2) ..... | 520 |
|--|-----|

## Q

### QFP-100

|  |   |
|--|---|
| Package Dimensions (QFP-100) .....     | 8 |
| Pin Assignment Diagram (QFP-100) ..... | 9 |

## R

### RCR

|   |     |
|---|-----|
| Reload/compare Register (ch.0/ch.1) (RCR0/RCR1) ..... | 274 |
|---|-----|

### RDR

|   |     |
|---|-----|
| Port Input Resistor Registers (RDR0,RDR1) ..... | 183 |
|---|-----|

### Read

|  |     |
|--|-----|
| Setting the Flash Memory to the Read/Reset State ..... | 495 |
|--|-----|

### Ready

|                      |     |
|----------------------|-----|
| Ready Function ..... | 171 |
|----------------------|-----|

### Register

|  |          |
|--|----------|
| 16-bit Reload Register (TMRLR) .....                                   | 302      |
| 16-bit Timer Register (TMR) .....                                      | 301      |
| 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR) .....       | 301      |
| Accessing Low-power Consumption Mode Control Register .....            | 130      |
| Address Register (IADR) .....  | 570      |
| Analog Input Enable Register (ADER) .....                              | 184      |
| Automatic Ready Function Selection Register (ARSR) .....               | 162      |
| Bus Control Register (IBCR).....                                       | 562      |
| Bus Control Signal Selection Register (EPCR) .....                     | 165      |
| Bus Status Register (IBSR) .....                                       | 560      |
| Chip Selection Active Level Register (CALR).....                       | 453      |
| Chip Selection Area MASK Register (CMRx).....                          | 450      |
| Chip Selection Area Register (CARx) .....                              | 451      |
| Chip Selection Control Register (CSCR).....                            | 452      |
| Clock Control Register (ICCR) .....                                    | 568      |
| Communication Prescaler Control Register (CDCR) .....                  | 419      |
| Communication Prescaler Control Register0/1 (SDCR0/SDCR1) .....        | 395      |
| Compare Clear Register (CPCLR) .....                                   | 224      |
| Condition Code Register (CCR).....                                     | 33       |
| Configuration of Clock Selection Register (CKSCR) .....                | 112      |
| Configuration of PLL Output Selection Register (PLLOS) .....           | 115      |
| Configuration of Watch Timer Control Register (WTC) .....              | 212      |
| Control Status Register 1 (ADCS1).....                                 | 360      |
| Control Status Register 2 (ADCS2).....                                 | 363      |
| Counter Control Register (ch.0) Upper (CCR0) .....                     | 265      |
| Counter Control Register (ch.0/ch.1) Lower (CCRL0/CCRL1) .....         | 269      |
| Counter Control Register (ch.1) Upper (CCR1) .....                     | 267      |
| Counter Status Register 0/1 (CSR0/CSR1) .....                          | 271      |
| Data Register (IDAR).....  | 571      |
| Data Registers (ADCR2 and ADCR1) .....                                 | 366      |
| Dedicated Registers .....  | 29       |
| Direct Page Register (DPR)<Initial Value: 01 <sub>H</sub> >.....       | 38       |
| Divide Ratio Control Register (DIVR0 to DIVR2) .....                   | 526      |
| DMA Control Status Register (DMACS) .....                              | 72       |
| EI <sup>2</sup> OS Status Register (ISCS) .....                        | 82       |
| External Address Output Control Register (HACR) .....                  | 164      |
| Flash Memory Control Status Register (FMCS) .....                      | 478, 480 |
| General-purpose Register .....   | 30       |
| General-purpose Register (Register Bank) .....                         | 39       |
| I/O Register Address Pointer (IOA) .....                               | 71, 81   |
| Input Capture Control Status Register (ICS01) .....                    | 234      |
| Input Capture Data Register (IPCP0,IPCP1) .....                        | 233      |
| Interrupt Control Register (ICR00 to ICR15) Function .....             | 50       |
| Interrupt Factors,Interrupt Vector,and Interrupt Control Register..... | 47       |

|   |          |
|---|----------|
| Interrupt Level Mask Register (ILM).....  | 34       |
| Interrupt/DTP Enable Register (ENIR: Enable Interrupt Request Register).....                              | 344      |
| Interrupt/DTP Source Register (EIRR: External Interrupt Request Register) .....                           | 345      |
| List of 8/16-bit PPG Timer Registers.....   | 322      |
| List of 8/16-bit Up/Down Counter/Timer Registers .....  | 264      |
| List of Free-run Timer Registers .....  | 224      |
| List of I <sup>2</sup> C Interface Registers .....  | 559      |
| List of Input Capture Registers .....   | 233      |
| List of Interrupt Control Registers .....   | 49       |
| List of $\mu$ DMAC Registers.....   | 66       |
| List of Output Compare Registers.....   | 229      |
| List of PWC Timer Registers.....  | 519      |
| List of Registers .....   | 161, 296 |
| List of Registers for 8/10-bit A/D Converter .....  | 359      |
| List of Registers for DTP/external Interrupt Unit.....  | 344      |
| List of Registers Used for the Chip Selection Facility .....  | 449      |
| List of UART Registers.....   | 411      |
| Notes on Using the Bus Control Register (IBCR) .....  | 567      |
| Output Compare Control Register (OCS01/OCS23/OCS45) .....   | 230      |
| Output Compare Register (OCCP0 to OCCP5) .....  | 230      |
| Port Input Resistor Registers (RDR0,RDR1).....  | 183      |
| Port Output Pin Registers (ODR7,ODR4).....  | 183      |
| PPG0 to PPG5 Output Control Registers (PPG01,PPG23,PPG45) .....   | 328      |
| PPG0/PPG2/PPG4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4) .....                                  | 323      |
| PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5) .....                                  | 325      |
| Program Address Detection Control Status Register (PACSR) .....   | 461      |
| Program Address Detection Registers (PADR0,PADR1) .....   | 463      |
| Program Counter Bank Register (PCB) <Initial Value: Value in Reset Vector> .....                          | 37       |
| PWC Data Buffer Register (PWCR0 to PWCR 2) .....  | 525      |
| Register Settings in Master/Slave Communication .....   | 434      |
| Register Settings in Two-way Communication .....  | 432      |
| Registers for I/O Port.....   | 179      |
| Registers of the ROM Mirror Function Selection Module .....   | 474      |
| Reload/Compare Register (ch.0/ch.1) (RCR0/RCR1) .....   | 274      |
| Request Level Setting Register (ELVR: External Level Register).....                                       | 345      |
| Serial Control Register (SCR).....  | 414      |
| Serial Data Register 0/1 (SDR0/SDR1).....   | 394      |
| Serial Input/Output Register (SIDR/SODR) .....  | 416      |
| Serial Mode Control Status Register0/1 (SMCS0/SMCS1) .....  | 390      |
| Serial Mode Register (SMR) .....  | 412      |
| Serial Status Register (SSR) .....  | 417      |
| Time-base Timer Control Register (TBTC).....  | 190      |
| Timer Control Status Register (TMCSR).....  | 297      |
| Timer Counter Control Status Register (TCCS) .....  | 225      |
| Timer Counter Data Register (TCDDT) .....   | 225      |
| Up/Down Count Register (ch.0/ch.1) (UDCR0/UDCR1) .....  | 273      |
| Up/Down Timer Input Enable Register (UDER) .....  | 184      |
| Watchdog Timer Control Register (WDTC) .....  | 201      |
| Configuration of Interrupt Control Register (ICR00 to ICR15).....   | 50       |
| Function of Each Bit in Interrupt Control Register (ICR00 to ICR15).....                                  | 51       |
| List of Registers and Initial Values of Address Match Detection Function .....                            | 460      |
| Low-power Consumption Mode Control Register (LPMCR) .....   | 128      |
| Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode..... | 150      |
| Port Direction Registers (DDR0 to DDRA) .....   | 181      |
| Port Registers (PDR0 to PDRA) .....   | 180      |
| Reload Registers (PRLL0 to PRLL5, PRLH0 to PRLH5) .....   | 330      |
| $\mu$ PG Control/Status Register (PGCSR) .....  | 552      |
| <b>Register Bank</b>  |          |
| Common Register Bank Prefix (CMR) .....   | 41       |
| General-purpose Register (Register Bank).....   | 39       |
| <b>Register Bank Pointer</b>  |          |
| Register Bank Pointer (RP).....   | 34       |
| <b>Relationship</b>   |          |
| Relationship Between Mode Pins and Mode Data (an Example Showing Recommended Relationship) .....          | 158      |
| Relationship Between Reload Value and Pulse Width .....   | 334      |
| <b>Reload</b>   |          |
| 16-bit Reload Register (TMRLR).....   | 302      |
| 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR) .....  | 301      |
| Operations of Internal Clock Mode (Reload Mode) .....   | 306      |
| Relationship Between Reload Value and Pulse Width .....   | 334      |
| Reload Function.....  | 280      |
| Reload Operation Mode .....   | 537      |
| Reload/Compare Register (ch.0/ch.1) (RCR0/RCR1) .....   | 274      |
| Selection of Reload and Compare Functions .....   | 280      |
| Timer Value and Reload Value.....   | 537      |
| Up/Down Count at any Width in Reload/Compare Function .....   | 281      |
| Reload Registers (PRLL0 to PRLL5, PRLH0 to PRLH5) .....   | 330      |
| <b>Reload Register</b>  |          |
| Write Timing to the Reload Register.....  | 337      |
| <b>Reload Timer</b>   |          |
| Block Diagram of Pin Related to 16-bit Reload Timer .....   | 295      |
| Block Diagram of the 16-bit Reload Timer.....   | 294      |
| Interrupt of 16-bit Reload Timer .....  | 303      |
| Interrupt of 16-bit Reload Timer,DMA Transfer,and EI <sup>2</sup> OS .....                                | 303      |

## Index

|   |               |
|---|---------------|
| Operation Modes of the 16-bit Reload Timer.....                                       | 292           |
| Pin Related to 16-bit Reload Timer .....  | 294           |
| Program Example of 16-bit Reload Timer.....   | 312           |
| Settings of the 16-bit Reload Timer .....   | 304           |
| <b>Repeated</b>   |               |
| One-shot Measurement and Repeated Measurement .....                                   | 540           |
| <b>Request</b>  |               |
| External Interrupt Request Level .....  | 351           |
| Request Level Setting Register<br>(ELVR: External Level Register).....                | 345           |
| <b>Reset</b>  |               |
| Block Diagram of External-Reset Pin .....   | 100           |
| Pin States During a Reset .....   | 105           |
| Reset Factors .....   | 96            |
| Reset Factors and Oscillation Stabilization Wait Time .....                           | 98            |
| Reset State Waiting for Stable Oscillation.....                                       | 99            |
| Setting the Flash Memory to the Read/Reset State .....                                | 495           |
| <b>Reset-Factor</b>   |               |
| Cautions about Reset-Factor Bits .....  | 104           |
| Correspondence Between Reset-Factor Bits and Reset<br>Factors .....                   | 104           |
| Reset-Factor Bits .....   | 103           |
| <b>Resetting</b>  |               |
| Overview of Resetting.....  | 101           |
| <b>Resistor</b>   |               |
| Port Input Resistor Registers (RDR0,RDR1) .....                                       | 183           |
| <b>Restart</b>  |               |
| Restart.....  | 536           |
| <b>Result Data</b>  |               |
| Measurement Result Data .....   | 540           |
| <b>ROM</b>  |               |
| Block Diagram of the ROM Mirror Function Selection<br>Module.....                     | 474           |
| Registers of the ROM Mirror Function Selection Module .....                           | 474           |
| ROMM (ROM Mirror Function Selection Register)<br>.....                                | 475           |
| <b>ROMM</b>   |               |
| ROMM (ROM Mirror Function Selection Register)<br>.....                                | 475           |
| <b>RP</b>   |               |
| Register Bank Pointer (RP) .....  | 34            |
| <b>S</b>  |               |
| <b>S</b>  |               |
| Setting Bits of Different Modes (S1,S0) .....   | 156           |
| <b>SCR</b>  |               |
| Serial Control Register (SCR).....  | 414           |
| <b>SDCR</b>   |               |
| Communication Prescaler Control Register0/1<br>(SDCR0/SDCR1).....                     | 395           |
| <b>SDR</b>  |               |
| Serial Data Register 0/1 (SDR0/SDR1) .....  | 394           |
| <b>Sector</b>   |               |
| Sector Configuration.....   | 479           |
| <b>Sector Erase</b>   |               |
| Chip/Sector Erase Operation .....   | 489           |
| Erasing Arbitrary Data in the Flash Memory (Sector Erase)<br>.....                    | 499           |
| Sector Erase Operation .....  | 493           |
| Sector Erase Suspend .....  | 490, 491, 493 |
| State Transitions of Sector Erase Timer Flag (DQ3)<br>.....                           | 493           |
| Write/Chip Sector Erase Operation .....   | 492           |
| Write/Chip Sector Erase Operations .....  | 491           |
| <b>Sector Erasure</b>   |               |
| Procedure for Sector Erasure .....  | 499           |
| Resuming the Sector Erasure of Flash Memory.....                                      | 502           |
| Suspending Sector Erasure for the Flash Memory<br>.....                               | 501           |
| <b>Security</b>   |               |
| Cancel of Security .....  | 503           |
| Operation in Security Permission .....  | 503           |
| Setting of Security .....   | 503           |
| <b>Serial</b>   |               |
| Serial Input/Output Register (SIDR/SODR) .....  | 416           |
| Serial Status Register (SSR).....   | 417           |
| <b>Serial Control</b>   |               |
| Serial Control Register (SCR).....  | 414           |
| <b>Serial Data</b>  |               |
| Operation During Serial Data Transfer .....   | 402           |
| Serial Data Register 0/1 (SDR0/SDR1).....   | 394           |
| <b>Serial I/O</b>   |               |
| Operational States of Serial I/O Units .....  | 399           |
| <b>Serial Mode</b>  |               |
| Serial Mode Control Status Register0/1 (SMCS0/SMCS1)<br>.....                         | 390           |
| Serial Mode Register (SMR) .....  | 412           |
| <b>Serial Programming</b>   |               |
| Examples of Serial Programming Connections .....                                      | 509           |
| <b>Setting</b>  |               |
| Setting and Operating State.....  | 468           |
| <b>Setting Bit</b>  |               |
| Bus Mode Setting Bits (M1,M0) .....   | 157           |
| Setting Bits of Different Modes (S1,S0).....  | 156           |
| <b>Shift Clock</b>  |               |
| External Shift Clock Mode .....   | 398           |
| Internal Shift Clock Mode .....   | 398           |
| <b>Shift Operation</b>  |               |
| Start/Stop Timing and Input/Output Timing of Shift<br>Operation.....                  | 401           |
| <b>SIDR</b>   |               |
| Serial Input/Output Register (SIDR/SODR) .....  | 416           |
| <b>Single Chip Mode</b>   |               |
| Pin State in Single Chip Mode .....   | 143           |
| <b>Single Mode</b>  |               |
| Example of $\mu$ DMAC Start in Single Mode .....                                      | 370           |
| <b>Single-Chip</b>  |               |
| Example of Connection in Single-Chip Mode<br>(When Using the User Power Supply) ..... | 510           |

|  |     |
|--|-----|
| <b>Slave</b>   |     |
| Communication Procedure of Master/Slave Communication Function .....                                       | 435 |
| Connection Between CPUs in Master/Slave Communication .....  | 434 |
| Register Settings in Master/Slave Communication.....   | 434 |
| <b>Sleep Mode</b>  |     |
| Canceling the Sleep Mode .....   | 134 |
| Change to Sleep Mode .....   | 133 |
| <b>SLP</b>   |     |
| Priority of STP,SLP,and TMD Bits .....   | 130 |
| <b>SMCS</b>  |     |
| Serial Mode Control Status Register0/1 (SMCS0/SMCS1) .....   | 390 |
| <b>SMR</b>   |     |
| Serial Mode Register (SMR) .....   | 412 |
| <b>SODR</b>  |     |
| Serial Input/Output Register (SIDR/SODR) .....   | 416 |
| <b>Software</b>  |     |
| Notes on Software Interrupt .....  | 65  |
| Return from Software Interrupt .....   | 64  |
| Software Interrupt Operation .....   | 65  |
| Start of Software Interrupt .....  | 64  |
| <b>SPB</b>   |     |
| Bank Select Prefix (PCB,DTB,ADB,SPB).....  | 40  |
| <b>SSP</b>   |     |
| User Stack Pointer (USP) and System Stack Pointer (SSP) .....  | 32  |
| <b>SSR</b>   |     |
| Serial Status Register (SSR) .....   | 417 |
| <b>Stack</b>   |     |
| Stack Area .....   | 90  |
| <b>Stack Operation</b>   |     |
| Stack Operation During Return from Interrupt Processing .....  | 89  |
| Stack Operation When Interrupt Processing Starts .....   | 89  |
| <b>Standby</b>   |     |
| Change to Standby Mode and Interrupts .....  | 148 |
| Operational States in Standby Mode.....  | 132 |
| Standby Mode.....  | 125 |
| <b>Standby Mode</b>  |     |
| Cancellation of Standby Mode by Interrupt.....   | 148 |
| Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode ..... | 150 |
| <b>Start</b>   |     |
| Start Condition.....   | 574 |
| <b>State Transition</b>  |     |
| State Transition Diagram .....   | 141 |
| State Transitions During Count Operation .....   | 305 |
| <b>Stop</b>  |     |
| Stop.....  | 536 |
| Stop Condition .....   | 574 |
| <b>Stop Mode</b>   |     |
| Canceling the Stop Mode.....   | 139 |
| Change to Stop Mode .....  | 139 |
| Example of $\mu$ DMAC Start in Stop Mode.....  | 374 |
| <b>STP</b>   |     |
| Priority of STP,SLP,and TMD Bits .....   | 130 |
| <b>Structure</b>   |     |
| Structure of Instruction Map .....   | 633 |
| <b>Sub Clock</b>   |     |
| Main Clock Mode,PLL Clock Mode,and Sub Clock Mode .....  | 117 |
| Sub Clock Oscillation Stabilization Wait Time Function .....   | 215 |
| <b>Sub Clock Mode</b>  |     |
| Precaution on Use of the Sub Clock Mode .....  | 482 |
| Setting Sub Clock Mode .....   | 483 |
| <b>Switching</b>   |     |
| Switching the Clock Mode .....   | 149 |
| <b>Synchronous Mode</b>  |     |
| Operation in CLK Synchronous Mode (Operation Mode 2) .....   | 430 |
| <b>System Stack Pointer</b>  |     |
| User Stack Pointer (USP) and System Stack Pointer (SSP) .....  | 32  |
| <b>T</b>   |     |
| <b>TBTC</b>  |     |
| Time-base Timer Control Register (TBTC) .....  | 190 |
| <b>TCCS</b>  |     |
| Timer Counter Control Status Register (TCCS) .....   | 225 |
| <b>TCDT</b>  |     |
| Timer Counter Data Register (TCDT) .....   | 225 |
| <b>Time-base Timer</b>   |     |
| Block Diagram of Time-base Timer.....  | 188 |
| Canceling the Time-base Timer Mode.....  | 135 |
| Change to Time-base Timer Mode .....   | 135 |
| Notes on Using Time-base Timer.....  | 196 |
| Operation of Interval Timer Function (Time-base Timer) .....   | 193 |
| Operation of Time-base Timer.....  | 195 |
| Sample Programs of Time-base Timer.....  | 197 |
| Time-base Timer Control Register (TBTC) .....  | 190 |
| Time-base Timer Interrupt.....   | 192 |
| Time-base Timer Interrupt and $\mu$ DMAC .....   | 192 |
| <b>Timer</b>   |     |
| 16-bit Timer Register (TMR) .....  | 301 |
| 16-bit Timer Register (TMR)/<br>16-bit Reload Register (TMRLR).....  | 301 |
| Block Diagram of 8/16-bit Up/Down Counter/Timer .....  | 261 |
| Block Diagram of $\mu$ PG Timer .....  | 550 |
| Block Diagram of Pin Related to 16-bit Input/<br>Output Timer .....  | 222 |
| Block Diagram of Pin Related to 8/16-bit PPG Timer .....   | 321 |
| Block Diagram of Pin Related to 8/16-bit Up/Down<br>Counter/Timer .....                                    | 263 |
| Block Diagram of Pin Related to $\mu$ PG Timer .....   | 551 |
| Block Diagram of Pin Related to PWC Timer .....  | 518 |
| Block Diagram of PWC Timer .....   | 517 |

## Index

|  |     |
|--|-----|
| Block Diagram of the 8/16-bit PPG Timer .....  | 319 |
| Clearing the Timer .....   | 537 |
| Functions of 16-bit Input/Output Timer.....  | 218 |
| Functions of 8/16-bit PPG Timer .....  | 318 |
| Interrupt of 16-bit Input/Output Timer .....   | 235 |
| Interrupt of 8/16-bit PPG Timer .....  | 331 |
| Interrupt of 8/16-bit Up/Down Counter/Timer .....                                      | 275 |
| Interrupt of PWC Timer, DMA Transfer, and EI <sup>2</sup> OS .....                     | 528 |
| Interrupt Source Related to PWC Timer .....  | 527 |
| Interrupts of the 8/16-bit PPG Timer .....   | 336 |
| List of 8/16-bit PPG Timer Registers .....   | 322 |
| List of 8/16-bit Up/Down Counter/Timer Registers .....                                 | 264 |
| List of PWC Timer Registers.....   | 519 |
| Major Functions of 8/16-bit Up/Down Counter/Timer .....                                | 260 |
| Operation and Timing of 16-bit Input/Output Timer .....                                | 237 |
| Operation of Timer Functions .....   | 530 |
| Outline of 8/16-bit PPG Timer Operation .....  | 333 |
| Outline of PWC Timer Operations .....  | 529 |
| Pin Related to 16-bit Input/Output Timer.....  | 221 |
| Pin Related to 8/16-bit PPG Timer.....   | 320 |
| Pin Related to 8/16-bit Up/Down Counter/Timer.....                                     | 262 |
| Pin Related to $\mu$ PG Timer .....  | 550 |
| Pin Related to PWC Timer .....   | 518 |
| Program Example of 8/16-bit Up/Down Counter/Timer .....                                | 285 |
| PWC Timer Functions.....   | 516 |
| Start and Stop of Timer/Pulse Width Measurement .....                                  | 535 |
| State Transitions of Sector Erase Timer Flag (DQ3) .....                               | 493 |
| Timer Control Status Register (TMCSR).....   | 297 |
| Timer Counter Control Status Register (TCCS).....                                      | 225 |
| Timer Counter Data Register (TCDT).....  | 225 |
| Timer Interval .....   | 538 |
| Timer Value and Reload Value .....   | 537 |
| Timing Chart of $\mu$ PG Timer .....   | 554 |
| Up/Down Timer Input Enable Register (UDER).....  | 184 |
| Interrupt of 8/16-bit PPG Timer,DMA Transfer, and EI <sup>2</sup> OS .....             | 332 |
| Interrupt of 8/16-bit Up/Down Counter/Timer,DMA Transfer, and EI <sup>2</sup> OS ..... | 276 |
| Program Example of 8/16-bit PPG Timer .....  | 338 |
| Timer Function for Oscillation Stabilization Wait Time .....                           | 193 |
| Timer Operation Flow.....  | 539 |
| <b>Timing</b>  |     |
| Capture Timing to Input Signal .....   | 245 |
| Change Timing of Output Pin .....  | 244 |
| End Timing of the Automatic Algorithm .....  | 482 |
| Notes on Timing.....   | 554 |
| Start/Stop Timing and Input/Output Timing of Shift Operation .....                     | 401 |
| State Transitions of the Timing Limit Excess Flag (DQ5) .....                          | 492 |
| Write Timing to the Reload Register .....  | 337 |
| <b>Timing Chart</b>  |     |
| Timing Chart of $\mu$ PG Timer .....   | 554 |

|                   |   |     |
|-------------------|---|-----|
| TMCSR             | Timer Control Status Register (TMCSR).....  | 297 |
| TMD               | Priority of STP,SLP,and TMD Bits.....   | 130 |
| TMR               | 16-bit Timer Register (TMR) .....   | 301 |
|                   | 16-bit Timer Register<br>(TMR)/16-bit Reload Register (TMRLR)<br>.....                | 301 |
| TMRLR             | 16-bit Reload Register (TMRLR) .....  | 302 |
|                   | 16-bit Timer Register<br>(TMR)/16-bit Reload Register (TMRLR)<br>.....                | 301 |
| Toggle Bit        | State Transitions of the Toggle Bit Flag (DQ6).....                                   | 491 |
| Transition        | State Transition Diagram.....   | 141 |
| Trigger           | Precautions When Starting by External Trigger/<br>Internal Trigger.....               | 378 |
| Two-way           | Communication Procedure for Two-way Communication<br>Function .....                   | 433 |
|                   | Connection Between CPUs in Two-way Communication<br>.....                             | 432 |
|                   | Register Settings in Two-way Communication .....                                      | 432 |
| U                 |   |     |
| UART              | Block Diagram of Pin Related to UART .....  | 410 |
|                   | Interrupt of UART .....   | 421 |
|                   | Interrupt of UART,DMA Transfer,and EI <sup>2</sup> OS .....                           | 422 |
|                   | Interrupt Source Related to UART.....   | 421 |
|                   | List of UART Registers .....  | 411 |
|                   | Pin Related to UART .....   | 410 |
|                   | Precautions on Use of the UART.....   | 437 |
|                   | Program Example of UART .....   | 438 |
|                   | UART Block Diagram.....   | 409 |
|                   | UART Clock Selection .....  | 424 |
|                   | UART Features.....  | 408 |
| UDCR              | Up/Down Count Register (ch.0/ch.1) (UDCR0/UDCR1)<br>.....                             | 273 |
|                   | Writing Data to UDCR .....  | 283 |
| UDER              | Up/down Timer Input Enable Register (UDER) .....                                      | 184 |
| Up/Down           | Up/Down Count at any Width in Reload/Compare Function<br>.....                        | 281 |
|                   | Up/Down Count Register (ch.0/ch.1) (UDCR0/UDCR1)<br>.....                             | 273 |
| Up/down           | Up/down Timer Input Enable Register (UDER) .....                                      | 184 |
| User Power Supply | Example of Connection in Single-Chip Mode<br>(When Using the User Power Supply) ..... | 510 |

|  |          |
|--|----------|
| Example of Minimum Connection with Flash<br>Microcontroller Programmer<br>(When Using the User Power Supply) ..... | 512      |
| <b>User Stack Pointer</b>  |          |
| User Stack Pointer (USP) and System Stack Pointer (SSP)<br>.....   | 32       |
| <b>USP</b>   |          |
| User Stack Pointer (USP) and System Stack Pointer (SSP)<br>.....   | 32       |
| <b>V</b>   |          |
| <b>Vector</b>  |          |
| Interrupt Vector.....  | 46       |
| <b>W</b>   |          |
| <b>Wait Time</b>   |          |
| Oscillation Stabilization Wait Time .....  | 121, 149 |
| Reset Factors and Oscillation Stabilization Wait Time<br>.....   | 98       |
| Sub Clock Oscillation Stabilization Wait Time Function<br>.....  | 215      |
| Timer Function for Oscillation Stabilization Wait Time<br>.....  | 193      |
| <b>Watch Counter</b>   |          |
| Watch Counter .....  | 214      |
| <b>Watch Mode</b>  |          |
| Canceling the Watch Mode .....   | 137      |
| Change to Watch Mode.....  | 137      |
| <b>Watch Timer</b>   |          |
| Block Diagram of Watch Timer .....   | 211      |
| Configuration of Watch Timer Control Register (WTC)<br>.....   | 212      |
| Functions of Watch Timer .....   | 210      |
| Interval Interrupt Function of Watch Timer.....  | 214      |
| <b>Watchdog Timer</b>  |          |
| Block Diagram of Watchdog Timer .....  | 203      |
| Clock Source for Watchdog Timer Specifying Function<br>.....   | 215      |
| Functions of Watchdog Timer .....  | 200      |
| Notes on Using Watchdog Timer.....   | 207      |
| Operation of Watchdog Timer .....  | 205      |
| Sample Programs for Watchdog Timer .....   | 208      |
| Watchdog Timer Control Register (WDTC).....  | 201      |
| <b>Waveform</b>  |          |
| Examples of Output Waveform.....   | 240      |
| <b>WDTC</b>  |          |
| Watchdog Timer Control Register (WDTC) .....   | 201      |
| <b>Width</b>   |          |
| Minimum Input Pulse Width .....  | 543      |
| Operational Flow of Pulse Width Measurement .....  | 545      |
| Pulse Width/Interval Calculation Method.....   | 543      |
| Range for Counting the Pulse Width/Interval .....  | 544      |
| Start and Stop of Timer/Pulse Width Measurement<br>.....   | 535      |
| <b>Write</b>   |          |
| Flash Memory Write/Erase .....   | 494      |
| Write Operation .....  | 489      |
| Write Timing to the Reload Register.....   | 337      |
| Write/Chip Sector Erase Operation.....   | 492      |
| Write/Chip Sector Erase Operations.....  | 491      |
| <b>Writing</b>   |          |
| Methods for Writing/Erasing Flash Memory .....   | 478      |
| Pins Used for Fujitsu Standard Serial On-board Writing<br>.....  | 507      |
| Writing Data to UDCR .....   | 283      |
| <b>WTC</b>   |          |
| Configuration of Watch Timer Control Register (WTC)<br>.....   | 212      |
| $\mu$  |          |
| <b><math>\mu</math>DMAC</b>  |          |
| Conversion Operation Using $\mu$ DMAC.....   | 369      |
| Example of $\mu$ DMAC Start in Continuous Mode .....   | 372      |
| Example of $\mu$ DMAC Start in Single Mode .....   | 370      |
| Example of $\mu$ DMAC Start in Stop Mode .....   | 374      |
| List of $\mu$ DMAC Registers .....   | 66       |
| Operation Flow of Conversion Data Protection Function<br>(when $\mu$ DMAC is Used) .....                           | 377      |
| Time-base Timer Interrupt and $\mu$ DMAC .....   | 192      |
| $\mu$ DMAC Functions .....   | 66       |
| $\mu$ DMAC Operations.....   | 68       |
| $\mu$ DMAC Processing Procedure .....  | 74       |
| $\mu$ DMAC Processing Time (Time Per One-time Transfer)<br>.....   | 75       |
| <b><math>\mu</math>PG</b>  |          |
| Block Diagram of $\mu$ PG Timer .....  | 550      |
| Block Diagram of Pin Related to $\mu$ PG Timer .....   | 551      |
| $\mu$ PG Control/Status Register (PGCSR) .....   | 552      |
| Pin Related to $\mu$ PG Timer .....  | 550      |
| Timing Chart of $\mu$ PG Timer.....  | 554      |



CM44-10121-7E

---

**FUJITSU MICROELECTRONICS • CONTROLLER MANUAL**

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

16-BIT MICROCONTROLLER

MB90480/485 Series

HARDWARE MANUAL

---

July 2008 the seventh edition

Published **FUJITSU MICROELECTRONICS LIMITED**

Edited Business & Media Promotion Dept.

---



