



---

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**  
**MB90570 series**  
**HARDWARE MANUAL**



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

## **16-BIT MICROCONTROLLER**

### **MB90570 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.e.fujitsu.com/micom/en-support/>



# PREFACE

## ■ Objectives and Intended Reader

Thank you for your interest in Fujitsu semiconductor products.

The MB90570 series was developed as one of the F<sup>2</sup>MC-16LX series general-purpose versions, or as a proprietary 16-bit one-chip microcontroller capable of supporting an application specific IC (ASIC).

This manual, which is intended for engineers designing products using this semiconductor, explains the functions and operations of the MB90570 series.

## ■ Trademark

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

Other system and product names in this manual are trademarks of respective companies or organizations.

The symbols ™ and ® are sometimes omitted in this manual.

## ■ The I<sup>2</sup>C Licence

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.

## ■ Structure of This Manual

This manual includes the following 27 chapters and an appendix.

### **CHAPTER 1 "OVERVIEW"**

This chapter describes an overview of the MB90570 series.

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

This chapter describes the CPU functions and operations.

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

This chapter describes the interrupt functions and operations.

### **CHAPTER 4 "CLOCK AND RESET"**

This chapter describes the functions and operations of the clock and reset.

### **CHAPTER 5 "LOW-POWER CONSUMPTION CONTROL CIRCUIT"**

This chapter describes the functions and operations of the low-power consumption control circuit.

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

This chapter describes the functions and operations of the low-power consumption mode.

### **CHAPTER 7 "MEMORY ACCESS MODE"**

This chapter describes the functions and operations of the memory access modes.

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

This chapter describes the functions and operations of the I/O port.

## **CHAPTER 9 "TIMEBASE TIMER"**

This chapter describes the functions and operations of the Timebase Timer.

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

This chapter describes the functions and operations of the Watchdog Timer.

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

This chapter describes the functions and operations of the Watch Timer.

## **CHAPTER 12 "16-BIT I/O TIMER"**

This chapter describes the functions and operations of the 16-bit I/O timer.

## **CHAPTER 13 "8/16-BIT I/O PPG"**

This chapter describes the functions and operations of the 8/16-bit PPG.

## **CHAPTER 14 "8/16-BIT UP/DOWN COUNTER/TIMER"**

This chapter describes the functions and operations of the 8/16-bit up/down counter/timer.

## **CHAPTER 15 "DTP/EXTERNAL INTERRUPT"**

This chapter describes the functions and operations of the DTP/external interrupt.

## **CHAPTER 16 "DELAYED INTERRUPT REQUESTING MODULE"**

This chapter describes the functions and operations of the delayed interrupt requesting module.

## **CHAPTER 17 "A/D CONVERTER"**

This chapter describes the functions and operations of the A/D converter.

## **CHAPTER 18 "D/A CONVERTER"**

This chapter describes the functions and operations of the D/A converter.

## **CHAPTER 19 "UART"**

This chapter describes the functions and operations of the UART.

## **CHAPTER 20 "EXTENDED SERIAL I/O INTERFACE"**

This chapter describes the functions and operations of the extended serial I/O interface.

## **CHAPTER 21 "I<sup>2</sup>C INTERFACE"**

This chapter describes the functions and operations of the I<sup>2</sup>C interface.

## **CHAPTER 22 "CHIP SELECT FUNCTION"**

This chapter describes the functions and operations of the chip select function.

## **CHAPTER 23 "CLOCK MONITOR FUNCTION"**

This chapter describes the functions and operations of the clock monitor function.

## **CHAPTER 24 "ADDRESS MATCH DETECTION FUNCTION"**

This chapter describes the functions and operation of the address match detection function.

## **CHAPTER 25 "ROM MIRROR FUNCTION SELECTION MODULE"**

This chapter describes the functions and operations of the ROM mirror function selection module.

## **CHAPTER 26 "2M-BIT FLASH MEMORY"**

This chapter describes the functions and operations of 2M-bit flash memory.

## **CHAPTER 27 "EXAMPLE OF MB90F574/A SERIAL PROGRAMMING CONNECTION"**

This chapter describes examples of serial programming connection with the AF220/AF210/AF120/AF110 flash microcomputer programmer manufactured by YDC Corporation.

## **APPENDIX**

This appendix describes an I/O map, an instructions list, and other information.



- 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 OVERVIEW .....</b>	<b>1</b>
1.1 Features .....	2
1.2 Product Lineup .....	5
1.3 Block Diagram .....	6
1.4 Pin Assignment .....	7
1.5 Package Dimensions .....	8
1.6 Pin Description .....	11
1.7 I/O Circuit Types .....	17
1.8 Notes on Handling Device .....	20
 <b>CHAPTER 2 CPU .....</b>	 <b>23</b>
2.1 Memory Space .....	24
2.2 Addressing .....	25
2.3 Allocating Many-Byte Length Data in a Memory Space .....	28
2.4 Dedicated Registers .....	30
2.4.1 Accumulator (A) .....	32
2.4.2 User Stack Pointer (USP) and System Stack Pointer (SSP) .....	34
2.4.3 Processor Status (PS) .....	36
2.4.4 Program Counter (PC) .....	38
2.4.5 Direct Page Register (DPR) .....	39
2.4.6 Bank Register .....	40
2.5 General-Purpose Registers .....	41
2.6 Prefix Codes .....	43
2.6.1 Restrictions on the Use of Prefix Instructions .....	46
2.6.2 Notes on Using "DIV A, Ri" and "DIVW A, RWi" Instructions .....	48
 <b>CHAPTER 3 INTERRUPT .....</b>	 <b>51</b>
3.1 Overview of the Interrupt .....	52
3.2 Interrupt Source .....	53
3.3 Interrupt Vector .....	55
3.4 Hardware Interrupt .....	57
3.4.1 Operation .....	60
3.4.2 Hardware Interrupt Operation Flowchart .....	63
3.4.3 Sample Use Procedure of Hardware Interrupt .....	64
3.5 Software Interrupts .....	65
3.6 Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	67
3.6.1 Interrupt Control Register (ICR) .....	70
3.6.2 Extended Intelligent I/O Service Descriptor (ISD) .....	73
3.6.3 Registers of the Extended Intelligent I/O Service Descriptor (ISD) .....	74
3.6.4 Operation .....	77
3.6.5 Procedure for Using the Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	78
3.6.6 EI <sup>2</sup> OS Execution Time .....	79
3.7 Exception .....	82

<b>CHAPTER 4</b>	<b>CLOCK AND RESET</b>	<b>83</b>
4.1	Clock Generator	84
4.2	Clock Supply Map	85
4.3	Reset Source	86
4.4	Operation after a Reset is Released	88
<b>CHAPTER 5</b>	<b>LOW-POWER CONSUMPTION CONTROL CIRCUIT</b>	<b>89</b>
5.1	Overview of the Low-Power Consumption Control Circuit	90
5.2	Block Diagram of the Low-Power Consumption Control Circuit	93
5.3	Registers of the Low-Power Consumption Control Circuit	94
5.3.1	Low-Power Consumption Mode Control Register (LPMCR)	95
5.3.2	Clock Selection Register (CKSCR)	97
5.4	Status Transition for Clock Selection	100
<b>CHAPTER 6</b>	<b>LOW-POWER CONSUMPTION MODE</b>	<b>103</b>
6.1	Low-Power Consumption Mode	104
6.1.1	Sleep Mode	107
6.1.2	Pseudo Watch Mode	108
6.1.3	Watch Mode	110
6.1.4	Stop Mode	112
6.1.5	Hardware Standby Mode	114
6.2	Status Transition in Low-Power Consumption Mode	115
6.3	Status Transition Diagram of Low-Power Consumption Mode	119
<b>CHAPTER 7</b>	<b>MEMORY ACCESS MODE</b>	<b>125</b>
7.1	Memory Access Modes	126
7.1.1	Mode Pins	127
7.1.2	Mode Data	128
7.1.3	Memory Space for Each Bus Mode	129
7.2	External Memory Access (External Bus Pin Control Circuit)	132
7.2.1	Automatic Ready Function Selection Register (ARSR)	134
7.2.2	External Address Output Control Register (HACR)	136
7.2.3	Bus Control Signal Selection Register (ECSR)	137
7.3	Operation of the External Memory Access Control Signal	140
7.3.1	Ready Function	143
7.3.2	Hold Function	145
<b>CHAPTER 8</b>	<b>I/O PORT</b>	<b>147</b>
8.1	Overview of the I/O Port	148
8.2	Registers of the I/O Port	150
8.2.1	Port Data Register (PDR)	152
8.2.2	Port Direction Register (DDR)	155
8.2.3	Output Pin Register (ODR)	157
8.2.4	Input Resistor Register (RDR)	158
8.2.5	Analog Input Enable Register (ADER)	160

<b>CHAPTER 9</b>	<b>TIMEBASE TIMER</b>	<b>161</b>
9.1	Overview of the Timebase Timer	162
9.2	Timebase Timer Control Register (TBTC)	164
9.3	Operation of the Timebase Timer	166
<b>CHAPTER 10</b>	<b>WATCHDOG TIMER</b>	<b>167</b>
10.1	Overview of the Watchdog Timer	168
10.2	Watchdog Timer Control Register (WDTTC)	170
10.3	Operation of the Watchdog Timer	172
<b>CHAPTER 11</b>	<b>WATCH TIMER</b>	<b>173</b>
11.1	Overview of the Watch Timer	174
11.2	Watch Timer Control Register (WTC)	176
11.3	Operation of the Watch Timer	178
<b>CHAPTER 12</b>	<b>16-BIT I/O TIMER</b>	<b>179</b>
12.1	Overview of the 16-Bit I/O Timer	180
12.2	Block Diagram of the 16-Bit I/O Timer	182
12.3	16-Bit Input/Output Timer Register	183
12.4	16-Bit Free Run Timer	185
12.4.1	Timer Counter Data Register (TCDT)	186
12.4.2	Timer Counter Control Status Register (TCCS)	187
12.5	Output Compare	190
12.5.1	Output Compare Register (OCCP0 to OCCP3)	192
12.5.2	Output Compare Control Status Register (OCS0 to OCS3)	193
12.6	Input Capture	196
12.6.1	Input Capture Data Register (IPCP0, IPCP1)	198
12.6.2	Input Capture Control Status Register (ICS01)	199
12.7	Operation of the 16-Bit Free Run Timer	201
12.8	Operation of the 16-Bit Output Compare	203
12.9	Operation of the 16-Bit Input Capture	206
<b>CHAPTER 13</b>	<b>8/16-BIT PPG</b>	<b>209</b>
13.1	Overview of the 8/16-Bit PPG	210
13.2	Block Diagrams of the 8/16-Bit PPG	211
13.3	Registers of the 8/16-Bit PPG	213
13.3.1	PPG0 Operating Mode Control Register (PPGC0)	214
13.3.2	PPG1 Operating Mode Control Register (PPGC1)	216
13.3.3	PPG0 and PPG1 Output Pin Control Register (PPG0E)	219
13.3.4	Reload Registers (PRL and PRLH)	221
13.4	Operation of the 8/16-Bit PPG	222
13.4.1	8/16-Bit PPG Operating Modes	223
13.4.2	8/16-Bit PPG Output Operation	224
13.4.3	Selecting the Count Clock for the 8/16-Bit PPG	226
13.4.4	Controlling the Pulse Pin Output of the 8/16-Bit PPG	227
13.4.5	Timing of Writing the Reload Registers in the 8/16-Bit PPG	228
13.4.6	8/16-Bit PPG Interrupt	229
13.4.7	Initial Value of Each Hardware Component in the 8/16-Bit PPG	230

<b>CHAPTER 14 8/16-BIT UP/DOWN COUNTER/TIMER .....</b>	<b>231</b>
14.1 Overview of the 8/16-Bit Up/Down Counter/Timer .....	232
14.2 Block Diagram of the 8/16-Bit Up/Down Counter/Timer .....	234
14.3 Registers of the 8/16-Bit Up/Down Counter/Timer .....	236
14.3.1 Up/Down Count Register Ch.0/1 (UDCR0/1) .....	237
14.3.2 Reload/Compare Register 0/1 (RCR0/1) .....	238
14.3.3 Counter Status Register 0/1 (CSR0/1) .....	239
14.3.4 Counter Control Register High Ch.0 (CCR0) .....	241
14.3.5 Counter Control Register High Ch.1 (CCR1) .....	243
14.3.6 Counter Control Register Low Ch.0/1 (CCLR0/1) .....	245
14.4 Count Mode Selection for 8/16-Bit Up/Down Counter/Timer .....	247
14.5 Reload Function and Compare Function of 8/16-Bit Up/Down Counter/Timer .....	250
14.6 Simultaneous Activation of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer .....	252
14.7 Writing 8/16-Bit Up/Down Counter/Timer Data to UDCR .....	254
<b>CHAPTER 15 DTP/EXTERNAL INTERRUPT .....</b>	<b>257</b>
15.1 Overview of the DTP/External Interrupt .....	258
15.2 Registers of the DTP/External Interrupt .....	259
15.2.1 DTP/Interrupt Enable Register (ENIR) .....	260
15.2.2 DTP/Interrupt Source Register (EIRR) .....	261
15.2.3 Request Level Setting Register (ELVR) .....	262
15.3 Operation of the DTP/External Interrupt .....	264
15.4 Notes on Using the DTP/External Interrupt .....	267
<b>CHAPTER 16 DELAYED INTERRUPT REQUESTING MODULE .....</b>	<b>269</b>
16.1 Overview of the Delayed Interrupt Requesting Module .....	270
16.2 Operation of the Delayed Interrupt Requesting Module .....	271
<b>CHAPTER 17 A/D CONVERTER .....</b>	<b>273</b>
17.1 Overview of the A/D Converter .....	274
17.2 Registers of the A/D Converter .....	276
17.2.1 Control Status Register (ADCS1, ADCS2) .....	277
17.2.2 Data Register (ADCR1, ADCR2) .....	282
17.3 Operation of the A/D Converter .....	284
17.3.1 Conversion Using EI <sup>2</sup> OS .....	286
17.3.2 Example of Activating EI <sup>2</sup> OS in the Single Mode .....	287
17.3.3 Example of Activating EI <sup>2</sup> OS in the Continuous Mode .....	289
17.3.4 Example of Activating EI <sup>2</sup> OS in the Stop Mode .....	291
17.4 Notes on Using the A/D Converter .....	293
17.5 Conversion Data Protection Function .....	294
<b>CHAPTER 18 D/A CONVERTER .....</b>	<b>297</b>
18.1 Overview of the D/A Converter .....	298
18.2 Registers of the D/A Converter .....	300
18.3 Operation of the D/A Converter .....	302

<b>CHAPTER 19</b>	<b>UART</b>	<b>303</b>
19.1	Overview of the UART	304
19.2	Block Diagram of the UART	305
19.3	Registers of the UART	306
19.3.1	Serial Mode Register (SMR)	307
19.3.2	Serial Control Register (SCR)	309
19.3.3	Serial Input Data Register (SIDR)/Serial Output Data Register (SODR)	312
19.3.4	Serial Status Register (SSR)	313
19.3.5	Communication Prescaler Control Register (CDCR)	316
19.4	UART Baud Rates	318
19.5	Operation of the UART	321
19.5.1	Asynchronous (Start-stop Synchronous) Mode	322
19.5.2	CLK Synchronous Mode	323
19.6	Flags and Interrupt Sources of the UART	325
19.6.1	Timing to Set an Interrupt and Flag of the UART	326
19.7	Applications of the UART and Precautions	329
<b>CHAPTER 20</b>	<b>EXTENDED SERIAL I/O INTERFACE</b>	<b>331</b>
20.1	Overview of the Extended Serial I/O Interface	332
20.2	Registers in the Extended Serial I/O Interface	334
20.2.1	Serial Mode Control Status Register (SMCS)	335
20.2.2	Serial Shift Data Register (SDR)	339
20.3	Operation of the Extended Serial I/O Interface	340
20.3.1	Shift Clock	341
20.3.2	Operating States of the Extended Serial I/O Interface	342
20.3.3	Operation Timings of the Extended Serial I/O Interface	344
20.3.4	Serial Data I/O Shift Timings	346
20.3.5	Interrupt Function of the Extended Serial I/O Interface	347
<b>CHAPTER 21</b>	<b>I<sup>2</sup>C INTERFACE</b>	<b>349</b>
21.1	Overview of the I <sup>2</sup> C Interface	350
21.2	Block Diagram of the I <sup>2</sup> C Interface	351
21.3	I <sup>2</sup> C Interface Registers	352
21.3.1	Bus Status Register (IBSR)	353
21.3.2	Bus Control Register (IBCR)	356
21.3.3	Clock Control Register (ICCR)	359
21.3.4	Address Register (IADR)	362
21.3.5	Data Register(IDAR)	363
21.4	Operation of the I <sup>2</sup> C Interface	364
21.4.1	Operation Flow of the I <sup>2</sup> C Interface	366
<b>CHAPTER 22</b>	<b>CHIP SELECT FUNCTION</b>	<b>369</b>
22.1	Overview of the Chip Select Function	370
22.2	Register of the Chip Select Function	371
22.3	Operation of the Chip Select Function	372
22.4	Decode Address Space of the Chip Select Function	373

<b>CHAPTER 23</b>	<b>CLOCK MONITOR FUNCTION</b>	<b>377</b>
23.1	Clock Monitor Function	378
23.2	Clock Output Enable Register (CLKR)	379
<b>CHAPTER 24</b>	<b>ADDRESS MATCH DETECTION FUNCTION</b>	<b>381</b>
24.1	Overview of the Address Match Detection Function	382
24.2	Registers of the Address Match Detection Function	383
24.2.1	Program Address Detection Registers (PADR0 and PADR1)	384
24.2.2	Program Address Detection Control Status Register (PACSR)	385
24.3	Operation of the Address Match Detection Function	386
24.4	Example of the Address Match Detection Function	387
24.5	Example and Flow of Program Patch Processing	389
<b>CHAPTER 25</b>	<b>ROM MIRROR FUNCTION SELECTION MODULE</b>	<b>391</b>
25.1	Overview of the ROM Mirror Function Selection Module	392
25.2	Register of the ROM Mirror Function Selection Module	393
<b>CHAPTER 26</b>	<b>2M-BIT FLASH MEMORY</b>	<b>395</b>
26.1	Overview of the 2M-Bit Flash Memory	396
26.2	Sector Configuration of the 2M-Bit Flash Memory	397
26.3	Flash Memory Control Status Register (FMCS)	398
26.4	Method for Activating Flash Memory Automatic Algorithm	401
26.5	Checking Automatic Algorithm Execution Status	402
26.5.1	Data Polling Flag (DQ7)	404
26.5.2	Toggle Bit Flag (DQ6)	406
26.5.3	Timing Limit Excess Flag (DQ5)	407
26.5.4	Sector Erase Timer Flag (DQ3)	408
26.6	Detailed Explanation of Flash Memory Write/Erase	409
26.6.1	Read/Reset	410
26.6.2	Data Write	411
26.6.3	Data Erase (All Chip Erase)	413
26.6.4	Data Erase (Sector Erase)	414
26.6.5	Sector Erase Temporary Stop	416
26.6.6	Sector Erase Restart	417
26.7	Example of Flash Memory Program	418
<b>CHAPTER 27</b>	<b>EXAMPLES OF MB90F574/A SERIAL PROGRAMMING CONNECTION</b>	<b>423</b>
27.1	Basic Configuration of MB90F574/A Serial Programming Connection	424
27.2	Example of Serial Programming Connection (User Power Supply Used)	427
27.3	Example of Serial Programming Connection (Power Supplied from the Programmer)	429
27.4	Example of Minimum Connection to Flash Microcomputer Programmer (User Power Supply Used)	431
27.5	Example of Minimum Connection to Flash Microcomputer Programmer (Power Supplied from the Programmer)	433

<b>APPENDIX .....</b>	<b>435</b>
APPENDIX A   I/O Map .....	436
APPENDIX B   INSTRUCTIONS .....	443
B.1   Instruction Types .....	444
B.2   Addressing .....	445
B.3   Direct Addressing .....	447
B.4   Indirect Addressing .....	453
B.5   Execution Cycle Count .....	461
B.6   Effective Address Field .....	464
B.7   How to Read the Instruction List .....	465
B.8   F2MC-16LX Instruction List .....	468
B.9   Instruction Map .....	482
<b>INDEX .....</b>	<b>505</b>





# Main changes in this edition

Page	Changes (For details, refer to main body.)
443 to 504	Changed the entire part of "APPENDIX B Instructions"

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



# CHAPTER 1    OVERVIEW

---

**This chapter describes an overview of the MB90570 series.**

---

- 1.1 "Features"
- 1.2 "Product Lineup"
- 1.3 "Block Diagram"
- 1.4 "Pin Assignment"
- 1.5 "Package Dimensions"
- 1.6 "Pin Description"
- 1.7 "I/O Circuit Types"
- 1.8 "Notes on Handling the Device"

## 1.1 Features

---

**MB90570 is a general-purpose Fujitsu 16-bit microcontroller designed for process control requiring high-speed realtime processing for welfare products.**

---

### ■ Features

In addition to inheriting the FMC series AT architecture, the MB90570 series instruction system adds high-level language support instructions, expands addressing modes, enhances multiplication and division instructions, and provides rich bit processing. The system also enables long-word data processing by mounting a 32-bit accumulator. The series contains an I<sup>2</sup>CBUS interface to enable simplified communication between devices and is suitable for auto audio and VTR systems.

○ **Minimum instruction execution time:**

62.5 ns/4 MHz, oscillation frequency multiplied by four (PLL clock multiplication system)

○ **Maximum memory space:**

16M bytes

○ **Instruction system optimized for the controller**

- Data types that can be handled: bits, bytes, words, and long words
- Standard addressing modes: 23 types
- Enhancing high-precision arithmetic operation using the 32-bit accumulator
- Enhancing signed multiplication and division instructions and RETI instructions

○ **Instruction system supporting the high-level language (C) and multitasking**

- Using the system stack pointer
- Instruction set symmetric and barrel shift instructions

○ **Program patch function (For two address pointers)**

○ **Execution speed improvement:**

4-byte queue

○ **Powerful interrupt function**

- Programmable setting of eight priority levels
- External interrupt input: 8 channels

○ **Data transfer function**

- Intelligent I/O service
- Maximum 16 channels
- DTP request input: 8 channels (bidirectional edge can be set for two. No level detection can be set.)

- **Internal ROM size and ROM type**
  - MASK-ROM: 128 KB/256 KB
  - FLASH-ROM: 256 KB
  - Internal RAM sizes
    - FLASH: 10 KB
    - EVA: Max. 10 KB
    - MASK-ROM: 6 KB/10 KB
- **General-purpose ports**
  - Up to 97 ports
  - Input pull-up resistor setting possible: 24 (ports 0, 1, 6)
  - Open-drain setting possible: 8 (with diode clamps for port 4)
- **I<sup>2</sup>CBUS interface:**
  - 1 channel
- **Chip select output:**
  - 8 (active level setting possible)
- **A/D converter (RC successive approximation type)**
  - Resolution: 8 or 10 bits: 8 channels (multiplexing)
  - Conversion time: 26.3  $\mu$ s
- **D/A converter (R-2R system)**
  - Resolution (8 bits): 2 channels (independent)
  - Setup time: 12.5  $\mu$ s
- **UART (full-duplex double buffer system):**
  - 2 channels
- **Extended I/O serial interface:**
  - 3 channels
- **8/16-bit up/down counter:**
  - 1 channel (8 bits x 2 channels)
- **8/16-bit PPG timer:**
  - 1 channel (8 bits x 2 channels)
- **I/O timer:**
  - 1 channel
    - 16-bit free-run timer
    - 16-bit input capture x 2 channels
    - 16-bit output compare x 4 channels

## CHAPTER 1 OVERVIEW

- **Clock output function**
- **Watch timer:**
  - 1 channel
- **18-bit timebase and watchdog timer (18 bits)**
- **Low-power consumption mode**
  - Sleep, stop, hardware standby, CPU intermittent operation mode function
- **Package**
  - LQFP-120
    - FPT-120P-M21 (lead pitch 0.5 mm): (None for MB90573 and MB90574)
    - FPT-120P-M05 (lead pitch 0.4 mm): (None for MB90574C)
  - QFP-120
    - FPT-120P-M13 (lead pitch 0.5 mm)
- **CMOS technology**

## 1.2 Product Lineup

**Table 1.2-1 "Product Lineup (MB90570 Series)" lists the products available. Products whose product numbers end with the letter A or C are different to those having product numbers without the letter A or C in that a return from standby mode is possible using an interrupt generated by a ch0 to ch1 edge request when a DTP/external interrupt occurs.**

### ■ Product Lineup

**Table 1.2-1 Product Lineup (MB90570 Series)**

	<b>MB90V570</b>	<b>MB90V570A</b>	<b>MB90F574/A</b>	<b>MB90573</b>	<b>MB90574</b>	<b>MB90574C</b>
ROM size	-	-	256 Kbyte	128 Kbyte	256 Kbyte	256 Kbyte
RAM size	10 Kbyte	10 Kbyte	10 Kbyte	6 Kbyte	10 Kbyte	10 Kbyte
Others	EVA products	EVA products	FLASH products	MASK products	MASK products	MASK products
Exclusive power supply for the emulator*	None	None	-	-	-	-

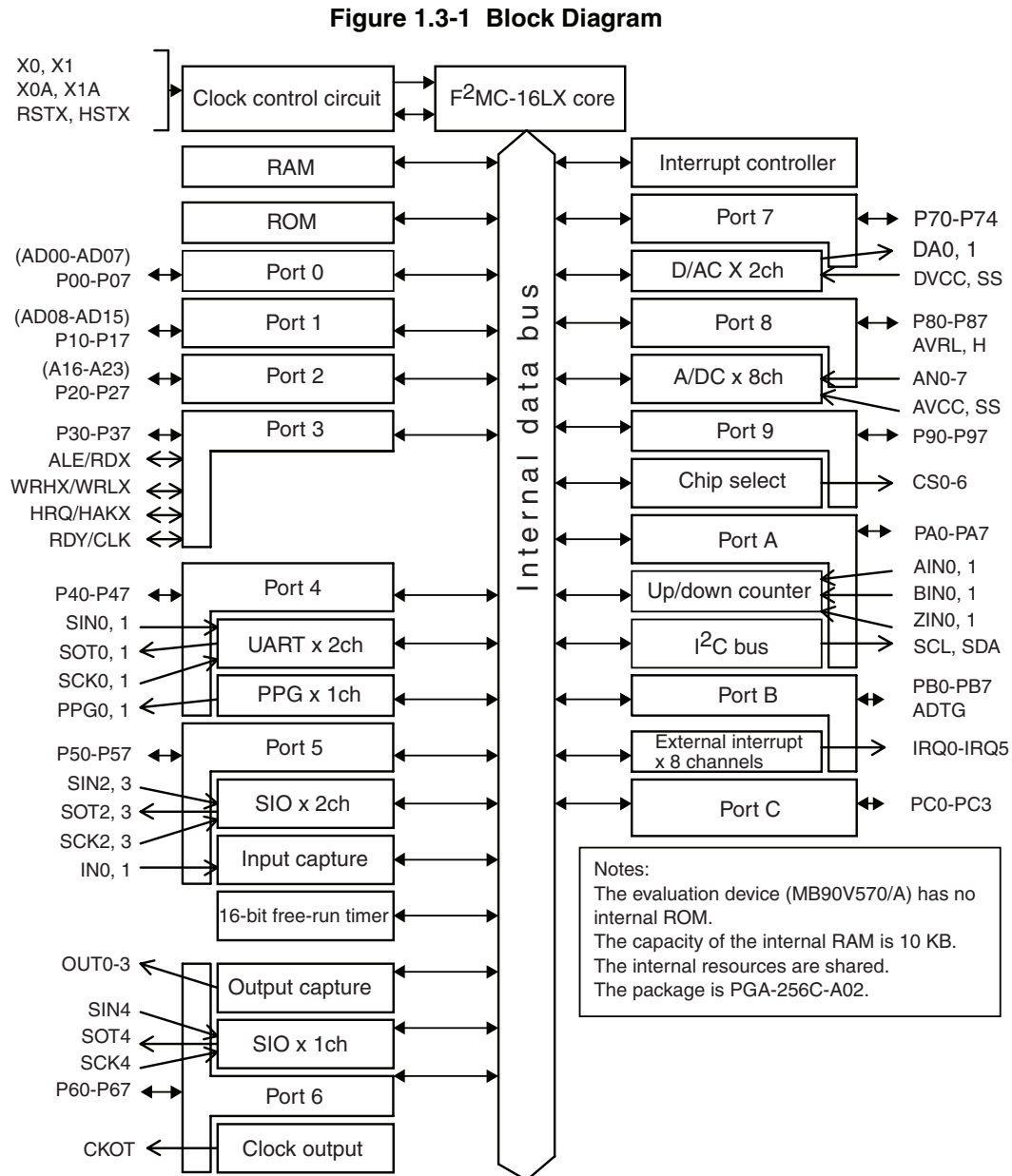
\* : Select this dip switch S2 setting when using the evaluation pod MB2145-507. For details, see the "MB2145-507 Hardware Manual (exclusive power supply pin for emulator)".



## 1.3 Block Diagram

Figure 1.3-1 "Block Diagram" shows the block diagram.

### ■ Block Diagram



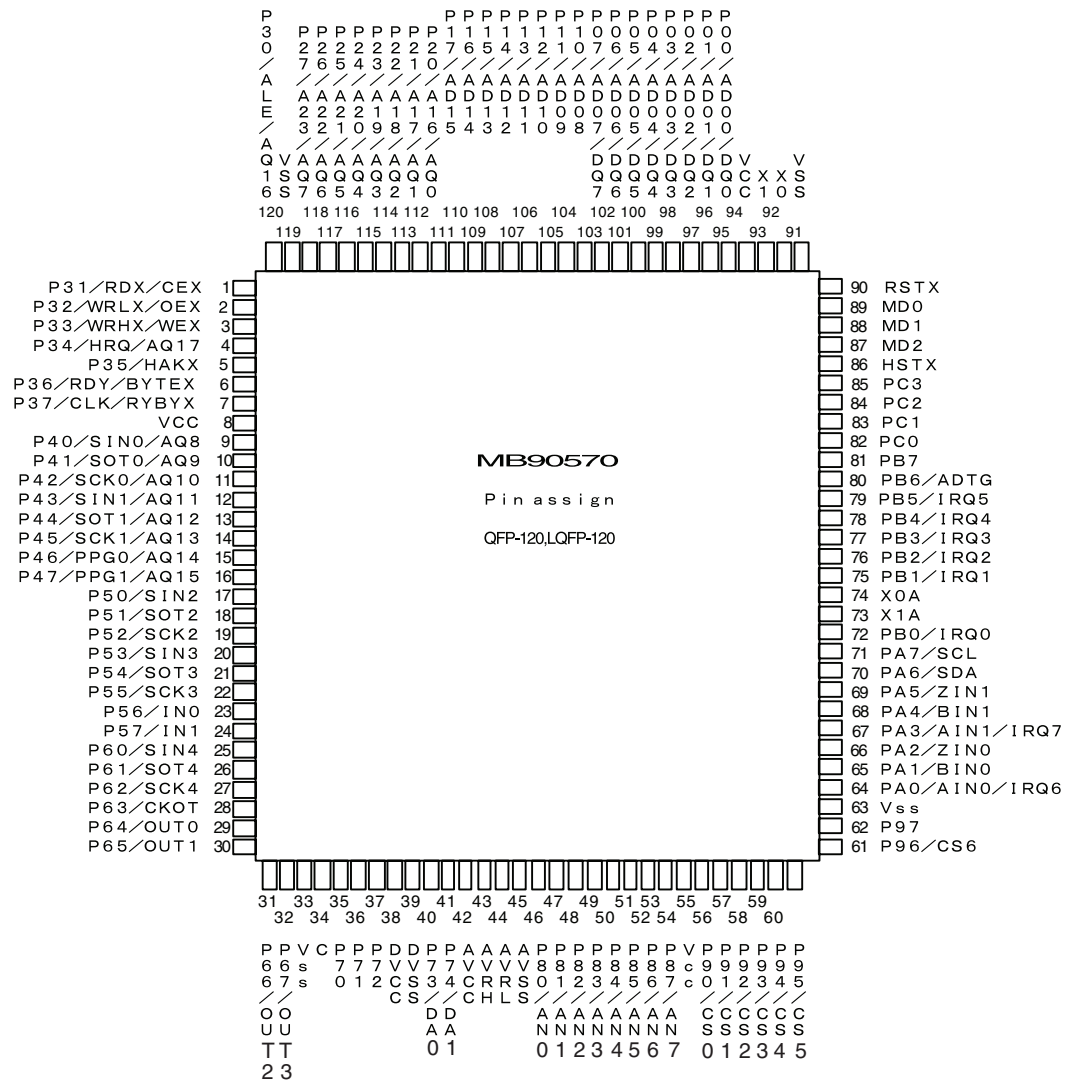
- P00-P07 (8): With the input pull-up resistor setting register
- P10-P17 (8): With the input pull-up resistor setting register
- P60-P67 (8): With the input pull-up resistor setting register
- P40-P47 (8): With the open-drain setting register

## 1.4 Pin Assignment

Figure 1.4-1 "Pin Assignment" shows the pin assignment.

### ■ Pin Assignment

Figure 1.4-1 Pin Assignment

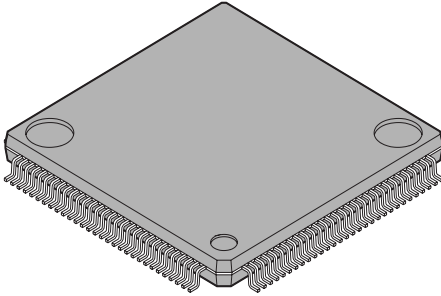


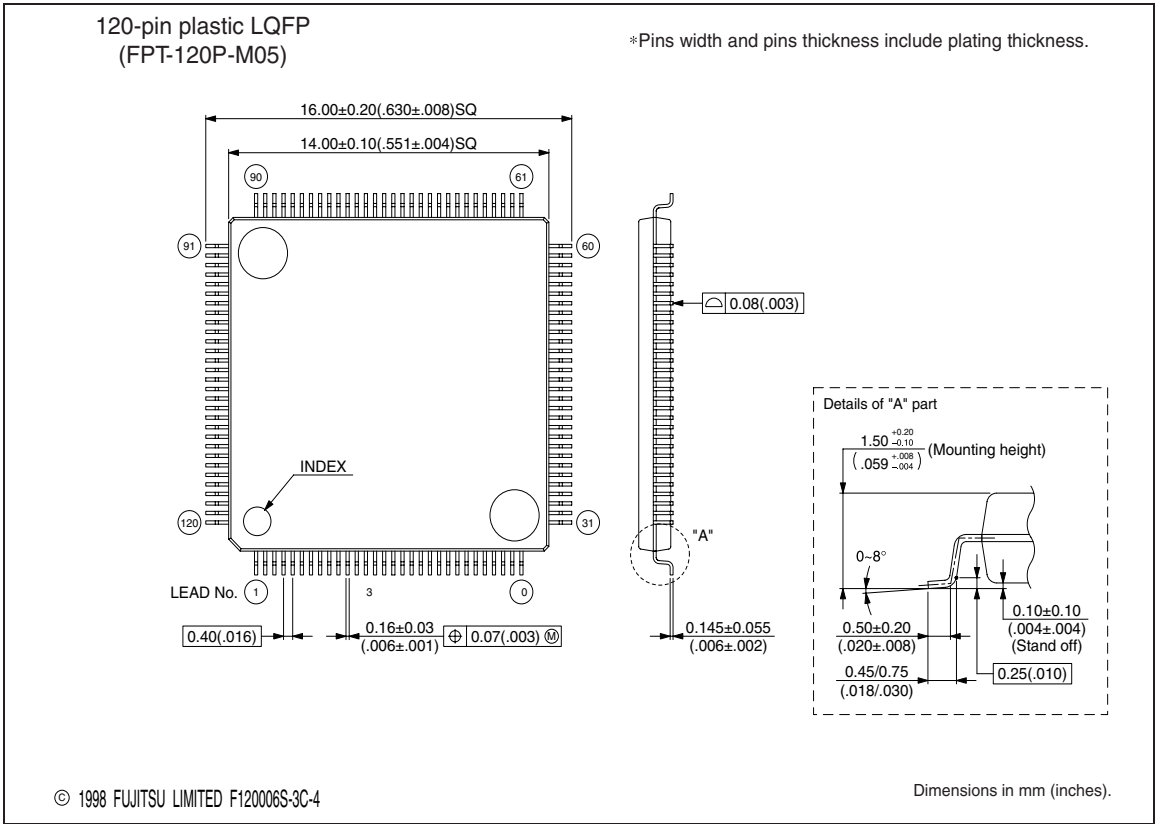
# 1.5 Package Dimensions

The MB90570 series supports three types of packages.

■ Package Dimensions of FPT-120P-M05

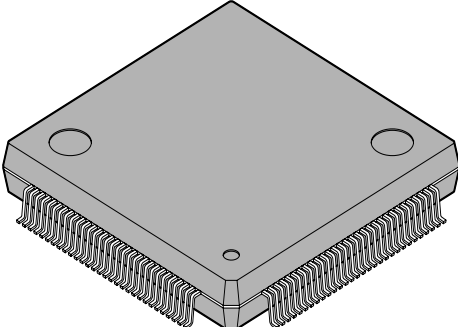
Figure 1.5-1 External Dimensions of FPT-120P-M05

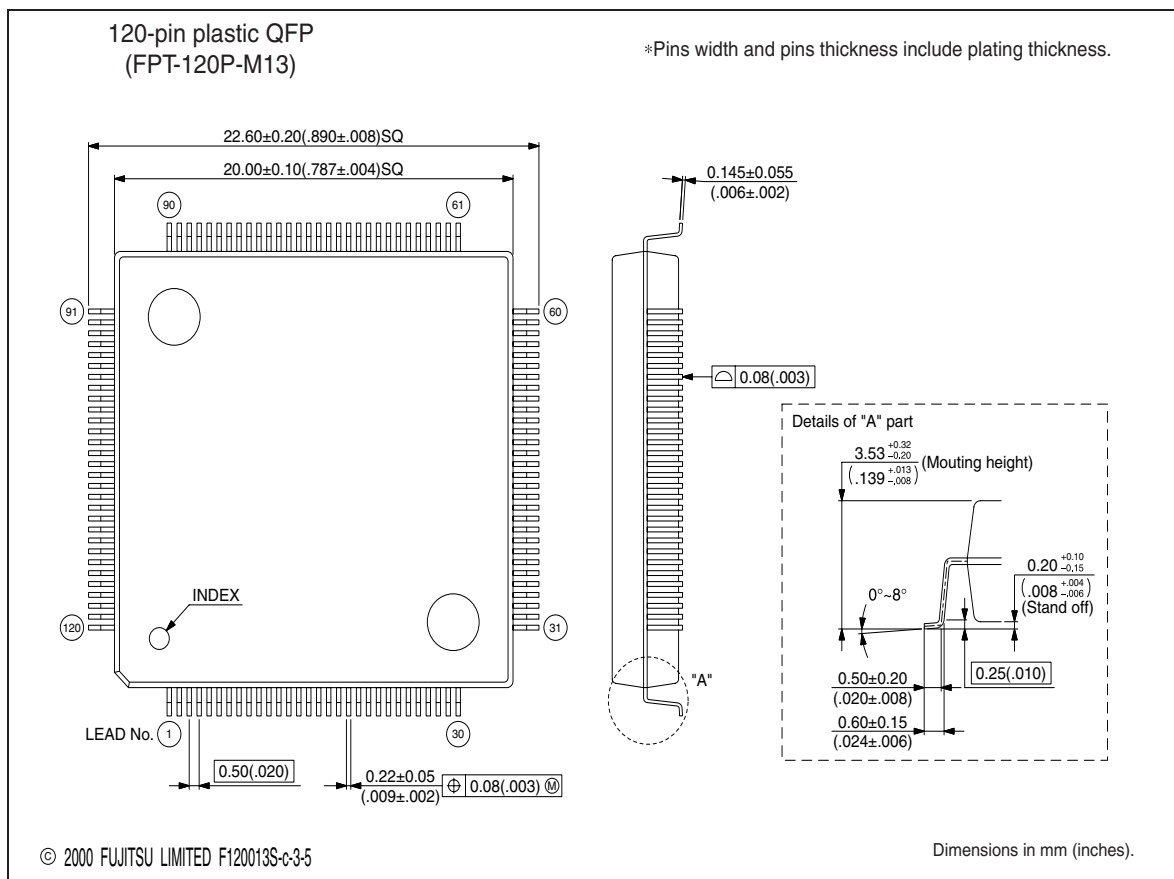
<div>120-pin plastic LQFP</div>  <div>(FPT-120P-M05)</div>	Lead pitch	0.40 mm
	Package width × package length	14.0 × 14.0 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	1.70 mm MAX
	Weight	0.62 g



# ■ Package Dimensions of FPT-120P-M13

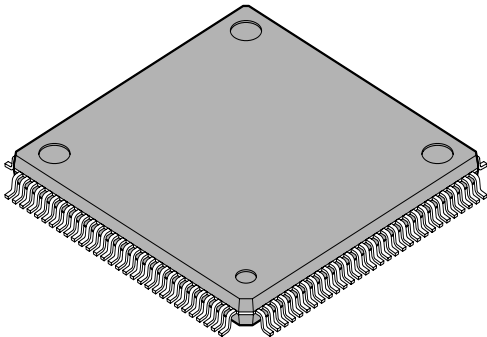
**Figure 1.5-2 External Dimensions of FTP-120P-M13**

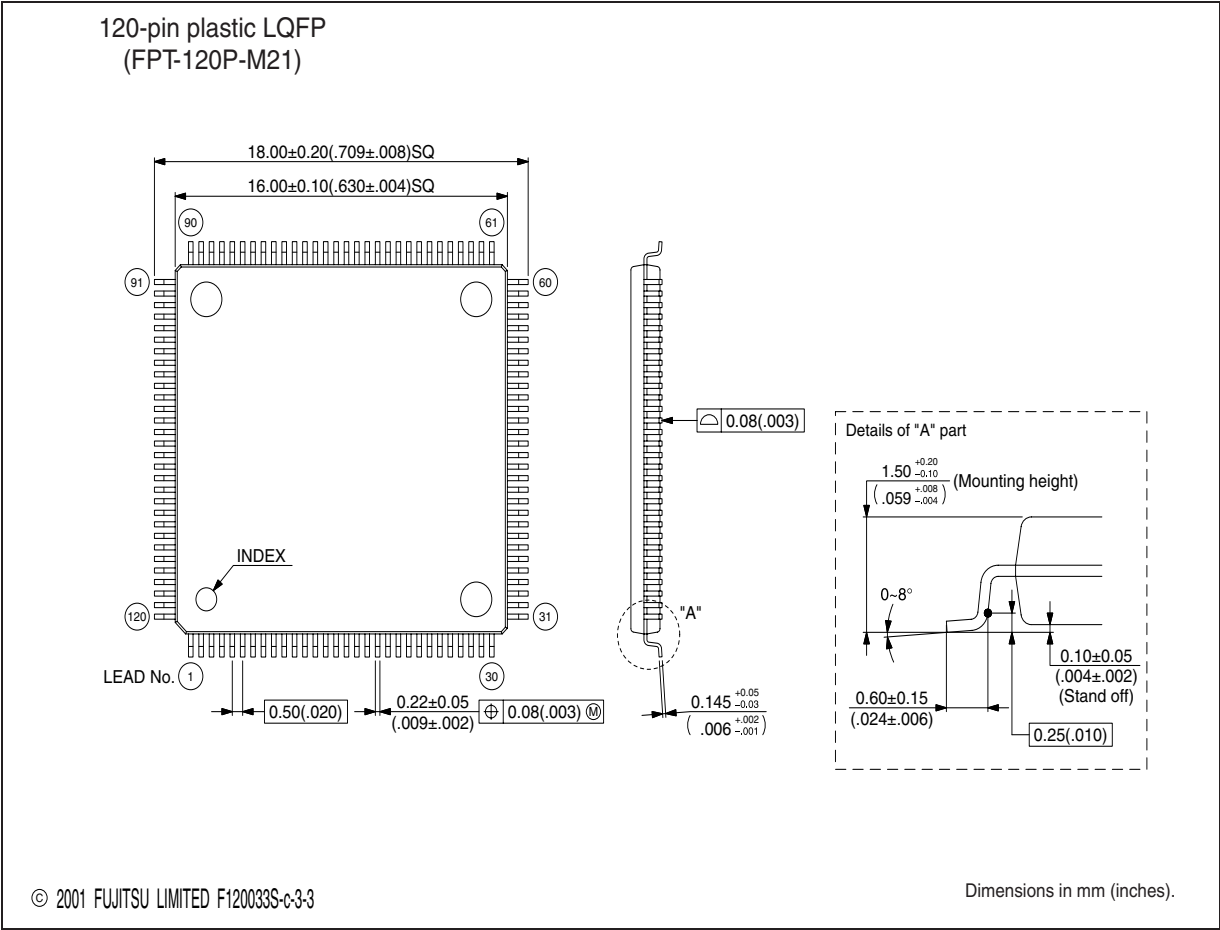
<div>120-pin plastic QFP</div>  <div>(FPT-120P-M13)</div>	Lead pitch	0.50 mm
	Package width × package length	20.0 × 20.0 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	3.85 mm MAX
	Weight	2.58g



■ Package Dimensions of FPT-120P-M21

Figure 1.5-3 External Dimensions of FPT-120P-M21

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



## 1.6 Pin Description

**Table 1.6-1 "Pin Description" provides a list of pin function explanations.**

### ■ Pin Description

**Table 1.6-1 Pin Description**

Pin No.	Pin name	Circuit type	Explanation of function
92/93	X0/X1	A	High-speed oscillation input pin
74/73	X0A/X1A	B	Low-speed oscillation input pin
90	RSTX	C	Reset input pin
86	HSTX	C	Hardware standby input pin
95 to 102	P00 to 07	D	General-purpose output port. At input setting, this port can be set with the pull-up resistor setting register (RDR0). At output setting, this port is invalidated.
	AD00 to 07		In external bus mode, function as lower address output and lower data I/O pins.
103 to 110	P10 to 17	D	General-purpose I/O port. At input setting, this port can be set with the pull-up resistor setting register (RDR1). At output setting, this port is invalidated.
	AD08 to 15		In external bus mode, function as the middle address output and higher data I/O pins.
111 to 118	P20 to 27	E	General-purpose I/O port
	A16 to 23		In external bus mode, function as the higher address output pins.
120	P30	E	General-purpose I/O port
	ALE		In external bus mode, used as an address latch enable signal output pin.
1	P31	E	General-purpose I/O port
	RDX		In external bus mode, used as a read strobe signal output pin.
2	P32	E	General-purpose I/O port
	WRLX		In external bus mode, used as the pin for outputting the data bus lower 8 bits write strobe signal.
3	P33	E	General-purpose I/O port
	WRHX		In external bus mode, used as the pin for outputting the data bus higher 8 bits write strobe signal.

## CHAPTER 1 OVERVIEW

**Table 1.6-1 Pin Description (Continued)**

Pin No.	Pin name	Circuit type	Explanation of function
4	P34	E	General-purpose I/O port
	HRQ		In external bus mode, used as a hold request signal input pin.
5	P35	E	General-purpose I/O port
	HAKX		In external bus mode, used as a hold acknowledge signal output pin.
6	P36	E	General-purpose I/O port
	RDY		In external bus mode, used as a ready signal input pin.
7	P37	E	General-purpose I/O port
	CLK		In external bus mode, used as a clock (CLK) signal output pin.
9	P40	F	General-purpose I/O port that can be set in the open-drain by the ODR4 register.
	SIN0		UART Ch.0 serial data input pin. Because this input pin can be used whenever UART Ch.0 is performing an input operation, the output function must be used for intentional use only. When using this pin together with other function outputs, place in a status in which output is disabled during a SIN operation.
10	P41	F	General-purpose I/O port. This port can be set in the open-drain with the ODR4 register.
	SOT0		UART Ch.0 serial data output pin. This pin is validated when UART Ch.0 allows the data output specification.
11	P42	F	General-purpose I/O port. This port can be set in the open-drain with the ODR4 register.
	SCK0		UART Ch.0 serial clock I/O pin. This pin is validated when UART Ch.0 allows the clock output specification.
12	P43	F	General-purpose I/O port. This port can be set in the open-drain with the ODR4 register.
	SIN1		UART Ch.1 serial data input pin. Because this input can be used whenever UART Ch.1 is performing an input operation, use the input function for intentional use only. To use this pin together with other function outputs, place in a status in which the output is disabled during a SIN operation.
13	P44	F	General-purpose I/O port. This port can be set in the open-drain with the ODR4 register.
	SOT1		UART Ch.1 serial data output pin that is validated when UART Ch.1 allows the data output specification.

Table 1.6-1 Pin Description (Continued)

Pin No.	Pin name	Circuit type	Explanation of function
14	P45	F	General-purpose I/O port. This port can be set in the open-drain with the ODR4 register.
	SCK1		UART Ch.1 serial clock I/O pin that is validated when UART Ch.1 allows the clock output specification.
15 to 16	P46 to 47	F	General-purpose I/O port. This port can be set in the open-drain with the ODR4 register.
	PPG0 to 1		PPG0 and PPG1 output pins that are valid when the output specifications of PPG0 and PPG1 are allowed.
17	P50	E	General-purpose I/O port
	SIN2		I/O serial Ch.0 data input pin. Because this input can be used whenever serial data is input, use the output for intentional use only.
18	P51	E	General-purpose I/O port
	SOT2		I/O serial Ch.0 data output pin that is valid when serial data output of serial Ch.0 is allowed.
19	P52	E	General-purpose I/O port
	SCK2		Serial clock I/O pin of I/O serial Ch.0 that is validated when the serial clock output of serial Ch.0 is allowed.
20	P53	E	General-purpose I/O port
	SIN3		I/O serial Ch.1 data input pin. Because this data input pin can be used whenever serial data is input, use the output for intentional use only.
21	P54	E	General-purpose I/O port
	SOT3		Data output pin of I/O serial Ch.1 that is validated when the serial data output of serial Ch.1 is allowed.
22	P55	E	General-purpose I/O port
	SCK3		Serial clock I/O pin of I/O serial Ch.1 that is validated when the serial clock output of serial Ch.1 is allowed.
23 to 24	P56 to 57	E	General-purpose I/O port
	IN0 to 1		Input capture Ch.0/1 trigger input pin. Because this trigger input pin function can be used whenever input capture Ch.0/1 is performing an input operation. Allow the output for intentional use only.
25	P60	D	General-purpose I/O port. At input setting, this port can be set with the pull-up resistor setting register (RDR6). At output setting, this port is invalid.
	SIN4		I/O serial Ch.2 data input pin. Because this data input pin can be used whenever serial data is input, use the output for intentional use only.



## CHAPTER 1 OVERVIEW

**Table 1.6-1 Pin Description (Continued)**

Pin No.	Pin name	Circuit type	Explanation of function
26	P61	D	General-purpose I/O port. At input setting, this port can be set with the pull-up resistor setting register (RDR6). At output setting, this port is invalid.
	SOT4		Data output pin of I/O serial Ch.2 that is validated when the serial data output of serial Ch.2 is allowed.
27	P62	D	General-purpose I/O port. At input setting, this port can be set with the pull-up resistor setting register (RDR6). At output setting, this port is invalid.
	SCK4		Serial clock I/O pin of I/O serial Ch.2 that is validated when the serial clock output of serial Ch.2 is allowed.
28	P63	D	General-purpose I/O port. At input setting, this port can be set with the pull-up resistor setting register (RDR6). At output setting, this port is invalid.
	CKOT		Clock monitor function output pin that is validated when the clock monitor output is allowed.
29 to 32	P64 to 67	D	General-purpose I/O port. At input setting, this port can be set with the pull-up resistor setting register (RDR6). At output setting, this port is invalid.
	OUT0 to 3		Output compare Ch.0-3 event output pins are validated when each channel attains output enable status.
34	C	G	Power supply stabilization capacity pin. Connect an external 0.1 $\mu$ ceramic capacitor. This connection is unnecessary for FLASH products (MB90F574/A) and for MB90574C.
35 to 37	P70 to 72	E	General-purpose I/O port
38	DV <sub>CC</sub>	H	D/A converter Vref input pin. Do not exceed V <sub>CC</sub> .
39	DV <sub>SS</sub>	H	D/A converter GND power supply pin. Set the same V <sub>SS</sub> voltage.
40 to 41	P73 to 74	I	General-purpose I/O port
	DA0 to 1		Analog signal output pins of D/A converter Ch.0,1
42	AV <sub>CC</sub>	H	V <sub>CC</sub> power supply input pin of the analog macro (D/A, A/D, and others)
43	AVRH	J	A/D converter Vref+ input pin. Do not exceed V <sub>CC</sub> .
44	AVRL	H	A/D converter Vref- input pin. Must not be equal to or less than V <sub>SS</sub> .
45	AV <sub>SS</sub>	H	V <sub>SS</sub> power supply input pin of the analog macro (D/A, A/D, and others)
46 to 53	P80 to 87	K	General-purpose I/O port
	AN0 to 7		A/D converter analog input pin is a function that is validated when the analog input specification is allowed.

Table 1.6-1 Pin Description (Continued)

Pin No.	Pin name	Circuit type	Explanation of function
55 to 62	P90 to 97	E	General-purpose I/O port
	CS0 to 7		Chip select output pin is a function that is validated when the chip select output is allowed.
64	PA0	E	General-purpose I/O port
	AIN0		Can be used as the count clock A input of 8/16-bit up-down counter Ch.0.
	IRQ6		Can be used as interrupt request input Ch.6.
65	PA1	E	General-purpose I/O port
	BIN0		Can be used as the count clock B input of 8/16-bit up-down counter Ch.0.
66	PA2	E	General-purpose I/O port
	ZIN0		Can be used as the control clock Z input of 8/16-bit up-down counter Ch.0.
67	PA3	E	General-purpose I/O port
	AIN1		Can be used as the count clock A input of 8/16-bit up-down counter Ch.1.
	IRQ7		Can be used as interrupt request input Ch.7.
68	PA4	E	General-purpose I/O port
	BIN1		Can be used as the count clock B input of 8/16-bit up-down counter Ch.1.
69	PA5	E	General-purpose I/O port
	ZIN1		Can be used as the control clock Z input of 8/16-bit up-down counter Ch.1.
70	PA6	L	General-purpose I/O port
	SDA		This is the data I/O pin of the I <sup>2</sup> C interface. Using this pin is effective when the operation of the I <sup>2</sup> C interface is allowed. Set the port output to input setting (DDRA: bit 6 = 0) while the I <sup>2</sup> C interface is operating.
71	PA7	L	General-purpose I/O port
	SCL		This is the clock I/O pin of the I <sup>2</sup> C interface. Using this pin is effective when the operation of the I <sup>2</sup> C interface is allowed. Set the port output to input setting (DDRA: bit 7 = 0) while the I <sup>2</sup> C interface is operating.

## CHAPTER 1 OVERVIEW

**Table 1.6-1 Pin Description (Continued)**

Pin No.	Pin name	Circuit type	Explanation of function
72, 75 to 79	PB0 to 5	E	General-purpose I/O port
	IRQ0 to 5		Because external interrupt input pins IRQ0 and IRQ1 can detect both edges but cannot detect interrupt sources based on levels, these pins cannot be used for the return from STOP in MB90V570, MB90F574, MB90573, and MB90574. However, for MB90V570A, MB90F574A, and MB90574C, a return from STOP is possible by the edge detection of IRQ0 and IRQ1.
80	PB6	E	General-purpose I/O port
	ADTG		Because the A/D converter external trigger input pin can be used whenever the A/D converter is performing an input operation, in cases other than intentional use, place this pin in output stop status.
81	PB7	E	General-purpose I/O port
82 to 85	PC0 to 3	E	General-purpose I/O port
88, 89	MD1, MD0	C	Operating mode selection input pin must be connected directly (direct coupling) to $V_{CC}$ or $V_{SS}$ .
87	MD2	C	Operating mode selection input pin must be connected directly (direct coupling) to $V_{CC}$ or $V_{SS}$ .
8, 54, 94	$V_{CC}$	-	Power supply (5 V) input pin
33, 63, 91, 119	$V_{SS}$	-	Power supply (0 V) input pin

## 1.7 I/O Circuit Types

Table 1.7-1 "I/O Circuit Types" lists the I/O circuit types.

### ■ I/O Circuit Types

Table 1.7-1 I/O Circuit Types

Classifi- cation	Circuit	Remarks
A		Oscillation circuit <ul style="list-style-type: none"> <li>High-speed oscillation feedback resistor = approx. 1 M<math>\Omega</math></li> </ul>
B		Oscillation circuit <ul style="list-style-type: none"> <li>Low-speed oscillation feedback resistor = approx. 1 M<math>\Omega</math></li> </ul>
C		Hysteresis input pin <ul style="list-style-type: none"> <li>Resistor value : approx. 50 K<math>\Omega</math> (TYP)</li> </ul>
D		CMOS hysteresis input pin with input pull-up control <ul style="list-style-type: none"> <li>CMOS level output</li> <li>CMOS hysteresis inputs (With the standby-time input shutdown function)</li> <li>Pull-up resistor value: Approx. 50 K<math>\Omega</math> (TYP), <math>I_{OL} = 4</math> mA</li> </ul>

Table 1.7-1 I/O Circuit Types (Continued)

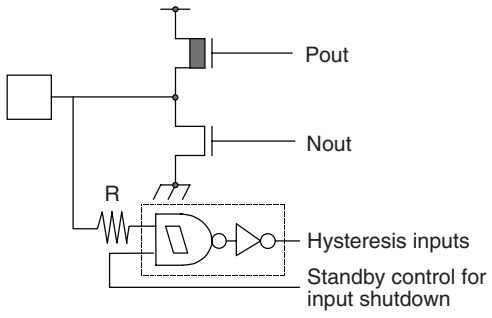
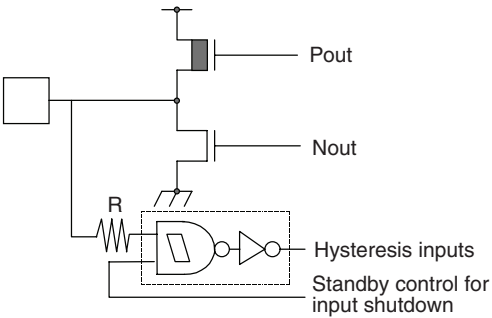
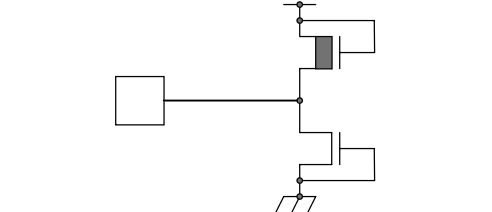
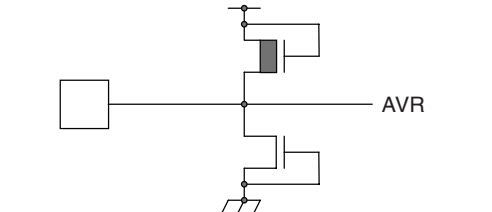
Classifi- cation	Circuit	Remarks
E		CMOS hysteresis I/O pin • CMOS level output • CMOS hysteresis inputs (With the standby-time input shutdown function), $I_{OL} = 4\text{ mA}$
F		CMOS hysteresis I/O pin • CMOS level output • CMOS hysteresis inputs (With the standby-time input shutdown function), $I_{OL} = 10\text{ mA}$ (Heavy-current port)
G		C pin output (Capacitor connection pin) N.C. pin for MB90F574/A
H		Analog power supply input protection circuit

Table 1.7-1 I/O Circuit Types (Continued)

Classifi- cation	Circuit	Remarks
I		<ul style="list-style-type: none"> <li>CMOS hysteresis I/O</li> <li>Pin for both analog and CMOS outputs (No CMOS output at analog output, Analog output given priority: DAE = 1)</li> <li>With the input shutdown standby control function, <math>I_{OL} = 4</math> mA</li> </ul>
J		<ul style="list-style-type: none"> <li>A/D converter ref+ (AVRH) power supply input pin, With the power supply protection circuit</li> </ul>
K		<ul style="list-style-type: none"> <li>Pin for both CMOS hysteresis and analog inputs</li> <li>CMOS output</li> <li>With the input shutdown function for the input shutdown standby</li> </ul>
L		<p>Hysteresis inputs N-ch open-drain output With the input shutdown standby control function, <math>I_{OL} = 4</math> mA</p>

## 1.8 Notes on Handling Device

---

Note in particular the following items when handling devices.

- Strict observance of the maximum rated voltage (Prevention of latchup)
  - Stabilization of power supply voltage
  - When the power is turned on
  - Handling of unused pins
  - Handling for the power supply pin of the A/D converter
  - When using the external clock
  - Power supply pin
  - Sequence of voltage application/cut-off to the power supply pins/analog input pins of the A/D converter
  - Notes on using "DIV A, Ri" and "DIVW A, RWi" instructions
  - If subclock mode is not used
  - If the output from ports 0 and 1 is indefinite
  - When REALOS is used
- 

### ■ Notes on Handling Device

#### ○ Strict observance of the maximum rated voltage (Prevention of latchup)

- In CMOS IC, a latchup phenomenon may occur if a voltage higher than  $V_{CC}$  or that lower than  $V_{SS}$  is applied to input pins or output pins, other than medium- to high-voltage, or a voltage exceeding the rated voltage is applied between  $V_{CC}$  and  $V_{SS}$ . If a latchup phenomenon occurs, the power supply current may increase rapidly, leading to thermal damage of circuit elements. Therefore, care must be taken, when using devices, so that the absolute maximum rating is not exceeded.
- When turning on or turning off the analog power supply, care must be taken to ensure that the analog power supply voltages ( $AV_{CC}$ ,  $AVRH$ ,  $DV_{CC}$ ) and analog input voltage do not exceed the digital power supply voltage ( $V_{CC}$ ).

#### ○ Stabilization of power supply voltage

If the power supply voltage varies acutely even within the operation assurance range of the  $V_{CC}$  power supply voltage, a malfunction may occur. The  $V_{CC}$  power supply voltage must therefore be stabilized.

As stabilization guidelines, stabilize the power supply voltage so that  $V_{CC}$  ripple fluctuations (peak to peak value) in the commercial frequencies (50 to 60 Hz) fall within 10% of the standard  $V_{CC}$  power supply voltage and the transient fluctuation rate becomes 0.1V/ms or less in instantaneous fluctuation for power supply switching.

#### ○ When the power is turned on

When turning on the power, secure 50ms (between 0.2V and 2.7V) or more for the rising time of the power supply voltage ( $V_{CC}$ ) to prevent the malfunctioning of the built-in step-down circuits.

### ○ Handling of unused pins

Leaving unused input pins open may lead to malfunction or permanent damage due to latchup. Therefore, take steps such as pull-up or pull-down via resistance of at least  $2k\Omega$ .

If there is any unused I/O pin, open the pin after setting it to the output status or take the same steps as those for the input pins after setting the pin to the input status.

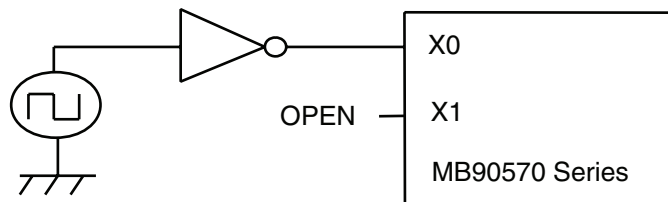
### ○ Handling for the power supply pin of the A/D converter

Even when the A/D converter is not used, connect it to ensure  $AV_{CC} = V_{CC}$ ,  $AV_{SS} = AVRH = AVRL = V_{SS}$ .

### ○ When using the external clock

When using the external clock, an oscillation stabilization time is required when releasing power-on reset, subclock mode, and stop mode. When using the external clock as shown in Figure 1.8-1 "Example of Using the External Clock", activate the X0 pin only and set the X1 pin to open.

**Figure 1.8-1 Example of Using the External Clock**



### ○ Power supply pin

- If there are multiple  $V_{CC}$  and  $V_{SS}$  pins, from the point of view of device design, pins to be of the same potential are connected the inside of the device to prevent such malfunctioning as latchup. To reduce unnecessary radiation, prevent malfunctioning of the strobe signal due to the rise of ground level, and observe the standard for total output current, be sure to connect the  $V_{CC}$  and  $V_{SS}$  pins to the power supply and ground externally.
- Connect  $V_{CC}$  and  $V_{SS}$  to the device from the current supply source at a low impedance.
- As a measure against power supply noise, connect a capacitor of about  $0.1\mu F$  as a bypass capacitor between  $V_{CC}$  and  $V_{SS}$  in the vicinity of  $V_{CC}$  and  $V_{SS}$  pins of the device.

### ○ Sequence of voltage application/cut-off to the power supply pins/analog input pins of the A/D converter

- Be sure to apply voltage to the power supplies ( $AV_{CC}$ ,  $AVRH$ , and  $AVRL$ ) of the A/D converter and the analog input pins ( $AN0$  to  $AN7$ ), after applying voltage to the digital power supply pins ( $V_{CC}$ ).
- To turn off the power, first turn off the A/D converter power and the analog input and then turn off the digital power. In this case, carry out voltage applications and power-off so that  $AVRH$  do not exceed  $AV_{CC}$ . If input ports are used by the pins which also serve as analog inputs, the input voltage do not exceed  $AV_{CC}$ . (Voltage can be applied to the analog power supply and digital power supply simultaneously or both types of power supplies can be turned off simultaneously)



### ○ Notes on using "DIV A, Ri" and "DIVW A, RWi" instructions

Use the signed multiplication/division instructions "DIV A,Ri" and "DIVW A,RWi" after setting the value of the corresponding bank register (DTB, ADB, USB, and SSB) to "00<sub>H</sub>".

If the value of the corresponding bank register (DTB, ADB, USB, and SSB) is set to a value other than "00<sub>H</sub>", the remainder obtained as a result of instruction execution is not stored in the register of the instruction operands.

For details, see Section 2.6.2 "Notes on Using DIV A, Ri and DIVW A, Rwi Instructions"

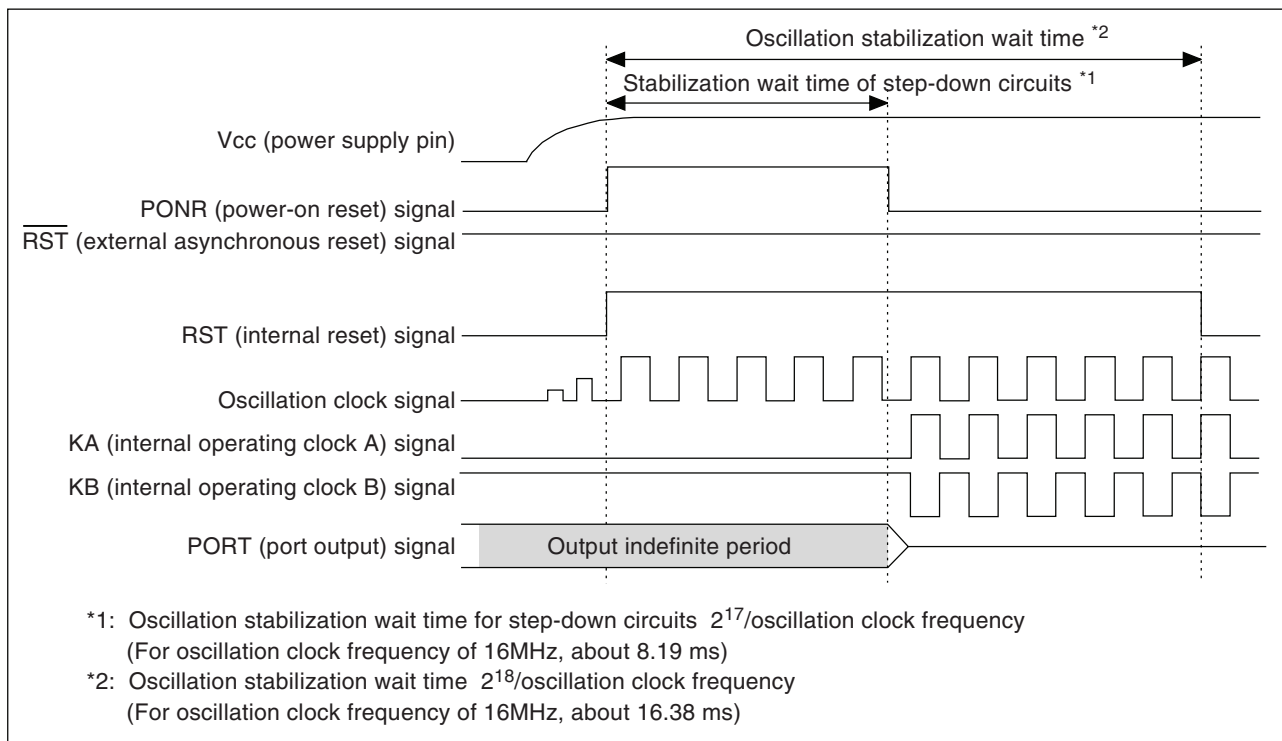
### ○ If subclock mode is not used

Even if subclock mode is not used, connect an oscillator to the X0A and X1A pins.

### ○ If the output from ports 0 and 1 is indefinite

After the power is turned on, indefinite is output from ports 0 and 1 in the oscillation stabilization wait time (during power-on reset) of the step-down circuit. Note that the timing of output is given as shown in Figure 1.8-2 "Chart of timing in which output from the ports 0 and 1 is indefinite". Still, if the type does not contain any step-down circuit, indefinite is not output because there is no oscillation stabilization wait time for the step-down circuit.

**Figure 1.8-2 Chart of timing in which output from the ports 0 and 1 is indefinite**



### ○ When REALOS is used

When REALOS is used, the extended intelligent I/O service (EI<sup>2</sup>OS) cannot be used.

# CHAPTER 2 CPU

---

**This chapter describes the CPU functions and operations.**

---

2.1 "Memory Space"

2.2 "Addressing"

2.3 "Allocating Many-Byte Length Data in a Memory Space"

2.4 "Dedicated Registers"

2.5 "General-Purpose Registers"

2.6 "Prefix Codes"

## 2.1 Memory Space

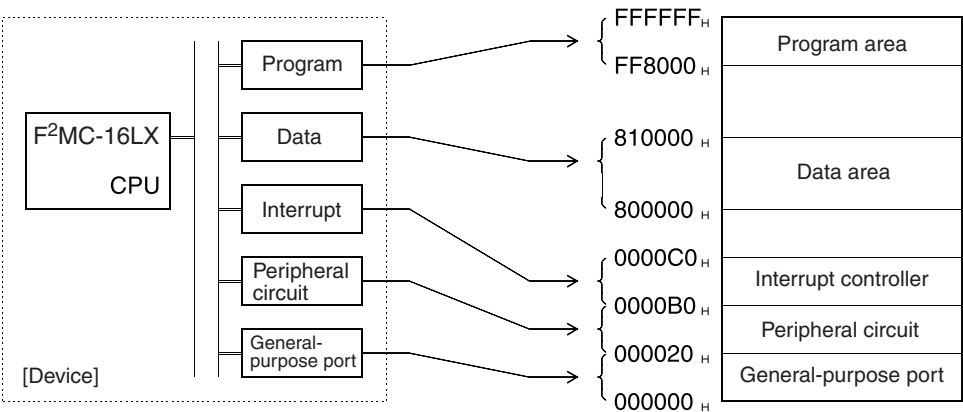
The F<sup>2</sup>MC-16LX CPU core is a 16-bit CPU designed for applications requiring high-speed real-time processing for products related to the general public (non-industrial) and products installed in vehicles. The F<sup>2</sup>MC-16LX instruction set is designed for controller applications and allows high-speed and high-efficiency processing of various types of control.

The internal 32-bit accumulator enables the F<sup>2</sup>MC-16LX to perform 32-bit data processing in addition to 16-bit data processing. The maximum amount of memory is 16M bytes (expandable), and it can be accessed by the linear or bank method. The instruction system, based on the A-T architecture of the F<sup>2</sup>MC-8L, has been enhanced by adding high-level language support instructions, expanding the addressing modes, enhancing the multiplication and division instructions, and providing rich bit processing.

■ Memory Space

All data items, programs, and I/O operations managed by F<sup>2</sup>MC-16LX CPU are allocated in the 16M-byte memory space of the F<sup>2</sup>MC-16LX CPU. Specifying these addresses with a 24-bit address bus enables the CPU to access each resource.

Figure 2.1-1 Example of the Relationship between the F<sup>2</sup>MC-16LX System and Memory Map



## 2.2 Addressing

The following two methods are used to specify the addresses of F<sup>2</sup>MC-16LX:

- **Linear method:** All 24-bit addresses are specified by instructions.
- **Bank method:** The higher 8 bits of an address are specified by the bank register associated with an application, and the lower 16 bits of the address are specified by an instruction.

### ■ Addressing using the linear method

The linear method is applied as follows:

#### ○ 24-bit operand specification:

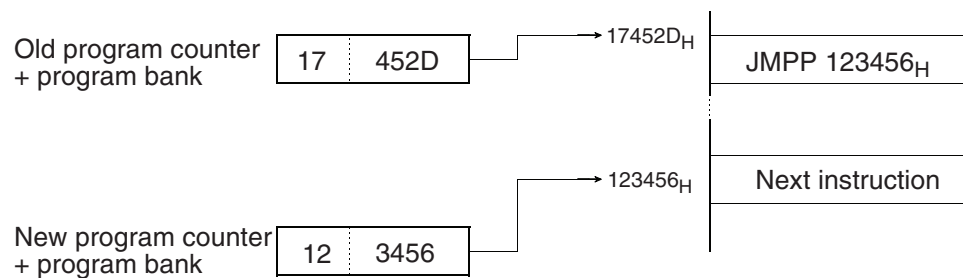
A 24-bit address is specified directly by an operand.

#### ○ 32-bit register indirect specification:

The lower 24 bits of the 32-bit general-purpose register are used as an address.

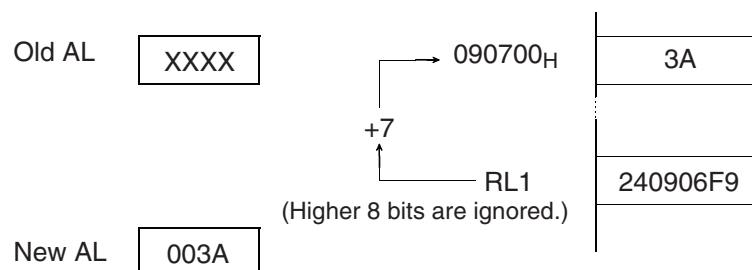
**Figure 2.2-1 Example of 24-Bit Operand Specification of the Linear Method**

JMPP 123456<sub>H</sub>



**Figure 2.2-2 Example of 32-Bit Register Indirect Specification of the Linear Method**

MOV A, @RL1+7



## ■ Addressing using the bank method

The bank method divides the 16M-byte space into 256 banks for each 64K bytes and specifies the bank associated with each space using the following five bank registers:

### ○ Program bank register (PCB): Reset-time initial value FF<sub>H</sub>

The 64K-byte bank specified by the PCB is called a program (PC) space. The program space contains primarily instruction codes, a vector table, and immediate data.

### ○ Data bank register (DTB): Reset-time initial value 00<sub>H</sub>

The 64K-byte bank specified by the DTB is called a data (DT) space. The data space contains primarily readable data, data that can be written, and the control and data registers of external and internal resources.

### ○ User stack bank register (USB): Reset initial value 00<sub>H</sub>

### ○ System stack bank register (SSB): Reset initial value 00<sub>H</sub>

The 64K-byte bank specified by the USB or SSB is called stack (SP) space. Stack space is accessed when stack access occurs at register saving of push and pop instructions and interrupts. The space to be used is dependent on an S flag in the condition code register.

### ○ Additional bank register (ADB): Reset-time initial value 00<sub>H</sub>

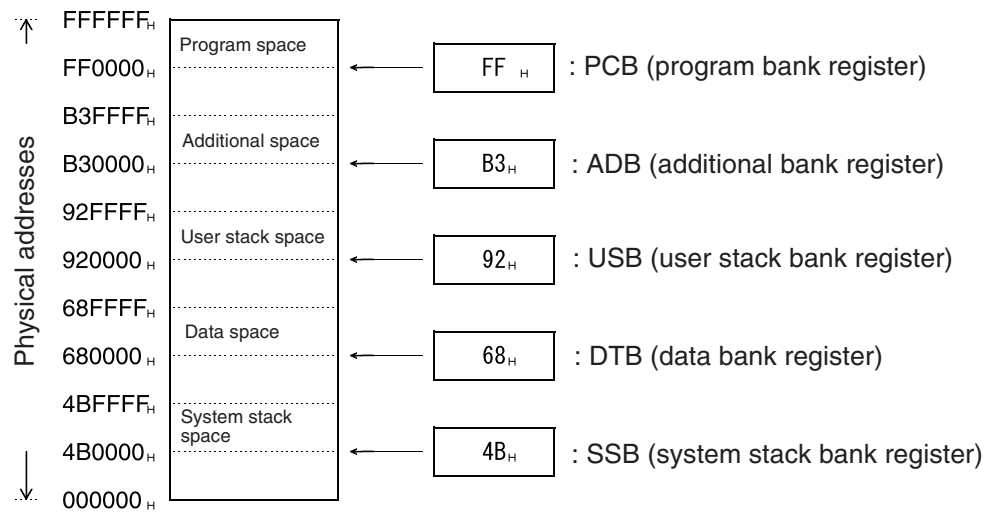
The 64K-byte bank specified by the ADB is called an additional (AD) space. The additional space contains primarily data that overflowed from the DT space.

To increase instruction code efficiency, default spaces as listed below are determined for instructions for each addressing mode. To use spaces other than default spaces when using an addressing mode, specify the prefix code associated with each bank prior to specifying the instruction, thereby enabling a bank space to be accessed associated with the prefix code.

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

**Table 2.2-1 Default Spaces**

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

**Figure 2.2-3 Example of the Physical Address of Spaces**

## 2.3 Allocating Many-Byte Length Data in a Memory Space

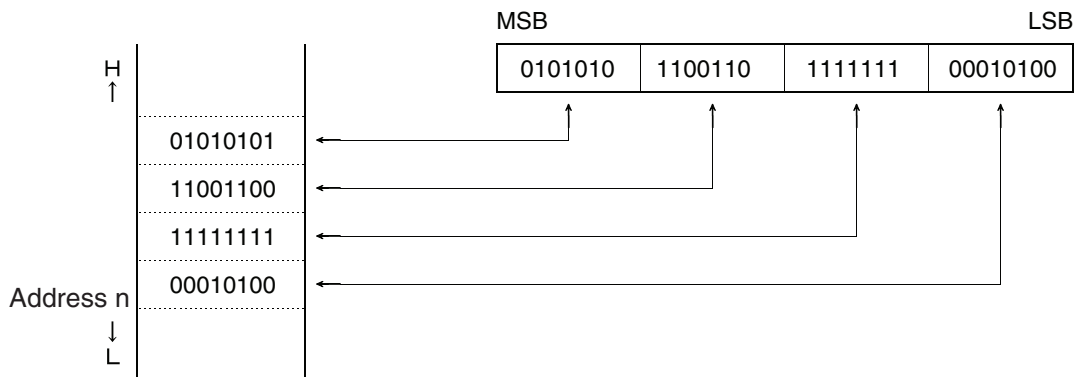
Data is written into the memory in ascending order of addresses. If data is 32 bits in length, the lower 16 bits are transferred before the higher 16 bits, and then the higher 16 bits are transferred.

If a reset signal is input immediately after the lower data is written, the higher data may not be written. To retain the data, the reset signal must be input after the higher data is written.

### ■ Allocating Many-Byte Length Data in a Memory Space

As shown in Figure 2.3-1 "Example of Allocating Many-Byte Length Data in the Memory", the lower eight bits of many-byte length data in a memory space are provided at address n. The remaining bits are provided sequentially at the addresses n+1, n+2, n+3, ...

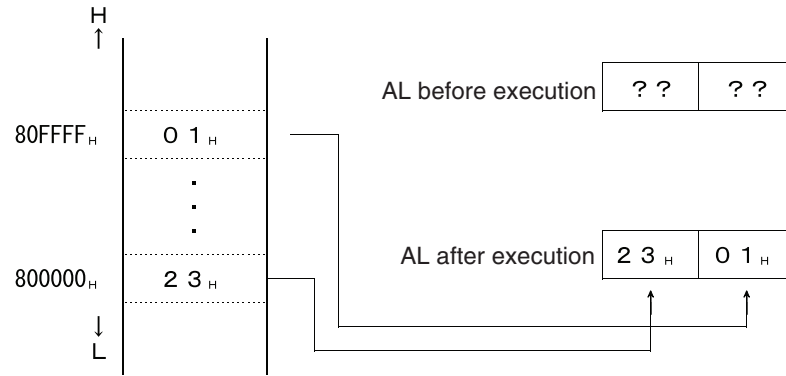
**Figure 2.3-1 Example of Allocating Many-Byte Length Data in the Memory**



### ■ Access to Many-Byte Length Data

As shown in Figure 2.3-2 "Example of Accessing Many-Byte Length Data (Execution of MOVW A, 080FFFF<sub>H</sub>)", all access operations are basically performed within the bank. For instructions that access many-byte length data, the address following address FFFF<sub>H</sub> is 0000<sub>H</sub> of the same bank.

**Figure 2.3-2 Example of Accessing Many-Byte Length Data (Execution of MOVW A, 080FFFF<sub>H</sub>)**





## 2.4 Dedicated Registers

---

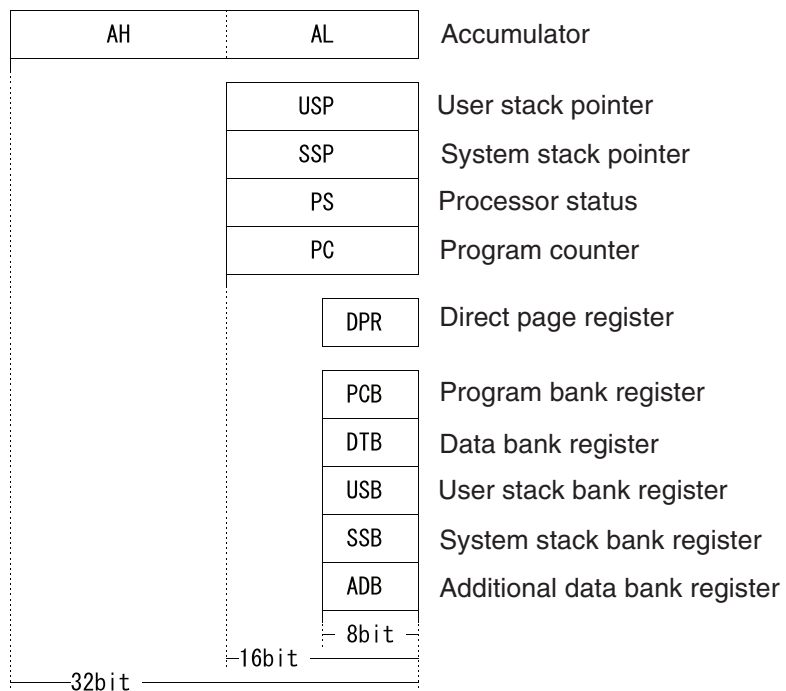
**A dedicated register exists in the CPU as dedicated hardware, and its use is restricted to the CPU architecture.**

---

### ■ Dedicated Registers

F<sup>2</sup>MC-16LX supports the following 11 dedicated registers:

- **Accumulator (A=AH:AL):**  
Two 16-bit accumulators (Can be used as the accumulator of 32 bits in total.)
- **User stack pointer (USP):**  
16-bit pointer indicating the user stack area
- **System stack pointer (SSP):**  
16-bit pointer indicating the system stack area
- **Processor status (PS):**  
16-bit register indicating the system status
- **Program counter (PC):**  
16-bit register having addresses at which the program is stored
- **Program bank register (PCB):**  
8-bit register indicating the PC space
- **Data bank register (DTB):**  
8-bit register indicating the DT space
- **User stack bank register (USB):**  
8-bit register indicating the user stack space
- **System stack bank register (SSB):**  
8-bit register indicating the system stack space
- **Additional bank register (ADB):**  
8-bit register indicating the AD space
- **Direct page register (DPR):**  
8-bit register indicating the direct page

**Figure 2.4-1 Dedicated Registers**

## 2.4.1 Accumulator (A)

The A consists of two 16-bit arithmetic operation registers AH and AL and is used as temporary storage for arithmetic operation results or for data transfer. For 32-bit data processing, the AH and AL in a combination are used. Only the AL is used for word processing of 16-bit data or for byte processing of 8-bit data.

■ Accumulator (A)

Data in the A can be used for arithmetic operations with data in the memory and registers (Ri, RWi, and RLi). As with F<sup>2</sup>MC-8, when F<sup>2</sup>MC-16LX transfers data equal to or less than a word length to the AL, data in the AL before the transfer is transferred automatically to the AH (data retention function), thereby ensuring that the data retention function and AL-AH arithmetic operations will increase processing efficiency.

Figure 2.4-2 Example of 32-bit Data Transfer

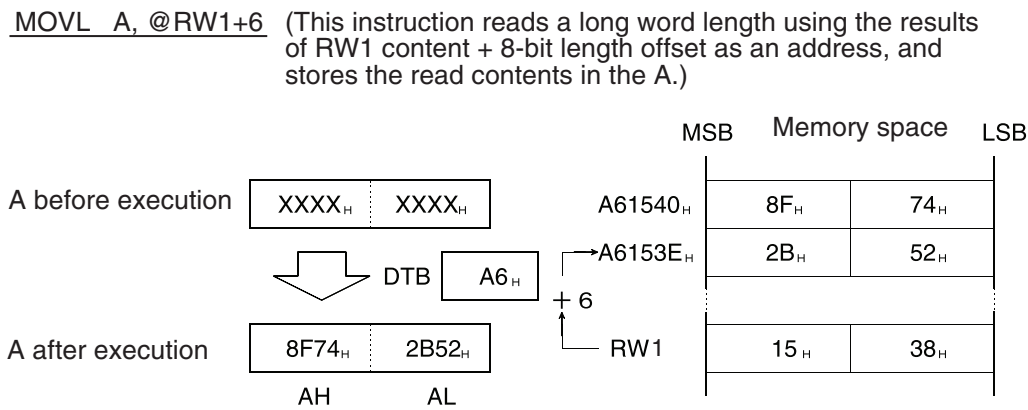
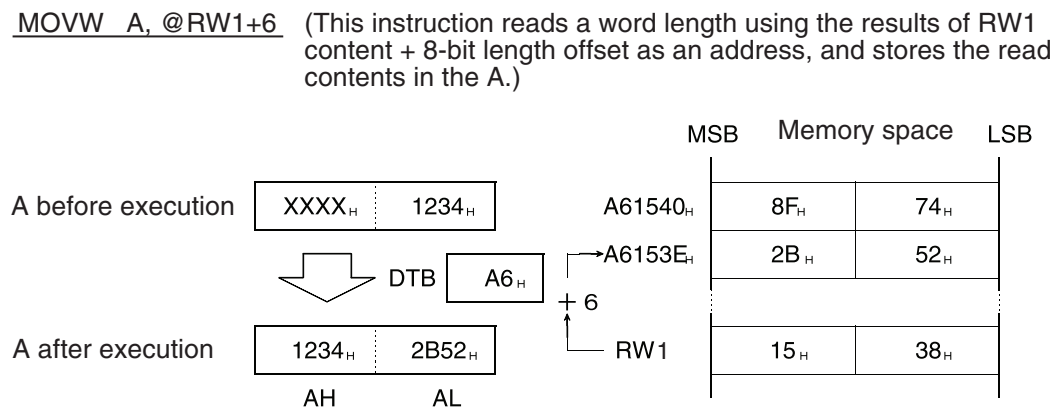


Figure 2.4-3 Example of AL-AH Transfer



As shown in Figures 2.4-4 and 2.4-5, when data with a length not exceeding one byte is transferred to the AL, it is zero-extended or sign-extended to 16 bits and is stored in the AL. Data in the AL is also handled in lengths of words or bytes.

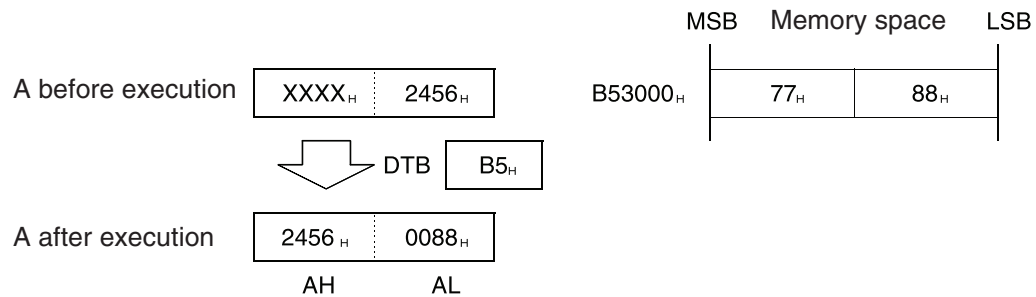
If an arithmetic operation instruction for byte processing is executed in the AL, the higher eight

bits of the AL before arithmetic operations are ignored. All higher eight bits of the arithmetic operation result are set to 0.

The A is not initialized by the reset, immediately after which the values become undefined.

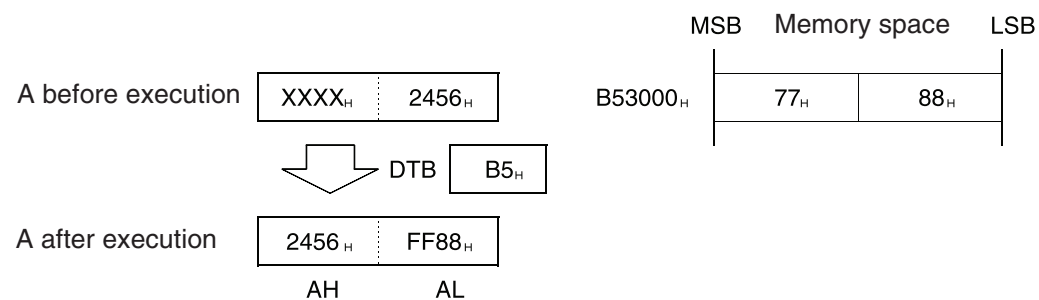
**Figure 2.4-4 Example of Executing Zero Extension**

MOV A, 3000H (This instruction extends the content at address 3000 by adding zeros, and stores the result in the AL.)



**Figure 2.4-5 Example of Executing the Sign Extension**

MOVX A, 3000H (This instruction extends the content at address 3000H with the sign and stores the result in the AL.)



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

The User Stack Pointer (USP) and the System Stack Pointer (SSP) are 16-bit registers that contain addresses for data saving and return at execution of a push/pop instruction or a subroutine.

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

The user stack pointer (USP) and the system stack pointer (SSP) are used by stack instructions. However, if the S flag in the processor status register is 0, the USP register is validated. If the S flag is 1, the SSP register is validated. (See Figure 2.4-6 "Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 0)" and Figure 2.4-7 "Stack Operation Instruction and Stack Pointers (Example of PUSHW A when the S Flag is 1)"

If an interrupt is accepted, the S flag is set, thereby ensuring that register saving at the interrupt will be performed in the memory area indicated by the SSP. The SSP is used for stack processing by the interrupt routine. The USP is used for stack processing by a routine other than the interrupt routine. Use only the SSP if the stack space does not have to be split. The higher eight bits of an address at stack processing are indicated by SSP--> SSB and USP --> USB. USP and SSP are not initialized at a reset, and their values become undefined instead.

Figure 2.4-6 Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 0)

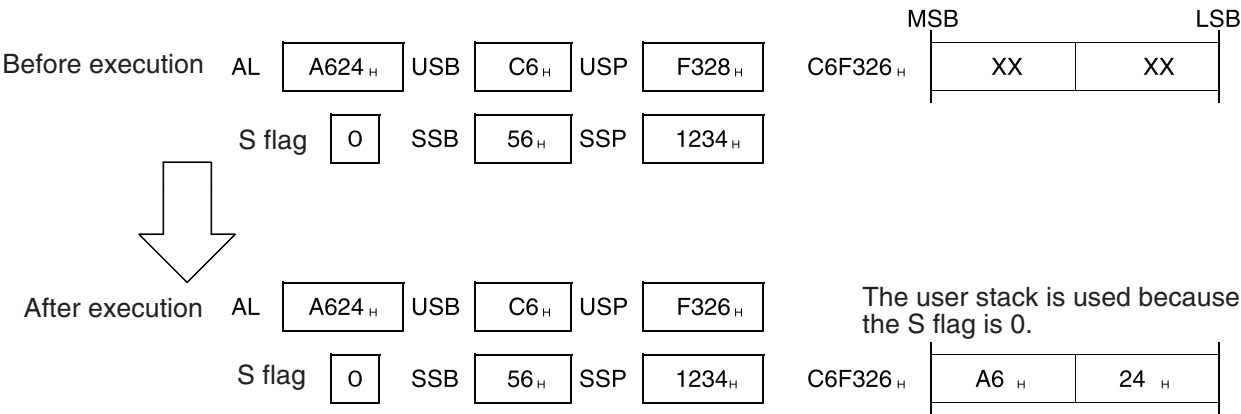
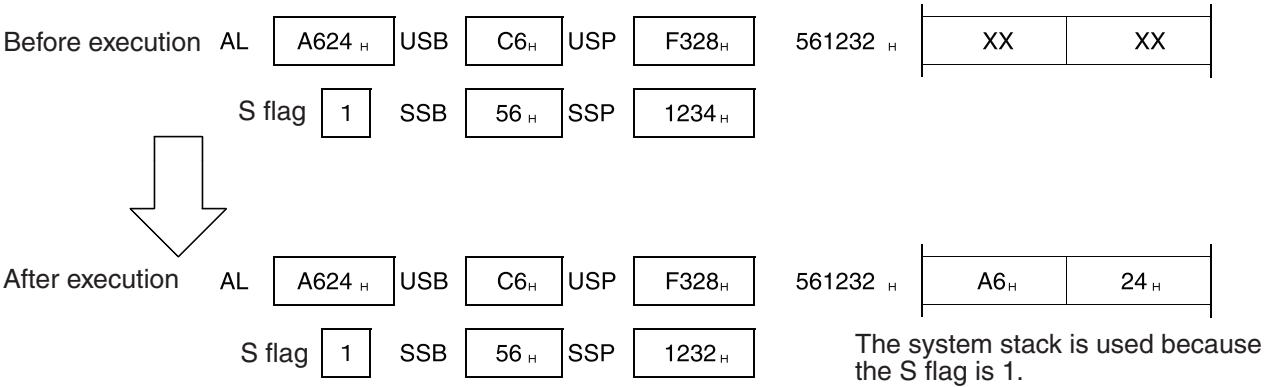


Figure 2.4-7 Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 1)



**Note:**  
In principle, use an even address as the value to be set in the stack pointer.

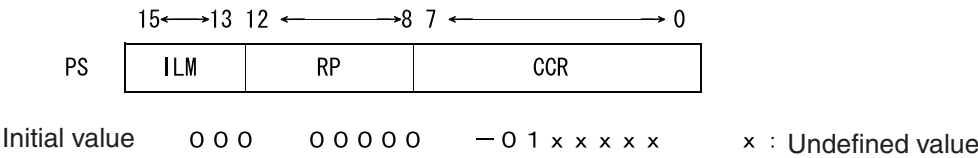
### 2.4.3 Processor Status (PS)

The PS consists of bits for controlling CPU operations and bits indicating CPU status.

■ Processor Status (PS)

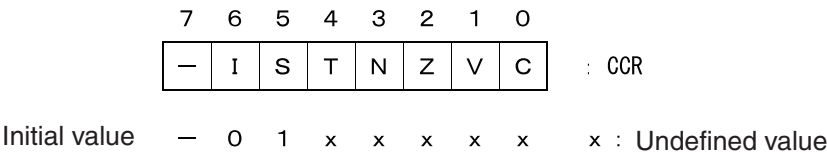
The higher bytes of the PS consist of the register bank pointer (RP), which indicates the first address of the register bank, and the interrupt level mask register (ILM). The lower bytes of the PS consist of the condition code register (CCR) formed by flags to be set or reset by instruction execution results or interrupt occurrence.

Figure 2.4-8 Structure of the PS



■ Condition Code Register (CCR)

Figure 2.4-9 Configuration of the Condition Code Register



**I: Interrupt enable flag:**

An interrupt is enabled when the I is 1 for all interrupt requests other than a software interrupt. If the I is 0, the interrupt is masked. The I is cleared at reset.

**S: Stack flag:**

When S is 0, the USP is valid as the stack operation pointer. If the S is 1, the SSP is valid.

The S is set when an interrupt is accepted or the reset is made.

**T: Sticky bit flag:**

This flag is 1 if the data shifted out from the carry contains at least one 1 after the logical right or arithmetic right shift instruction is executed. In other cases, the flag is 0. The flag is also 0 when the shift amount is 0.

**N: Negative flag:**

This flag is set when the MSB of the arithmetic operation result is 1 and is cleared if the MSB is 0.

**Z: Zero flag:**

This flag is set when all arithmetic operation results are 0 and is cleared in other cases.

**V: Overflow flag:**

This flag is set when an overflow occurs to indicate a signed numeric value as a result of an arithmetic operation and is cleared if no overflow occurs.

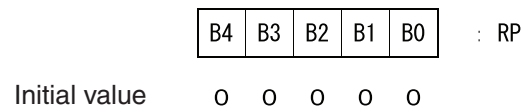
**C: Carry flag:**

This flag is set when a carry-up or carry-down is generated from the MSB as a result of arithmetic operation and is cleared if no carry-up or carry-down occurs.

**■ Register Bank Pointer (RP)**

The RP register indicates the relationship between the general-purpose register of F<sup>2</sup>MC-16LX and the address of the internal RAM in which the general-purpose register exists. The RP indicates the first memory address of the used register bank as the conversion expression  $[000180_H + (RP) \cdot 10_H]$ . The RP consists of 5 bits and can accept a value from 00H to 1FH and can provide a register bank in the memory of 000180<sub>H</sub> to 00037F<sub>H</sub>.

However, if the memory is not an internal RAM even if this range is satisfied, the register cannot be used as a general-purpose register. The contents of the RP are all initialized to 0 by the reset. An 8-bit immediate value can be transferred to the RP, though only the lower five bits of the data are actually used.

**Figure 2.4-10 Configuration of the Register Bank Pointer (RP)****■ Interrupt Level Mask Register (ILM)**

The ILM consists of three bits to indicate the level of the CPU interrupt mask. Only interrupt requests whose levels are higher than the level indicated by these three bits are accepted. Level 0 is defined as the highest and level 7 is defined as the lowest. Therefore, to enable an interrupt to be accepted, a value smaller than the value retained by the current ILM must be requested. When the interrupt is accepted, its level value is set in the ILM and any subsequent interrupt whose priority is equal or lower is not accepted. The contents of the ILM are all initialized to 0 by the reset. An 8-bit immediate value can be transferred to the ILM, though only the lower three bits of the data are actually used.

**Figure 2.4-11 Configuration of the Interrupt Level Mask Register (ILM)****Table 2.4-1 High-Low of the Levels Indicated by the Interrupt Level Mask Register (ILM)**

ILM2	ILM1	ILM0	Level value	Interrupt level to be allowed
0	0	0	0	Interrupt prohibited
0	0	1	1	0 only
0	1	0	2	Level whose value is less than 1
0	1	1	3	Level whose value is less than 2
1	0	0	4	Level whose value is less than 3
1	0	1	5	Level whose value is less than 4
1	1	0	6	Level whose value is less than 5
1	1	1	7	Level whose value is less than 6



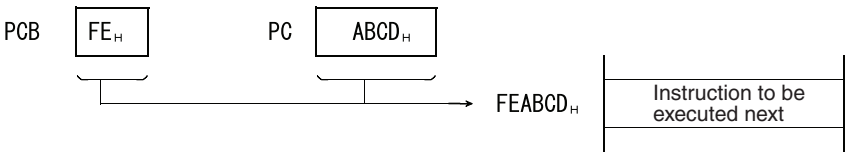
### 2.4.4 Program Counter (PC)

The PC is a 16-bit counter that indicates the lower 16 bits of the memory address of the instruction code to be executed by the CPU. The higher eight addresses are indicated by the PCB.

■ Program Counter (PC)

The PC is updated by a conditional branch instruction, subroutine call instruction, interrupt, or reset and can be used as the base pointer at operand access.

Figure 2.4-12 Configuration of the Program Counter



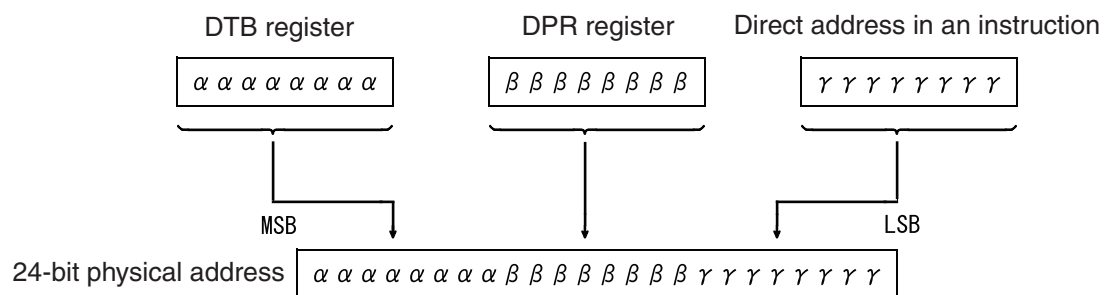
## 2.4.5 Direct Page Register (DPR)

The DPR specifies operands `addr8` to `addr15` when a direct addressing instruction is executed. The length of the DPR is eight bits and is initialized to 01H by the reset. The DPR can be read or written by an instruction.

### ■ Direct Page Register (DPR)

Figure 2.4-13 "Generating a Physical Address by Direct Addressing" shows a physical address to be created by direct addressing.

**Figure 2.4-13 Generating a Physical Address by Direct Addressing**



## 2.4.6 Bank Register

---

The five types of bank register are as follows:

- **Program Counter Bank Register (PCB)** <initial value: value in the reset vector>
  - **Data Bank Register (DTB)** <initial value: 00<sub>H</sub>>
  - **User Stack Bank Register (USB)** <initial value: 00<sub>H</sub>>
  - **System Stack Bank Register (SSB)** <initial value: 00<sub>H</sub>>
  - **Additional Data Bank Register (ADB)** <initial value: 00<sub>H</sub>>
- 

### ■ Bank Register

The bank registers indicate the memory banks in which the PC space, DT space, SP space (user), SP space (system), and AD space are allocated. All bank registers have a byte length. The PCB is initialized to 00<sub>H</sub> with the reset vector at reset. A bank register other than the PCB is readable and can be written. The PCB is readable but cannot be written.

The PCB is rewritten when an interrupt occurs or at execution of JMPP, CALLP, RETP, RETIQ, or RETF instructions branching to all 16M-byte spaces.

For register operations, see Section 2.1 "Memory Space".

## 2.5 General-Purpose Registers

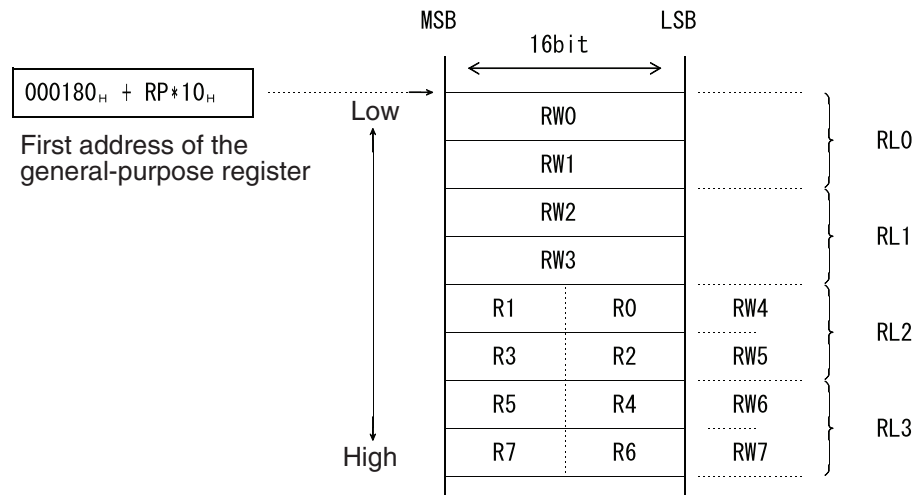
Like a normal memory, a user can specify the use of a general-purpose register. A general-purpose register is the same as a dedicated register in that it coexists with the RAM in the address space of the CPU and can be accessed without specifying an address.

### ■ General-Purpose Registers

The general-purpose registers of F<sup>2</sup>MC-16LX exist at 000180<sub>H</sub> to 00037F<sub>H</sub> (maximum) of the main storage. They specify which part of an address previously mentioned is the register bank being used through the register bank pointer (RP). Each bank contains the three types of registers listed below. These registers are not independent of one another and are related follows:

- 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 2.5-1 General-Purpose Registers**



The relationship between higher and lower bytes of the byte and word registers is obtained from the following expression:

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

The relationship between the higher and lower bytes of the RL<sub>i</sub> and the RW can be obtained from the following expression:

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

### ■ Register Banks

As shown in Table 2.5-1 "Register Bank Functions", a register bank consists of eight words and can be used for arithmetic operations as a general-purpose register of byte registers R0 to R7, word registers RW0 to RW7, and long word registers RL0 to RL3. A bank can also be used to store an instruction pointer. RL0 to RL3 can also be used as a linear pointer that accesses all spaces directly. As with the ordinary RAM, the contents of the register bank are not initialized by the reset and the status prior to the reset is retained. The values are undefined at power-on.

**Table 2.5-1 Register Bank Functions**

Register	Function
R0 ~ R7	Used as operands of the instructions.
RW0 ~ RW7	Used as operands of the pointer and instructions.
RL0 ~ RL3	Used as operands of the long pointer and instructions.

**Note:**

R0 is also used as the barrel shift counter and normalize instruction counter.

RW0 is also used as the string instruction counter.

## 2.6 Prefix Codes

Three types of prefix codes are as follows: bank select prefixes, common register bank prefixes, and flag change suppression prefixes.

A part of an instruction operation can be changed by prefixing a prefix code to the instruction.

### ■ Bank Select Prefixes

The memory space to be used at data access is determined for each addressing mode.

The memory space for data access by an instruction can be selected arbitrarily by prefixing the bank select prefix to the instruction regardless of an addressing mode.

Table 2.6-1 "Bank Select Prefixes" lists the bank select prefixes and the memory areas to be selected.

**Table 2.6-1 Bank Select Prefixes**

Bank select prefix	Space to be selected
PCB	PC space
DTB	Data space
ADB	Additional area
SPB	Either the SSP or USP space is used depending on the content of the stack flag.

When using the bank select prefixes, note the following instructions:

○ **String instructions [MOVS, MOVSW, SCEQ, SCWEQ, FILS, and FILSW]**

The bank register specified by an operand is used regardless of whether there is a prefix.

○ **Stack operation instructions [PUSHW, POPW]**

The SSB or USB is used according to the S flag regardless of whether there is a prefix.

○ **I/O access instructions**

$$\left[ \begin{array}{l} \text{MOV A, io / MOV io, A / MOVX A, io / MOVW A, io / MOVW io, A / MOV io, \#imm8} \\ \text{MOVW io, \#imm16 / MOVB A, io:bp / MOVB io:bp, A / SETB io:bp / CLRB io:bp} \\ \text{BBC io:bp, rel / BBS io:bp, rel / WBTC, WBTS} \end{array} \right]$$

The I/O space of a bank is used regardless of whether there is a prefix.

○ **Flag change instructions [AND CCR, \#imm8, OR CCR, \#imm8]**

The instruction operations are normal, though the effect of a prefix continues until the next instruction.

### ○ **POPW ps**

The SSB or USB is used according to the S flag regardless of whether there is a prefix.

The effect of the prefix continues until the next instruction.

### ○ **MOV ILM, #imm8**

The operation instructions are normal, though the effect of a prefix continues until the next instruction.

### ○ **RETI**

The SSB is used regardless of whether there is a prefix.

## ■ **Common Register Bank Prefix (CMR)**

To simplify data exchange among two or more tasks, it is necessary to use a means of accessing the same register bank that was easily determined by comparison regardless of the RP value set. All register accesses of instructions accessing a register bank can be changed to the common bank in 000180<sub>H</sub> to 00018F<sub>H</sub> (register bank selected for RP = 0) by prefixing the CMR to the instruction regardless of the current RP value.

Note the following instructions:

### ○ **String instructions [MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW]**

If an interrupt request occurs during execution of a string instruction to which a prefix code was added, a malfunction occurs for a string instruction executed after the return of the interrupt because the prefix is invalid. Do not add the CMR prefix to the above string instructions.

### ○ **Flag change instruction [AND CCR, #imm8, OR CCR, #imm8, POPW PS]**

The instruction operations are normal, though the effect of the prefix continues until the next instruction.

### ○ **MOV ILM, #imm8**

The instruction operations are normal, though the effect of the prefix continues until the next instruction.

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

To suppress a flag change, use the flag change suppression prefix code (NCC). A flag change accompanied by instruction execution can be suppressed by prefixing an unnecessary flag change to the instruction.

Note the following instructions:

### ○ **String instructions [MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW]**

If an interrupt request occurs during execution of a string instruction to which a prefix code was added, a malfunction occurs for a string instruction executed after the return of the interrupt because the prefix is invalid. Do not add the NCC prefix to the above string instructions.

### ○ **Flag change instruction [AND CCR, #imm8, OR CCR, #imm8, POPW PS]**

The instruction operations are normal, though the effect of the prefix continues until the next instruction.

- **Interrupt instructions [INT #vct8, INT9, INT addr16, INTP addr24, RETI]**

The CCR changes according to instruction specifications regardless of whether there is a prefix.

- **JCTX @A**

The CCR changes according to instruction specifications regardless of whether there is a prefix.

- **MOV ILM, imm8**

The instruction operations are normal, though the effect of the prefix continues until the next instruction.



## 2.6.1 Restrictions on the Use of Prefix Instructions

The following restrictions apply on the use of prefix instructions:

- No interrupt/hold request is accepted during execution of prefix codes and interrupt/hold suppression instructions.
- If a prefix code is prefixed to an interrupt/hold instruction, the effect of the prefix code is delayed.
- When competing prefix codes are consecutive, the last prefix code is valid.

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

The following restriction apply to the interrupt/hold suppression instructions listed in Table 2.6-2 "Prefix Codes and Interrupt/Hold Suppression Instruction":

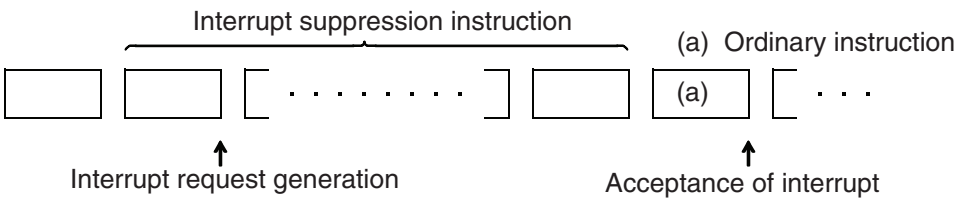
**Table 2.6-2 Prefix Codes and Interrupt/Hold Suppression Instructions**

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

#### ○ Interrupt/hold suppression

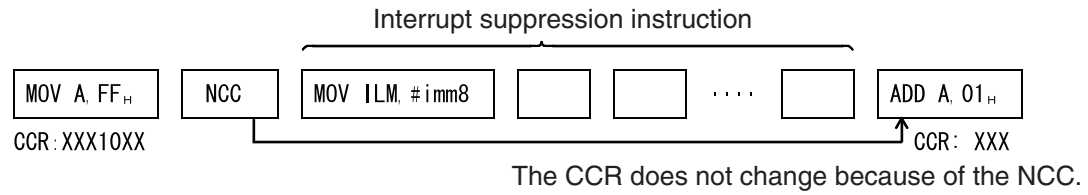
As shown in Figure 2.6-1 "Interrupt/Hold Suppression", any generated interrupts or hold requests are not accepted during execution of prefix codes and interrupt/hold suppression instructions. In such cases, an interrupt or hold request is accepted when an instruction other than a prefix code or interrupt/hold suppression instruction has been executed.

**Figure 2.6-1 Interrupt/Hold Suppression**



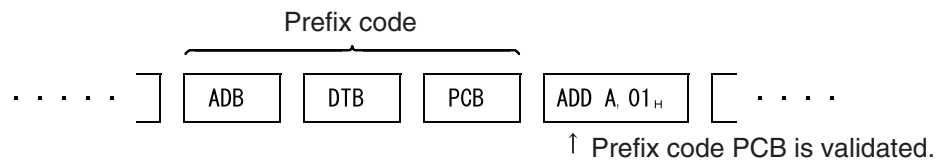
#### ○ Delaying the effect of the prefix code

As shown in Figure 2.6-2 "Interrupt/Hold Suppression Instruction and Prefix Codes", if a prefix code is prefixed to an interrupt or hold suppression instruction, the effect of the prefix code is validated for the first subsequent instruction following the interrupt/hold suppression instruction.

**Figure 2.6-2 Interrupt/Hold Suppression Instructions and Prefix Codes**

### ■ When Prefix Codes are Consecutive

As shown in Figure 2.6-3 "Consecutive Prefix Codes", when the prefix codes (PCB, ADB, DTB, and SPB) in contention are consecutive, the last prefix code is valid.

**Figure 2.6-3 Consecutive Prefix Codes**

## 2.6.2 Notes on Using "DIV A, Ri" and "DIVW A, RWi" Instructions

The remainders of instruction execution are stored at the address (lower 16-bit) equivalent to the register of the instruction operand in the memory bank area (higher 8-bit) in accordance with the contents listed in Table 2.6-3 "Note on Using "DIV A, Ri" and "DIVW A, RWi" Instruction". Set the value of the corresponding bank register to "00<sub>H</sub>" when using "DIV A, Ri" and "DIVW A, RWi" instructions.

### ■ Notes on Using "DIV A, Ri" and "DIVW A, RWi" Instructions

Table 2.6-3 Notes on Using "DIV A, Ri" and "DIVW A, RWi" Instructions

Instruction	Name of the bank register affected when executing the instructions shown in the table	Address at which the remainder is stored
DIV A, R0	DTB	(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 8 <sub>H</sub> : Lower 16-bit)
DIV A, R1		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 9 <sub>H</sub> : Lower 16-bit)
DIV A, R4		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + C <sub>H</sub> : Lower 16-bit)
DIV A, R5		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + D <sub>H</sub> : Lower 16-bit)
DIVW A, RW0		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 0 <sub>H</sub> : Lower 16-bit)
DIVW A, RW1		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 2 <sub>H</sub> : Lower 16-bit)
DIVW A, RW4		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 8 <sub>H</sub> : Lower 16-bit)
DIVW A, RW5		(DTB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + A <sub>H</sub> : Lower 16-bit)
DIV A, R2	ADB	(ADB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + A <sub>H</sub> : Lower 16-bit)
DIV A, R6		(ADB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + E <sub>H</sub> : Lower 16-bit)
DIVW A, RW2		(ADB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 4 <sub>H</sub> : Lower 16-bit)
DIVW A, RW6		(ADB: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + E <sub>H</sub> : Lower 16-bit)
DIV A, R3	USB SSB*1	(USB*2: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + B <sub>H</sub> : Lower 16-bit)
DIV A, R7		(USB*2: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + F <sub>H</sub> : Lower 16-bit)
DIVW A, RW3		(USB*2: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + 6 <sub>H</sub> : Lower 16-bit)
DIVW A, RW7		(USB*2: Higher 8-bit) + (0180 <sub>H</sub> + RP x 10 <sub>H</sub> + E <sub>H</sub> : Lower 16-bit)

\*1 Depending on the S bit of the CCR register

\*2 When the S bit of the CCR register is 0

If the value of corresponding bank registers (DTB, ADB, USB, SSB) is set to "00<sub>H</sub>", the remainder of division results are stored in the register of the instruction operand. If the value is set to a value other than "00<sub>H</sub>", the higher 8-bit address is specified by the bank register corresponding to the register of the instruction operand. The lower 16-bit address then becomes the same address as that of the register of the instruction operand at which the remainder is stored.

**[Example]**

If "DIV A, R0" instruction is executed in the case of DTB=053<sub>H</sub>/RP=003<sub>H</sub>, the address of R0 is as follows: 0180<sub>H</sub>+RP (003<sub>H</sub>) x 10<sub>H</sub>+08<sub>H</sub> (address equivalent to R0) = 0001B8<sub>H</sub>

The bank register to be specified by "DIV A, R0" is DTB, so the address to which bank-specified 053<sub>H</sub> is added, that is 05301B8<sub>H</sub>, is the address at which the remainder of a result is stored. (For an explanation of Ri and RWi registers, see Section 2.5 "General-purpose Register".

■ **Evasion of notes**

In order to evade notes of the "DIV A, Ri" and "DIVW A, RWi" instructions during program development, the compiler modifies the program so that the respective instructions are not generated. The assembler then replaces these instructions by functions equivalent to the instruction strings.

Use the following compiler and assembler.

- Compiler: Version V02L06 of cc907 or later, and version V30L02 of fcc907s or later
- Assembler: Version V03L04 of asm907a or later, and version V30L04 (Rev.30004) of fasm907s or later



## CHAPTER 3    INTERRUPT

---

**This chapter describes the interrupt functions and operations.**

---

- 3.1 "Overview of the Interrupt"
- 3.2 "Interrupt Source"
- 3.3 "Interrupt Vector"
- 3.4 "Hardware Interrupt"
- 3.5 "Software Interrupts"
- 3.6 "Extended Intelligent I/O Service (EI<sup>2</sup>OS)"
- 3.7 "Exception"

## 3.1 Overview of the Interrupt

---

**The F<sup>2</sup>MC-16LX has an interrupt function that interrupts processing if a suitable event occurs and passes control to a program defined separately.**

---

### ■ Overview of the Interrupt

The interrupt functions support the following four types of interrupts:

- Hardware interrupt: Interrupt processing activated by the occurrence of an internal resource event
- Software interrupt: Interrupt processing activated by the occurrence of an event occurrence instruction
- Extended intelligent I/O service (EI<sup>2</sup>OS): Transfer processing activated by the occurrence of an internal resource event
- Exception: Interrupt processing activated by the occurrence of an operation execution

This chapter describes these four types of interrupt functions.

## 3.2 Interrupt Source

Table 3.2-1 "Interrupt Source, Interrupt Vector, Interrupt Control Register" shows the interrupt source, the interrupt vector, and the interrupt control register.

### ■ Interrupt Source

Table 3.2-1 Interrupt Source, Interrupt Vector, Interrupt Control Register

Interrupt source	EI <sup>2</sup> OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Reset	×	# 08	FFFFDCH	—	—
INT9 instruction	×	# 09	FFFFD8H	—	—
Exception	×	# 10	FFFFD4H	—	—
A/D converter	Y2	# 11	FFFFD0H	ICR00	0000B0H
Input capture 0 fetch	Y2	# 12	FFFFCCH		
DTP0 (external division 0)	Y2	# 13	FFFFC8H	ICR01	0000B1H
Input capture 1 fetch	Y2	# 14	FFFFC4H		
Output compare 0 matching	Y2	# 15	FFFFC0H	ICR02	0000B2H
Output compare 1 matching	Y2	# 16	FFFFBCH		
Output compare 2 matching	Y2	# 17	FFFFB8H	ICR03	0000B3H
Output compare 3 matching	Y2	# 18	FFFFB4H		
I/O serial 0	Y2	# 19	FFFFB0H	ICR04	0000B4H
Free run timer	×	# 20	FFFFACH		
I/O serial 1	Y2	# 21	FFFFA8H	ICR05	0000B5H
Watch timer	×	# 22	FFFFA4H		
I/O serial 2	Y2	# 23	FFFFA0H	ICR06	0000B6H
DTP1 (external division 1)	Y2	# 24	FFFF9CH		
DTP2/3 (external division 2/3)	Y2	# 25	FFFF98H	ICR07	0000B7H
8/16-bit PPG0 counter borrow	×	# 26	FFFF94H		
DTP4/5 (external division 4/5)	Y2	# 27	FFFF90H	ICR08	0000B8H
8/16-bit PPG1 counter borrow	×	# 28	FFFF8CH		



## CHAPTER 3 INTERRUPT

**Table 3.2-1 Interrupt Source, Interrupt Vector, Interrupt Control Register (Continued)**

Interrupt source	EI <sup>2</sup> OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Up/down counter 0 borrow/overflow/reverse	Y2	# 29	FFFF88H	ICR09	0000B9H
Up/down counter 0 compare matching	Y2	# 30	FFFF84H		
Up/down counter 1 borrow/overflow/reverse	Y2	# 31	FFFF80H	ICR10	0000BAH
Up/down counter 1 compare matching	Y2	# 32	FFFF7CH		
DTP6 (external division 6)	Y2	# 33	FFFF78H	ICR11	0000BBH
Time base timer	×	# 34	FFFF74H		
DTP7 (external division 7)	Y2	# 35	FFFF70H	ICR12	0000BCH
I2C interface	×	# 36	FFFF6CH		
UART1 reception completion	Y1	# 37	FFFF68H	ICR13	0000BDH
UART1 sending completion	Y2	# 38	FFFF64H		
UART0 reception completion	Y1	# 39	FFFF60H	ICR14	0000BEH
UART0 sending completion	Y2	# 40	FFFF5CH		
Flash memory	×	# 41	FFFF58H	ICR15	0000BFH
Delay interrupt	×	# 42	FFFF54H		

Y1: The interrupt request flag is cleared by the EI<sup>2</sup>OS interrupt clear signal. There is a stop request.

Y2: The interrupt request flag is cleared by the EI<sup>2</sup>OS interrupt clear signal.

×

## 3.3 Interrupt Vector

**Table 3.3-1 "List of Interrupt Vectors" lists the interrupt vectors.**

### ■ Interrupt Vectors

**Table 3.3-1 List of Interrupt Vectors**

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 0	FFFFFC <sub>H</sub>	FFFFFD <sub>H</sub>	FFFFFE <sub>H</sub>	Not used	#0	None
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 7	FFFFE0 <sub>H</sub>	FFFFE1 <sub>H</sub>	FFFFE2 <sub>H</sub>	Not used	#7	None
INT 8	FFFFDC <sub>H</sub>	FFFFDD <sub>H</sub>	FFFFDE <sub>H</sub>	FFFFDF	#8	(RESET vector)
INT 9	FFFFD8 <sub>H</sub>	FFFFD9 <sub>H</sub>	FFFFDA <sub>H</sub>	Not used	#9	None
INT 10	FFFFD4 <sub>H</sub>	FFFFD5 <sub>H</sub>	FFFFD6 <sub>H</sub>	Not used	#10	<Exception>
INT 11	FFFFD0 <sub>H</sub>	FFFFD1 <sub>H</sub>	FFFFD2 <sub>H</sub>	Not used	#11	A/D converter
INT 12	FFFFCC <sub>H</sub>	FFFFCD <sub>H</sub>	FFFFCE <sub>H</sub>	Not used	#12	ICAP0
INT 13	FFFFC8 <sub>H</sub>	FFFFC9 <sub>H</sub>	FFFFCA <sub>H</sub>	Not used	#13	DTP0
INT 14	FFFFC4 <sub>H</sub>	FFFFC5 <sub>H</sub>	FFFFC6 <sub>H</sub>	Not used	#14	ICAD1
INT 15	FFFFC0 <sub>H</sub>	FFFFC1 <sub>H</sub>	FFFFC2 <sub>H</sub>	Not used	#15	Output compare #0
INT 16	FFFFBC <sub>H</sub>	FFFFBD <sub>H</sub>	FFFFBE <sub>H</sub>	Not used	#16	Output compare #1
INT 17	FFFFB8 <sub>H</sub>	FFFFB9 <sub>H</sub>	FFFFBA <sub>H</sub>	Not used	#17	Output compare #2
INT 18	FFFFB4 <sub>H</sub>	FFFFB5 <sub>H</sub>	FFFFB6 <sub>H</sub>	Not used	#18	Output compare #3
INT 19	FFFFB0 <sub>H</sub>	FFFFB1 <sub>H</sub>	FFFFB2 <sub>H</sub>	Not used	#19	Extended I/O serial 0
INT 20	FFFFAC <sub>H</sub>	FFFFAD <sub>H</sub>	FFFFAE <sub>H</sub>	Not used	#20	Free run timer
INT 21	FFFFA8 <sub>H</sub>	FFFFA9 <sub>H</sub>	FFFFAA <sub>H</sub>	Not used	#21	Extended I/O serial 1
INT 22	FFFFA4 <sub>H</sub>	FFFFA5 <sub>H</sub>	FFFFA6 <sub>H</sub>	Not used	#22	Watch timer
INT 23	FFFFA0 <sub>H</sub>	FFFFA1 <sub>H</sub>	FFFFA2 <sub>H</sub>	Not used	#23	Extended I/O serial 2
INT 24	FFFF9C <sub>H</sub>	FFFF9D <sub>H</sub>	FFFF9E <sub>H</sub>	Not used	#24	DTP1
INT 25	FFFF98 <sub>H</sub>	FFFF99 <sub>H</sub>	FFFF9A <sub>H</sub>	Not used	#25	DTP2/3

## CHAPTER 3 INTERRUPT

**Table 3.3-1 List of Interrupt Vectors (Continued)**

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 26	FFFF94 <sub>H</sub>	FFFF95 <sub>H</sub>	FFFF96 <sub>H</sub>	Not used	#26	PPG0
INT 27	FFFF90 <sub>H</sub>	FFFF91 <sub>H</sub>	FFFF92 <sub>H</sub>	Not used	#27	DTP4/5
INT 28	FFFF8C <sub>H</sub>	FFFF8D <sub>H</sub>	FFFF8E <sub>H</sub>	Not used	#28	PPG1
INT 29	FFFF88 <sub>H</sub>	FFFF89 <sub>H</sub>	FFFF8A <sub>H</sub>	Not used	#29	Up/down 0 borrow
INT 30	FFFF84 <sub>H</sub>	FFFF85 <sub>H</sub>	FFFF86 <sub>H</sub>	Not used	#30	Up/down 0 compare
INT 31	FFFF80 <sub>H</sub>	FFFF81 <sub>H</sub>	FFFF82 <sub>H</sub>	Not used	#31	Up/down 1 borrow
INT 32	FFFF7C <sub>H</sub>	FFFF7D <sub>H</sub>	FFFF7E <sub>H</sub>	Not used	#32	Up/down 1 compare
INT 33	FFFF78 <sub>H</sub>	FFFF79 <sub>H</sub>	FFFF7A <sub>H</sub>	Not used	#33	DTP6
INT 34	FFFF74 <sub>H</sub>	FFFF75 <sub>H</sub>	FFFF76 <sub>H</sub>	Not used	#34	Time base
INT 35	FFFF70 <sub>H</sub>	FFFF71 <sub>H</sub>	FFFF72 <sub>H</sub>	Not used	#35	DTP7
INT 36	FFFF6C <sub>H</sub>	FFFF6D <sub>H</sub>	FFFF6E <sub>H</sub>	Not used	#36	I <sup>2</sup> C interface
INT 37	FFFF68 <sub>H</sub>	FFFF69 <sub>H</sub>	FFFF6A <sub>H</sub>	Not used	#37	UART1 reception completion
INT 38	FFFF64 <sub>H</sub>	FFFF65 <sub>H</sub>	FFFF66 <sub>H</sub>	Not used	#38	UART1 sending completion
INT 39	FFFF60 <sub>H</sub>	FFFF61 <sub>H</sub>	FFFF62 <sub>H</sub>	Not used	#39	UART0 reception completion
INT 40	FFFF5C <sub>H</sub>	FFFF5D <sub>H</sub>	FFFF5E <sub>H</sub>	Not used	#40	UART0 sending completion
INT 41	FFFF58 <sub>H</sub>	FFFF59 <sub>H</sub>	FFFF5A <sub>H</sub>	Not used	#41	Flash memory
INT 42	FFFF54 <sub>H</sub>	FFFF55 <sub>H</sub>	FFFF56 <sub>H</sub>	Not used	#42	Delay interrupt
INT 43	FFFF50 <sub>H</sub>	FFFF51 <sub>H</sub>	FFFF52 <sub>H</sub>	Not used	#43	None
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 254	FFFC04 <sub>H</sub>	FFFC05 <sub>H</sub>	FFFC06 <sub>H</sub>	Not used	#254	None
INT 255	FFFC00 <sub>H</sub>	FFFC01 <sub>H</sub>	FFFC02 <sub>H</sub>	Not used	#255	None

## 3.4 Hardware Interrupt

---

**A hardware interrupt has a function that interrupts a program being executed by the CPU according to an interrupt request signal from an internal resource and passes control to user-defined interrupt processing program.**

---

### ■ Overview of Hardware Interrupts

The interrupt level of an interrupt request is compared with the interrupt level mask register (ILM) of the CPU PS and the I flag in the CCR is referenced by hardware to determine if interrupt occurrence conditions are satisfied. If the conditions are satisfied, a hardware interrupt occurs.

When a hardware interrupt occurs, the CPU responds as follows:

- Saves the descriptions of the PC, PS, A, PCB, DTB, ADB, and DPR registers in the CPU in the system stack.
- Sets the ILM in the PS register (the ILM automatically attains the same interrupt level as that of the interrupt request being processed).
- Fetches the contents of the corresponding interrupt vector and branches control thereto.

### ■ Hardware Interrupt Mechanism

The following four mechanisms are related to hardware interrupts:

#### ○ Internal resource

Interrupt enable bit, interrupt request bit: control of an interrupt from a resource

#### ○ Interrupt controller

ICR: Level assignment to an interrupt, priority decision of interrupts requested concurrently

#### ○ CPU

I, ILM: Comparison of an interrupt level with the current level, interrupt enable state identification

Micro code: Interrupt processing step

#### ○ Addresses $\text{FFFC00}_\text{H}$ to $\text{FFFFFF}_\text{H}$ in memory

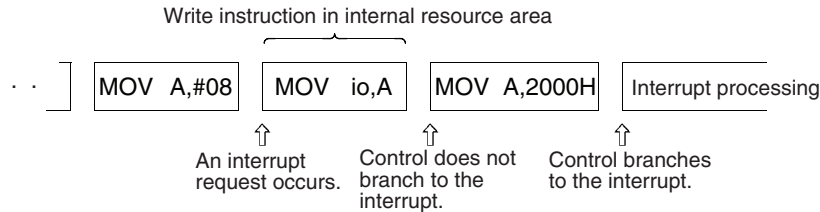
Interrupt vector table: The interrupt vector table to be referenced during interrupt processing is allocated to addresses and shared with software interrupts.

The internal resource reflects on the resource control register, the interrupt controller, the ICR, the CPU, and the CCR description. To use hardware interrupts, these three mechanisms must be set by software in advance. For the ICR, see Interrupt Control Register (ICR) in Section 3.6.1 "Interrupt Control Register (ICR)"

### ■ Interrupt Request during Writing in an Internal Resource Area

No hardware interrupt requests are accepted during writing in an internal resource area, thereby preventing the CPU from operating incorrectly for an interrupt request made while resource-interrupt-control-register-related writing is executed. The internal resource area is not an I/O addressing area (000000<sub>H</sub> to 0000FF<sub>H</sub>) but are areas allocated to the control register and data register of the internal resource.

**Figure 3.4-1 Hardware Interrupt Request during Writing in Internal Resource Area**



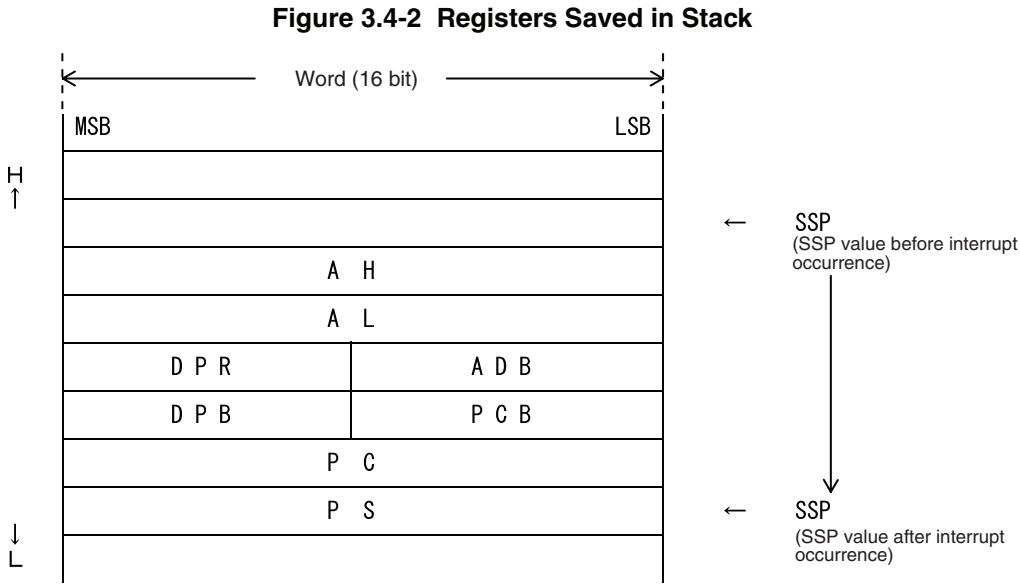
### ■ Interrupt Suppress Instruction

The F<sup>2</sup>MC-16LX provides an interrupt suppress instruction not to detect absence/presence of hardware interrupts. See Section 2.6.1 "Interrupt Suppression Instructions".

### ■ Multiple Interrupts

The F<sup>2</sup>MC-16LX CPU supports multiple interrupts. If an interrupt having an interrupt level higher than that of the executed instruction occurs, control passes to the former instruction following termination of the latter instruction. Following termination of the higher level instruction, control returns to the previous interrupt processing. If an interrupt having an interrupt level equal to or lower than that of the executed interrupt processing occurs, the new interrupt request is held until the executed interrupt processing terminates if the processing is not changed by the ILM description or I flag instruction. Multiple extended intelligent I/O services are not activated concurrently. While one extended intelligent I/O service is being processed, other interrupt requests and extended intelligent I/O service requests are held.

■ Registers Saved on Stack When an Interrupt Occurs



■ Notes on Hardware Interrupts

Some internal resources clear interrupt requests by a read operation of control registers and data registers. If an interrupt request is cleared by the read operation before control passes to interrupt processing hardware, a malfunction occurs.

Therefore, if an internal resource that clears interrupt requests by a register read operation is to be used, prevent the internal resource from reading registers at an interrupt occurrence.

## 3.4.1 Operation

---

**This section describes the operations from the occurrence of hardware interrupt request to the completion of the interruption processing.**

---

### ■ Hardware Interrupt Operation

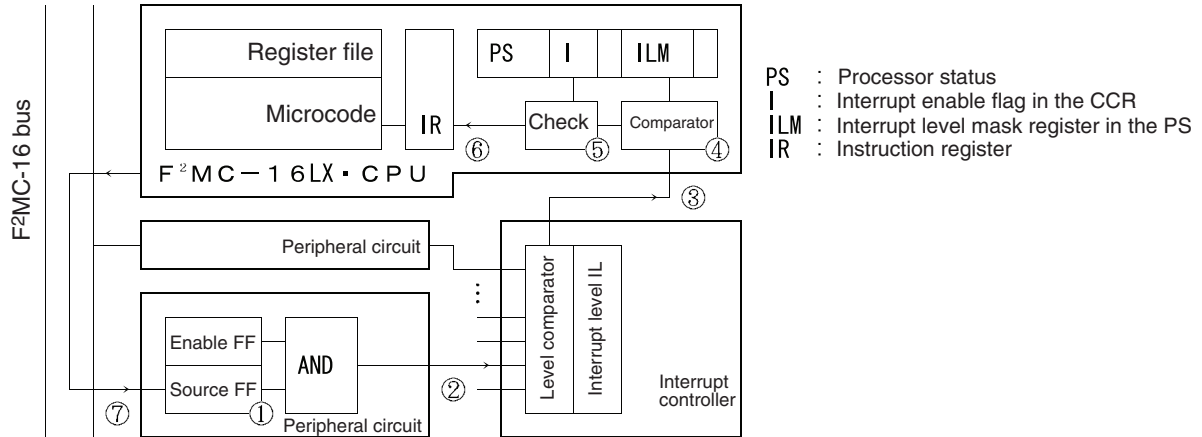
The internal resource with the hardware interrupt request function has an interrupt request flag indicating whether an interrupt is requested and an interrupt enable flag indicating whether the interrupt is requested to the CPU. The interrupt request flag is set when an event peculiar to the internal resource occurs. The resource issues an interrupt to the interrupt controller if the interrupt enable flag indicates enable.

In the ICRs, the interrupt controller compares the interrupt levels (IL) of the interrupt requests received concurrently. The controller selects the request with the highest level (the smallest IL value) and posts the request to the CPU. If two or more requests have the same level, the request with the smallest interrupt number is selected. The relationship between interrupts and between ICRs is dependent on hardware.

The CPU compares the received interrupt level (IL) with the interrupt mask register (ILM) in the processor status register (PS). If the I-bit in the condition code register (CCR) is set to 1, the CPU activates the interrupt processing macrocode following termination of the executed instruction. The ISE bit in the interrupt controller register (ICR) is referenced at the beginning of the interrupt processing macrocode to ensure that the bit is set to 0. The CPU then activates interrupt processing.

The interrupt processing first saves 12 bytes of PS, PC, PCB, DTB, ADB, DPR, and A on the system stack (memory indicated by SSB and SSP). Then, it loads the vector in the interrupt vector program counter (PC and PCB), updates the ILM in the PS to the level of the accepted interrupt request and sets the S flag to 1. ① to ⑦ in Figure 3.4-3 "Hardware Interrupt Operation to Release" are explained below.

Figure 3.4-3 Hardware Interrupt Operation



- ① An interrupt source occurs in the peripheral circuit.
- ② The interrupt enable bit in the peripheral circuit is checked to determine if an interrupt is to be enabled. If so, the peripheral circuit requests an interrupt to the interrupt controller.
- ③ Upon receiving the interrupt request, the interrupt controller checks the priorities of interrupts requested concurrently and sends the interrupt level of the request with the highest priority to the CPU.
- ④ The CPU compares the interrupt level requested from the interrupt controller with the ILM bit in the processor status register.
- ⑤ The CPU checks the contents of the I flag in the processor status register only if the priority of the requested interrupt is higher than the current interrupt level.
- ⑥ The CPU sets the ILM bit to the requested interrupt level (IL) only if the I flag checked in check ⑤ indicates the interrupt enabled status. The CPU then processes the interrupt following termination of the executed instruction and passes control to the interrupt processing routine.
- ⑦ The interrupt request terminates following clearance of the interrupt source that occurred in ① by software in the user interrupt processing routine.

The execution time of the interrupt processing performed by the CPU in ⑥ and ⑦ is shown below. The time elapsed before control is passed to the interrupt sequence is dependent on the address indicated by the stack pointer.

### ■ Processing Time for Hardware Interrupt

When an interrupt request was generated, a wait time for sampling interrupts and time for interrupt handling are required before the interrupt is accepted and the interrupt handling routine activated.

#### ○ Wait time for sampling interrupt requests

This is the time from the generation of the interrupt request to the termination of the instruction during execution. The interrupt request sampled in the final cycle of each instruction is used to determine whether an interrupt request was generated. Consequently, the CPU does not detect an interrupt request during the execution of an instruction, and a wait time occurs.

The wait time for sampling interrupt requests is longest when an interrupt request is generated immediately after starting POPW, and after PW0 to PW7 instructions, which have the longest execution cycle (45 machine cycles).



○ **Interrupt handling time (time required for preparation for interrupt processing)**

- Interrupt activation:  $24 + 6 \times Z$  machine cycles
- Return from interrupt:  $11 + 6 \times Z$  machine cycles (RETI instruction)

**Table 3.4-1 Correction Values of Number of Cycles of Interrupt Processing Time**

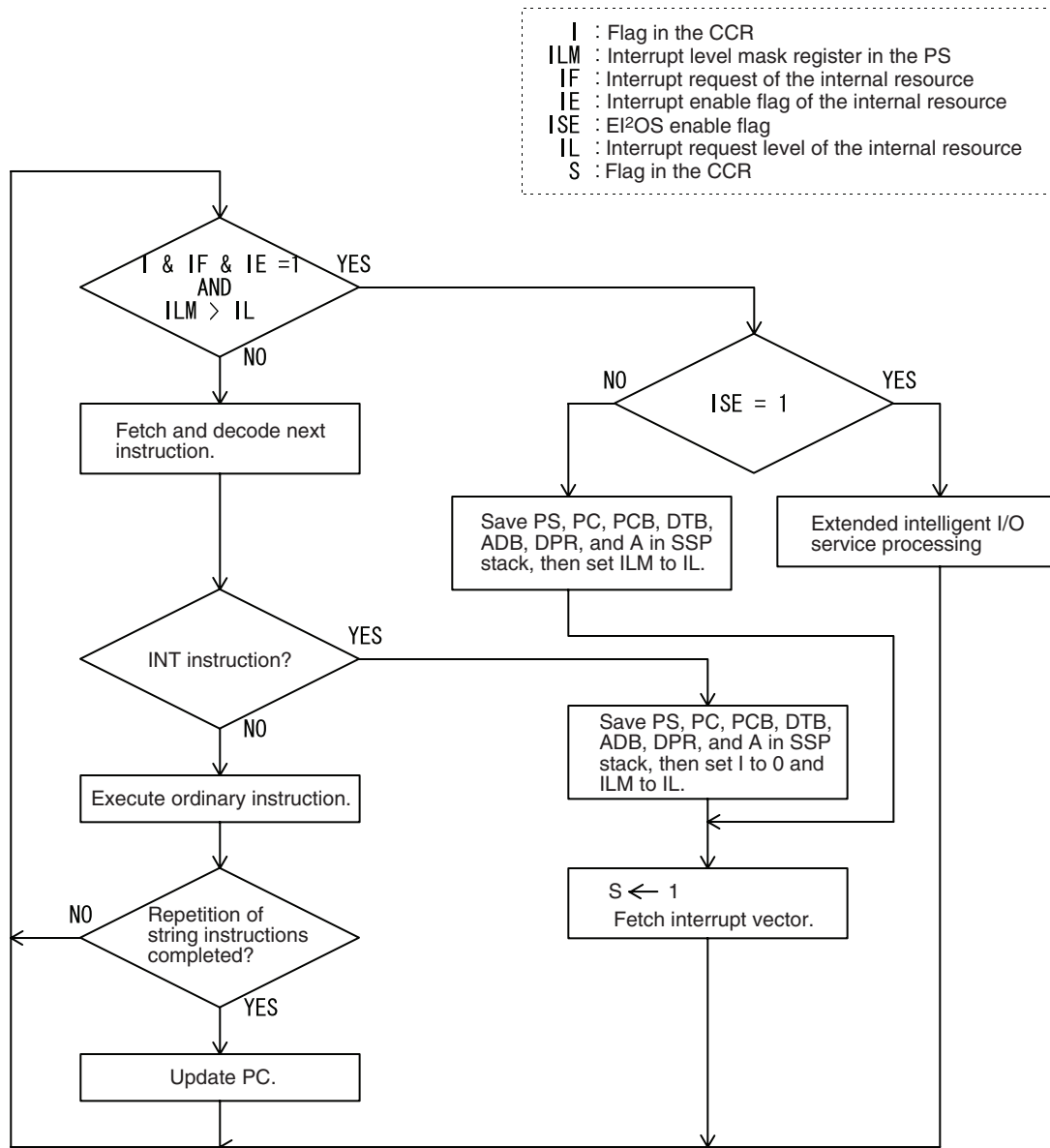
Address indicated by stack pointer	Correction value of the number of cycles (Z)
8-bit data bus in external area	+4
Even-numbered address in external area	+1
Odd-numbered address in external area	+4
Even-numbered address in internal area	0
Odd-numbered address in internal area	+2

## 3.4.2 Hardware Interrupt Operation Flowchart

Figure 3.4-4 "Hardware Interrupt Operation Flowchart" shows the operational flowchart for hardware interrupts.

### ■ Hardware Interrupt Operation Flowchart

Figure 3.4-4 Hardware Interrupt Operation Flowchart



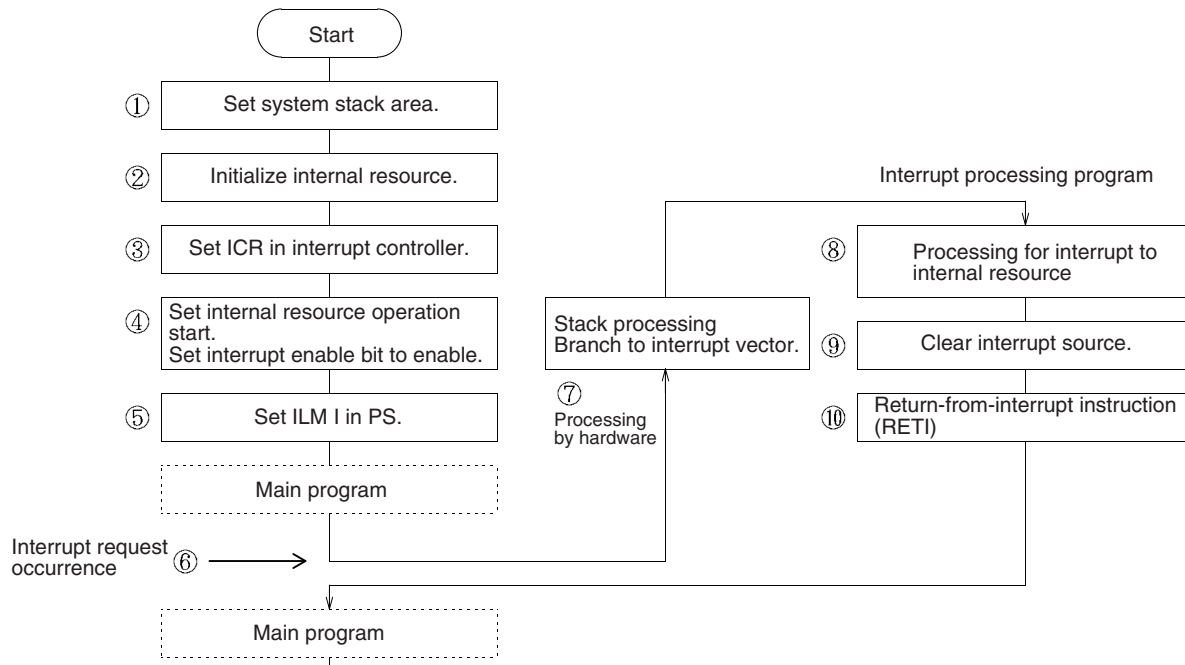
### 3.4.3 Sample Use Procedure of Hardware Interrupt

To use the hardware interrupt, a system stack area, peripheral features, and an interrupt control register (ICR) must be set.

#### ■ Flow Chart of Sample Use Procedure

Figure 3.4-5 "Sample Use Procedure of Hardware Interrupt" shows how a hardware interrupt is used.

**Figure 3.4-5 Sample Use Procedure of Hardware Interrupt**



- ① Set system stack area.
- ② Initialize internal resource that can generate an interrupt request.
- ③ Set ICR in interrupt controller.
- ④ Place internal resource in operation start state. Set interrupt enable bit to enable.
- ⑤ Set the I flag of the interrupt level mask register (ILM) in the processor status register (PS) and the I flag in the condition code register (CCR) to interrupt enabled.
- ⑥ Internal resource interrupt occurs, causing hardware interrupt request to occur.
- ⑦ Register is saved by interrupt processing hardware and control branches to interrupt processing program.
- ⑧ Perform processing of internal resource for interrupt occurrence by interrupt processing program
- ⑨ Release interrupt request of internal resource circuit.
- ⑩ Execute return-from-interrupt instruction and return to program executed before branch.

## 3.5 Software Interrupts

---

**The software interrupt function passes control to a user-defined interrupt processing program from a program being executed by the CPU according to execution of a dedicated instruction.**

---

### ■ Overview of Software Interrupts

The software interrupt function is always activated by executing a software interrupt instruction.

If a software interrupt occurs, the CPU responds as follows:

- Saves the descriptions of the PC, PS, A, PCB, DTB, ADB, and DPR registers in the CPU in the system stack.
- Sets the I flag in the condition code register (CCR). Interrupts are disabled automatically.
- Fetches the contents of the corresponding interrupt vector and branches control.

An interrupt caused by execution of an INT instruction (software interrupt) is not requested by interrupt request flag or interrupt enable flag. This interrupt is always requested when an INT instruction is executed.

Because the INT instruction has no interrupt level, the INT instruction does not update the ILM and sets the I flag to 0, thereby placing the following interrupt requests in a hold state.

### ■ Software Interrupt Mechanism

All mechanisms related to software interrupts are installed in the CPU.

#### ○ CPU

Microcode: Interrupt processing step

To use a software interrupt, the corresponding instruction must be executed.

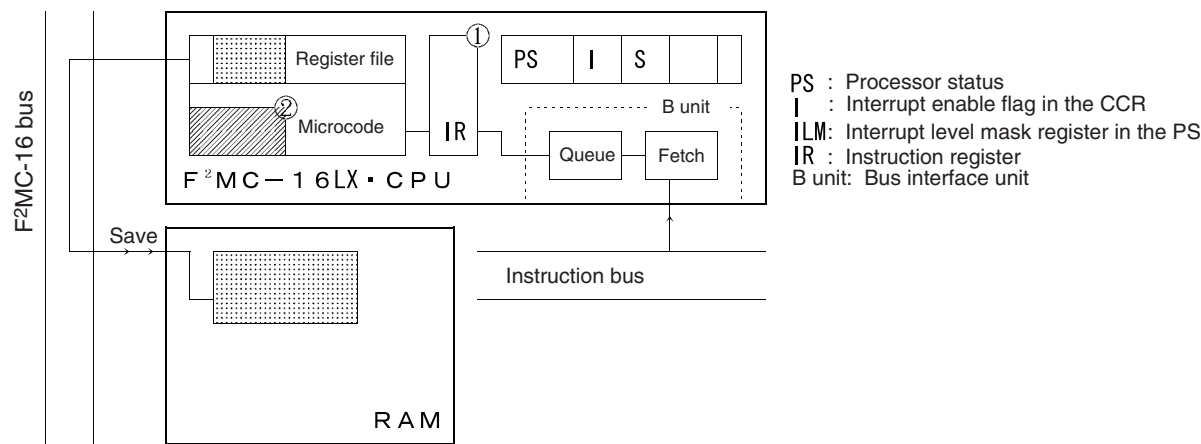
For interrupt vectors, as listed in Table 3.3-1 "List of Interrupt Vectors", hardware interrupts and software interrupts share the same area. For example, because interrupt request number INT 11 is used for an A/D converter interrupt (hardware interrupt) and for software interrupt INT #11, the A/D converter and INT #11 call the same interrupt processing routine for interrupt processing.

### ■ Software Interrupt Operation

When the CPU fetches and executes a software interrupt instruction, the microcode for software interrupt processing is activated. The microcode saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) in the memory areas indicated by SSB and SSP, reads a 3-byte interrupt vector and loads the vector in the program counters (PC and PCB). The microcode also sets the I flag to 0 and the S flag to 1 in the condition code register (CCR), thereby making the user-defined interrupt handling program the next instruction to be executed.

Figure 3.5-1 "Software Interrupt Occurrence to Release" shows the processing from the time a software interrupt is generated to the completion of interrupt processing.

Figure 3.5-1 Software Interrupt Occurrence to Release



- ① Executes a software interrupt instruction.
- ② Saves the dedicated registers in the CPU in the register file according to the microcode corresponding to the software interrupt instruction.
- ③ Interrupt processing terminates by a RETI instruction in the user interrupt processing routine.

■ Note on Software Interrupts

If the program bank register (PCB) indicates FFH, the vector area of a CALLV instruction overlaps with the INT #vct8 instruction table. Generate a program such that a CALLV instruction and an INT #vct8 instruction do not use the same address.

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

---

The extended intelligent I/O service (EI<sup>2</sup>OS) is a type of hardware interrupt operation that transfers data automatically between I/O and memory. It allows a data transfer between I/O and memory as direct memory access (DMA) with the interrupt processing program.

---

### ■ Overview of EI<sup>2</sup>OS Interrupt Processing

Compared with the conventional methods applied in interrupt processing programs, the extended intelligent I/O service (EI<sup>2</sup>OS) has the following advantages:

- The size of the program can be reduced because no program for data transfer requires description.
- Because internal registers are not used for data transfer, register saving is not required and transfer speeds are increased.
- Because I/O can stop data transfer for convenience, unnecessary data is not transferred.
- The incrementing or updating of buffer addresses can be selected.
- The incrementing or updating of I/O register addresses can be selected. (If buffer address update is specified)

Upon terminating data transfer, the EI<sup>2</sup>OS sets termination conditions to S1 and S0 bits in the interrupt control register (ICR) and branches control to an interrupt processing routine automatically, thereby enabling the user to determine the type of EI<sup>2</sup>OS termination conditions.

To implement the EI<sup>2</sup>OS, hardware is separated into two blocks, each containing the following register and descriptor.

#### ○ Interrupt control register (ICR):

When installed in the interrupt controller, indicates the status at EI<sup>2</sup>OS activation, EI<sup>2</sup>OS channel specification, and EI<sup>2</sup>OS termination.

#### ○ Extended intelligent I/O service descriptor (ISD):

Installed in RAM. Holds the transfer mode, I/O addresses and transfer counts, and buffer addresses.

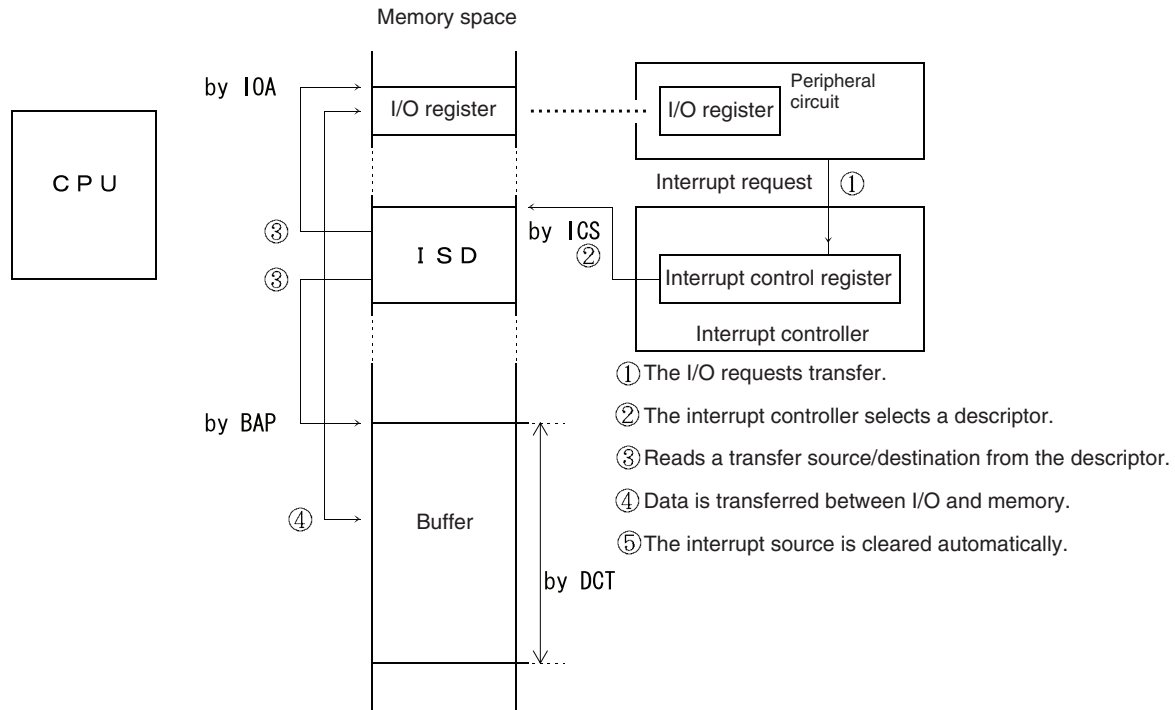
#### Note:

When REALOS is used, the extended intelligent I/O service (EI<sup>2</sup>OS) cannot be used.

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

Figure 3.6-1 "Outline of the Extended Intelligent I/O Service" outlines the EI<sup>2</sup>OS operation.

**Figure 3.6-1 Outline of the Extended Intelligent I/O Service**



**Note:**

The IOA can specify areas 000000<sub>H</sub> to 00FFFF<sub>H</sub>.

The BAP can specify areas 000000<sub>H</sub> to FFFFFF<sub>H</sub>.

The DCT can specify the transfer count up to 65536.

#### ■ Structure of the extended intelligent I/O service (EI<sup>2</sup>OS)

EI<sup>2</sup>OS-related structures can be divided into the following 4 categories:

- **Internal resource**

Interrupt enable bit, interrupt request bit: Control interrupt requests from resources

- **Interrupt controller**

ICR: Assigns levels to interrupts, decides priorities of concurrent interrupt requests, and selects EI<sup>2</sup>OS operation.

- **CPU**

I, ILM: Compares requested interrupt levels with the current level and identifies the interrupt enable state.

Microcode: EI<sup>2</sup>OS processing step

- **RAM**

Descriptor: Describes EI<sup>2</sup>OS transfer information.



### 3.6.1 Interrupt Control Register (ICR)

Interrupt control registers are installed in the interrupt controller. An ICR register corresponds to each I/O having an interrupt function. An ICR register has the following functions:

- Setting an interrupt level of the corresponding peripheral circuit
- Selecting an interrupt from the corresponding peripheral circuit as an ordinary interrupt or an extended intelligent I/O service
- Selecting the channel of the extended intelligent I/O service

Accessing this register by a read modify write instruction will result in a malfunction.

#### ■ Interrupt control register (ICR)

**Figure 3.6-2 Interrupt Control Register (ICR)**

Interrupt control register (ICR)	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	⇔ Bit number
Address: B0 <sub>H</sub> to BF <sub>H</sub>	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	For write operation
Read/write ⇔	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇔	(0)	(0)	(0)	(0)	(0)	(1)	(1)	(1)	
Address: B0 <sub>H</sub> to BF <sub>H</sub>	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	⇔ Bit number
	—	—	S1	S0	ISE	IL2	IL1	IL0	For read operation
Read/write ⇔	(—)	(—)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇔	(X)	(X)	(0)	(0)	(0)	(1)	(1)	(1)	

**Note:**

ICS3 to ICS0 are valid only when the EI<sup>2</sup>OS is to be activated. To activate the EI<sup>2</sup>OS, set ISE to 1. To not activate the EI<sup>2</sup>OS, set ISE to 0. When the EI<sup>2</sup>OS is not activated, ICS3 to ICS0 can be set as required.

ICS1 and ICS0 are valid for writing only. S1 and S0 are valid for reading only.

**[Bits 15 to 12][Bits 7 to 4]: ICS3 to ICS0 (extended intelligent I/O service channel select bits)**

EI<sup>2</sup>OS channel select bits. These bits are provided for writing only. EI<sup>2</sup>OS channels are specified by these bits. The address of an extended intelligent I/O service descriptor in memory is determined by the value set in the bits. The ICS bits are initialized to 0000 by a reset.

**Table 3.6-1 Correspondence between EI<sup>2</sup>OS Channel Selection Bits and Descriptor Addresses**

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

**[Bits 13 and 12][Bits 5 and 4]: S0, S1 (extended intelligent I/O service status)**

EI<sup>2</sup>OS termination status bit. These bits are provided for reading only. By checking the value of these bits at termination of the EI<sup>2</sup>OS, the termination conditions can be determined.

The bits are initialized to 00 at a reset.

**Table 3.6-2 Relationship between EI<sup>2</sup>OS Status Bits and EI<sup>2</sup>OS Status**

SI	S0	Termination condition
0	0	EI <sup>2</sup> OS is in operation or not activated.
0	1	Stop status due to count termination
1	0	Reserved
1	1	Stop status due to a request from an internal resource

**[Bit 11][Bit 3]: ISE (extended intelligent I/O service enable bit)**

EI<sup>2</sup>OS enable bit. If this bit is set to 1 at occurrence of an interrupt request, the EI<sup>2</sup>OS is activated. If the bit is set to 0, the interrupt sequence is activated. The ISE bit is set to 0 when the EI<sup>2</sup>OS terminates (due to count termination or a request from an internal resource).

If the corresponding internal resource has no EI<sup>2</sup>OS function, the ISE bit must be set to 0 by software. This bit is readable and can be written. The bit is initialized to 0 by a reset.

**[Bits 10 to 8][Bits 2 to 0]: IL0, IL1, IL2 (interrupt level setting bits)**

Interrupt level setting bits. These bits specify the level of a corresponding internal resource interrupt. These bits are readable and can be written. These bits are initialized to level 7 (no interrupt) by a reset.

**Table 3.6-3 Correspondence between Interrupt Level Setting Bits and Interrupt Levels**

IL2	IL1	IL0	Interrupt level
0	0	0	0 (highest-level interrupt)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (lowest-level interrupt)
1	1	1	7 (No interrupt)

## 3.6.2 Extended Intelligent I/O Service Descriptor (ISD)

The extended intelligent I/O service descriptor (ISD) is located at 000100<sub>H</sub> to 00017F<sub>H</sub> in internal RAM and contains the following data:

- Various types of control data for data transfer
- Status data
- Buffer address pointers

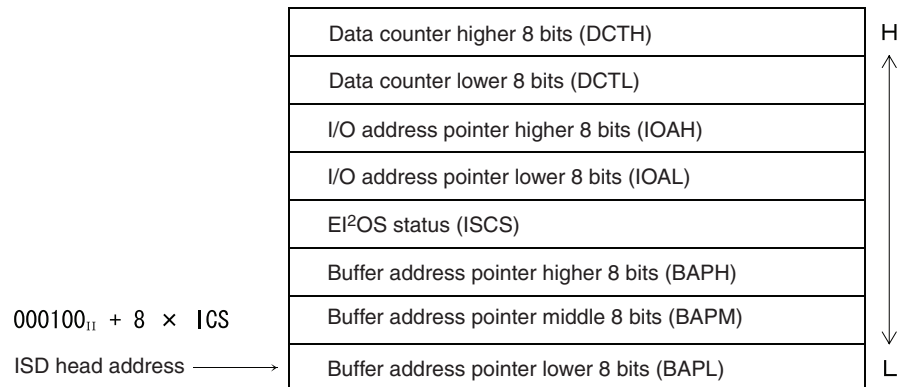
### ■ Extended intelligent I/O service descriptor (ISD)

The ISD is in 000100<sub>H</sub> to 00017F<sub>H</sub> in internal RAM and contains:

- Various types of control data for data transfer
- Status data
- Buffer address pointer

Figure 3.6-3 "Structure of Extended Intelligent I/O Service Descriptor" shows the structure of the extended intelligent I/O service descriptor.

**Figure 3.6-3 Structure of Extended Intelligent I/O Service Descriptor**



### 3.6.3 Registers of the Extended Intelligent I/O Service Descriptor (ISD)

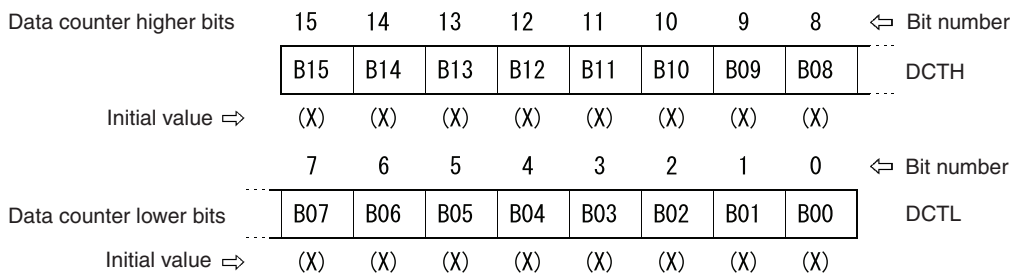
The extended intelligent I/O service descriptor (ISD) contains the following registers:

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

#### ■ Data Counter (DCT)

A 16 bit register used as a counter for counting transfer data items. After data transfer, this counter is decremented by one. When this counter is 0, the EI<sup>2</sup>OS terminates.

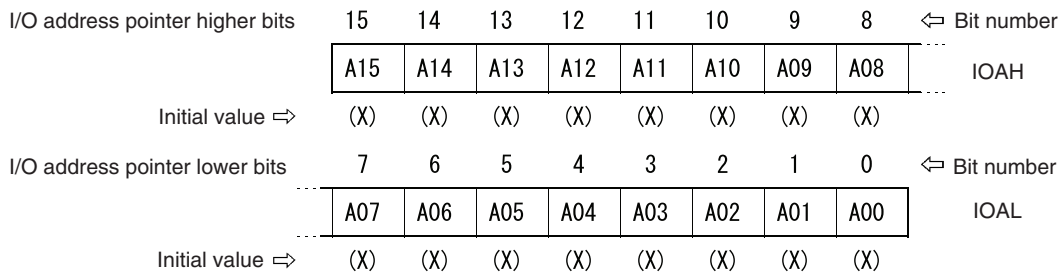
**Figure 3.6-4 Structure of Data Counter (DTC)**



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

A 16 bit register that indicates a lower address (A15 to A0) of the I/O register with which a buffer switches data. All higher address bits (A23 to A16) are set to 0. An I/O from 000000<sub>H</sub> to 00FFFF<sub>H</sub> can be specified.

**Figure 3.6-5 Structure of I/O Register Address Pointer (IOA)**



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

An 8 bit register that updates/fixes the buffer address pointer and I/O register address pointer and indicates a unit of transfer data length (bytes/words) and transfer direction.

**Figure 3.6-6 Structure of EI<sup>2</sup>OS Status Register (ISCS)**

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

	7	6	5	4	3	2	1	0	↔ Bit number
	Reserved	Reserved	Reserved	IF	BW	BF	DIR	SE	
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)	

#### [Bits 7 to 5]: Reserved bits

Write 0.

#### [Bit 4]: IF

Specifies whether the I/O register address pointer is to be updated or fixed.

**Table 3.6-4 IF Bit Functions**

IF	Function
0	The I/O register address pointer is updated (incremented) after data transfer.
1	The I/O register address pointer is fixed after data transfer.

#### [Bit 3]: BW

Specifies a unit of transfer data length.

**Table 3.6-5 BW Bit Functions**

BW	Function
0	Bytes
1	Words

#### [Bit 2]: BF

Specifies whether the buffer address pointer is to be updated or fixed.

**Table 3.6-6 BF Bit Functions**

BF	Function
0	The buffer address pointer is updated (incremented) after data transfer.
1	The buffer address pointer is fixed after data transfer.

#### Note:

Only the buffer address pointer can be incremented because only the lower 16 bits are changed.

### [Bit 1]: DIR

Specifies the direction of data transfer.

**Table 3.6-7 DIR Bit Functions**

DIR	Function
0	I/O address pointer --> Buffer address pointer
1	Buffer address pointer --> I/O address pointer.

### [Bit 0]: SE

Controls termination of the extended intelligent I/O service on request from an internal resource.

**Table 3.6-8 EI<sup>2</sup>OS Termination Control Bits**

SE	Function
0	The service is not terminated by a request from an internal resource.
1	The service is terminated by a request from an internal resource.

## ■ Buffer Address Pointer (BAP)

A 24 bit register that holds an address to be used next for transfer by the EI<sup>2</sup>OS. Because a BAP is installed for each EI<sup>2</sup>OS channel separately, an EI<sup>2</sup>OS channel can be used for transfer with anywhere in the 16-Mbyte space.

### **Note:**

If the BF bit in the EI<sup>2</sup>OS status register (ISCS) is set to "update", the lower 16 bits (BARM and BAPL) of the BAP change, but the higher 8 bits (BAPH) do not change.

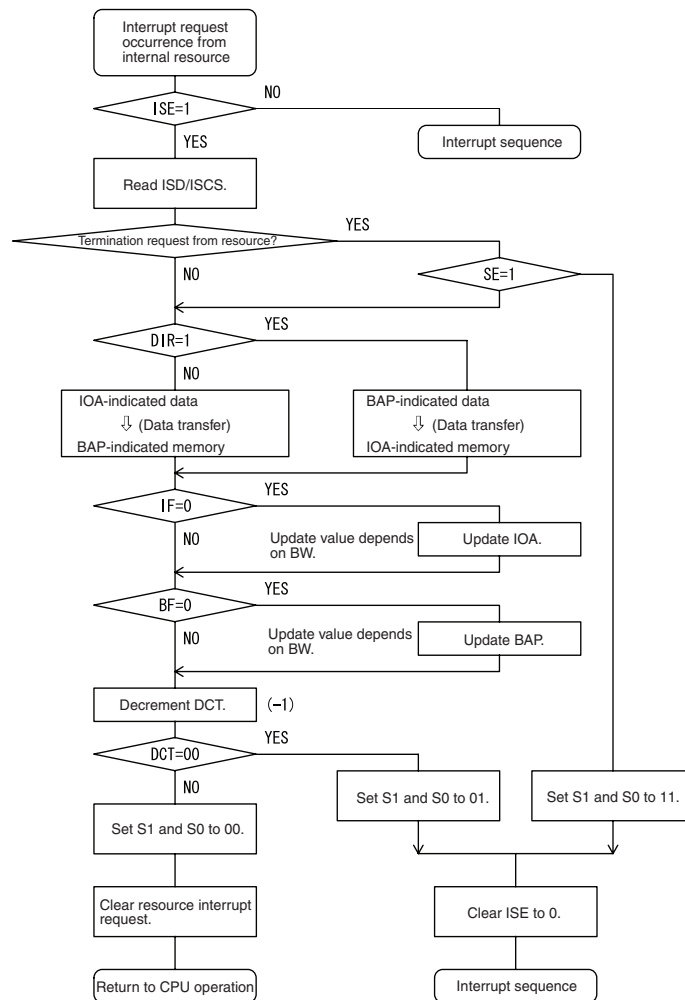
## 3.6.4 Operation

When an interrupt request is issued from the peripheral functions and the EI<sup>2</sup>OS activation is specified in corresponding interrupt register (ICR), the CPU transfers the data by EI<sup>2</sup>OS. When all the specified number of data transfers are completed, the hardware interrupt processing automatically starts.

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

Figure 3.6-7 "EI<sup>2</sup>OS Operation Flow" shows the operation flow of EI<sup>2</sup>OS operation using CPU-internal microcode.

Figure 3.6-7 EI<sup>2</sup>OS Operation Flow



BAP : Buffer address pointer  
 IOA : I/O register address pointer  
 ISD : EI<sup>2</sup>OS descriptor  
 ISCS : EI<sup>2</sup>OS status  
 DCT : Data counter  
 ISE : EI<sup>2</sup>OS enable bit  
 S1, S0 : EI<sup>2</sup>OS termination status  
 IF : IOA update/fix selection bit of EI<sup>2</sup>OS status register (ISCS)  
 BW: Transfer data length specification bit of EI<sup>2</sup>OS status register (ISCS)  
 BF: BAP update/fix selection bit of EI<sup>2</sup>OS status register (ISCS)  
 DIR: Data transfer direction specification bit of EI<sup>2</sup>OS status register (ISCS)  
 SE : EI<sup>2</sup>OS termination control bit of EI<sup>2</sup>OS status register (ISCS)



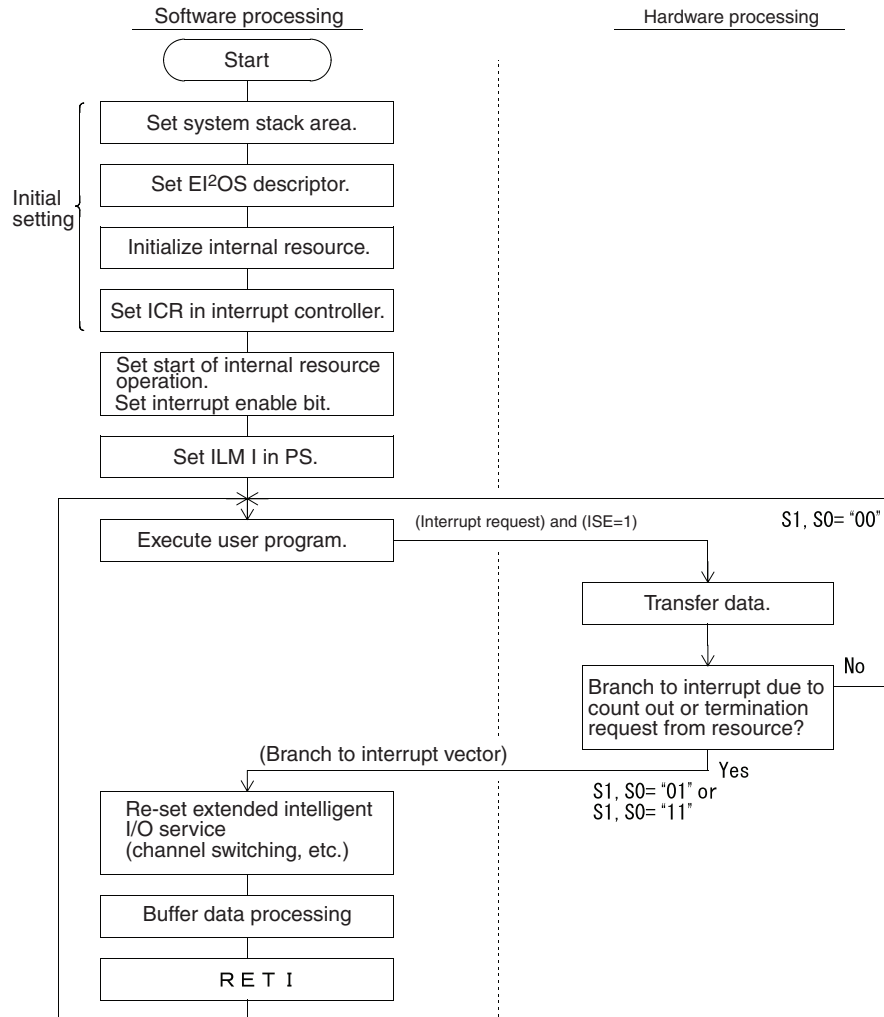
### 3.6.5 Procedure for Using the Extended Intelligent I/O Service (EI<sup>2</sup>OS)

Use of the extended intelligent I/O service (EI<sup>2</sup>OS) requires that the system stack area, extended intelligent I/O service (EI<sup>2</sup>OS) descriptor, peripheral functions, and interrupt control register (ICR) be specified.

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

Figure 3.6-8 "EI<sup>2</sup>OS Use Procedure Flow" shows the processing of the extended intelligent I/O service (EI<sup>2</sup>OS) in terms of hardware and software.

**Figure 3.6-8 EI<sup>2</sup>OS Use Procedure Flow**



ISE : EI<sup>2</sup>OS enable bit of interrupt control register (ICR)

S1, S0: EI<sup>2</sup>OS status of interrupt control register (ICR)

### 3.6.6 EI<sup>2</sup>OS Execution Time

The execution time for the extended intelligent I/O service (EI<sup>2</sup>OS) changes depending on the following:

- Setting of the EI<sup>2</sup>OS status register (ISCS)
- 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 operations
- Data length of transfer data

Moreover, an interrupt handling time is added because a hardware interrupt is activated when the EI<sup>2</sup>OS terminates data transfer.

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

##### ○ When data transfer continues (the stop conditions are not met)

The EI<sup>2</sup>OS execution time when data transfer continues is determined by the EI<sup>2</sup>OS status register (ISCS) setting, as shown in Table 3.6-9 "Execution Time when EI<sup>2</sup>OS Continues".

**Table 3.6-9 Execution Time when EI<sup>2</sup>OS Continues**

Settings of EI <sup>2</sup> OS termination control bits (ISCS and SE)		Set to 0		Set to 1	
I/O address pointer		Fixed	Updated	Fixed	Updated
Setting of BAP address update/fix selection bit (BF)	Fixed	32	34	33	35
	Updated	34	36	35	37

Unit: Machine cycle (1 machine cycle is equivalent to 1 clock cycle of the machine clock ( $\phi$ )).

Values require correction depending on the conditions at the time of EI<sup>2</sup>OS execution, as shown in Table 3.6-10 "Correction Value of Data Transfer of EI<sup>2</sup>OS Execution Time".

**Table 3.6-10 Correction Value of Data Transfer of 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

Even: Even-numbered address/word transfer

8: 8-bit external bus width/word transfer

Odd: Odd-numbered address/word transfer

○ **At the time of count termination of the data counter (DCT) (at the time of the final data transfer)**

An interrupt handling time is added because a hardware interrupt is activated when the EI<sup>2</sup>OS terminates data transfer. EI<sup>2</sup>OS execution time at the time of count termination is calculated (using the formula) as follows:

EI<sup>2</sup>OS execution time at the time of count termination = EI<sup>2</sup>OS execution time at the time of data transfer  
+ interrupt handling time (21+ 6 x Z) machine cycles

The interrupt handling time depends on the address indicated by the stack pointer. Table 3.6-11 "Correction Value (Z) of Interrupt Handling Time" lists correction value (Z) of interrupt handling time.

**Table 3.6-11 Correction Value (Z) of Interrupt Handling Time**

Address indicated by the stack pointer	Correction value (Z)
External 8-bit	+4
Even address at external access	+1
Odd address at external access	+4
Even address at internal access	0
Odd address at internal access	+2

#### ○ At termination following a termination request from the peripheral functions (I/O)

When data transfer by the EI<sup>2</sup>OS terminated in the middle of the execution (ICR: S1, S0 = 11) following a termination request from peripheral functions (I/O), no data is transferred and only a hardware interrupt is thrown. In this case, the EI<sup>2</sup>OS execution time can be calculated using the following formula. Z in the formula indicates the correction value of the interrupt processing time. (See Table 3.6-11 "Correction Value (Z) of Interrupt Handling Time".)

EI<sup>2</sup>OS execution time when processing terminates midway through =  $36 + 6 \times Z$  machine cycles

**Note:**

1 machine cycle is equivalent to 1 clock cycle of the machine clock ( $\phi$ ).

## 3.7 Exception

---

In the F<sup>2</sup>MC-16LX, the execution of an undefined instruction results in an exception and exception processing, which is basically the same as interrupt processing. When an exception is detected at the boundary of an instruction, control passes from ordinary processing to exception processing. Because exception processing generally occurs if an undefined operation is executed, exception processing should be used only for debugging or as an emergency measure to activate recovery software.

---

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

The F<sup>2</sup>MC-16LX treats all codes not defined in the instruction map as undefined instructions. If an undefined instruction is executed, processing equivalent to software interrupt instruction INT 10 is performed. That is, the AL, AH, DPR, DTB, ADB, PCB, PC, and PS descriptions are saved in the system stack. The I flag of the condition code register (CCR) is set to 0 and the S flag is set to 1, and control branches to the routine indicated by the vector of interrupt number 10. The PC value saved in the stack indicates the address storing the undefined instruction. For an instruction code of two bytes or more, because the address stores the code that can be recognized as an undefined code, control can be returned by a RETI instruction, though it is meaningless because the exception occurs again.

# CHAPTER 4    CLOCK AND RESET

---

**This chapter describes the functions and operations of the clock and reset.**

---

4.1 "Clock Generator"

4.2 "Clock Supply Map"

4.3 "Reset Source"

4.4 "Operation after a Reset is Released"

## 4.1 Clock Generator

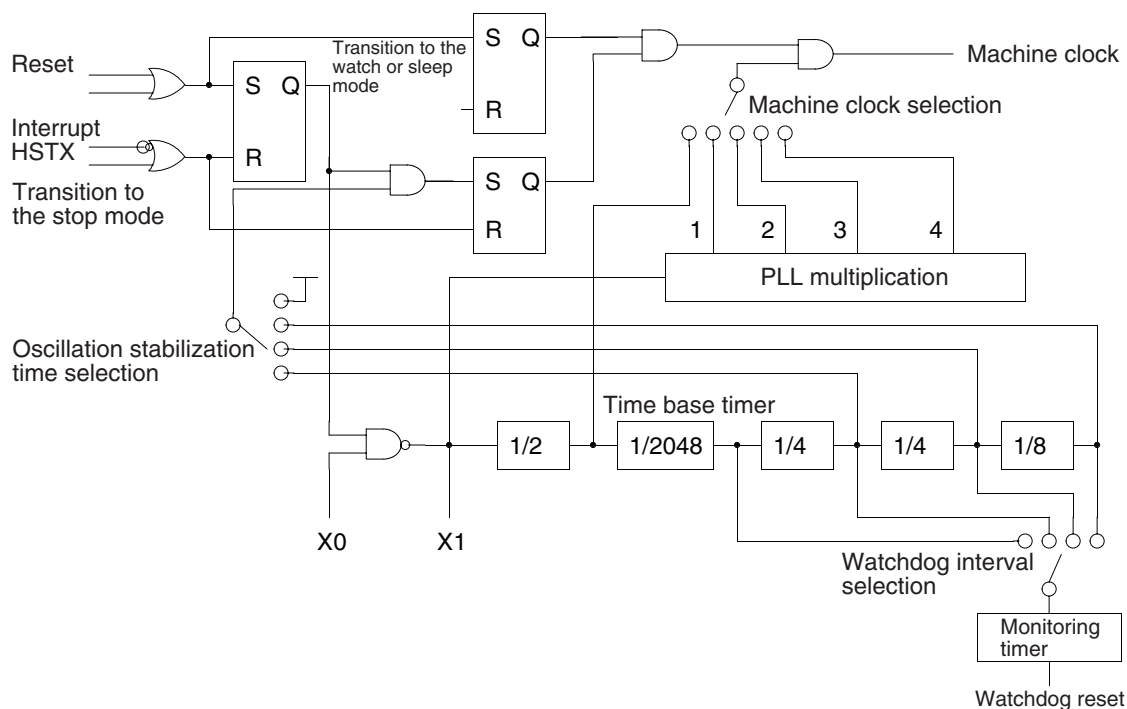
The clock generator controls operation of the internal clock, including the sleep, watch, and stop modes and the PLL clock multiplication functions. The internal clock is called a machine clock; one cycle of which is used as a machine cycle. The clock generated by OSC oscillation is called an oscillation clock. The clock generated by internal VCO oscillation is called a PLL clock.

### ■ Notes for the Clock Generator

When the operating voltage is 5 V, the OSC oscillation frequency range is from 3 to 16 MHz, but the maximum operating frequency of the CPU and peripheral circuits is 16 MHz. If the frequency generated by the specified multiplication factor exceeds the maximum operating frequency, the CPU and peripheral circuits do not operate normally. For example, if the frequency of OSC oscillation is 16 MHz, only 1 can be specified as the multiplication factor.

The minimum operating frequency of VCO oscillation is 4 MHz; any frequency less than 4 MHz cannot be specified.

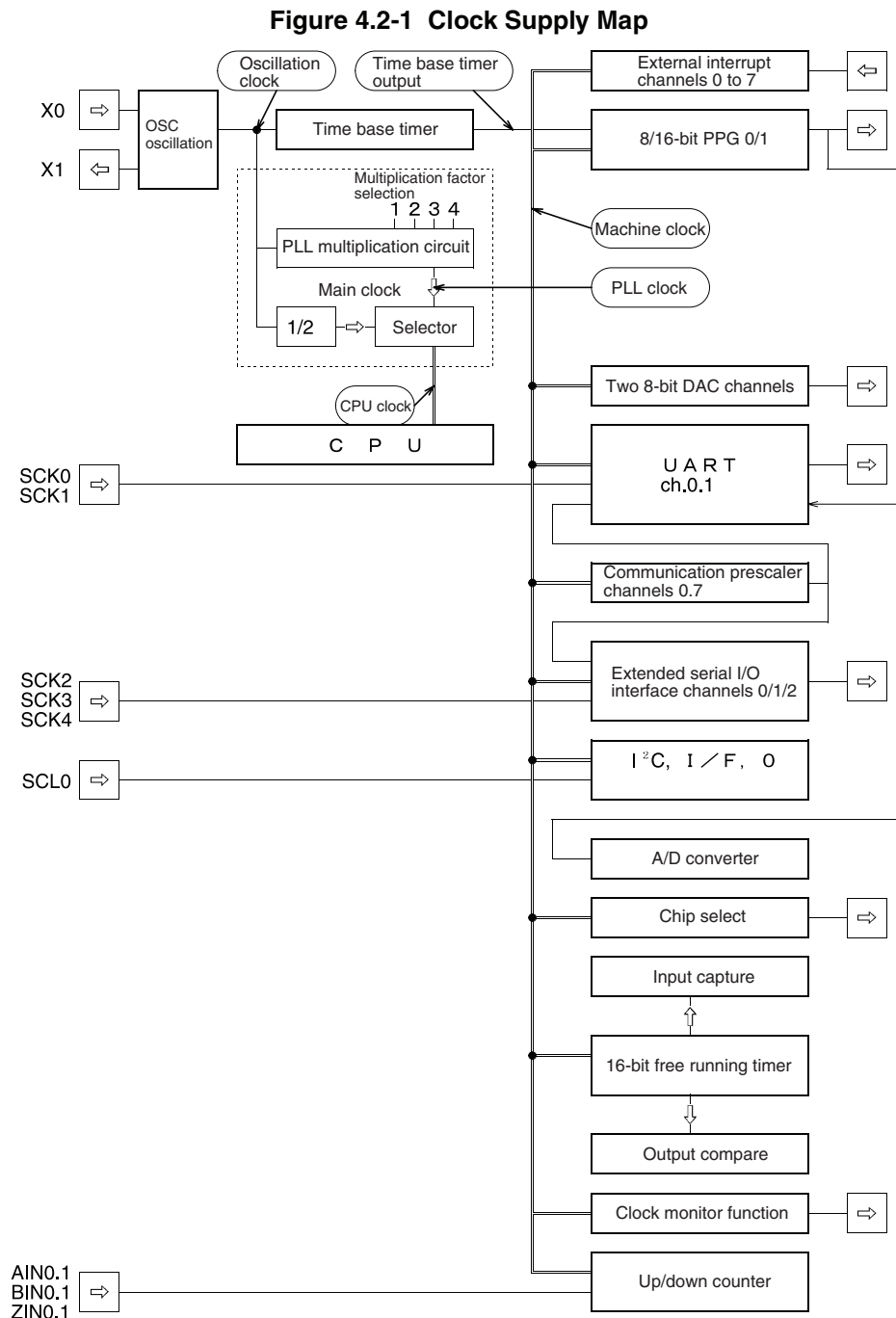
**Figure 4.1-1 Clock Generator Block Diagram**



## 4.2 Clock Supply Map

Figure 4.2-1 "Clock Supply Map" shows the clock supply map.

### ■ Clock Supply Map





## 4.3 Reset Source

The five types of reset sources are as follows:

- Occurrence of a power-on reset
- Release of the hardware standby state
- Watchdog timer overflow
- Occurrence of an external reset request by the RSTX pin
- Occurrence of a reset request by the software

### ■ Reset Source

When the stop mode is released or a power-on reset occurs, operation starts following elapse of the oscillation stabilization time.

When a reset factor occurs, the F<sup>2</sup>MC-16LX immediately stops executing current processing and enters the reset release wait state.

The machine clock and watchdog function initial states differ depending on the reset factor.

The reset factor bits in the watchdog control register can be checked to determine the reset factor.

#### Note:

Because the external reset input is sampled in synchronization with the internal clock in a mode other than the stop mode, no reset input is accepted when the externally supplied clock stops.

When the external bus is used and a reset factor occurs, the address generated by each device during reset is undefined. All external bus access signals, such as RDX and WRX, become inactive.

**Table 4.3-1 Reset Factors**

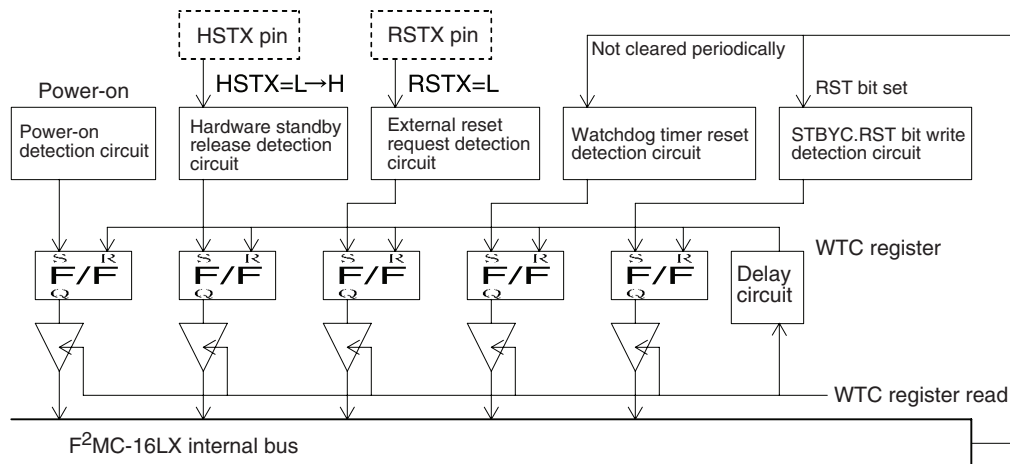
Reset	Factor	Machine clock	Watchdog timer	Oscillation stabilization time
Power-on	The power is turned on.	Main clock	Stopped	Required
Hardware standby	The HSTX pin input becomes low.	Main clock	Stopped	Required
Watchdog timer	The watchdog timer overflows.	Main clock	Stopped	Required
External pin	The RSTX pin input becomes low.	Retains the pre-reset status.	Retains the pre-reset status.	Not required
Software	"0" is written in the RST bit in the STBYC.	Retains the pre-reset status.	Retains the pre-reset status.	Not required

- When the reset input is received in the stop or hardware standby mode, the oscillation stabilization time is required for a reset factor.
- These bits are not initialized by a reset other than a power-on reset. When a power-on reset

occurs, these bits are initialized to "11", and so the oscillation stabilization time for the main clock at power-on is about  $2^{18}$  OSC oscillation cycles for evaluation products and about  $2^{17}$  OSC oscillation cycles for FLASK/MASK products. Other oscillation stabilization times are determined by the WS1/WS0 ratio of the clock selection register (CKSCR).

There is a flip-flop corresponding to each reset factor. The status of each flip-flop can be checked by reading the watchdog control register. To identify the reset factor after releasing a reset, ensure that branching of an appropriate program occurs after the value read from the watchdog control register is processed by software.

**Figure 4.3-1 Reset Factor Bit Block Diagram**



When multiple reset factors occur, the corresponding reset factor bits in the watchdog control register are set to "1". When an external reset request and watchdog reset occur simultaneously, the ERST and WRST bits are both set to "1".

This rule does not apply to a power-on reset. When the PONR bit is "1", the values of the other bits do not indicate normal reset factors, and so a software program should be created that ignores the values of other bits when the PONR bit is "1".

**Table 4.3-2 Correspondence between the Reset Factors and the Values of the Reset Factor Bits**

Reset factor	PONR	STBR	WRST	ERST	SRST
Power-on	1	—	—	—	—
Hardware standby	*	1	*	*	*
Watchdog timer	*	*	1	*	*
External pin	*	*	*	1	*
RST bit	*	*	*	*	1

(An asterisk (\*) indicates that the value before reset is retained.)

**Note:**

The reset factor bits are cleared only when the watchdog control register is read. The reset factor bit corresponding to a reset factor that has occurred once remains set to "1" when another reset factor occurs.

For the configuration of the watchdog control register and the reset factor bits, see CHAPTER 10 "WATCHDOG TIMER".

## 4.4 Operation after a Reset is Released

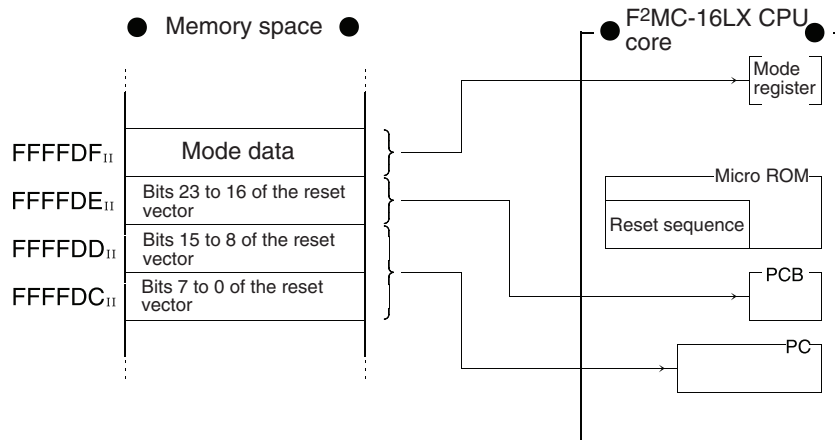
When the reset factor is removed, the F<sup>2</sup>MC-16LX immediately outputs the address at which the reset vector is stored and fetches the reset vector and mode data. A 4-byte area at FFFFDC<sub>H</sub> to FFFFDF<sub>H</sub> is allocated for the reset vector and mode data, which are transferred to the corresponding registers by hardware following a reset release.

### ■ When Reset is Released

Use the mode pins to specify the internal ROM or external memory from which to read the reset vector and mode data. When the external vector mode is specified using the mode pins, the reset vector and mode data are read from the external memory not the internal ROM, and so when the microcontroller is to be used in single chip mode or internal ROM and external bus mode, specifying the internal vector mode using the mode pins is recommended.

The bus mode after the reset vector and mode data are read is specified by mode data.

**Figure 4.4-1 Locations and Destinations of the Reset Vector and of Mode Data**



### Note:

The contents of the mode register in the figure above are undefined immediately after reset. Store any mode data in memory space in advance to ensure that a write operation will be performed.

# CHAPTER 5    LOW-POWER CONSUMPTION CONTROL CIRCUIT

---

**This chapter describes the functions and operations of the low-power consumption control circuit.**

---

- 5.1 "Overview of the Low-Power Consumption Control Circuit"
- 5.2 "Block Diagram of the Low-Power Consumption Control Circuit"
- 5.3 "Registers of the Low-Power Consumption Control Circuit"
- 5.4 "Status Transition for Clock Selection"

## 5.1 Overview of the Low-Power Consumption Control Circuit

---

The low-power consumption control circuit normally uses the low-power consumption mode as the operating mode. The intermittent CPU operation function and oscillation stabilization time can be set by setting the corresponding register bits.

In the entire block diagram, the low-power consumption control circuit is part of the clock control circuit (see Section 1.3 "Block Diagram" of the MB90570 Series).

---

### ■ Operating Modes of the Low-Power Consumption Control Circuit

The operating modes are as follows:

- PLL clock mode
- PLL sleep mode
- PLL watch mode
- Pseudo watch mode
- Main clock mode
- Main clock sleep mode
- Main clock watch mode
- Main clock stop mode
- Subclock mode
- Subclock sleep mode
- Subclock watch mode
- Subclock stop mode
- Hardware standby mode

Operating modes other than the PLL clock mode are categorized as low-power consumption modes.

### ■ Intermittent CPU Operation Function

The intermittent CPU operation function stops the clock provided to the CPU and delays the start of the internal bus cycle when a register, internal memory, internal peripheral, or external bus is accessed. Processing can be performed with low-power consumption because CPU execution speed is reduced with the provision of a high-speed clock to the internal peripherals. Use the CG1 and CG0 bits in the LPMCR to select the temporary stop cycle count for the clock provided to the CPU.

The external bus operation itself is performed using the same clock as that used for the peripherals.

The execution time required when the intermittent CPU operation function is used can be obtained as follows:

- Adds the compensation value obtained by multiplying the number of accesses of registers, internal memories, internal peripherals, and external buses by the temporary stop cycle count of the normal execution time.

### ■ Setting of the Oscillation Stabilization Time for the Main Clock

Use the WS1 and WS0 in the CKSCR bits to select the oscillation stabilization time for the main clock required when the stop or hardware standby mode is released. Select the oscillation stabilization time according to the types and characteristics of the oscillation device and the oscillator connected to the X0 and X1 pins.

These bits are not initialized by a reset other than a power-on reset. When a power-on reset occurs, these bits are initialized to "11". and so the oscillation stabilization time for the main clock at power-on is about  $2^{18}$  OSC oscillation cycles for evaluation products and about  $2^{17}$  OSC oscillation cycles for FLASK/MASK products.

### ■ Switching of the Machine Clock

#### ○ Switching between the main and the PLL clocks

Writing to the MCS bit in the CKSCR register switches between the main and the PLL clocks.

Setting the MCS bit from "1" to "0" switches the machine clock from the main clock to the PLL clock after the oscillation stabilization time for the PLL clock ( $2^{12}$  machine clock cycles) has elapsed.

Setting the MCS bit from "0" to "1" switches the machine clock from the PLL clock to the main clock. This switch occurs when a PLL clock edge coincides with a main clock edge (after one to eight PLL clock cycles).

When the MCS bit is rewritten, the machine clock is not switched immediately. Operate each peripheral according to the machine clock after referencing the MCM bit in the CKSCR to confirm that the machine clock has been switched.

### ○ Switching between the main clock and subclock

Writing to the SCS bit in the CKSCR register switches between the main clock and subclock.

Setting the SCS bit from "1" to "0" switches the machine clock from the main clock to the subclock. This switch occurs when a subclock edge is detected.

Setting the SCS bit from "0" to "1" switches the machine clock from the subclock to the main clock after the oscillation stabilization time for the main clock has elapsed.

When the SCS bit is rewritten, the machine clock is not switched immediately. Operate each peripheral after checking whether it is dependent on the machine clock.

#### **Note:**

When tune on the power or hardware standby mode or stop mode is released, the subclock oscillation stabilization time (about 2 seconds) is generated. In the meantime, when switching from the main clock mode to the subclock mode, the oscillation stabilization time is generated.

### ○ Machine clock initialization

The MCS and SCS bits are not initialized by a reset by an external pin or the RST bit in the LPMCR, but are initialized to "1" by other resets.

## ■ PLL Clock Multiplication Function

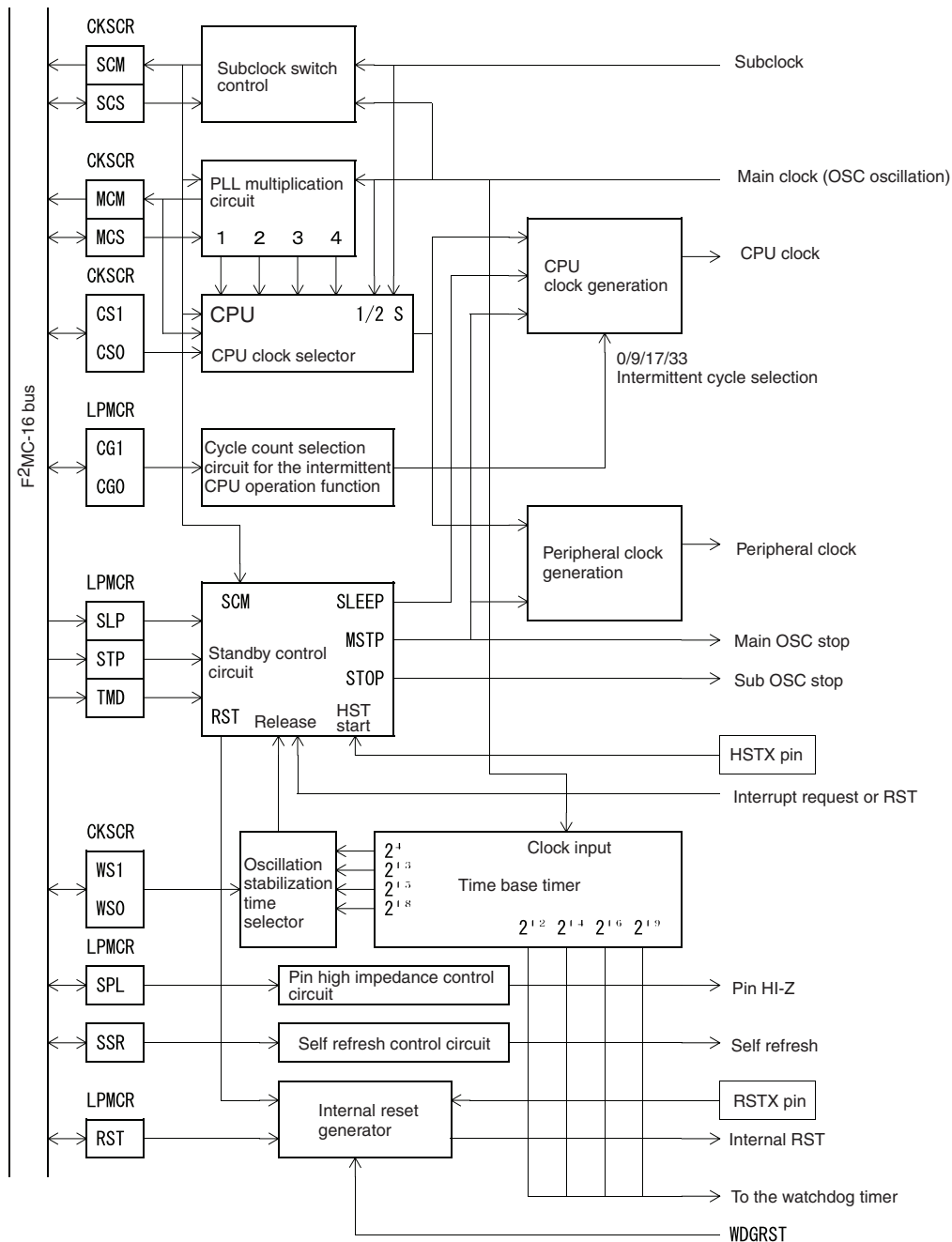
A PLL clock multiplication factor can be selected among 2, 4, 6, and 8 using the CS1 and CS0 bits. The clock obtained by dividing the selected clock by 2 is used as the machine clock.

## 5.2 Block Diagram of the Low-Power Consumption Control Circuit

Figure 5.2-1 "Block Diagram of the Low-Power Consumption Control Circuit and Clock Generator" shows a block diagram.

### ■ Block Diagram of the Low-Power Consumption Control Circuit

Figure 5.2-1 Block Diagram of the Low-Power Consumption Control Circuit and Clock Generator





### 5.3 Registers of the Low-Power Consumption Control Circuit

The following two types of low-power consumption control circuit register are provided:

- Low-power consumption mode control register
- Clock selection register

■ Registers in the Low-Power Consumption Control Circuit

Figure 5.3-1 Registers in the Low-Power Consumption Control Circuit

○ Low-power consumption mode control register

	7	6	5	4	3	2	1	0	⇔ Bit number
Address : 0000A0 <sub>11</sub>	STP	SLP	SPL	RST	TMD	CG1	CG0	SSR	LPMCR
Read/write ⇔	(W)	(W)	(R/W)	(W)	(W)	(R/W)	(R/W)	(R/W)	
Initial value ⇔	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

○ Clock selection register

	15	14	13	12	11	10	9	8	⇔ Bit number
Address : 0000A1 <sub>11</sub>	SCM	MCM	WS1	WS0	SCS	MCS	CS1	CS0	CKSCR
Read/write ⇔	(R)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇔	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	

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

This section describes the bit configuration and functions of the low-power consumption mode control register (LPMCR)

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

**Figure 5.3-2 Low-Power Consumption Mode Control Register (LPMCR)**

Low-Power Consumption Mode Control Register (LPMCR)

	7	6	5	4	3	2	1	0	⇐ Bit number
Address : 0000A0 <sub>11</sub>	STP	SLP	SPL	RST	TMD	CG1	CG0	SSR	LPMCR
Read/write ⇨	(W)	(W)	(R/W)	(W)	(W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

#### [Bit 7] STP

Writing "1" causes a transition to the pseudo watch mode (when MCS is 0 and SCS is 1 in the CKSCR) or stop mode (when MCS is 1 or SCS is 0 in the CKSCR). Writing "0" does not start an operation. When a reset occurs or the watch or stop mode is released, this bit is cleared to "0" and is a write-only bit. The read value is always "0".

#### [Bit 6] SLP

Writing "1" causes a transition to the sleep mode. Writing "0" does not start an operation. When a reset occurs or the sleep or stop mode is released, this bit is cleared to "0".

Writing "1" in the STP and SLP bits simultaneously causes a transition to the pseudo watch mode or stop mode. This bit is a write-only bit. The read value is always "0".

#### [Bit 5] SPL

When this bit is "0", the external pin levels are retained in the watch or stop mode. When the bit is "1", the external pins are set to high impedance in the watch or stop mode. When a reset occurs, this bit is cleared to "0" and is a read/write bit.

#### [Bit 4] RST

Writing "0" generates the internal reset signal for three machine cycles. Writing "1" does not start an operation. When this bit is read, the value is "1".

#### [Bit 3] TMD

Writing "0" causes a transition to the watch mode. Writing "1" does not start an operation. When a reset occurs or the watch or stop mode is released, this bit is cleared to "1" and is a write-only bit. The read value is always "1".

#### [Bits 2 and 1] CG1 and CG0

These bits set the temporary stop cycle count for the intermittent CPU operation function.

When a reset is caused by power-on, hardware standby, or watchdog, these bits are initialized to "00". These bits are not initialized by a reset as a result of another reset factor and are read/write bits.

Table 5.3-1 "CG Bit Settings" lists the CG bit settings.

**Table 5.3-1 CG Bit Settings**

CG1	CG0	Temporary stop cycle count for the CPU clock
0	0	0 cycle (CPU clock = peripheral clock)
0	1	9 cycles (CPU clock = peripheral clock = 1:about 3 to 4)
1	0	17 cycles (CPU clock = peripheral clock = 1:about 5 to 6)
1	1	33 cycles (CPU clock = peripheral clock = 1:about 9 to 10)

**[Bit 0] SSR**

When this bit is "1", DRAMC self refresh control is exercised in the sleep (main or PLL), watch, or stop mode. When a reset occurs, this bit is cleared to "0" and is a read/write bit.

■ **Accessing the Low-Power Consumption Mode Control Register (LPMCR)**

Setting the low-power consumption mode control register enters the low-power consumption mode. Use the instructions listed in Table 5.3-2 "Instructions to Be Used for Switching to Low-Power Consumption Mode" for this purpose. Using other instructions to start low-power consumption mode may cause a malfunction. Any instruction can be used to control functions other than switching to low-power consumption mode from the low-power mode control register.

To use word length to write data to the low-power mode control register, be sure that even addresses are used. Writing with odd addresses to start low-power consumption mode may cause a malfunction.

**Table 5.3-2 Instructions to Be Used for Switching to Low-Power Consumption Mode**

MOV io,#imm8	MOV dir,#imm8	MOV eam,#imm8	MOV eam,#immRi
MOV io,A	MOV dir,A	MOV addr16,A	MOV eam,A
MOV RLi+dip8,A	MOV P addr24,A		
MOVW io,#imm16	MOVW dir,#imm16	MOVW eam,#imm16	MOVW eam,RWi
MOVW io,A	MOVW dir,A	MOVW addr16,A	MOVW eam,RWi
MOVW RLi+dip8,A	MOPW addr24,A		
SETB io:bp	SETB dir:bp	SETB addr16:bp	

## 5.3.2 Clock Selection Register (CKSCR)

This section describes the bit configuration and functions of the clock selection register (CKSCR).

### ■ Clock Selection Register (CKSCR)

**Figure 5.3-3 Clock Selection Register (CKSCR)**

	15	14	13	12	11	10	9	8	↔ Bit number
Address : 0000A1 <sub>11</sub>	SCM	MCM	WS1	WS0	SCS	MCS	CS1	CS0	CKSCR
Read/write ↔	(R)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↔	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	

#### [Bit 15] SCM

This bit indicates whether the main clock or subclock is selected as the machine clock. When this bit is "0", the subclock is selected. When this bit is "1", the main clock is selected. When SCS is 1 and SCM is 1, the main clock is in the oscillation stabilization wait state.

Setting the SCS bit in the CKSCR from "1" to "0" switches the machine clock from the main clock to the subclock mode. This switch occurs in synchronization with the subclock (about 130  $\mu$ s).

Setting the SCS bit in the CKSCR from "0" to "1" switches the machine clock from the subclock to the main clock mode after the oscillation stabilization time for the main clock has elapsed. The timebase timer is automatically cleared.

#### [Bit 14] MCM

This bit indicates whether the main or the PLL clock is selected as the machine clock. When this bit is "0", the PLL clock is selected. When this bit is "1", the main clock is selected. When MCS is "0" and MCM is "1", the PLL clock is in the oscillation stabilization wait state.

The oscillation stabilization time for the PLL clock is fixed to  $2^{13}$  main clock cycles.

#### [Bits 13 and 12] WS1 and WS0

These bits set the oscillation stabilization time for the main clock after the stop or hardware standby mode is released.

These bits are initialized to "11" by a power-on reset, are not initialized by a reset as a result of another reset factor, and are read/write bits.

Table 5.3-3 "WS Bit Settings" lists the WS bit settings.

**Table 5.3-3 WS Bit Settings**

WS1	WS0	Oscillation stabilization time (OSC oscillation: 4 MHz)
0	0	No oscillation stabilization time
0	1	About 2.05 ms ( $2^{13}$ OSC oscillation cycles)

Table 5.3-3 WS Bit Settings (Continued)

WS1	WS0	Oscillation stabilization time (OSC oscillation: 4 MHz)
1	0	About 8.19 ms ( $2^{15}$ OSC oscillation cycles)
1	1	About 65.54 ms ( $2^{18}$ OSC oscillation cycles) [Initial value]

**Note:**

When tune on the power or hardware standby mode or stop mode is released, the subclock oscillation stabilization time (about 2 seconds) is generated. In the meantime, when switching from the main clock mode to the subclock mode, the oscillation stabilization time is generated.

**[Bit 11] SCS**

This bit specifies whether the main clock or subclock is to be selected as the machine clock. Writing "0" selects the subclock. Writing "1" selects the main clock. When this bit is "0", writing "1" switches the machine clock from the main clock to the subclock mode. This switch occurs in synchronization with the subclock (about 130  $\mu$ s).

When this bit is "1", writing "0" switches the machine clock from the subclock to the main clock mode after the oscillation stabilization time for the main clock has elapsed. At this time, automatically clears the timebase timer. When both SCS and MCS are "0", the value set for SCS is used and the subclock is selected.

**[Bit 10] MCS**

This bit specifies whether the main or the PLL clock is to be selected as the machine clock. Writing "0" selects the PLL clock. Writing "1" selects the main clock. When this bit is "1", writing "0" automatically clears the time base timer to generate the oscillation stabilization time for the PLL clock. The oscillation stabilization time for the PLL clock is fixed to  $2^{13}$  main clock cycles.

The clock obtained by dividing the main clock by 2 is used as the operating clock when the main clock is selected (when the OSC oscillation frequency is 4 MHz, the operating clock frequency is 2 MHz).

This bit is initialized to "1" by a power-on, hardware standby, or watchdog reset.

**Note:**

Before setting the MCS bit from "1" to "0", ensure that the time base timer interrupt is masked by the TBIE bit of the time base timer control register (TBTC) or interrupt level mask register (ILM).

Because it may not be possible for eight machine cycles to set the MCS bit to "0" after it was set to "1", set it to "0" after waiting eight or more machine cycles.

**[Bits 9 and 8] CS1 and CS0**

These bits select a multiplication factor for the PLL clock and are not initialized by a reset caused by an external pin or the RST bit. The bits are initialized to "00" by a power-on, hardware standby, or watchdog reset.

When the MCS bit is "0", a write operation is suppressed. Set the MCS bit to "1" (main clock mode), then write the CS bits. The CS bits are read/write bits.

### 5.3 Registers of the Low-Power Consumption Control Circuit

Table 5.3-4 "CS Bit Settings" lists the CS bit settings.

**Table 5.3-4 CS Bit Settings**

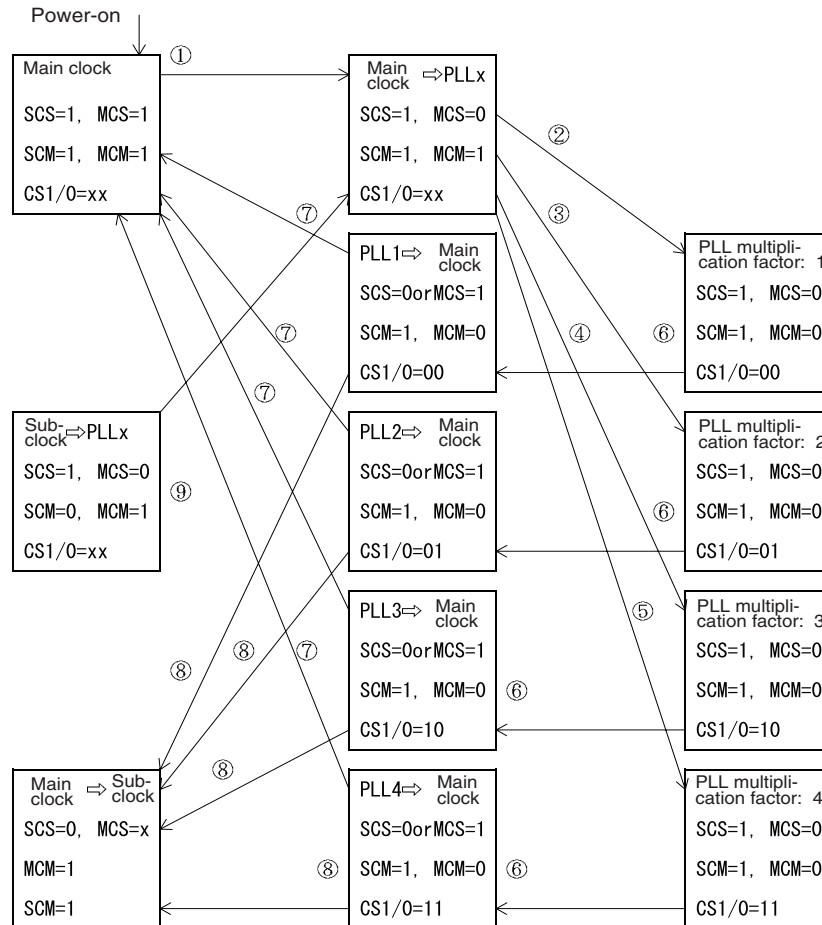
<b>CS1</b>	<b>CS0</b>	<b>Machine clock (OSC oscillation: 4 MHz)</b>
0	0	4 MHz (The operating frequency is equal to the OSC oscillation frequency.)
0	1	8 MHz (The operating frequency is equal to the frequency obtained by multiplying the OSC oscillation frequency by 2.)
1	0	12 MHz (The operating frequency is equal to the frequency obtained by multiplying the OSC oscillation frequency by 3.)
1	1	16 MHz (The operating frequency is equal to the frequency obtained by multiplying the OSC oscillation frequency by 4.)

## 5.4 Status Transition for Clock Selection

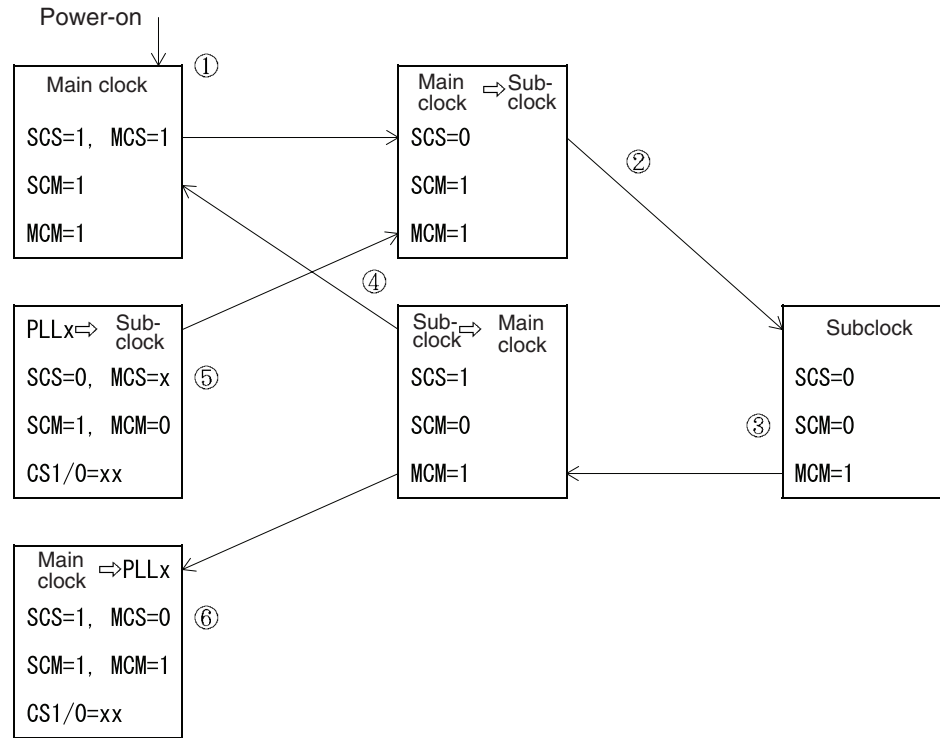
Figure 5.4-1 "Status Transition Diagram for Clock Selection (1)" and Figure 5.4-2 "Status Transition Diagram for Clock Selection (2)" show status transition for clock selection.

### ■ Status Transition for Clock Selection

Figure 5.4-1 Status Transition Diagram for Clock Selection (1)



- ① The MCS bit is cleared and the SCS bit is set.
- ② The oscillation stabilization time for the PLL clock has elapsed and CS1 and CS0 are 00.
- ③ The oscillation stabilization time for the PLL clock has elapsed and CS1 and CS0 are 01.
- ④ The oscillation stabilization time for the PLL clock has elapsed and CS1 and CS0 are 10.
- ⑤ The oscillation stabilization time for the PLL clock has elapsed and CS1 and CS0 are 11.
- ⑥ The MCS bit is set or the SCS bit is cleared.
- ⑦ The PLL clock synchronizes with the main clock and SCS is 1.
- ⑧ The PLL clock synchronizes with the main clock and SCS is 0.
- ⑨ The oscillation stabilization time for the main clock has elapsed and MCS is 0.

**Figure 5.4-2 Status Transition Diagram for Clock Selection (2)**

- ① The SCS bit is cleared.
- ② The subclock edge is detected.
- ③ The SCS bit is set.
- ④ The oscillation stabilization time for the main clock has elapsed and MCS is 1.
- ⑤ The PLL clock synchronizes with the main clock and SCS is 0.
- ⑥ The oscillation stabilization time for the main clock has elapsed and MCS is 0.





## CHAPTER 6    LOW-POWER CONSUMPTION MODE

---

**This chapter describes the functions and operations of the low-power consumption mode.**

---

6.1 "Low-Power Consumption Mode"

6.2 "Status Transition in Low-Power Consumption Mode"

6.3 "Status Transition Diagram of Low-Power Consumption Mode"

## 6.1 Low-Power Consumption Mode

---

There are the following operating modes:

- PLL clock mode
- PLL sleep mode
- PLL watch mode
- Pseudo watch mode
- Main clock mode
- Main clock sleep mode
- Main clock watch mode
- Main clock stop mode
- Subclock mode
- Subclock sleep mode
- Subclock watch mode
- Subclock stop mode
- Hardware standby mode

Operating modes other than the PLL clock mode are categorized as low-power consumption modes.

---

### ■ Low-Power Consumption Mode

#### ○ Main clock mode and main clock sleep mode

The microcontroller operates using the main clock (main OSC oscillation clock) and subclock (sub OSC oscillation clock). The clock obtained by dividing the main clock by 2 is used as the operating clock and the subclock (clock obtained by dividing the sub OSC oscillation clock by 4) is used as the watch clock. The PLL clock (VCO oscillation clock) is stopped.

#### ○ Subclock mode and subclock sleep mode

The microcontroller operates using only the subclock. The clock obtained by dividing the subclock by 4 is used as the operating clock. The main and PLL clocks are stopped.

#### ○ PLL sleep mode and main clock sleep mode

Only the CPU operating clock is stopped. Clocks other than the CPU clock are operating.

#### ○ Pseudo watch mode

The pseudo watch mode is the mode for operating the watch or time base timers only.

#### ○ PLL watch mode, main clock watch mode, and subclock watch mode

Only the clock timer is operating. The microcontroller operates using the clock obtained by dividing the subclock by 4. The main and PLL clocks are stopped.

In the PLL watch mode, main clock watch mode, and subclock watch mode, only operating modes differ when they are returned from an interrupt (PLL clock mode, main clock mode, and subclock mode). Operation is the same in these watch modes.

### ○ Main clock stop mode, subclock stop mode, and hardware standby mode

Oscillation is stopped. Data can be retained with the lowest power consumption.

The main clock stop mode and subclock stop mode differ in terms of the operating mode when returned from an interrupt (main clock mode and subclock mode). Operation is the same in these stop modes.

### ○ Intermittent CPU operation function

The intermittent CPU operation function intermittently operates the clock provided to the CPU when registers, internal memory, internal peripherals, and the external bus are accessed. Processing can be performed with low-power consumption because the CPU execution speed is reduced with the provision of a high-speed clock to the internal peripherals.

A PLL clock multiplication factor can be selected among 2, 4, 6, and 8 using the CS1 and CS0 bits. The clock obtained by dividing the selected clock by 2 is used as the machine clock.

## ■ Operating States in the Low-Power Consumption Mode

Table 6.1-1 "Operating States in the Low-Power Consumption Mode" lists the status of chip blocks in operating modes.

**Table 6.1-1 Operating States in the Low-Power Consumption Mode**

State	Transition condition	Sub OSC oscillation	Main OSC oscillation	Clock	CPU	Peripherals	Pins	Released by
Subclock	SCS=0 MCS=x	Operating	Stopped	Operating	Operating	Operating	Operating	Reset interrupt
Subclock sleep	SCS=0 MCS=x SLP=1	Operating	Stopped	Operating	Stopped	Operating	Operating	Reset interrupt
Main clock sleep	SCS=1 MCS=1 SLP=1	Operating	Operating	Operating	Stopped	Operating	Operating	Reset interrupt
PLL sleep	SCS=1 MCS=0 SLP=1	Operating	Operating	Operating	Stopped	Operating	Operating	Reset interrupt
Pseudo watch (SPL=0)	SCS=1 MCS=0 STP=1	Operating	Operating	Stopped	Stopped	Stopped	Retained	Reset interrupt
Pseudo watch (SPL=1)	SCS=1 MCS=0 STP=1	Operating	Operating	Stopped	Stopped	Stopped	HI-Z	Reset interrupt
Watch (SPL=0)	SCS=x MCS=x TMD=0	Operating	Stopped	Stopped	Stopped	Stopped	Retained	Reset interrupt
Watch (SPL=1)	SCS=x MCS=x TMD=0	Operating	Stopped	Stopped	Stopped	Stopped	HI-Z	Reset interrupt

## CHAPTER 6 LOW-POWER CONSUMPTION MODE

**Table 6.1-1 Operating States in the Low-Power Consumption Mode (Continued)**

State	Transition condition	Sub OSC oscillation	Main OSC oscillation	Clock	CPU	Peripherals	Pins	Released by
Stop (SPL=0)	MCS=1 or SCS=0 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Retained	Reset interrupt
Stop (SPL=1)	MCS=1 or SCS=0 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	Reset interrupt
Hardware standby	HSTX=L	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	HSTX=H

**Note:**

When tune on the power or hardware standby mode or stop mode is released, the subclock oscillation stabilization time (about 2 seconds) is generated. In the meantime, when switching from the main clock mode to the subclock mode, the oscillation stabilization time is generated.

## 6.1.1 Sleep Mode

---

**In the sleep mode, only the clock provided to the CPU is stopped. The CPU stops and the peripheral circuits continue operation.**

---

### ■ Transition to the Sleep Mode

Writing "1" in the SLP and TMD bits, and "0" in the STP bit in the low-power consumption mode control register sets the standby control circuit to the sleep mode.

When "1" is written in the SLP bit in the LPMCR, an interrupt request may have occurred, in which case the standby control circuit does not enter the sleep mode. The CPU executes the next instruction in the interrupt disable state or immediately causes a branch to the interrupt processing routine in the interrupt enable state.

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

### ■ Releasing the Sleep Mode

The standby control circuit releases the sleep mode when a reset is input or an interrupt occurs. After this circuit releases the sleep mode by a reset factor, it enters the reset state.

When a peripheral circuit or internal peripheral generates an interrupt request at interrupt level 7 or higher in the sleep mode, the standby control circuit releases the sleep mode. After the sleep mode is released, the interrupt is processed in the same way as normal interrupts. The interrupt enable state may be set by the settings of the I flag, ILM, and interrupt control register (ICR), in which case the CPU executes the not-interrupt-pending instruction following the standby interrupt instruction, then executes interrupt processing. In the interrupt disable state, the CPU continues processing from the instructions that follow the instructions causing the sleep mode.

## 6.1.2 Pseudo Watch Mode

---

**In the pseudo watch mode, an operation other than OSC oscillation (main and sub), watch timer, and time base timer is stopped and most chip functions stop.**

---

### ■ Transition to the Pseudo Watch Mode

Under the following conditions, the standby control circuit is set to pseudo watch mode: When the SCS bit is set to "1" and the MCS bit in the clock selection register (CKSCR) to "0" while the TMD and STP bits in the low-power consumption mode control register (LPMCR) are set to "1".

Whether to retain the status of each I/O pin before the pseudo watch mode or set it to high impedance in the pseudo watch mode can be specified using the SPL bit in the low-power consumption mode control register.

When "1" is written in the STP bit in the LPMCR, an interrupt request may occur, in which case the standby control circuit does not enter the pseudo watch mode.

In the pseudo watch mode, the contents of dedicated registers, such as the accumulator and internal RAM, are retained.

### ■ Releasing the Pseudo Watch Mode

The standby control circuit releases the pseudo watch mode when a reset is input or an interrupt occurs. When this circuit releases the pseudo watch mode by a reset factor, it enters the reset state.

At return from the pseudo watch mode, the standby control circuit first releases the pseudo watch mode, then waits for PLL clock oscillation to become stable. For this reason, the reset sequence is also performed using the main clock when the pseudo watch mode is released by a reset factor.

When a peripheral circuit generates an interrupt request at interrupt level 7 or higher in the pseudo watch mode, the standby control circuit releases the pseudo watch mode. After the pseudo watch mode is released, the interrupt is processed in the same way as normal interrupts. The interrupt enable state may be set by the settings of the I flag, ILM, or interrupt control register (ICR), in which case the CPU executes the not-interrupt-pending instruction following the standby write instruction, then executes interrupt processing. In the interrupt disable state, the CPU continues processing from the next instruction before the pseudo watch mode.

#### Note:

- For the MB90V570, MB90F574, MB90573, and MB90574
  - When releasing the pseudo watch mode by an external interrupt, set the external interrupt request to "H" level. If the external interrupt request is set to "L" level, a malfunction may occur. The pseudo watch mode does not release if an edge of the external interrupt request is set.
- For the MB90V570A, MB90F574A, and MB90574C
  - The pseudo watch mode can be released by inputting the external interrupt request set before the standby control circuit enters the pseudo watch mode. The interrupt request of ch2 to ch7 can be selected "H" or "L" level. The interrupt request of ch0 and ch1 can be selected rising edge or falling edge.



### 6.1.3 Watch Mode

---

**In the watch mode, an operation other than the sub OSC oscillation and clock timer is stopped. Most chip functions stop.**

---

#### ■ Transition to the Watch Mode

Writing "0" in the TMD bit in the low-power consumption mode control register sets the standby control circuit to the watch mode.

Whether to retain the status of each I/O pin before the watch mode or set it to high impedance in the pseudo watch mode can be specified using the SPL bit in the low-power consumption mode control register.

When "1" is written in the TMD bit in the LPMCR, an interrupt request may occur, in which case the standby control circuit does not enter the watch mode.

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

## ■ Releasing the Watch Mode

The standby control circuit releases the watch mode when a reset is input or an interrupt occurs. When this circuit releases the watch mode by a reset factor, it enters the reset state.

At return from the subclock watch mode, the standby control circuit releases the watch mode and then immediately enters the subclock mode. For this reason, the reset sequence is also performed using the subclock when the subclock watch mode is released by a reset factor.

At return from the main clock or PLL watch mode, the standby control circuit first releases the watch mode, then waits for main clock oscillation to become stable. For this reason, the reset sequence is also performed using the subclock when the watch mode is released by a reset factor.

When a peripheral circuit generates an interrupt request at interrupt level 7 or higher in the watch mode, the standby control circuit releases the watch mode. After the watch mode is released, the interrupt is processed in the same way as normal interrupts. The interrupt enable state may be set by the settings of the I flag, ILM, or interrupt control register (ICR), in which case the CPU executes the not-interrupt-pending instruction following the standby write instruction, then executes interrupt processing. In the interrupt disable state, the CPU continues processing from the next instruction before the watch mode.

### Note:

- For the MB90V570, MB90F574, MB90573, and MB90574
  - When releasing the watch mode by an external interrupt, set the external interrupt request to "H" level. If the external interrupt request is set to "L" level, a malfunction may occur. The watch mode does not release if an edge of the external interrupt request is set.
- For the MB90V570A, MB90F574A, and MB90574C
  - The watch mode can be released by inputting the external interrupt request set before the standby control circuit enters the watch mode. The interrupt request of ch2 to ch7 can be selected "H" or "L" level. The interrupt request of ch0 and ch1 can be selected rising edge or falling edge.

## 6.1.4 Stop Mode

---

**In the stop mode, OSC oscillation (main and sub) is stopped. All chip functions stop. Therefore, data can be retained with the lowest power consumption.**

---

### ■ Transition to the Stop Mode

Under the following conditions, the standby control circuit is set to stop mode: When the SCP bit is set to "0" or the MCS bit in the clock selection register (CKSCR) is set to "1" while the STP bit in the low-power consumption mode control register (LPMCR) is set to "1".

Whether to retain the status of each I/O pin before the stop mode or set it to high impedance in the pseudo watch mode can be specified using the SPL bit in the low-power consumption mode control register (LPMCR).

When "1" is written in the STP bit in the LPMCR, an interrupt request may occur, in which case the standby control circuit does not enter the stop mode.

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

### ■ Releasing the Stop Mode

The standby control circuit releases the stop mode when a reset is input or an interrupt occurs. When this circuit releases the stop mode by a reset factor, it enters the reset state.

At return from the subclock stop mode, the standby control circuit first waits for subclock oscillation to become stable, then releases the stop mode. For this reason, the reset sequence is also performed after the oscillation stabilization time for the subclock when the stop mode is released by a reset factor.

At return from the main clock stop mode, the standby control circuit first waits for main clock oscillation to become stable, then releases the stop mode. For this reason, the reset sequence is also performed after the oscillation stabilization time for the main clock when the stop mode is released by a reset factor.

When a peripheral circuit generates an interrupt request at interrupt level 7 or higher in the stop mode, the standby control circuit releases the stop mode. After the subclock stop mode is released, the interrupt is processed in the same way as normal interrupts when the oscillation stabilization time for the subclock has elapsed. The interrupt enable state may be set by the settings of the I flag, ILM, or interrupt control register (ICR), in which case the CPU executes the not-interrupt-pending instruction following the standby write instruction, then executes interrupt processing. In the interrupt disable state, the CPU continues processing from the next instruction before the stop mode.

After the main clock stop mode is released, the interrupt is processed in the same way as normal interrupts when the oscillation stabilization time for the main clock has elapsed. The oscillation stabilization time is specified by the WS1 and WS0 bits in the CKSCR. The interrupt enable state may be set by the settings of the I flag, ILM, or interrupt control register (ICR), in which case the CPU executes the not-interrupt-pending instruction following the standby write instruction, then executes interrupt processing. In the interrupt disable state, the CPU continues processing from the next instruction before the stop mode.

**Note:**

- For the MB90V570, MB90F574, MB90573, and MB90574
  - When releasing the stop mode by an external interrupt, set the external interrupt request to "H" level. If the external interrupt request is set to "L" level, a malfunction may occur. The stop mode does not release if an edge of the external interrupt request is set.
- For the MB90V570A, MB90F574A, and MB90574C
  - The stop mode can be released by inputting the external interrupt request set before the standby control circuit enters the stop mode. The interrupt request of ch2 to ch7 can be selected "H" or "L" level. The interrupt request of ch0 and ch1 can be selected rising edge or falling edge.

## 6.1.5 Hardware Standby Mode

---

**In the hardware standby mode, when the HSTX pin is low, oscillation is stopped and all I/O pins are set to high impedance regardless of other statuses, including resets.**

---

### ■ Transition to the Hardware Standby Mode

In any state, driving the HSTX pin low can set the standby control circuit to the hardware standby mode.

In the hardware standby mode, the contents of internal RAM are retained, but dedicated registers such as the accumulator are initialized.

### ■ Releasing the Hardware Standby Mode

The hardware standby mode can be released using the HSTX pin only. When the HSTX pin is driven high, the standby control circuit releases the hardware standby mode, enables the internal reset signal, then enters the oscillation stabilization wait state. When the oscillation stabilization time for the main clock has elapsed, the standby control circuit releases the internal reset. The CPU then starts execution from the reset sequence.

## 6.2 Status Transition in Low-Power Consumption Mode

In low-power consumption mode, transition to each state occurs according to the settings in the clock selection and low-power consumption mode control registers.

### ■ Status Transition in Low-Power Consumption Mode

Table 6.2-1 "List of Transition Conditions" lists the status transition conditions.

The symbols in Table 6.2-1 have the following meanings:

- MCS: MCS bit (Clock selection register) (PLL clock mode selected with MSC=0)
- SCS: SCS bit (Clock selection register) (Subclock mode selected with SCS=0)
- STP: STP bit (Low-power consumption mode control register) (Stop mode selected with STP=0)
- SLP: SLP bit (Low-power consumption mode control register) (Sleep mode selected with SLP=0)
- TMD: TMD bit (Low-power consumption mode control register) (Clock mode selected with TMD=0)
- MCM: MCM bit (Clock selection register) (PLL clock is used with MCM=0)
- SCM: SCM bit (Clock selection register) (Subclock is used with SCM=0)
- SCD: Subclock oscillation stopped (Subclock oscillation is stopped with SCD=1)
- MCD: Main clock oscillation stopped (Main clock oscillation is stopped with MCD=1)
- PCD: PLL clock oscillation stopped (PLL clock oscillation is stopped with PCD=1)

**Table 6.2-1 List of Transition Conditions**

Status before transition	Transition conditions	Status after transition
Power-on	01 Termination of main clock oscillation stabilization time wait	Main mode
Main clock oscillation stabilization	05 Termination of main clock oscillation stabilization time wait	Main mode
Main mode	06 SCS=0 Writing	MS transition mode
	07 SCS=1 • MCS=0 Writing	MP transition mode
	31 TMD=1 • STP=0 • SLP=1 Writing	Main sleep
	32 TMD=0 Writing	Main clock transition
	33 TMD=1 • STP=1 Writing	Main stop

## CHAPTER 6 LOW-POWER CONSUMPTION MODE

**Table 6.2-1 List of Transition Conditions (Continued)**

Status before transition	Transition conditions	Status after transition
PLL mode	21 SCS=0 Writing	PS transition mode
	20 SCS=1 • MCS=1 Writing	PM transition mode
	59 TMD=1 • STP=0 • SLP=1 Writing	PLL sleep
	58 TMD=0 Writing	PLL clock transition P
	57 TMD=1 • STP=1 Writing	Pseudo watch transition
Subclock mode	10 SCS=1 • MCS=1 Writing	SM transition mode
	12 SCS=1 • MCS=0 Writing	SP transition mode
	11 Reset activation	Main clock oscillation stabilization
	42 TMD=1 • STP=0 • SLP=1 Writing	Subclock sleep
	43 TMD=0 Writing	Subclock
	44 TMD=1 • STP=1 Writing	Subclock stop
PM transition mode	13 PLL--> Termination of main clock switching timing wait	Main mode
	38 TMD=1 • STP=0 • SLP=1 Writing	PM transition sleep
	39 TMD=0 Writing & PLL--> Termination of main clock switching wait	Main clock transition
	40 TMD=1 • STP=1 Writing & PLL--> Termination of main clock switching wait	Main stop
SM transition mode	02 Termination of main clock oscillation stabilization time wait	Main mode
	03 Reset activation or Interrupt	Main clock oscillation stabilization
	04 SCS=0 Writing	Subclock mode
	27 TMD=1 • STP=0 • SLP=1 Writing	SM transition sleep
	28 TMD=0 Writing & Termination of main clock oscillation stabilization time wait	Main clock
	29 TMD=1 • STP=1 Writing & Termination of main clock oscillation stabilization time wait	Main stop

Table 6.2-1 List of Transition Conditions (Continued)

Status before transition	Transition conditions	Status after transition
MP transition mode	16 Termination of PLL oscillation stabilization time wait	PLL mode
	14 SCS=1 • MCS=1 Writing	Main mode
	15 SCS=0 Writing	MS transition mode
	68 TMD=1 • STP=0 • SLP=1 Writing	MP transition sleep
	70 TMD=0 Writing	PLL clock transition M
	69 TMD=1 • STP=1 Writing & Termination of main clock oscillation stabilization time wait	Pseudo watch mode
SP transition mode	17 Termination of main clock oscillation stabilization time wait	MP transition mode
	18 MCS=1 Writing	SM transition mode
	19 Reset activation	Main clock oscillation stabilization
	75 TMD=1 • STP=0 • SLP=1 Writing	SP transition sleep
	76 TMD=0 Writing	PLL clock
	78 TMD=1 • STP=1 Writing & Termination of main clock oscillation stabilization time wait	Pseudo watch mode
MS transition mode	09 Main-->Termination of subclock switching timing wait	Subclock mode
	08 Reset activation	Main mode
	51 TMD=1 • STP=0 • SLP=1 Writing	MS transition sleep
	52 TMD=0 Writing & Termination of Main -->Subclock switching wait	Subclock
	53 TMD=1 • STP=1 Writing & Termination of Main -->Subclock switching wait	Subclock stop
PS transition mode	23 PPL-->Termination of main clock switching timing wait	MP transition mode
	22 SCS=1 Writing	PS transition mode
	56 TMD=1 • STP=0 • SLP=1 Writing	PS transition sleep
Main sleep	26 Interrupt or reset activation	Main mode
SM transition sleep	24 Termination of main clock oscillation stabilization time wait	Main sleep
	25 Interrupt or reset activation	SM transition mode
PM transition sleep	34 PLL-->Termination of main clock switching timing wait	Main sleep
	35 Interrupt or reset activation	PM transition mode
PLL sleep	63 Interrupt or reset activation	PLL mode



## CHAPTER 6 LOW-POWER CONSUMPTION MODE

**Table 6.2-1 List of Transition Conditions (Continued)**

Status before transition	Transition conditions		Status after transition
MP transition sleep	66	Termination of PLL oscillation stabilization time wait	PLL sleep
	67	Interrupt or reset activation	MP transition mode
SP transition sleep	73	Termination of main clock oscillation stabilization time wait	MP transition sleep
	74	Interrupt or reset activation	SP transition mode
Subclock sleep	46	Interrupt or reset activation	Subclock mode
MS transition sleep	49	Main-->Termination of subclock switching timing wait	Subclock sleep
	50	Interrupt or reset activation	MS transition mode
PS transition sleep	54	PLL-->Termination of main clock switching timing wait	MS transition sleep
	55	Interrupt or reset activation	PS transition mode
Main clock	30	Interrupt or reset activation	SM transition mode
Main clock transition	36	Main-->Termination of subclock switching timing wait	Main clock
	37	Interrupt or reset activation	Main mode
PLL clock	77	Interrupt or reset activation	SP transition mode
PLL clock transition M	72	Main-->Termination of subclock switching timing wait	PLL clock
	71	Interrupt or reset activation	MP transition mode
PLL clock transition P	65	PLL-->Termination of main clock switching timing wait	PLL clock transition M
	64	Interrupt or reset activation	PLL mode
Subclock	47	Interrupt or reset activation	Subclock mode
Main stop	41	Interrupt or reset activation	Stabilization of main clock oscillation
Pseudo watch	62	Interrupt or reset activation	MP transition mode
Pseudo watch transition	61	PLL-->Termination of main clock switching timing wait	Pseudo watch mode
	60	Interrupt or reset activation	PLL mode
Subclock stop	48	Interrupt	Stabilization of subclock oscillation
	79	Reset activation	Stabilization of main clock oscillation
Stabilization of subclock oscillation	45	Termination of subclock oscillation stabilization wait	Subclock mode
	80	Reset activation	Stabilization of main clock oscillation

## 6.3 Status Transition Diagram of Low-Power Consumption Mode

---

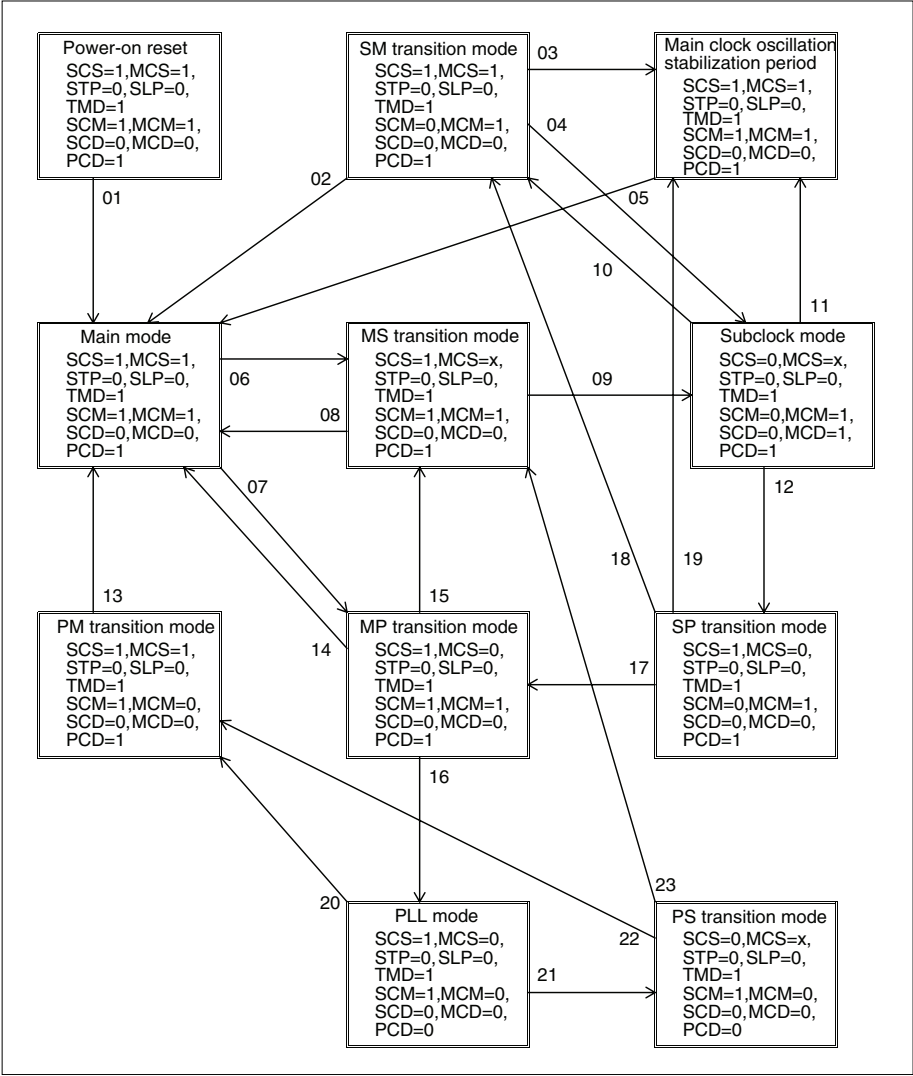
Figure 6.3-1 "Low-Power Consumption Mode Status Transition Diagram A" to Figure 6.3-4 "Low-Power Consumption Mode Status Transition Diagram" show a status transition diagram of step-by-step status transitions in accordance with simultaneous events.

---

### ■ Status Transition Diagram of Low-Power Consumption Mode

For example, assume that MCS and SLP are set simultaneously to 1 in PLL clock mode. In the status transition diagram, a transition to PM transition mode occurs before a transition to PM transition sleep mode. The transition from PLL clock mode to PM transition sleep mode occurs immediately. It is assumed that a reset is activated in subclock sleep mode. In the diagram, a transition to subclock mode occurs before a transition to the main clock oscillation stabilization period. In fact, these transitions occur at the same time.

Figure 6.3-1 Low-Power Consumption Mode Status Transition Diagram A



**Figure 6.3-2 Low-Power Consumption Mode Status Transition Diagram B**

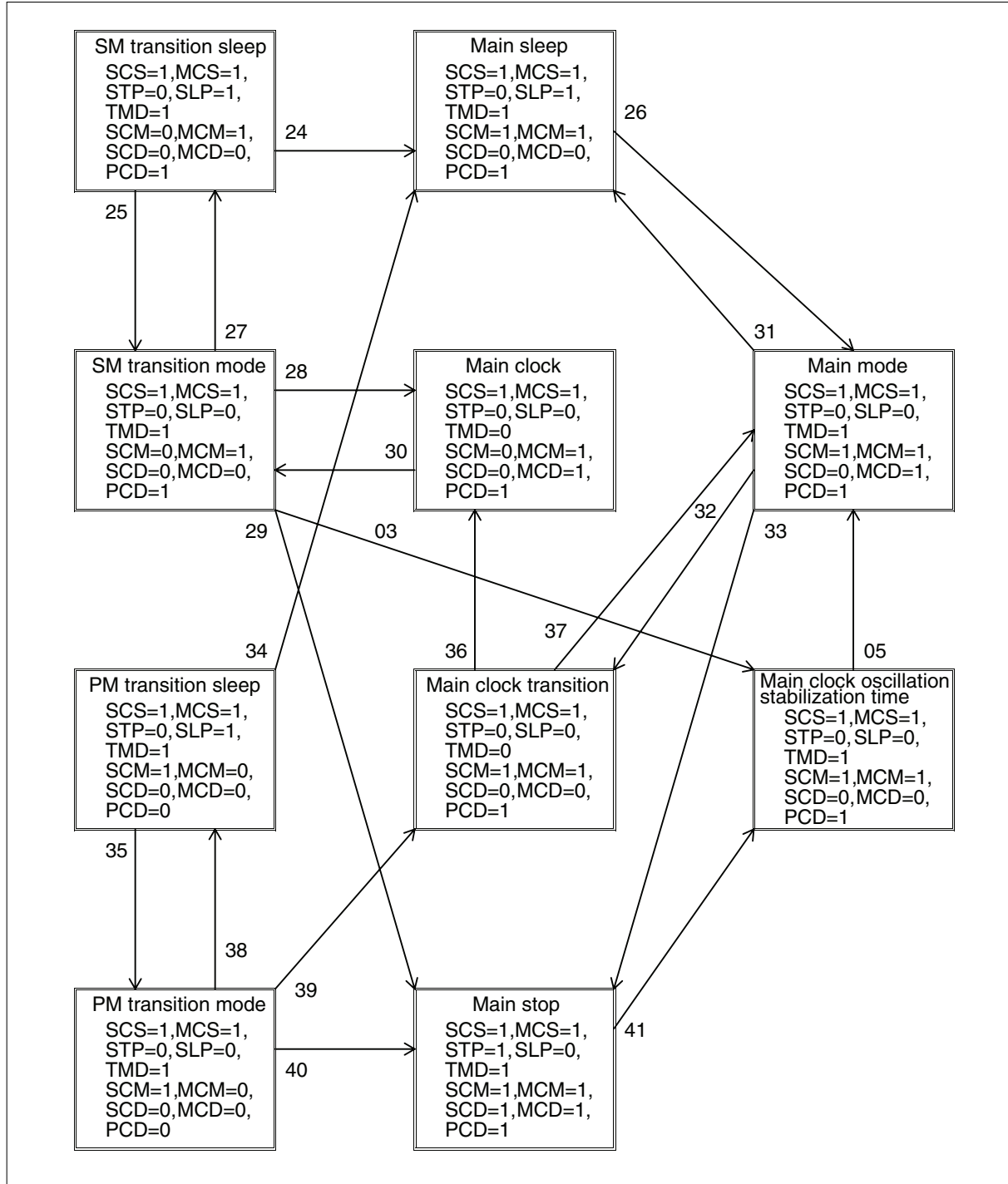
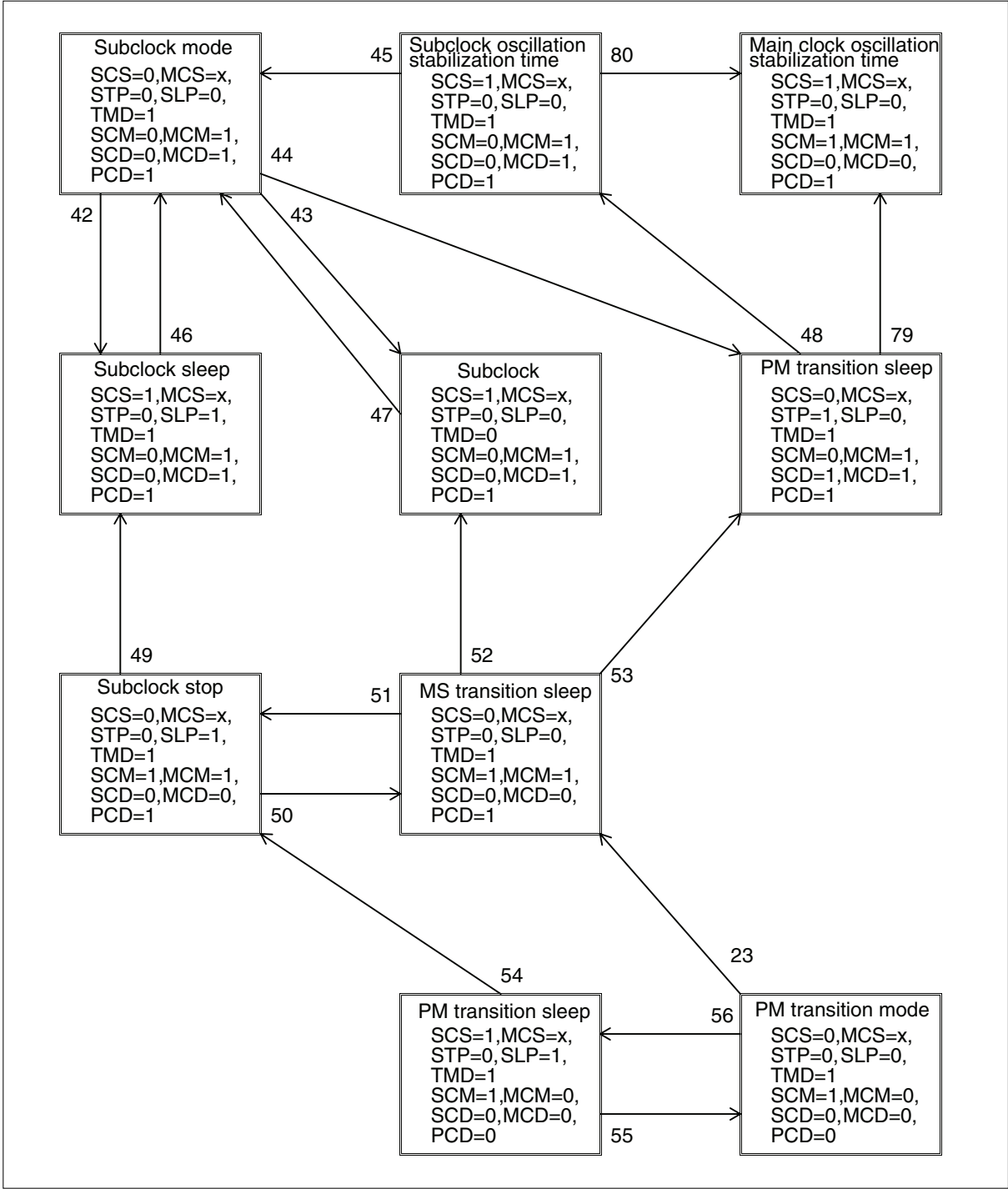
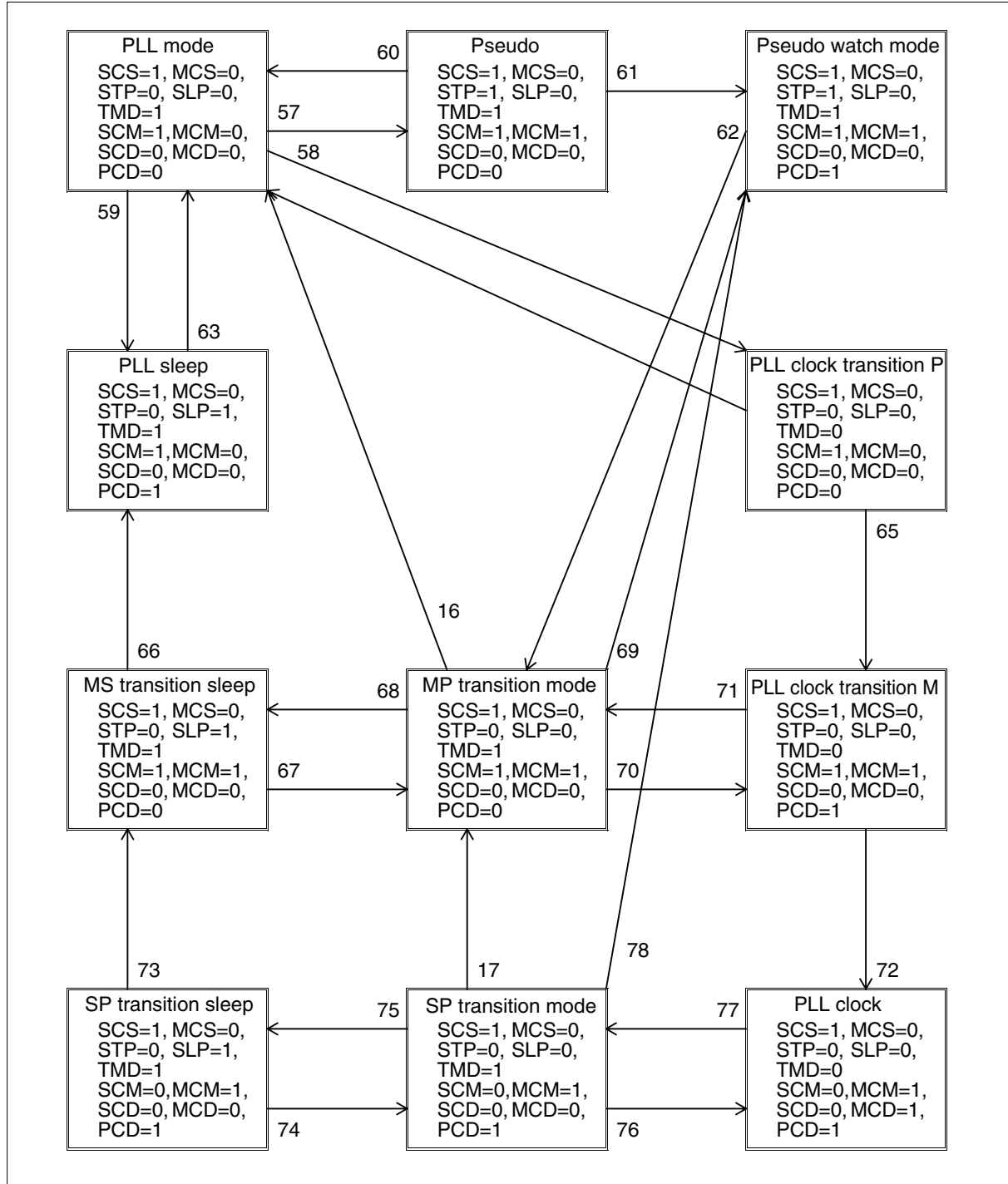


Figure 6.3-3 Low-Power Consumption Mode Status Transition Diagram C



**Figure 6.3-4 Low-Power Consumption Mode Status Transition Diagram**





# CHAPTER 7    MEMORY ACCESS MODE

---

**This chapter describes the functions and operations of the memory access modes.**

---

7.1 "Memory Access Modes"

7.2 "External Memory Access (External Bus Pin Control Circuit)"

7.3 "Operation of the External Memory Access Control Signal"



## 7.1 Memory Access Modes

---

**F<sup>2</sup>MC-16LX provides various modes for each access method and access area.**

---

### ■ Memory Access Modes

**Table 7.1-1 Memory Access Modes**

Operating mode	Bus mode	Access mode (external data bus width)
RUN	Single chip	—
	Internal ROM external bus	8 bits
		16 bits
	External ROM external bus	8 bits
		16 bits

### ■ Operating Mode

The operating mode indicates the mode for controlling device operation status and must be specified by the mode setting pin (MDx).

Selecting the operating mode enables the activation of ordinary operation and writing to the flash memory.

### ■ Bus Mode

The bus mode indicates the mode for controlling internal ROM operations and external access function operations and must be specified by the mode setting pin (MDx) and the Mx bit in mode data.

The mode setting pin (MDx) specifies the bus mode used when the reset vector or mode data is read. The Mx bit in the mode data specifies the bus mode at normal operation.

### ■ Access Mode

The access mode indicates the mode for controlling the external data bus width and must be specified by the mode setting pin (MDx) and the S0 bit in mode data.

Specify eight bits or 16 bits as the external data bus width by selecting an access mode.

## 7.1.1 Mode Pins

A mode can be specified by combining three external pins MD2 to MD0.

### ■ Mode Pins

Table 7.1-2 Relationship among Mode Pins and Modes to Be Set

Mode pin setting			Mode name	Reset vector access area	External data bus width	Remarks
MD2	MD1	MD0				
0	0	0	External vector mode 0	External	8 bits	—
0	0	1	External vector mode 1	External	16 bits	Access to the reset vector 16-bit bus width
0	1	0	(Specification prohibited)			
0	1	1	Internal vector mode	Internal	(Mode data)	Controlled with mode data after the reset sequence.
1	0	0	(Specification prohibited)			
1	0	1				
1	1	0	Flash serial writing(*1)	—	—	—
1	1	1	Flash memory mode	—	—	Used when the parallel writer is used.

#### Note:

Even if external vector mode 0 is selected, the initial values of IOBS and LMBS of the bus control selection register are set to 1. Therefore, a 16-bit width is set for the 0000C0<sub>H</sub> to 0000FF<sub>H</sub> and 002000<sub>H</sub> to 7FFFFFF<sub>H</sub> areas. To set the 8-bit width for these areas, write 0 in IOBS and LMBS of the bus control selection register.

For external vector mode 1, the HMBS bit is set to 0 to enable access to an interface with the 16-bit bus width.

Setting the mode pins only is not sufficient for serial writing to flash memory; other pins must be set as well. For details, see CHAPTER 27 "Example of MB90F574/A Serial Programming Connection".

\*1 Serial writing to the flash memory cannot be performed simply by setting the mode pin (other pins must be set). For details, see an example of flash serial writing connection.

### 7.1.2 Mode Data

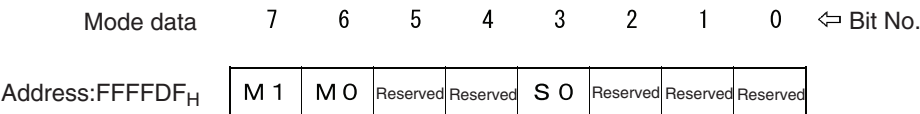
The mode data is allocated in the main storage FFFFDF<sub>H</sub> and is used to control CPU operations. This data is fetched during reset sequence execution and is stored in the mode register within the device. The reset sequence only can change the contents of the mode register.

The setting by this register is validated after the reset sequence.

A reserved bit must be set to 0.

■ Mode Data

Figure 7.1-1 Mode Data



[Bits 7, 6]: M1, M0 Bus mode setting bits

Used to specify an operating mode after the reset sequence ends.

M1	M0	Function
0	0	Single-chip mode
0	1	Internal ROM external bus mode
1	0	External ROM external bus mode
1	1	(Setting prohibited)

[Bit 3]: S0 Access mode setting bit

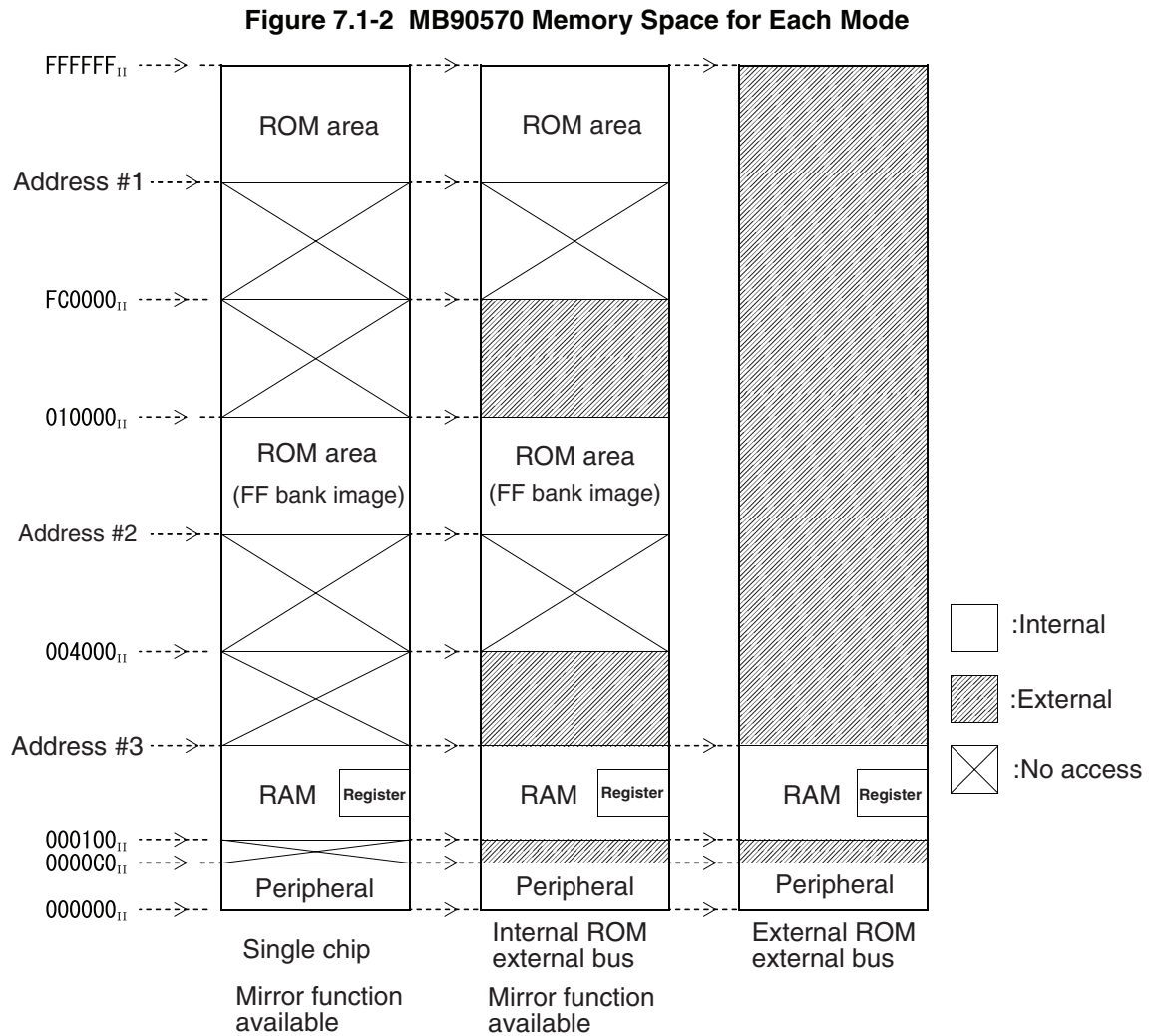
Used to specify the bus mode and access mode after the reset sequence ends.

S0	Function
0	External data bus 8-bit mode
1	External data bus 16-bit mode

### 7.1.3 Memory Space for Each Bus Mode

The following figure shows the correspondence between the access areas and physical addresses by specifying bus modes:

#### ■ Memory Space for Each Mode



Part No.	Address #1	Address #2	Address #3
MB90573	FE0000 <sub>II</sub>	004000 <sub>II</sub>	001900 <sub>II</sub>
MB90574/B	FC0000 <sub>II</sub>	004000 <sub>II</sub>	002900 <sub>II</sub>
MB90F574/A	FC0000 <sub>II</sub>	004000 <sub>II</sub>	002900 <sub>II</sub>
MB90V570/A	(FC0000 <sub>II</sub> )	004000 <sub>II</sub>	002900 <sub>II</sub>

### Note:

If "No ROM mirror function" is selected, see CHAPTER 25 "ROM MIRROR FUNCTION SELECTION MODULE"

The ROM of the FF bank can be seen as an image in the higher position of the 00 bank. However, the purpose of this method is to make effective use of the small model of the C compiler. The lower 16 bits are made equal, and so the table in the ROM can be referenced without "far" specified in the pointer declaration.

For example, if 00C000<sub>H</sub> is accessed, the contents of the ROM at FFC000<sub>H</sub> is actually supposed to be accessed. The ROM area of the FF bank here exceeds 48K bytes, and so the entire area cannot be seen in the 00 bank image. Therefore, the ROM data from FF4000<sub>H</sub> to FFFFFFF<sub>H</sub> is seen as the image from 004000<sub>H</sub> to 00FFFF<sub>H</sub>, and so storing the ROM data table in the area from FF4000<sub>H</sub> to FFFFFFF<sub>H</sub> is recommended.

### ■ Examples of Recommended Settings

**Table 7.1-3 Examples of Recommended Settings for Mode Pins and Mode Data**

Example of setting	MD2	MD1	MD0	M1	M0	S0
Single chip	0	1	1	0	0	X
Internal ROM external bus and 8-bit bus	0	1	1	0	1	0
Internal ROM external bus and 16-bit bus	0	1	1	0	1	1
External ROM external bus and 16-bit bus	0	0	1	1	0	1
External ROM external bus and 8-bit bus	0	0	0	1	0	0

For an external pin, the signal to be input or output varies depending on the type of mode.

**Table 7.1-4 Operations of External Pins Relating to Modes**

Pin name	Function	
	Single chip	External bus extension
		8 bits      16 bits
P07~00	Port	AD07~00
P17~10		A15~08
P27~20		A23~16*      AD15~08
P30		ALE
P31		RDX
P32		WRLX*
P33		Port
P34		HRQ*      HRHX*
P35		HAKX*
P36		RDY*
P37		CLK*

**Note:**

The higher address, WRLX, WRHX, HAKX, HRQ, RDY, and CLK can be used as ports by selecting a function. For details, see Section 7.2 "External Memory Access (External Bus Pin Control Circuit)".

## 7.2 External Memory Access (External Bus Pin Control Circuit)

---

The external bus pin control circuit controls the external bus pins used to externally extend the address and data buses of the CPU.

---

### ■ External Memory Access (External Bus Pin Control Circuit)

To access the external memory and peripherals of the device, F<sup>2</sup>MC-16LX provides the following addresses, data, and control signals:

- **CLK (P37):**  
Outputs the machine cycle clock (KBP).
- **RDY (P36):**  
External ready input pin
- **WRHX (P33):**  
Signal for writing higher eight bits of the data bus
- **WRLX (P32):**  
Signal for writing lower eight bits of the data bus
- **RDX (P31)**  
Read signal
- **ALE (P30):**  
Address latch enable signal

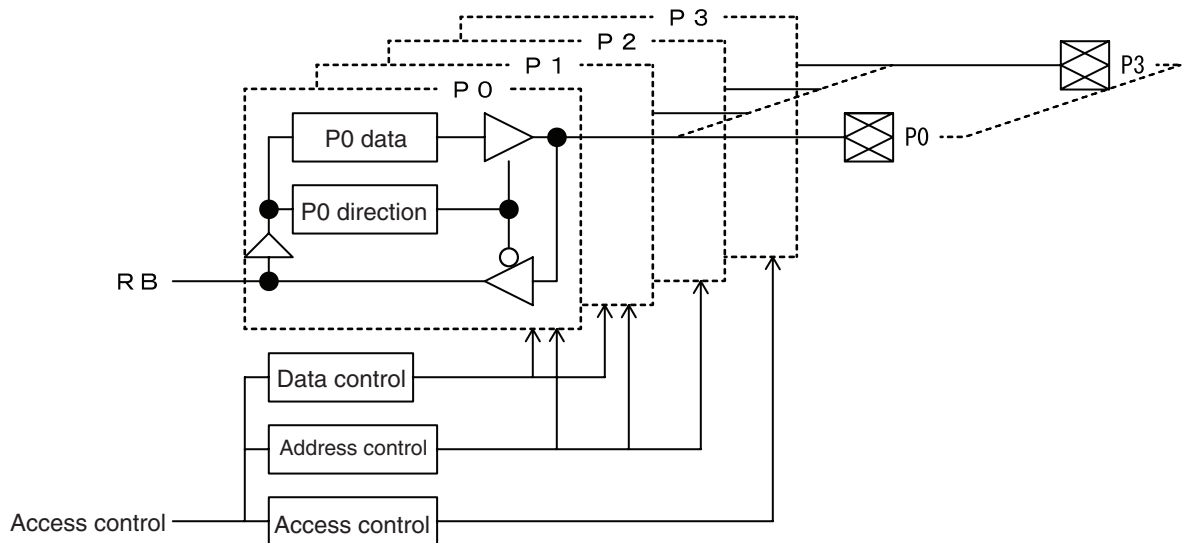
### ■ Configuration of External Memory Access Registers

**Figure 7.2-1 Configuration of External Memory Access Registers**

Automatic ready function selection register	15	14	13	12	11	10	9	8	⇔ Bit No.
Address:0000A5 <sub>H</sub>	IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write ⇔	(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
Initial value ⇔	(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	
External address output control register	7	6	5	4	3	2	1	0	⇔ Bit No.
Address:0000A6 <sub>H</sub>	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇔	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇔	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Bus control signal selection register	15	14	13	12	11	10	9	8	⇔ Bit No.
Address:0000A7 <sub>H</sub>	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇔	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	
Initial value ⇔	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(-)	

### ■ Block Diagram for External Memory Access

**Figure 7.2-2 Block Diagram for External Memory Access**





## 7.2.1 Automatic Ready Function Selection Register (ARSR)

The automatic ready function selection register (ARSR) sets the automatic wait time of memory access for each area when accessing the external memory.

### ■ Automatic Ready Function Selection Register (ARSR)

**Figure 7.2-3 Automatic Ready Function Selection Register (ARSR)**

Automatic ready function selection register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 0000A5 <sub>H</sub>	IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write ⇒	(W)	(W)	(W)	(W)	(—)	(—)	(W)	(W)	
Initial value ⇒	(0)	(0)	(1)	(1)	(—)	(—)	(0)	(0)	

#### [Bits 15, 14]: IOR1, IOR0

Specify the automatic wait function when an external access is made to the area from 0000C0<sub>H</sub> to 0000FF<sub>H</sub>. The following settings are obtained by combining two bits:

**Table 7.2-1 Setting of IOR1 and IOR0 Bits**

IOR1	IOR0	Setting
0	0	Automatic wait prohibited [Initial value]
0	1	At external access, an automatic wait of a 1 machine cycle is inserted.
1	0	At external access, an automatic wait of 2 machine cycles is inserted.
1	1	At external access, an automatic wait of 3 machine cycles is inserted.

#### [Bits 13, 12]: HMR1, HMR0

Specify the automatic wait function when an external access is made to the area from 800000<sub>H</sub> to FFFFFFFF<sub>H</sub>. The following settings are obtained by combining two bits:

**Table 7.2-2 Setting of HMR1 and HMR0 Bits**

HMR1	HMR0	Setting
0	0	Automatic wait prohibited
0	1	At external access, an automatic wait of a 1 machine cycle is inserted.
1	0	At external access, an automatic wait of 2 machine cycles is inserted.
1	1	At external access, an automatic wait of 3 machine cycles is inserted. [Initial value]

## 7.2 External Memory Access (External Bus Pin Control Circuit)

### [Bits 9, 8]: LMR1, LMR0

Specify the automatic wait function when an external access is made to the area from 002000<sub>H</sub> to 7FFFFFF<sub>H</sub>. The following settings are obtained by combining two bits:

**Table 7.2-3 Setting of LMR1 and LMR0 Bits**

LMR1	LMR0	Setting
0	0	Automatic wait prohibited [Initial value]
0	1	At external access, an automatic wait of a 1 machine cycle is inserted.
1	0	At external access, an automatic wait of 2 machine cycles is inserted.
1	1	At external access, an automatic wait of 3 machine cycles is inserted.

### 7.2.2 External Address Output Control Register (HACR)

The external address output control register (HACR) controls the external output of the addresses A23 to A16. The bits of the register correspond to addresses A23 to A16, respectively, and control the address output pins, as shown in Table 7.2-4 "Control of the External Address Output Control Register (bits A23 to A16)".

■ External Address Output Control Register (HACR)

Figure 7.2-4 External Address Output Control Register (HACR)

External address output control register	7	6	5	4	3	2	1	0	⇔ Bit No.
Address:0000A6 <sub>H</sub>	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇔	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇔	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

This register cannot be accessed when the device is in single-chip mode. In this case, all pins function as the I/O port regardless of the value of this register.

All bits of this register are dedicated to writing and are set to 1 for reading.

Table 7.2-4 Control of the External Address Output Control Register (bits A23 to A16)

A23~16	Control
0	The associated pin is the address output (AXX). [Initial value]
1	The associated pin is the I/O port (PXX).

### 7.2.3 Bus Control Signal Selection Register (ECSR)

This register sets the bus operation control function in an external bus mode. This register cannot be accessed when the device is in single-chip mode. In this case, all pins function as the I/O port regardless of the value of this register. All bits of this register are dedicated to writing and are set to 1 for reading.

#### ■ Bus Control Signal Selection Register (ECSR)

**Figure 7.2-5 Bus Control Signal Selection Register (ECSR)**

Bus control signal selection register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 0000A7 <sub>H</sub>	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(—)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(—)	

#### [Bit 15]: CKE

Controls the output of the external clock (CLK) as shown below.

**Table 7.2-5 Control of the CKE Bit**

CKE	Control
0	I/O port (P37) operation (clock output prohibited) [Initial value]
1	Clock signal (CLK) output allowed

#### [Bit 14]: RYE

Controls the input of the external ready (RDY) as shown below.

**Table 7.2-6 Control of the RYE Bit**

RYE	Control
0	I/O port (P36) operation (external RDY input prohibited) [Initial value]
1	External ready (RDY) input allowed

#### [Bit 13]: HDE

Specifies that I/O of hold pins is allowed. Setting this bit controls the hold request input (HRQ) and hold acknowledge output (HAKX) as shown below.

**Table 7.2-7 Control of the HDE Bit**

HDE	Control
0	I/O port (P35, P34) operation (hold function I/O prohibited [Initial value])
1	Hold request (HRQ) input allowed or hold acknowledge (HAKX) output allowed

**[Bit 12]: IOBS**

Specifies the bus size used when an external access is made to the 0000C0<sub>H</sub> to 0000FF<sub>H</sub> area in the external data bus 16-bit mode. Setting this bit enables the following controls:

**Table 7.2-8 Setting of the the IOBS Bit**

IOBS	Control
0	16-bit bus size access [Initial value]
1	8-bit bus size access

**[Bit 11]: HMBS**

Specifies the bus size used when an external access is made to the 800000<sub>H</sub> to FFFFFFF<sub>H</sub> area in the external data bus 16-bit mode. Setting this bit enables the following controls:

**Table 7.2-9 Control of the HMBS Bit**

HMBS	Control
0	16-bit bus size access [Initial value for external vector mode 1]
1	8-bit bus size access [Initial value for external vector mode 0]

**[Bit 10]: WRE**

Controls the output of the external write signal (WRHX and WRLX pins in the 16-bit bus mode, WRLX pin in the 8-bit bus mode) as shown below.

**Table 7.2-10 Control of the WRE Bit**

WRE	Control
0	I/O port (P33, P32) operation (write signal output prohibited) [Initial value]
1	Output of the write strobe signal (WRHX and WRLX, or WRLX only) allowed

In the external data bus 8-bit mode, P33 functions as the I/O port regardless of the value of this bit.

**[Bit 9]: LMBS**

Specifies the bus size used when an external access was made to the 002000<sub>H</sub> to 7FFFFFF<sub>H</sub> area in the external data bus 16-bit mode. Setting this bit enables the following controls:

Table 7.2-11 Control of the LMBS Bit

LMBS	Control
0	16-bit bus size access [Initial value]
1	8-bit bus size access

**Note:**

To enable the WRHX and WRLX functions using the WRE bit in the 16-bit bus mode, place P33 and P32 in the input mode. (Set bits 3 and 2 of DDR3 to 0.)

To enable the WRX function using the WRE bit in the 8-bit bus mode, place P32 in the input mode. (Set bit 2 of DDR3 to 0.)

Even if the RDY or HRQ input is allowed using the RYE or HDE bit, the I/O port function of the port is validated. The bit corresponding to the port in DDR3 must be set to 0 (input mode).

## 7.3 Operation of the External Memory Access Control Signal

The external memory is accessed in three cycles when the ready function is not used. The 8-bit bus width access in the external 16-bit bus mode is used to read or write a peripheral chip 8 bits in width when peripheral chips 8 bits in width and 16 bits in width are connected to the external bus.

MB90570 series support various modes for access methods and access areas. See Section 7.1 "Memory Access Modes".

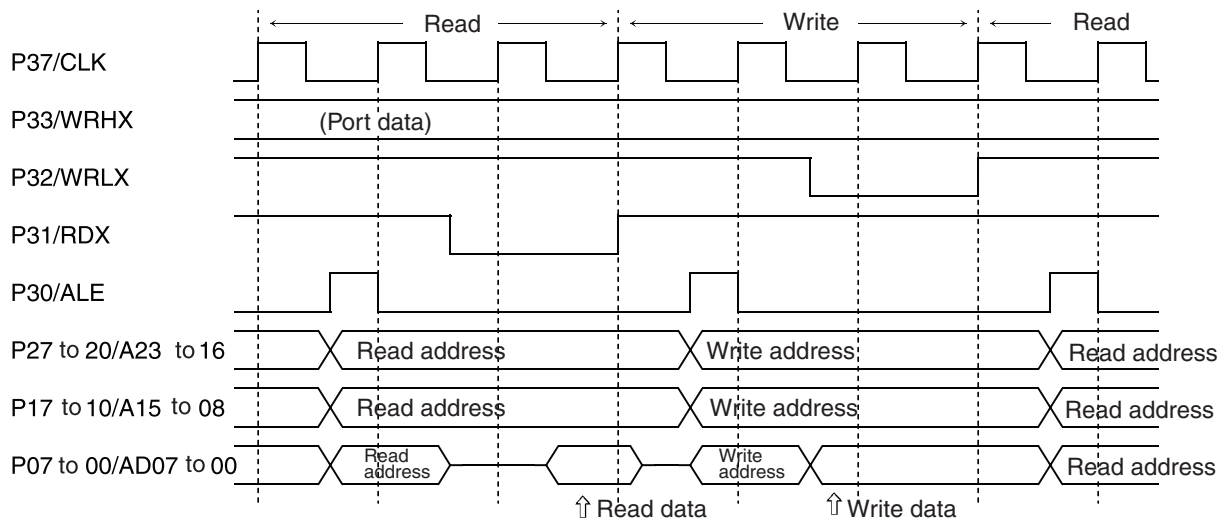
### ■ External Memory Access Control Signal

Because the 8-bit bus width access is executed using the lower eight bits of the data bus, the peripheral chip 8 bits in width must be connected to the lower eight bits of the data bus.

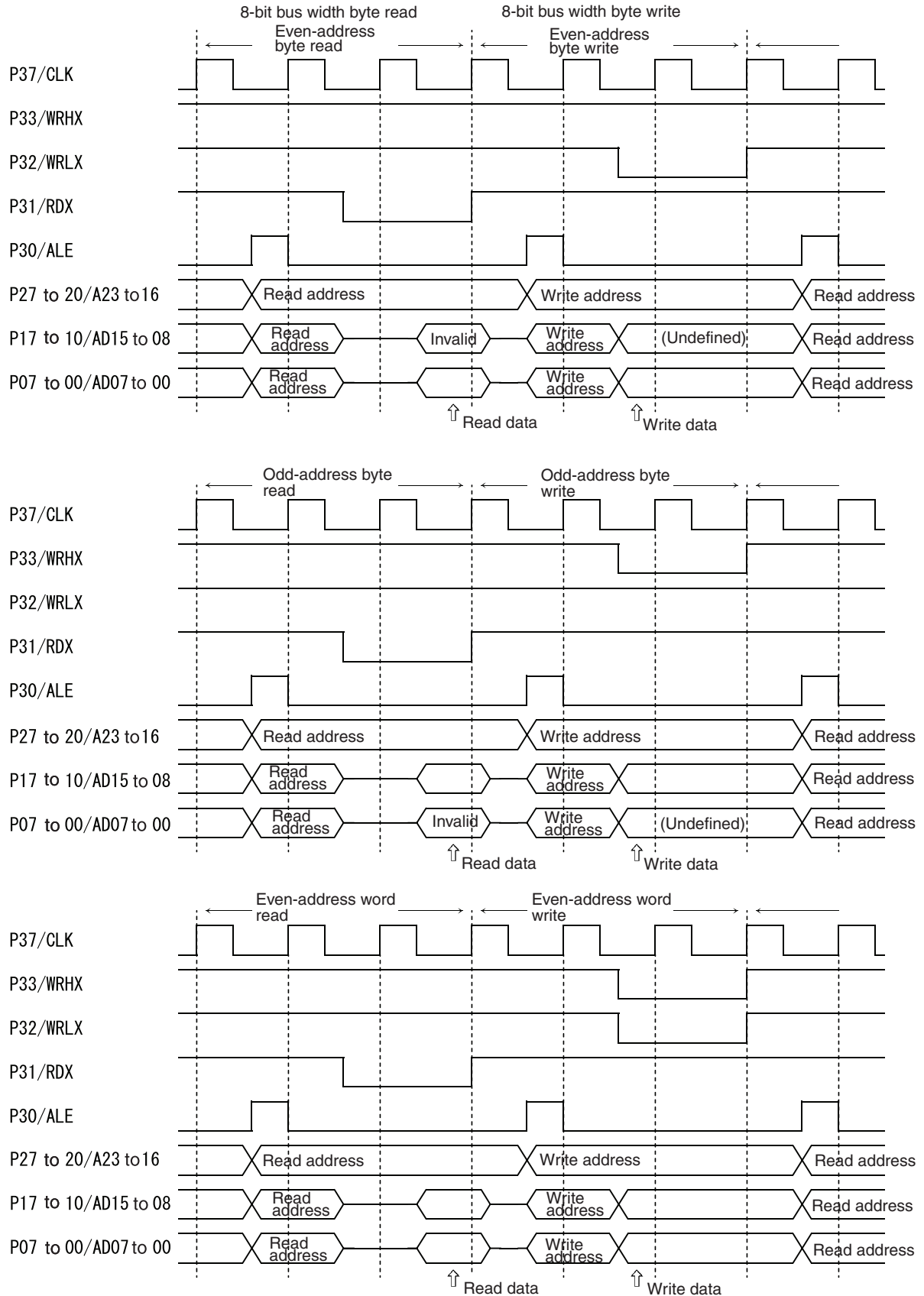
Use the HMBS, LMBS, and IOBS bits of the EPCR to specify whether a 16-bit bus width access or an 8-bit bus width access is to be made in the external 16-bit bus mode.

A bus operation may not be performed because RDX, WRLX, or WRHX is not asserted. A peripheral chip must not be accessed by the ALE signal only.

**Figure 7.3-1 External Memory Access Timing Chart (External 8-bit Bus Mode)**



**Figure 7.3-2 External Memory Access Timing Chart (External 16-bit Bus Mode)**





**Note:**

Design the external circuit so that it is always read with words.

Low-speed memory or peripheral circuits can be accessed by setting the P36/RDY pin or the automatic ready function selection register (ARSR).

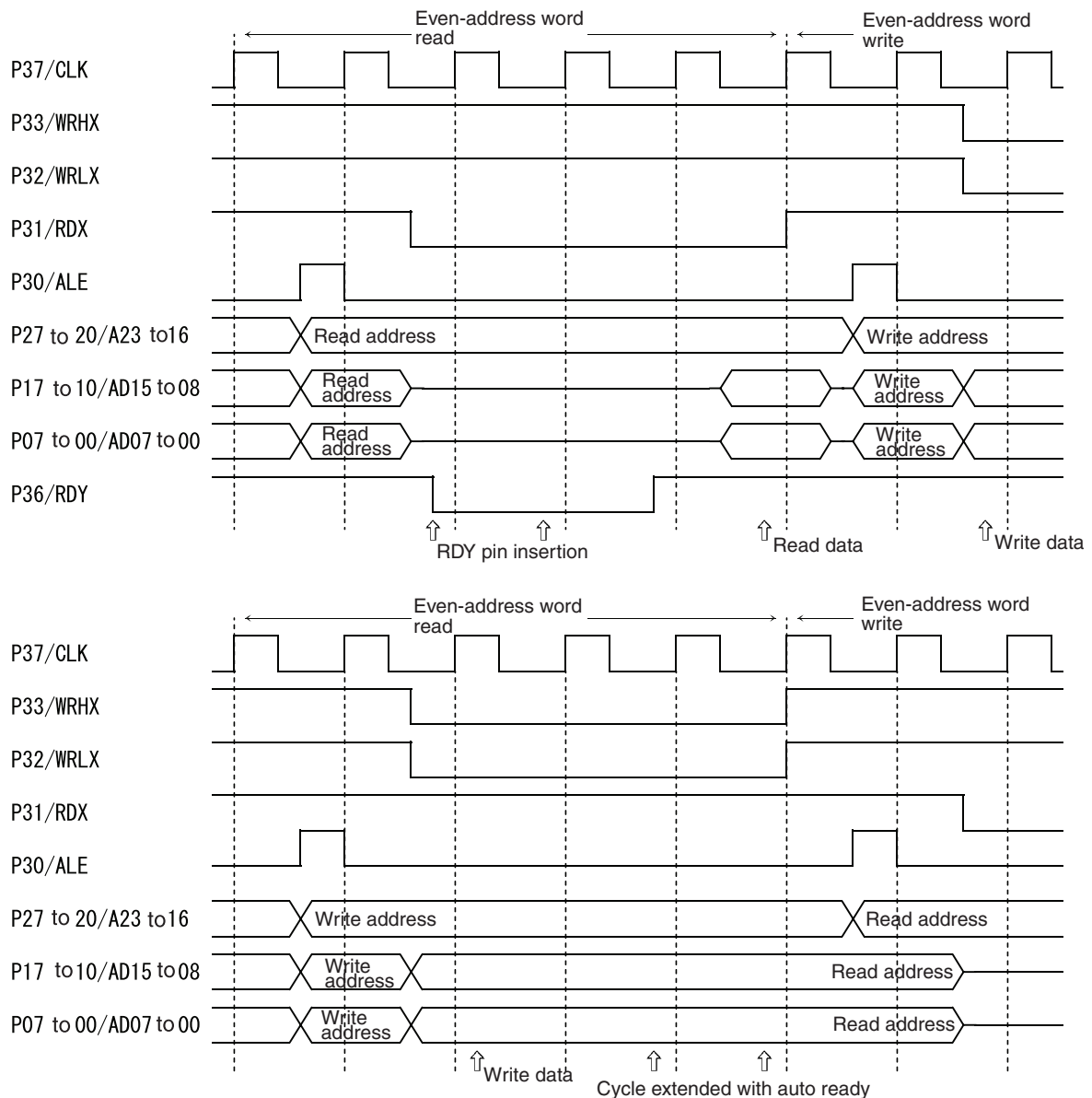
## 7.3.1 Ready Function

Low-speed memory or peripheral circuits can be accessed by setting the P36/RDY pin or the automatic ready function selection register (ARSR).

If the RYE bit in the bus control signal selection register (EPCR) is set to 1, a wait cycle occurs while the L level is input in the P36/RDY pin at access to an external area. The access cycle can be extended.

### ■ Ready Function

Figure 7.3-3 Ready Timing Chart



F<sup>2</sup>MC-16LX contains two types of auto ready functions for external memory. The auto ready function can automatically insert one to three wait cycles without an external circuit to extend the access cycle in the following cases: generation of an access to a lower-address external area provided at addresses 002000<sub>H</sub> to 7FFFFFF<sub>H</sub> and generation of a higher-address external area provided at addresses 800000<sub>H</sub> to FFFFFFF<sub>H</sub>. This function is activated by setting the LMR1 and LMR0 bits in the ARSR (lower-address external area) and the HMR1 and HMR0 bits in the ARSR (higher-address external area).

F<sup>2</sup>MC-16LX contains the auto ready function for external I/O independent of the auto ready function for the memory.

The auto ready function for external I/O can automatically insert one to three wait cycles without an external circuit to extend the access cycle when accessing an external area between addresses 0000C0<sub>H</sub> to 0000FF<sub>H</sub>. The function is activated by setting the IOR1 and IOR0 bits in the ARSR.

Assume that the RYE bit in the EPCR is set to 1 for either the external memory auto ready or the external I/O auto ready. In this case, if the L level is input in the P36/RDY pin after the wait cycle generated by the above auto ready function ends, the wait cycle continues.

## 7.3.2 Hold Function

If the HDE bit in the EPCR is set to 1, the external bus hold function by the P34/HRQ and P35/HAKX pins is validated.

### ■ Hold Function

If the H level is input in the P34/HRQ pin, the hold status is entered when the CPU instruction ends (for the string instruction when 1-element data processing ends). The following pins attain high-impedance status by outputting the L level from the P35/HAKX pin.

- Address output: P23/A19~P20/A16
- Address and data I/O: P17/D15~P00/D00
- Bus control signal: P30/ALE , P31/RDX ,P32/WRLX, P33/WRHX

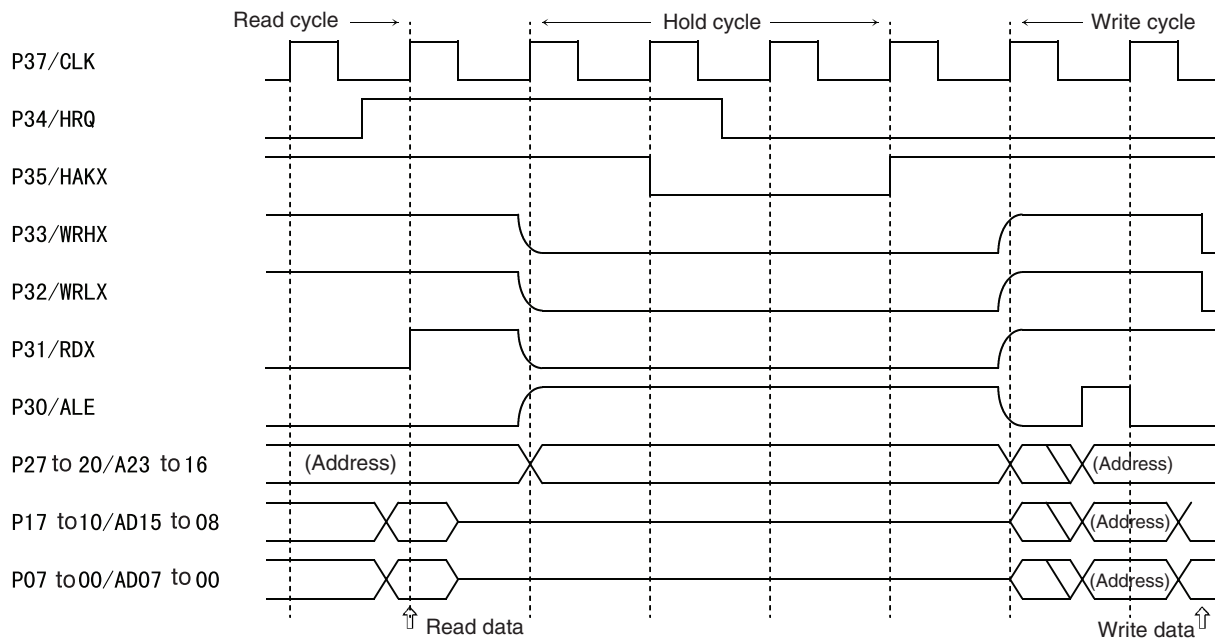
This enables external buses to be used from the external circuit of the device.

If the L level is input in the P34/HRQ pin, the P35/HAKX pin attains H-level output and the external pin status is rendered active so as to restart CPU operation.

No hold request input is accepted in the STOP status.

Figure 7.3-4 "Hold Timing (in the External 16-bit Mode)" shows the hold timing in external bus 16-bit mode.

**Figure 7.3-4 Hold Timing (in the External 16-bit Mode)**





## CHAPTER 8 I/O PORT

---

**This chapter describes the functions and operations of the I/O port.**

---

8.1 "Overview of the I/O Port"

8.2 "Registers of the I/O Port"

## 8.1 Overview of the I/O Port

---

**Input or output can be specified by setting the port direction register (PDR) individually for the pins in each port if the corresponding peripheral circuit is not using the pin.**

---

### ■ Overview of the I/O Port

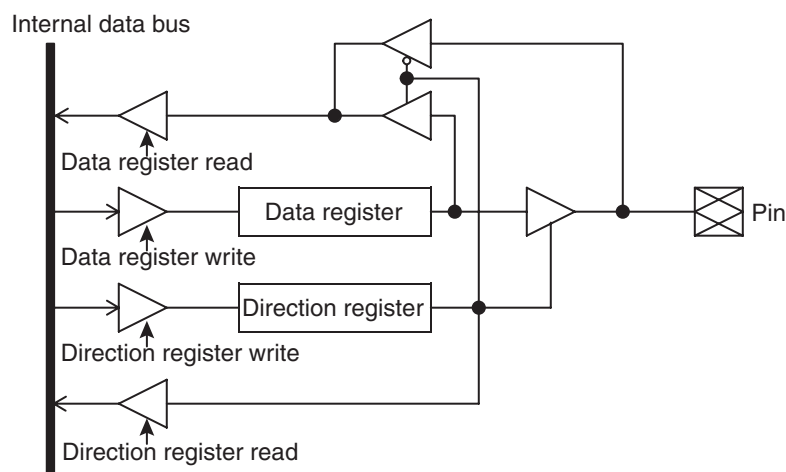
Input or output can be specified by setting the port direction register (PDR) individually for the pins in each port if the corresponding peripheral circuit is not using the pin. If a port data register (PDR) is read for data input, the value corresponding to the level of the pin is read. If a port data register (PDR) is read for data output, the latch value of the PDR register is read. This applies also for reading in a read modify write operation.

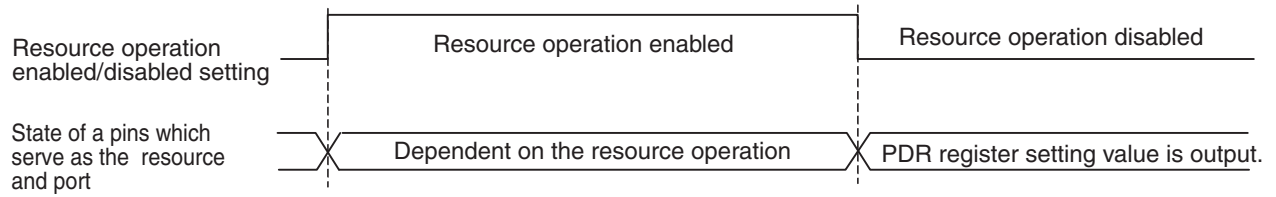
When the pin is used for control output, the signal output as control output is read if a PDR register is read, regardless of the value of the PDR register.

Note the following when a pin specified as input pin is changed to become an output pin: If a read modify write instruction (e.g., bit set) is used for presetting output data in a PDR register, input data from the pin, not the latch value of the data, is read out.

Figure 8.1-1 "Block Diagram of I/O Port" shows the block diagram of an I/O port.

**Figure 8.1-1 Block Diagram of I/O Port**



**■ Operation of a Port Used as a Resource When Port Output is Set by a PDR Register****Figure 8.1-2 State of Pin of Port Used as a Resource When Resource Operation is Enabled/Disabled**



## 8.2 Registers of the I/O Port

Figure 8.2-1 "Configuration of I/O Port Registers" shows the bit configurations of the I/O port registers.

### ■ Configuration of I/O Port Registers

Figure 8.2-1 Configuration of I/O Port Registers

Bit No. ⇒	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000000H	P07	P06	P05	P04	P03	P02	P01	P00	Port 0 data register (PDR0)
Address : 000001H	P17	P16	P15	P14	P13	P12	P11	P10	Port 1 data register (PDR1)
Address : 000002H	P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address : 000003H	P37	P36	P35	P34	P33	P32	P31	P30	Port 3 data register (PDR3)
Address : 000004H	P47	P46	P45	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address : 000005H	P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address : 000006H	P67	P66	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address : 000007H	—	—	—	P74	P73	P72	P71	P70	Port 7 data register (PDR7)
Address : 000008H	P87	P86	P85	P84	P83	P82	P81	P80	Port 8 data register (PDR8)
Address : 000009H	P97	P96	P95	P94	P93	P92	P91	P90	Port 9 data register (PDR9)
Address : 00000AH	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	Port A data register (PDRA)
Address : 00000BH	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	Port B data register (PDRTB)
Address : 00000CH	—	—	—	—	PC3	PC2	PC1	PC0	Port C data register (PDRTC)

Bit No. ⇒	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000010H	D07	D06	D05	D04	D03	D02	D01	D00	Port 0 direction register (DDR0)
Address : 000011H	D17	D16	D15	D14	D13	D12	D11	D10	Port 1 direction register (DDR1)
Address : 000012H	D27	D26	D25	D24	D23	D22	D21	D20	Port 2 direction register (DDR2)
Address : 000013H	D37	D36	D35	D34	D33	D32	D31	D30	Port 3 direction register (DDR3)
Address : 000014H	D47	D46	D45	D44	D43	D42	D41	D40	Port 4 direction register (DDR4)
Address : 000015H	D57	D56	D55	D54	D53	D52	D51	D50	Port 5 direction register (DDR5)
Address : 000016H	D67	D66	D65	D64	D63	D62	D61	D60	Port 6 direction register (DDR6)
Address : 000017H	—	—	—	D74	D73	D72	D71	D70	Port 7 direction register (DDR7)
Address : 000018H	D87	D86	D85	D84	D83	D82	D81	D80	Port 8 direction register (DDR8)
Address : 000019H	D97	D96	D95	D94	D93	D92	D91	D90	Port 9 direction register (DDR9)
Address : 00001AH	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	Port A direction register (DDRA)
Address : 00001BH	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Port B direction register (DDRB)
Address : 00001CH	—	—	—	—	DC3	DC2	DC1	DC0	Port C direction register (DDRC)

Bit No. ⇒	15	14	13	12	11	10	9	8	
Address : 00001D <sub>H</sub>	OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	Port 4 output pin register (ODR4)

Bit No. ⇒	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 00008C <sub>H</sub>	RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	Port 0 resistor register (RDR0)
Address : 00008D <sub>H</sub>	RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	Port 1 resistor register (RDR1)
Address : 00008E <sub>H</sub>	RD67	RD66	RD65	RD64	RD63	RD62	RD61	RD60	Port 6 resistor register (RDR6)

Bit No. ⇒	15	14	13	12	11	10	9	8	
Address : 00001E <sub>H</sub>	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 8 analog input enable register (ADER)

## 8.2.1 Port Data Register (PDR)

The following figure shows the bit configuration of the port data register (PDR).

### ■ Port Data Register (PDR)

**Figure 8.2-2 Port Data Register (PDR)**

Bit No. ⇨	7	6	5	4	3	2	1	0	Initial value	Access
PDR0 Address : 000000H	P07	P06	P05	P04	P03	P02	P01	P00	Undefined	R/W *
Bit No. ⇨	15	14	13	12	11	10	9	8		
PDR1 Address : 000001H	P17	P16	P15	P14	P13	P12	P11	P10	Undefined	R/W *
Bit No. ⇨	7	6	5	4	3	2	1	0		
PDR2 Address : 000002H	P27	P26	P25	P24	P23	P22	P21	P20	Undefined	R/W *
Bit No. ⇨	15	14	13	12	11	10	9	8		
PDR3 Address : 000003H	P37	P36	P35	P34	P33	P32	P31	P30	Undefined	R/W *
Bit No. ⇨	7	6	5	4	3	2	1	0		
PDR4 Address : 000004H	P47	P46	P45	P44	P43	P42	P41	P40	Undefined	R/W *
Bit No. ⇨	15	14	13	12	11	10	9	8		
PDR5 Address : 000005H	P57	P56	P55	P54	P53	P52	P51	P50	Undefined	R/W *
Bit No. ⇨	7	6	5	4	3	2	1	0		
PDR6 Address : 000006H	P67	P66	P65	P64	P63	P62	P61	P60	Undefined	R/W *
Bit No. ⇨	15	14	13	12	11	10	9	8		
PDR7 Address : 000007H	—	—	—	P74	P73	P72	P71	P70	Undefined	R/W *

Bit No. ⇨	7	6	5	4	3	2	1	0	Initial value	Access
PDR8 Address : 000008H	P87	P86	P85	P84	P83	P82	P81	P80	Undefined	R/W *
Bit No. ⇨	15	14	13	12	11	10	9	8		
PDR9 Address : 000009H	P97	P96	P95	P94	P93	P92	P91	P90	Undefined	R/W *
Bit No. ⇨	7	6	5	4	3	2	1	0		
PDRA Address : 00000AH	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	Undefined	R/W *
Bit No. ⇨	7	6	5	4	3	2	1	0		
PDRB Address : 00000BH	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0		
Bit No. ⇨	7	6	5	4	3	2	1	0		
PDRC Address : 00000CH	—	—	—	—	PC3	PC2	PC1	PC0		

Note: Note that I/O port read/write operations are different than those of memory as follows:

[Input mode]

Read: The level of a corresponding pin is read out.

Write: Data is written in an output latch.

[Output mode]

Read: The value of a data register latch is read out.

Write: Data is output to a corresponding pin

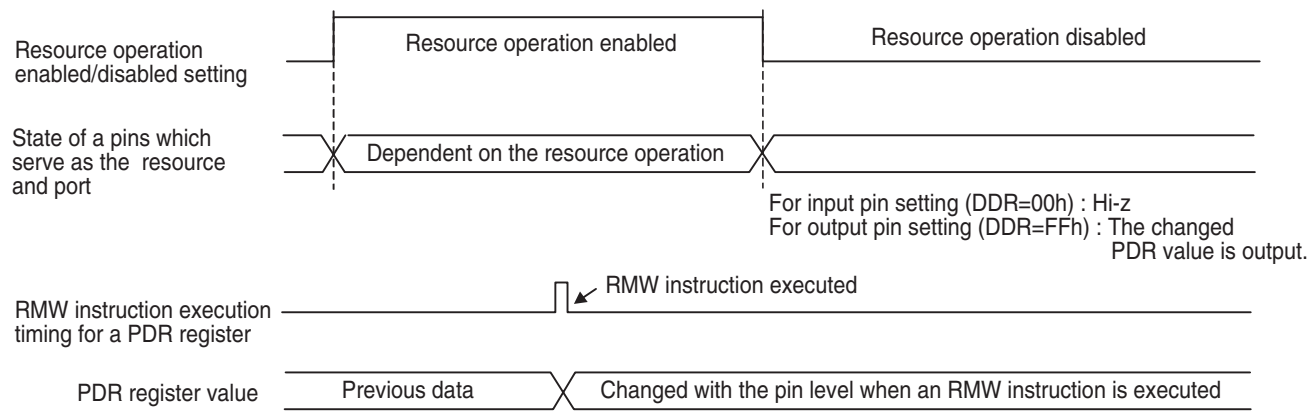
**Note:**

- In port input mode, the pin level is read when reading data if an RMW instruction is executed for a port data register (PDR). Note therefore that values of bits used as signal input ports other than the values involved in the bit operation may change.
- If an RMW instruction is executed for the data register (PDR) of a port also used as a resource during a resource operation, the pin level is read for the pin operated as a resource when reading. Note therefore that the values of bits used as signal input ports other than the values involved in the bit operation may change.

**Table 8.2-1 Data to be Read When an RMW Instruction is Executed for a PDR Register**

	When resource operation is enabled	When resource operation is disabled
When port input is set DDR = 00h	Pin level	Pin level
When port output is set DDR = FFh	Pin level	PDR register value

Figure 8.2-3 State of Port Used also as Resource When an RMW Instruction is Executed



For details on RMW instructions, see Appendix B.8 "List of F<sup>2</sup>MC-16LX Instructions".

## 8.2.2 Port Direction Register (DDR)

Figure 8.2-4 "Port Direction Register (DDR)" shows the bit configuration of the port direction register (DDR).

### ■ Port Direction Register (DDR)

Figure 8.2-4 Port Direction Register (DDR)

	Bit No. ⇒	7	6	5	4	3	2	1	0	Initial value	Access
DDR0 Address : 000010 <sub>H</sub>		D07	D06	D05	D04	D03	D02	D01	D00	00000000 <sub>B</sub>	R/W
DDR1 Address : 000011 <sub>H</sub>	Bit No. ⇒	15	14	13	12	11	10	9	8	00000000 <sub>B</sub>	R/W
		D17	D16	D15	D14	D13	D12	D11	D10		
DDR2 Address : 000012 <sub>H</sub>	Bit No. ⇒	7	6	5	4	3	2	1	0	00000000 <sub>B</sub>	R/W
		D27	D26	D25	D24	D23	D22	D21	D20		
DDR3 Address : 000013 <sub>H</sub>	Bit No. ⇒	15	14	13	12	11	10	9	8	00000000 <sub>B</sub>	R/W
		D37	D36	D35	D34	D33	D32	D31	D30		
DDR4 Address : 000014 <sub>H</sub>	Bit No. ⇒	7	6	5	4	3	2	1	0	00000000 <sub>B</sub>	R/W
		D47	D46	D45	D44	D43	D42	D41	D40		
DDR5 Address : 000015 <sub>H</sub>	Bit No. ⇒	15	14	13	12	11	10	9	8	00000000 <sub>B</sub>	R/W
		D57	D56	D55	D54	D53	D52	D51	D50		
DDR6 Address : 000016 <sub>H</sub>	Bit No. ⇒	7	6	5	4	3	2	1	0	00000000 <sub>B</sub>	R/W
		D67	D66	D65	D64	D63	D62	D61	D60		
DDR7 Address : 000017 <sub>H</sub>	Bit No. ⇒	15	14	13	12	11	10	9	8	-----000 <sub>B</sub>	R/W
		—	—	—	D74	D73	D72	D71	D70		

## CHAPTER 8 I/O PORT

Bit No. ⇒	7	6	5	4	3	2	1	0	Initial value	Access
DDR8 Address : 000018 <sub>H</sub>	D87	D86	D85	D84	D83	D82	D81	D80	00000000 <sub>B</sub>	R/W
Bit No. ⇒	15	14	13	12	11	10	9	8		
DDR9 Address : 000019 <sub>H</sub>	D97	D96	D95	D94	D93	D92	D91	D90	00000000 <sub>B</sub>	R/W
Bit No. ⇒	7	6	5	4	3	2	1	0		
DDRA Address : 00001A <sub>H</sub>	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	--000000 <sub>B</sub>	R/W
Bit No. ⇒	15	14	13	12	11	10	9	8		
DDRB Address : 00001B <sub>H</sub>	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	00000000 <sub>B</sub>	R/W
Bit No. ⇒	7	6	5	4	3	2	1	0		
DDRC Address : 00001C <sub>H</sub>	—	—	—	—	DC3	DC2	DC1	DC0	----0000 <sub>B</sub>	R/W

While a pin operates as a port, the corresponding pin is controlled as follows:

0: Input mode

1: Output mode

Set to 0 by a reset

### 8.2.3 Output Pin Register (ODR)

Figure 8.2-5 "Bit Configuration of the Output Pin Register (ODR)" shows the bit configuration of the output pin register (ODR). Figure 8.2-6 "Block Diagram of the Output Pin Register (ODR)" is a block diagram.

#### ■ Output Pin Register (ODR)

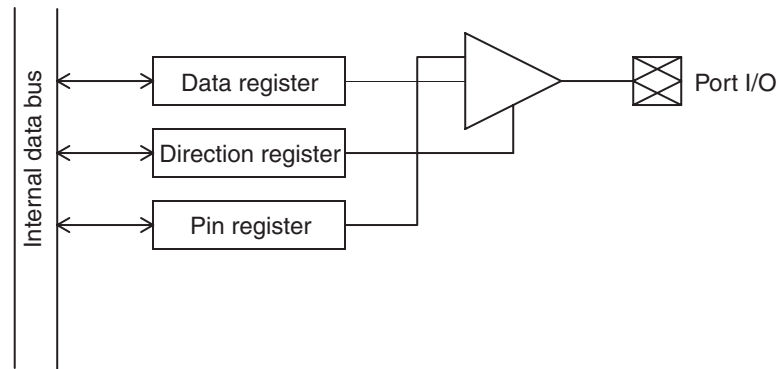
**Figure 8.2-5 Bit Configuration of the Output Pin Register (ODR)**

Port 4 output pin register (ODR4)

Bit No. ⇒	7	6	5	4	3	2	1	0	Initial value
Address: 00001D <sub>H</sub>	OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	00000000 <sub>B</sub>

#### ■ Block Diagram of the Output Pin Register (ODR)

**Figure 8.2-6 Block Diagram of the Output Pin Register (ODR)**



#### ■ Notes on the Output Pin Register (ODR)

The output pin register (ODR: Readable and writable) effects open drain control in output mode.

- 0: Standard output port in output mode
- 1: Open drain output port in output mode
- Meaningless in input mode (output Hi-z)
- The input/output mode is determined by the direction register (DDR).
- No pull-up resistor is provided while hardware is standby or stopped (SPL = 1)(high impedance).
- This function is inhibited for use in an external bus. Do not write this register.



### 8.2.4 Input Resistor Register (RDR)

Figure 8.2-7 "Bit Configuration of Input Resistor Register (RDR)" shows the bit configuration of the input resistor register (RDR). Figure 8.2-8 "Block Diagram of the Input Resistor Register (RDR)" is a block diagram.

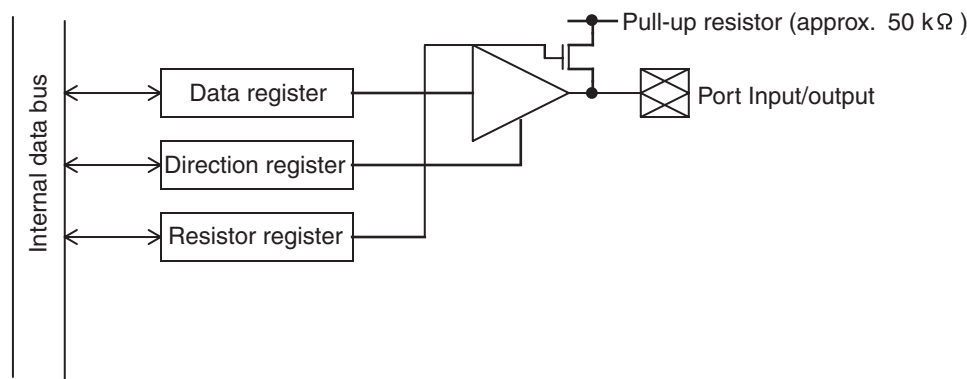
■ Input Resistor Register (RDR)

Figure 8.2-7 Bit Configuration of Input Resistor Register (RDR)

Bit No. ⇒	7	6	5	4	3	2	1	0	
Address : 00008C <sub>H</sub>	RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	Port 0 resistor register (RDR0) Initial value : 00000000 <sub>B</sub>
Bit No. ⇒	15	14	13	12	11	10	9	8	
Address : 00008D <sub>H</sub>	RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	Port 1 resistor register (RDR1) Initial value : 00000000 <sub>B</sub>
Bit No. ⇒	7	6	5	4	3	2	1	0	
Address : 00008E <sub>H</sub>	RD67	RD66	RD65	RD64	RD63	RD62	RD61	RD60	Port 6 resistor register (RDR6) Initial value : 00000000 <sub>B</sub>

■ Block Diagram of the Input Resistor Register

Figure 8.2-8 Block Diagram of the Input Resistor Register (RDR)



■ Notes on the Input Resistance Register (PDR)

The input resistance register (PDR: Readable and writable) effects pull-up resistor control in input mode.

- 0: Without pull-up resistor in input mode
- 1: With pull-up resistor in input mode

- Meaningless in output mode (without pull-up resistor).
- The input/output mode is determined by the direction register (DDR).
- No pull-up resistor is provided while hardware is standby or stopped (LPMCR: SPL = 1)(high impedance).
- This function is inhibited for use in an external bus. Do not write this register.

### 8.2.5 Analog Input Enable Register (ADER)

Figure 8.2-9 "Bit Configuration of the Analog Input Enable Register (ADER)" shows the bit configuration of the analog input enable register (ADER).

■ Analog Input Enable Register (ADER)

Figure 8.2-9 Bit Configuration of the Analog Input Enable Register (ADER)

Bit No. ⇨	15	14	13	12	11	10	9	8	Initial value
Address : 00001EH	ADER7	ADER6	ADER5	ADER4	ADER3	ADER2	ADER1	ADER0	11111111B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Controls port 8 as follows:

- 0: Port input mode
- 1: Analog input mode

Set to 1 by a reset.

## CHAPTER 9    TIMEBASE TIMER

---

**This chapter describes the functions and operations of the Timebase Timer.**

---

- 9.1 "Overview of the Timebase Timer"
- 9.2 "Timebase Timer Control Register (TBTC)"
- 9.3 "Operation of the Timebase Timer"

# 9.1 Overview of the Timebase Timer

The timebase timer consists of an 18-bit timer and a circuit for controlling the interval interrupt. The timebase timer uses the oscillation clock regardless of the setting of the MCS or SCS bit in the clock selection register (CKSCR).

■ Configuration of the Timebase Timer Register

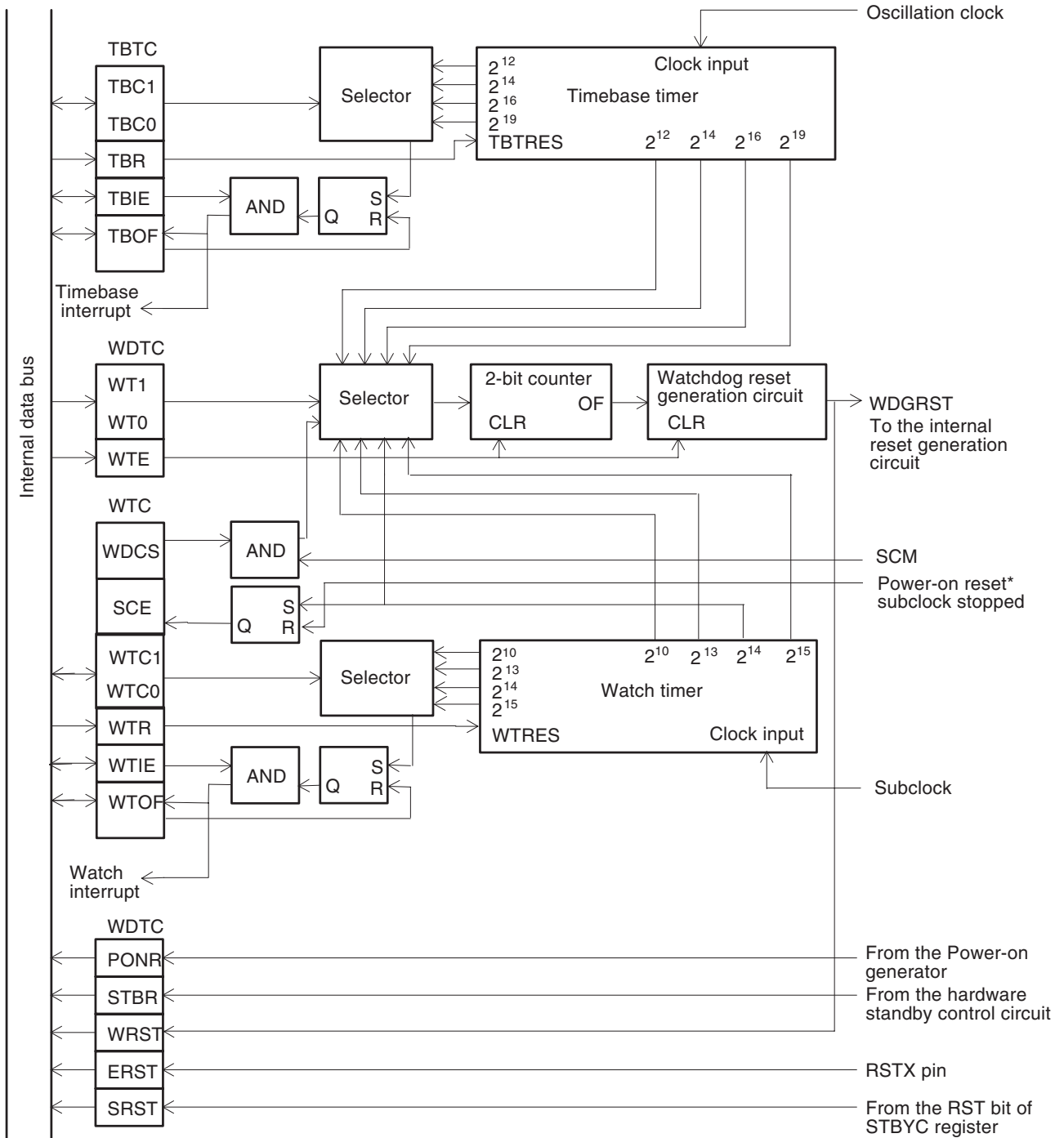
Figure 9.1-1 Configuration of Timebase Timer Register

Timebase timer control register

		15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 0000A9H		Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	TBTC
Read/Write	⇒	(-)	(-)	(-)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value	⇒	(1)	(-)	(-)	(0)	(0)	(1)	(0)	(0)	

## ■ Block Diagram of the Timebase Timer

Figure 9.1-2 Timebase Timer Block Diagram



## 9.2 Timebase Timer Control Register (TBTC)

The timebase timer control register (TBTC) controls the operation of the timebase timer and the interval interrupt time.

### ■ Timebase Timer Control Register (TBTC)

**Figure 9.2-1 Timebase Timer Control Register (TBTC) Configuration**

Timebase timer control register	15	14	13	12	11	10	9	8	⇔ Bit No.
Address : 0000A9H	Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	TBTC
Read/write ⇔	(—)	(—)	(—)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value ⇔	(1)	(—)	(—)	(0)	(0)	(1)	(0)	(0)	

**Note:**

As accessing with a read modify instruction may result in the execution of an incorrect operation, this instruction must not be used.

**[Bit 15] Testing bit**

This bit is used for testing. Always write 1 to this bit.

**[Bits 14 and 13] Unused bit**

Unused

**[Bit 12] TBIE**

This bit is used to enable interval interrupts from the timebase timer. Writing 1 to this bit enables interrupts; writing 0 disables interrupts. This bit is initialized to 0 upon a reset and is a read/write bit.

**[Bit 11] TBOF**

This bit is a flag indicating an interrupt request from the timebase timer. When the TBIE bit is 1, an interrupt request is made if the TBOF bit is set to 1. This bit is set to 1 whenever the interval set with the TBC1 and TBC0 bits elapses. The bit is cleared by writing 0 to the bit or upon a transition to stop or hardware standby mode or a reset. Writing 1 to this bit does not affect operation.

When a read modify write instruction is used, 1 is read from this bit.

**[Bit 10] TBR**

This bit is used to clear all bits of the timebase counter to 0. Writing 0 to this bit clears the timebase counter. Writing 1 to this bit does not affect operation. The number 1 is always read from this bit.

**[Bits 9 and 8] TBC1, TBC0**

These bits are used to set the timebase timer interval. The bits are initialized to 00 upon a reset. The bits are read/write bits.

## 9.2 Timebase Timer Control Register (TBTC)

Table 9.2-1 "Timebase Timer Interval Selection" lists interval settings.

**Table 9.2-1 Timebase Timer Interval Selection**

<b>TBC1</b>	<b>TBC0</b>	<b>Interval time for oscillation at 4 MHz</b>
0	0	1.024 ms
0	1	4.096 ms
1	0	16.384 ms
1	1	131.072 ms



## 9.3 Operation of the Timebase Timer

---

The timebase timer functions as a clock source for the watchdog timer, a timer for the oscillation stabilization time of the main and PLL clocks, and an interval timer for generating interrupts periodically.

---

### ■ Timebase Counter

The timebase timer includes an 18-bit counter that counts entered oscillator clocks from which machine clocks are created. The count operation continues while oscillator clocks are being input. The timebase timer is cleared following a power-on reset, a transition to stop or hardware standby mode, a switch from the main clock to PLL clock with the MCS bit of the CKSCR register, a switch from the main clock to subclock with the SCS bit of the CKSCR register, and writing 0 to the TBR bit of the TBTC register.

Clearing the timebase timer affects the watchdog counter and interval interrupts using outputs from the timebase timer.

### ■ Interval Interrupt Function of the Timebase Timer

Interrupts are generated periodically by the carry signal from the timebase counter. The TBOF flag is set whenever the interval time set with the TBC1 and TBC0 bits of the TBTC register elapses. The flag is set based on the time the timebase timer was last cleared.

When a transition from main clock mode to PLL clock mode occurs, the timebase timer is cleared because it is used as the timer for the oscillation stabilization of the PLL clock.

When a transition from main clock mode to subclock mode occurs, the timebase timer is cleared because it is used as the timer for the oscillation stabilization of the subclock.

Upon a transition to stop or hardware standby mode, the TBOF flag is cleared because the timebase timer is used as the timer for the oscillation stabilization time for a return.

# CHAPTER 10 WATCHDOG TIMER

---

**This chapter describes the functions and operations of the Watchdog Timer.**

---

- 10.1 "Overview of the Watchdog Timer"
- 10.2 "Watchdog Timer Control Register (WDTC)"
- 10.3 "Operation of the Watchdog Timer"

# 10.1 Overview of the Watchdog Timer

The watchdog timer consists of a 2-bit watchdog counter that uses the carry signal of an 18-bit time base timer or 15-bit watch timer as a clock source, a control register, and a watchdog reset control section.

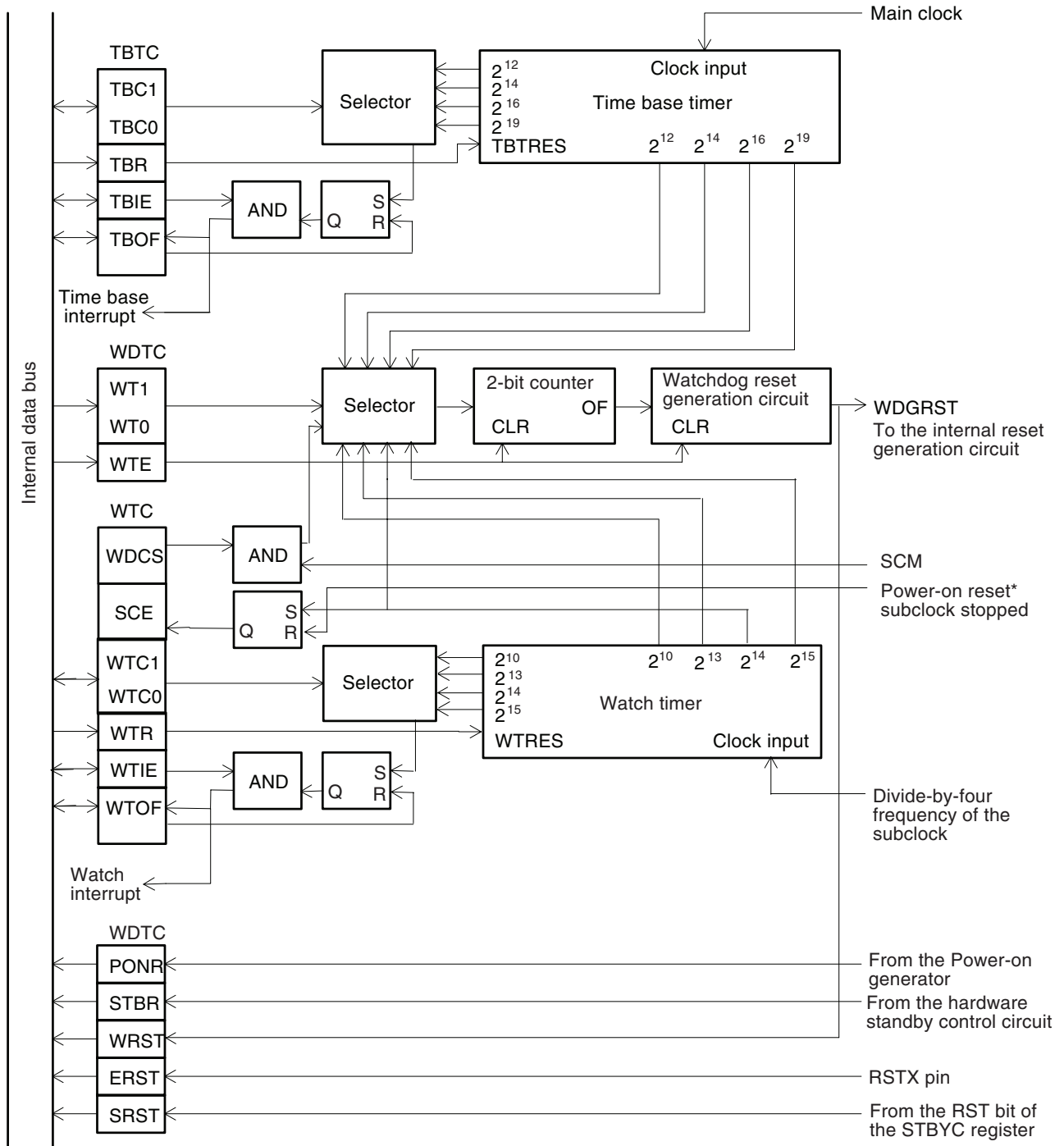
■ Configuration of the Watchdog Timer Control Register

Figure 10.1-1 Configuration of Watchdog Timer Control Register

Watchdog timer control register								
	7	6	5	4	3	2	1	0 ⇐ Bit No.
Address: 0000A8 <sub>H</sub>	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0
Read/Write	⇒ (R)	⇒ (R)	⇒ (R)	⇒ (R)	⇒ (R)	⇒ (W)	⇒ (W)	⇒ (W)
Initial value	⇒ (X)	⇒ (X)	⇒ (X)	⇒ (X)	⇒ (X)	⇒ (X)	⇒ (X)	⇒ (X)

# Block Diagram of the Watchdog Timer

Figure 10.1-2 Watchdog Timer Block Diagram



## 10.2 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) contains bits to control the watchdog timer and bits to identify reset factors.

### ■ Watchdog Timer Control Register (WDTC)

**Figure 10.2-1 Watchdog Timer Control Register (WDTC)**

Watchdog timer control register	7	6	5	4	3	2	1	0	↔ Bit No.
Address : 0000A8 <sub>H</sub>	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	WDTC
Read/write ⇒	(R)	(R)	(R)	(R)	(R)	(W)	(W)	(W)	
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

#### [Bits 7 to 3] PONR, STBR, WRST, ERST, SRST

The bits are flags indicating reset factor. When a reset factor occurs, these bits are set as shown in Table 10.2-1 "Correspondence between Bit Values and Reset Factors". These bits are read-only and set to 0 after a WDTC register read operation. Because the values of reset factor bits other than the PONR bit are not assured at Power-on, ensure during software design that the values of bits other than the PONR bit are ignored when this bit is 1.

**Table 10.2-1 Correspondence between Bit Values and Reset Factors**

Reset factor	PONR	STBR	WRST	ERST	SRST
Power-on	1	—	—	—	—
Hardware standby	*	1	*	*	*
Watchdog timer	*	*	1	*	*
External pin	*	*	*	1	*
RST bit	*	*	*	*	1

\* The previous value is retained.

#### [Bit 2] WTE

While the watchdog timer is being stopped, setting this bit to 0 enables the operation of the watchdog timer. Setting this bit again to 0 clears the watchdog timer counter. Setting it to 1 does not affect the operation.

The watchdog timer is stopped upon the reset by the reset factor bits of power-on, hardware standby, or watchdog timer. The number 1 is always read from this bit.

#### [Bits 1 and 0] WT1, WT0

These bits are write-only bits used to select an interval time for the watchdog timer. Only data written when the watchdog timer is started is valid; data written under other conditions is ignored. A clock to be input to the watchdog timer is selected according to the result of

## 10.2 Watchdog Timer Control Register (WDTC)

ANDing the WDSCS bit of WTC and the SCM bit of LPMCR. When the WDSCS is set to 1, the interval time of the time base timer is selected if the main clock and PLL clock are selected as machine clocks. When WDSCS is set to 0 or when the subclock is selected, the interval time of the watch timer is selected. Table 10.2-2 "Watchdog Timer Interval Selection Bits" lists interval time settings.

**Table 10.2-2 Watchdog Timer Interval Selection Bits (WT1 and WT0)**

WDSCS SCM	WT1	WT0	Interval time (Oscillation: main clock at 4 MHz, subclock at 32 KHz)	
			Min.	Max.
1	0	0	Approx. 3.58 ms	Approx. 4.61 ms
1	0	1	Approx. 14.33 ms	Approx. 18.43 ms
1	1	0	Approx. 57.23 ms	Approx. 73.73 ms
1	1	1	Approx. 458.75 ms	Approx. 589.82 ms
0	0	0	0.438 s	0.563 s
0	0	1	3.500 s	4.500 s
0	1	0	7.000 s	9.000 s
0	1	1	14.00 s	18.00 s

**Note:**

The above maximum interval values apply when the time base counter or watch counter is not reset during a watchdog operation.

### 10.3 Operation of the Watchdog Timer

The watchdog timer can be used to detect program crashes. If 0 is not written to the WTE bit of the watchdog timer within a predetermined time due to a program crash or otherwise, the watchdog timer issues a watchdog reset request.

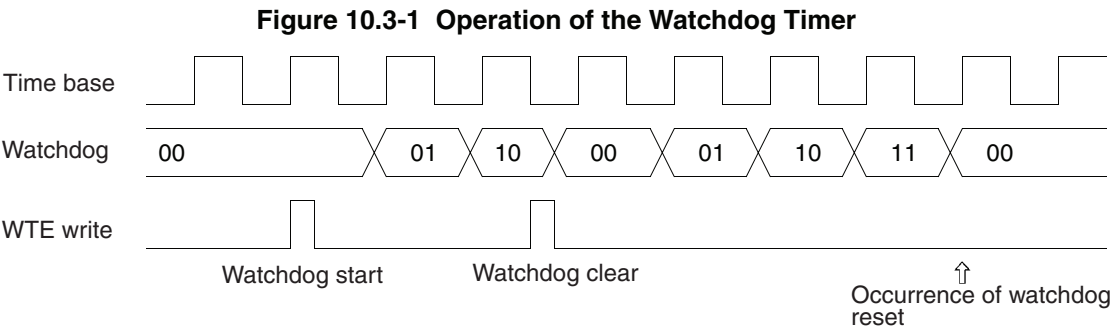
■ Starting the Watchdog Timer

When the watchdog timer has been stopped, 0 can be written to the WTE bit of the WDTC register to start the watchdog timer. At this time, the WT1 and WT0 bits are used to set the interval for watchdog timer reset occurrence. Only data written when the watchdog timer is started is valid as an interval setting.

■ Preventing Watchdog Timer Reset

Once the watchdog timer has started, the 2-bit watchdog counter must be cleared regularly with the program. Specifically, 0 must be written to the WTE bit of the WDTC register regularly. The watchdog counter is configured as a 2-bit counter that uses the carry signal from the time base counter as the clock source. Therefore, when the time base timer is cleared, the interval for watchdog reset occurrence may be longer than the set interval.

Figure 10.3-1 "Operation of the Watchdog Timer" shows the operation of the watchdog timer.



■ Stopping the Watchdog

Once started, the watchdog timer is stopped and initialized only upon the reset by the reset factor bits of power-on, hardware standby, or watchdog timer. A reset due to an external pin or software clears the watchdog counter but does not stop the watchdog function.

■ Clearing the Watchdog Counter

The watchdog counter is cleared by a reset, a transition to sleep or stop mode, a hold acknowledge signal, or a writing to the WTE bit. (The counter is not cleared by a transition to watch mode.)

# CHAPTER 11 WATCH TIMER

---

**This chapter describes the functions and operations of the Watch Timer.**

---

11.1 "Overview of the Watch Timer"

11.2 "Watch Timer Control Register (WTC)"

11.3 "Operation of the Watch Timer"



# 11.1 Overview of the Watch Timer

The watch timer consists of a 15-bit timer and an interval interrupt control circuit. The watch timer uses a subclock regardless of the value of the MCS or SCS bit in the clock selection register (CKSCR).

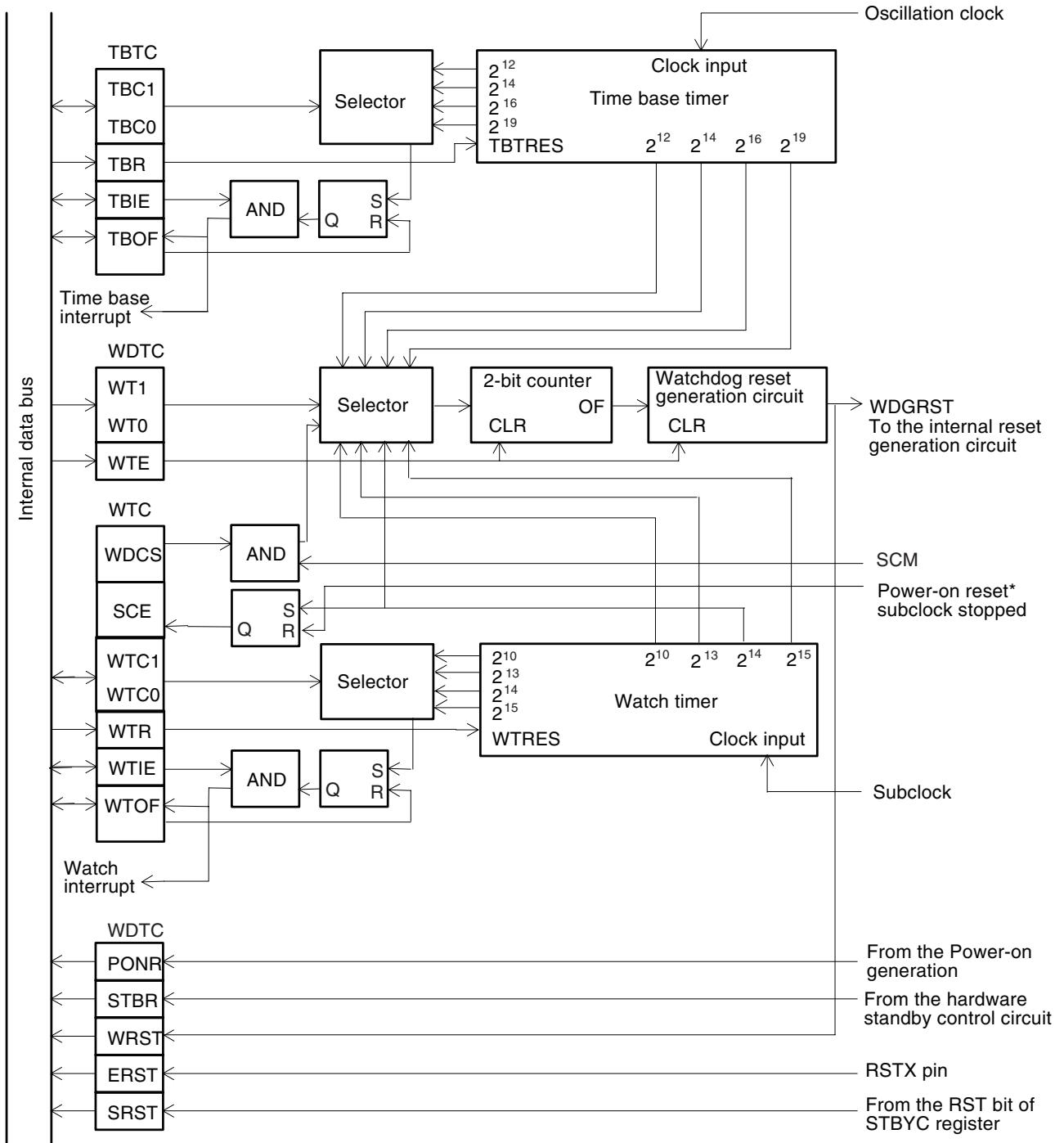
■ Configuration of the Watch Timer Control Register

Figure 11.1-1 Watch Timer Control Register (WTC)

Watch timer control register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 0000AA <sub>H</sub>	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	WTC
Read/Write ⇒	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(1)	(X)	(0)	(0)	(0)	(0)	(0)	(0)	

## ■ Block Diagram of the Watch Timer

Figure 11.1-2 Time Base Timer Block Diagram



## 11.2 Watch Timer Control Register (WTC)

The watch timer control register (WTC) controls the operation of the watch timer and the interval interrupt time.

### ■ Watch Timer Control Register (WTC)

**Figure 11.2-1 Watch Timer Control Register (WTC)**

Watch timer control register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address : 0000AAH	WDSCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	WTC
Read/write ⇐	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇐	(1)	(X)	(0)	(0)	(0)	(0)	(0)	(0)	

#### [Bit 7] WDSCS

When the main clock and PLL clock are selected, this bit is used to select the clock from the watch timer or time base timer as a clock to be input to the watchdog timer. When this bit is 0, the clock from the watch timer is selected; when this bit is 1, the clock from the time base timer is selected. That is, by setting WDSCS to 1, the time base timer output can be selected if the main clock and PLL clock are selected, or the watch timer output can be selected if the subclock is selected.

This bit is initialized to 1 following a power-on reset.

#### Note:

When WDSCS is set to 1, the count by the watchdog timer may be incremented because the time base timer output is asynchronous to the watch timer output. Therefore, when WDSCS is set to 1, the watchdog timer must be cleared before and after the clock mode is changed.

#### [Bit 6] SCE

This bit indicates that the oscillation stabilization period of the subclock has ended. When this bit is 0, the oscillation stabilization period continues. The oscillation stabilization period is fixed to  $2^{16}$  cycles (subclock).

This bit is initialized to 0 following a power-on reset, stop, or watchdog reset.

#### [Bit 5] WTIE

This bit is used to enable interval interrupts from the watch timer. Writing 1 to this bit enables interrupts; writing 0 disables interrupts. This bit is initialized to 0 upon a reset and is a read/write bit.

#### [Bit 4] WTOF

This bit is a flag indicating an interrupt request from the watch timer. When the WTIE bit is 1, an interrupt request is made if the WTOF bit is set to 1. This bit is set to 1 whenever the interval set with the WTC1 and WTC0 bits elapses. The bit is cleared by writing 0 to the bit or upon a transition to stop or hardware standby mode or a reset. Writing 1 to this bit does

not affect operation.

When a read, modify, or write instruction is used, 1 is read from this bit.

### [Bit 3] WTR

This bit is used to clear all bits of the watch timer counter to 0. Writing 0 to this bit clears the watch counter. Writing 1 to this bit does not affect operation. The number 1 is always read from this bit.

### [Bits 2, 1, and 0] WTC2, WTC1, WTC0

These bits are used to set the watch timer interval. Table 11.2-1 "Watch Timer Interval Selection" lists interval settings. The bits are initialized to 000 following a reset and are read/write bits.

When values are written to these bits, clear bit 4 (WTOF).

**Table 11.2-1 Watch Timer Interval Selection**

WTC2	WTC1	WTC0	Interval time *
0	0	0	62.5 ms
0	0	1	125 ms
0	1	0	250 ms
0	1	1	500 ms
1	0	0	1.0 s
1	0	1	2.0 s
1	1	0	4.0 s
1	1	1	—

\*: The interval time is the value when the subclock is set to 32KHz (for an operation clock frequency of 8KHz).

## 11.3 Operation of the Watch Timer

---

The watch timer functions as a clock source for the watchdog counter, a timer for the oscillation stabilization time of the subclock, and an interval timer for generating interrupts periodically.

---

### ■ Watch Counter

The watch timer includes a 15-bit counter that counts entered oscillator clocks from which machine clocks are created. The count operation continues while oscillator clocks are being input. The watch timer is cleared following a power-on reset, a transition to stop or hardware standby mode, and writing 0 to the WTR bit of the WTC register.

Clearing the watch timer affects the watchdog counter and interval interrupts using outputs from the watch timer.

### ■ Interval Interrupt Function of the Watch Timer

Interrupts are generated periodically by the carry signal from the watch counter. The WTOF flag is set whenever the interval time set with the WTC1 and WTC0 bits of the WDTC register elapses. The flag is set based on the time the watch timer was last cleared.

Upon a transition to stop or hardware standby mode, the WTOF flag is cleared because the watch timer is used as the timer for the oscillation stabilization time for a return.

# CHAPTER 12 16-BIT I/O TIMER

---

**This chapter describes the functions and operations of the 16-bit I/O timer.**

---

- 12.1 "Overview of the 16-Bit I/O Timer"
- 12.2 "Block Diagram of the 16-Bit I/O Timer"
- 12.3 "16-Bit Input/Output Timer Register"
- 12.4 "16-Bit Free Run Timer"
- 12.5 "Output Compare"
- 12.6 "Input Capture"
- 12.7 "Operation of the 16-Bit Free Run Timer"
- 12.8 "Operation of the 16-Bit Output Compare"
- 12.9 "Operation of the 16-Bit Input Capture"

## 12.1 Overview of the 16-Bit I/O Timer

---

The 16-bit I/O timer consists of one 16-bit free run timer, four output compare modules, and two input capture modules.

Using this function enables two independent waveforms to be output based on the 16-bit free run timer, enables the input pulse width to be measured, and enables the external clock cycle to be measured.

---

### ■ 16-Bit Free Run Timer (x 1)

The 16-bit free run timer consists of a 16-bit up counter, control register, and prescaler. The output value of this timer counter is used as the basic time (base timer) for the output compare and input capture.

#### ○ The counter operation clock (can be selected from the four types)

Four types of internal clock ( $\phi/4$ ,  $\phi/16$ ,  $\phi/32$ , and  $\phi/64$ )

#### ○ Interrupt

An interrupt can be generated by a counter value overflow or matching by comparison with compare register 0. (The mode must be set for compare match interrupt.)

#### ○ Initialization

The counter can be initialized to 0000<sub>H</sub> by a reset, software clearance, or matching by comparison with compare register 0.

### ■ Output Compare (x 4)

The output compare consists of two 16-bit compare registers, a compare output latch, and a control register. If the value of the 16-bit free run timer matches the compare register value, the pin output level is reverted and an interrupt occurs.

- The four compare registers operate independently.
  - An output pin and interrupt flag are provided for each compare register.
- The four compare registers can be paired to control the output pin.
  - The four compare registers are used to revert the output pin.
- The initial value of the output pin can be set.
- An interrupt can be generated when the values match by comparison.

### ■ Input Capture (x 2)

The input capture consists of capture registers, and control register. Detecting an edge of the signal input from the external input pin enables the value of the 16-bit free run timer to be retained in the capture register and an interrupt to be generated simultaneously.

- An external input signal edge can be selected.
  - The external input signal edge can be selected from the rising edge, falling edge, or both edges.

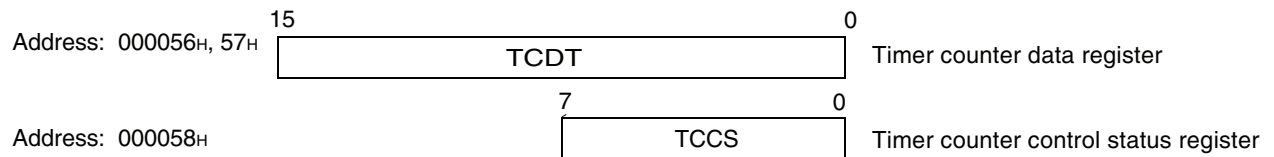
- The two input captures operate independent of each other.
- The valid edge of an external input signal can be used to generate an interrupt.
  - An input capture interrupt starts the intelligent IO service.

## Registers of Entire 16-Bit I/O Timer Section

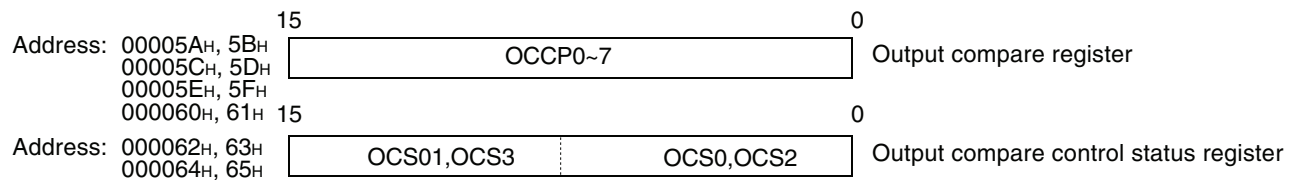
Figure 12.1-1 "Configuration of 16-Bit Input Capture Registers" shows the configuration of registers in the 16-bit input/output timer section.

**Figure 12.1-1 Configuration of 16-Bit Input Capture Registers**

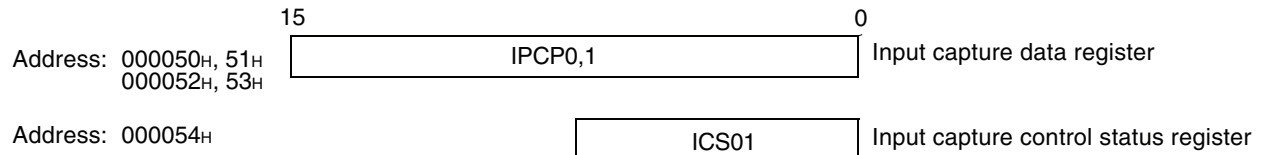
### 16-bit free run timer registers



### 16-bit output compare registers



### 16-bit input capture registers



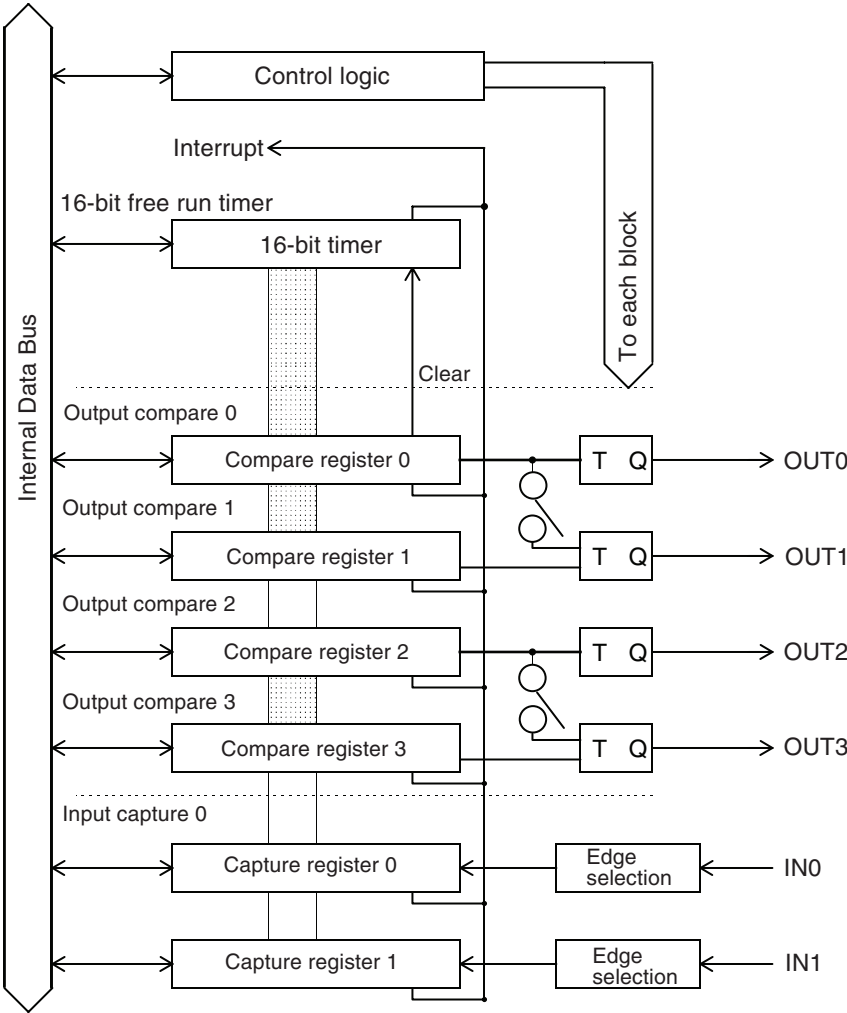


## 12.2 Block Diagram of the 16-Bit I/O Timer

Figure 12.2-1 "Block Diagram of the 16-Bit I/O Timer" is a block diagram of the 16-bit I/O timer.

■ Block Diagram of the 16-Bit I/O Timer

Figure 12.2-1 Block Diagram of the 16-Bit I/O Timer



## 12.3 16-Bit Input/Output Timer Register

There are the following 6 types of 16-bit input/output timer registers:

- Timer counter data register (TCDT)
- Timer counter control status register (TCCS)
- Output compare register (OCCP0 to OCCP3)
- Output compare control status register (OCS0 to OCS3)
- Input capture data register (IPCP0, IPCP1)
- Input capture control status register (ICS01)

### ■ 16-Bit Input/Output Timer Register

**Figure 12.3-1 16-Bit Input/Output Timer Registers (To be continued)**

Higher bits of the timer counter data register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000057H	T15	T14	T13	T12	T11	T10	T09	T08	TCDT(High)
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)	

Lower bits of the timer counter data register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000056H	T07	T06	T05	T04	T03	T02	T01	T00	TCDT(Low)
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Timer counter control status register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000058H	Reserved	IVF	IVFE	STOP	MODE	CLR	CLK1	CLK0	TCCS
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)	

Higher bits of the output compare register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: ch0 00005BH ch1 00005DH ch2 00005FH ch3 000061H	C15	C14	C13	C12	C11	C10	C09	C08	OCCP0~3(High)
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)	

Lower bits of the output compare register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: ch0 00005AH ch1 00005CH ch2 00005EH ch3 000060H	C07	C06	C05	C04	C03	C02	C01	C00	OCCP0~3(Low)
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)	

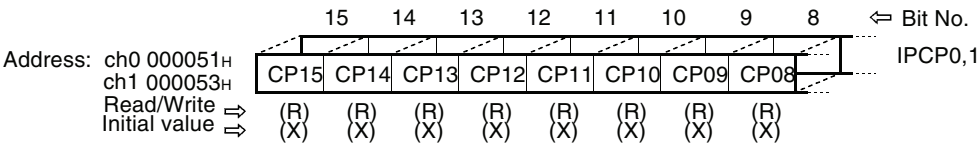
Higher bits of the output compare control status register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: ch1 000063H ch3 000065H	—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1,3
Read/Write ⇒	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	

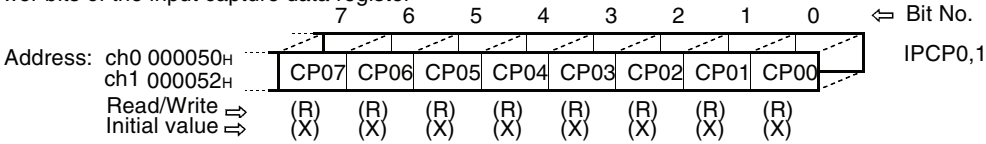
Lower bits of the output compare control status register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: ch0 000062H ch2 000064H	ICP1	ICP0	ICE1	ICE0	—	—	CST1	CST0	OCS0,2
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(-)	(-)	(0)	(0)	

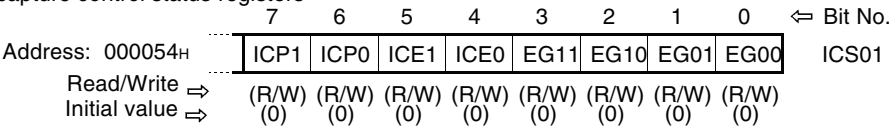
Higher bits of the input capture data register



Lower bits of the input capture data register



Input capture control status registers



## 12.4 16-Bit Free Run Timer

The 16-bit free run timer consists of the following two components.

- Timer Counter Data Register(TCDT)
- Timer Counter Control Status Register (TCCS)

### ■ 16-Bit Free Run Timer

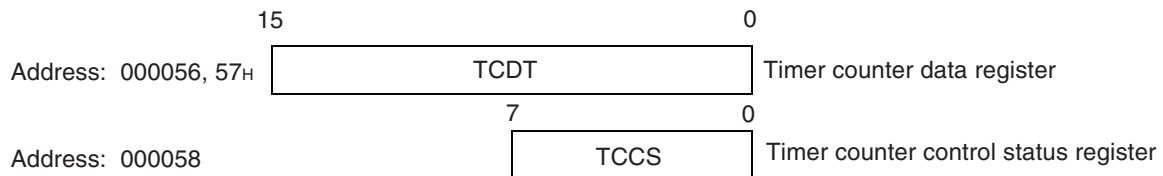
The count value of the 16-bit free run timer is used as the basic time (base timer) for the output compare and input capture.

- The count clock can be selected from the four types.
- A counter overflow interrupt can be generated.
- Depending on the mode setting, the counter can be initialized when the value matches the value of the compare register 0 of the output compare.

### ■ 16-Bit Free Run Timer Registers

The configuration of the 16-bit free run timer registers is shown below.

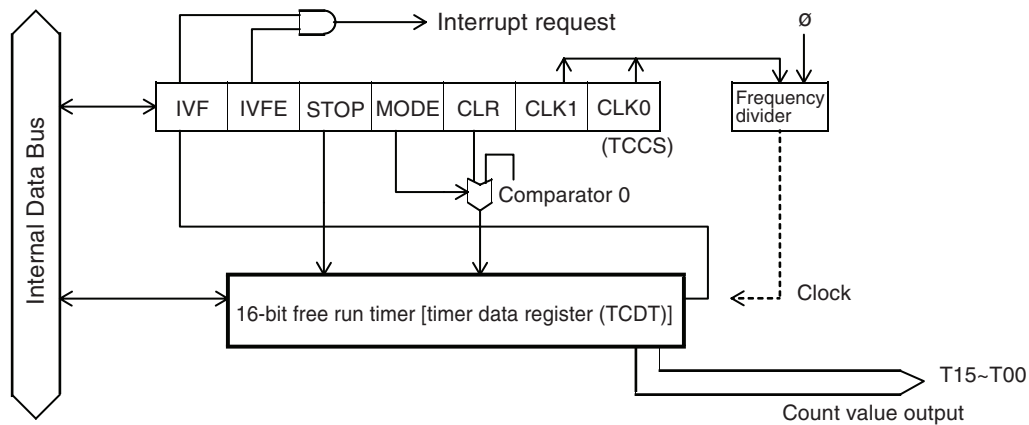
**Figure 12.4-1 16-Bit Free Run Timer Registers**



### ■ Block Diagram of 16-Bit Free Run Timer

Figure 12.4-2 "Block Diagram of 16-Bit Free Run Timer" is a block diagram of the 16-bit free run timer.

**Figure 12.4-2 Block Diagram of 16-Bit Free Run Timer**



### 12.4.1 Timer Counter Data Register (TCDT)

The timer counter data register (TCDT) can read the count value of the 16-bit free run timer.

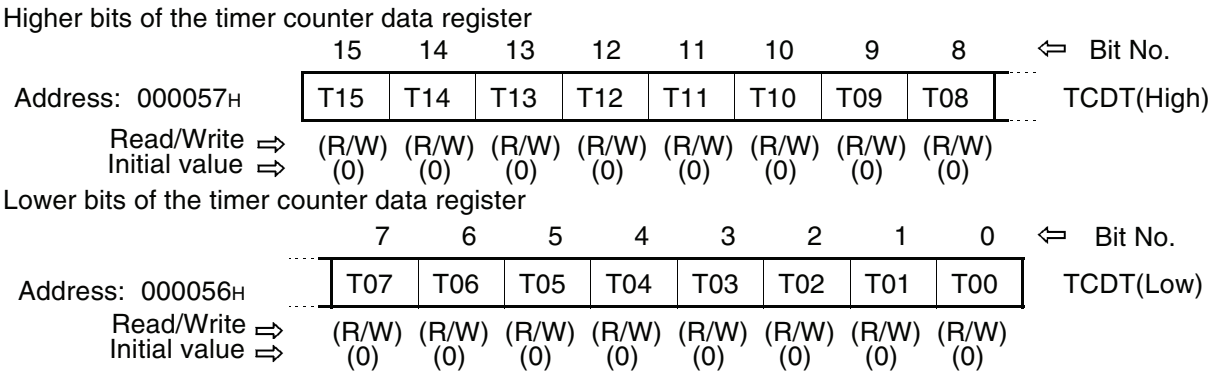
■ Timer Counter Data Register (TCDT)

In the timer counter data register (TCDT), the counter value is set to "0000<sub>H</sub>" at reset. Writing data to this register enables to set the timer value. This must be performed in stop status (STOP=1).

The 16-bit free run timer is initialized when the following factors occur.

- Initialized by a reset.
- Initialized by the clear bit (CLR) of the control status register.
- Initialized when the value of the compare register 0 of output compare and the timer counter value match (the mode must be set).

Figure 12.4-3 Timer Counter Data Register (TCDT)



**Note:**  
To access this register, use word access.

## 12.4.2 Timer Counter Control Status Register (TCCS)

The timer counter control status register (TCCS) controls the timer counter of the 16-bit free run timer.

### ■ Timer Counter Control Status Register (TCCS)

**Figure 12.4-4 Timer Counter Control Status Register (TCCS)**

Timer counter control status register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000058H	Reserved	IVF	IVFE	STOP	MODE	CLR	CLK1	CLK0	TCCS
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)	

#### [Bit 7] Reserved bit

Bit 7 is a reserved bit.

0 must be written in this bit.

#### [Bit 6] IVF

This bit is a 16-bit free run timer interrupt request flag.

When an overflow occurs in the 16-bit free run timer or when the counter is cleared as a result of matching by comparison with compare register 0 according to the mode setting, this bit is set to 1.

If the interrupt request enable bit (bit 5: IVFE) is set, an interrupt occurs.

This bit is cleared when 0 is written. Writing 1 is ineffectual.

Read-modify instructions can read 1.

**Table 12.4-1 IVF (Interrupt Request Flag) Function**

IVF	Function
0	Interrupt request is not present (initial value)
1	Interrupt request is present

#### [Bit 5] IVFE

This bit is the 16-bit free run timer interrupt enable bit. When the interrupt flag (bit 5: IVF) is set to 1 when the bit is 1, an interrupt occurs.

**Table 12.4-2 IVFE (Interrupt Enable Bit) Function**

IVFE	Function
0	Interrupt disabled (initial value)
1	Interrupt enabled

**[Bit 4] STOP**

This bit is for stopping the 16-bit free run timer from counting.

When 1 is written, the timer stops counting. When 0 is written, the timer starts counting.

**Table 12.4-3 STOP (Count Stop Bit) Function**

STOP	Function
0	Counting enabled (active) (initial value)
1	Counting disabled (stop)

**Note:**

When the 16-bit free run timer stops counting, the output compare also stops operation.

**[Bit 3] MODE**

This bit is for setting the initialization condition of the 16-bit free run timer.

When this bit is set to 0, the counter value can be initialized by a reset and clear bit (bit 2: CLR).

When this bit is set to 1, the counter value can be initialized by a reset, by a clear bit (bit 2: CLR), and by matching with the value of compare register 0 of output compare.

**Table 12.4-4 MODE (Initialization Condition Setting Bit) Function**

MODE	Function
0	Initialized by reset and clear bit (initial value)
1	Initialized by reset, clear bit, compare register 0

**Note:**

The counter value is initialized at the point at which the count value changes.

**[Bit 2] CLR**

This bit is for initializing the active 16-bit free run timer to 0000.

When 1 is written, the counter value is initialized to 0000. Writing 0 is ineffectual.

The read value is consistently 0. The counter value is initialized at the point at which the count value changes.

**Table 12.4-5 CLR (Initializing Bit) Function**

CLR	Function
0	Ineffectual (initial value)
1	Counter value is initialized to 0000

**Note:**

To initialize when the timer is stopped, write 0000 in the data register.

**[Bits 1 and 0] CLK1, CLK0**

These bits are for selecting the count clock of the 16-bit free run timer. The clock changes immediately after the bits are written. Therefore, the clock should be changed when the output compare and input capture are stopped.

**Table 12.4-6 CLK1, CLK0 (Count Clock Selection Bit)**

CLK1	CLK0	Count clock	$\phi=16\text{MHz}$	$\phi=8\text{MHz}$	$\phi=4\text{MHz}$	$\phi=1\text{MHz}$
0	0	$\phi/4$	0.25 $\mu\text{s}$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	4 $\mu\text{s}$
0	1	$\phi/16$	1 $\mu\text{s}$	2 $\mu\text{s}$	4 $\mu\text{s}$	16 $\mu\text{s}$
1	0	$\phi/64$	4 $\mu\text{s}$	8 $\mu\text{s}$	16 $\mu\text{s}$	64 $\mu\text{s}$
1	1	$\phi/256$	16 $\mu\text{s}$	32 $\mu\text{s}$	64 $\mu\text{s}$	256 $\mu\text{s}$

**Note:**

$\phi$ : Machine clock



# 12.5 Output Compare

The output compare consists of the following two components.

- Output Compare Register (OCCP0 to OCCP3)
- Output Compare Control Status Register (OCS0 to OCS3)

## ■ Output Compare

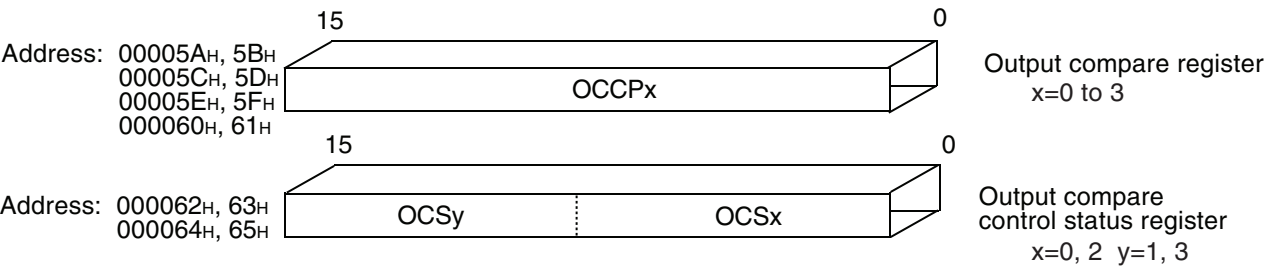
If the value set in the compare register matches the value of the 16-bit free run timer, the pin output level is reverted and an interrupt occurs.

- Two compare registers are provided and can operate independent of each other. Depending on the setting, the two compare registers can be used to control pin output.
- The pin output initial value can be set.
- Matching by comparison can generate an interrupt.

## ■ Output Compare Registers

The configuration of the output compare registers is shown below.

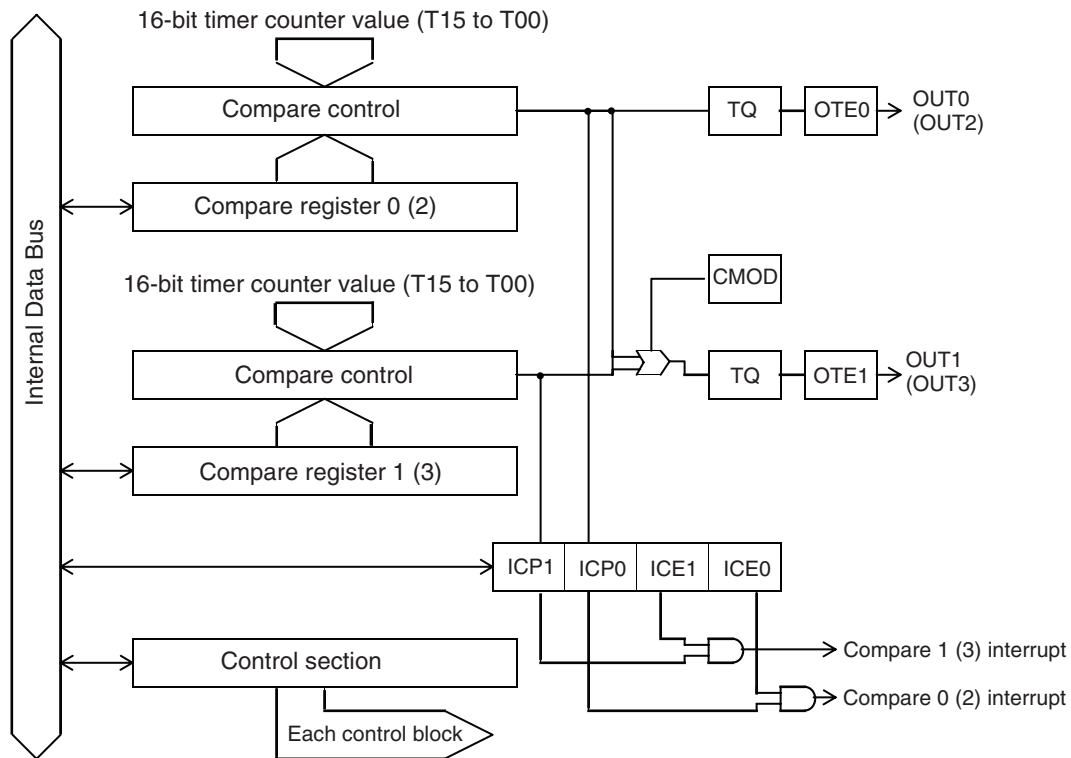
Figure 12.5-1 Output Compare Register Configuration



### ■ Block Diagram of Output Compare

Figure 12.5-2 "Block Diagram of Output Compare" is a block diagram of the output compare.

**Figure 12.5-2 Block Diagram of Output Compare**



### 12.5.1 Output Compare Register (OCCP0 to OCCP3)

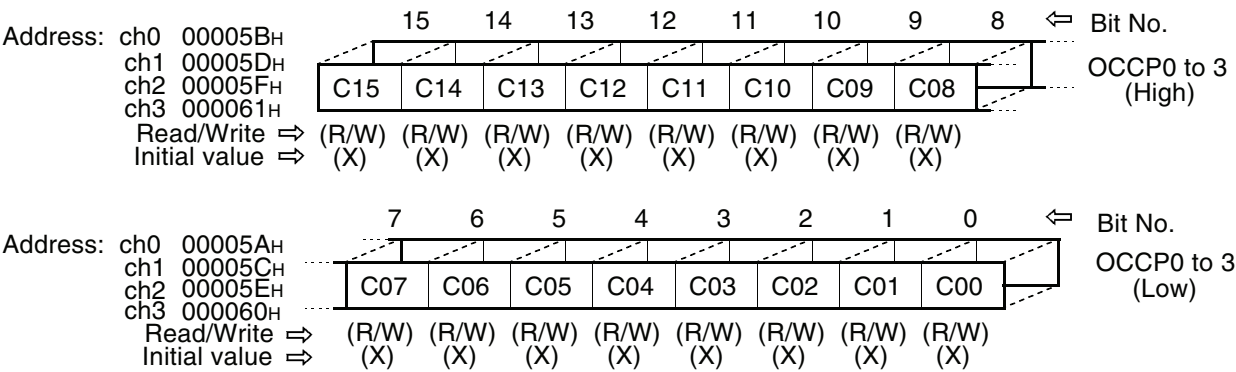
The output compare register (OCCP0 to OCCP3) is a 16-bit compare register for comparison with the 16-bit free run timer.

■ Output Compare Register (OCCP0 to OCCP3)

The initial value of the register value is undefined and must be set before activation is enabled. When this register value and the value of the 16-bit free run timer match, a compare signal is generated and the output compare interrupt flag is set. If output is enabled, the output level corresponding to the compare register is reverted.

Figure 12.5-3 Output Compare Register (OCCP0 to OCCP3)

Higher bits of the output compare register



**Note:**  
To access this register, use word access.

## 12.5.2 Output Compare Control Status Register (OCS0 to OCS3)

The output compare control status register (OCS0 to OCS3) controls the 16-bit free run timer.

### ■ Output Compare Control Status Register (OCS0 to OCS3)

**Figure 12.5-4 Output Compare Control Status Register (OCS0 to OCS3)**

Higher bits of the output compare control status register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: ch1 000063H	—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1,3
ch3 000065H									
Read/Write ⇐	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇐	(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	

Lower bits of the output compare control status register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: ch0 000062H	ICP1	ICP0	ICE1	ICE0	—	—	CST1	CST0	OCS0,2
ch2 000064H									
Read/Write ⇐	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇐	(0)	(0)	(0)	(0)	(-)	(-)	(0)	(0)	

#### [Bits 15, 14, and 13]

Unused bits

#### [Bit 12] CMOD

If pin output is enabled (OTE1 = 1 or OTE0 = 1), the pin output level revert operation mode is switched at matching by comparison.

#### ○ When CMOD is 0 (Initial value)

When CMOD is 0 (initial value), the pin output level corresponding to the compare register is reverted.

- OUT0/2: The level is reverted when the value matches compare register 0.
- OUT1/3: The level is reverted when the value matches compare register 1.

#### ○ When CMOD is 1

When CMOD is 1, the output level for compare register 0 is reverted as when CMOD is 0, but the pin (OUT1) output level corresponding to compare register 1 is reverted when the value of compare register 0 matches the value of compare register 1. If compare registers 0 and 1 have the same value, the operation is the same as that when there is only one compare register.

- OUT0/2: The level is reverted when the value matches compare register 0.
- OUT1/3: The level is reverted when compare registers 0 and 1 match.

**[Bits 11 and 10] OTE1, OTE0**

These are output compare pin output enable bits. The initial value of these bits is 0.

**Table 12.5-1 OTE1, OTE0 (Pin Output Enable Bit) Function**

OTE1, OTE0	Funtion
0	Operates as a general-purpose port (initial value)
1	Becomes an output compare pin output

**Note:**

OTE1: Corresponds to output compare 1/3.

OTE0: Corresponds to output compare 0/2.

**[Bits 9 and 8] OTD1, OTD0**

These bits are used for changing the pin output level when the output compare pin output is enabled. The initial value of compare pin output is 0. To write these bits, compare operation must be stopped. When these bits are read, the output compare pin output value can be read.

**Table 12.5-2 OTD1, OTD0 (Pin Output Level Change Bit) Function**

OTD1, OTD0	Funtion
0	Compare pin output becomes 0 (initial value)
1	Compare pin output becomes 1

**Note:**

OTD1: Corresponds to output compare 1/3.

OTD0: Corresponds to output compare 0/2.

**[Bits 7 and 6] ICP1, ICP0**

These bits are output compare interrupt flags. When the compare register value matches the value of the 16-bit free run timer, the bits are set to 1. When the bits are set when the interrupt request bits (ICE1, ICE0) are enabled, an output compare interrupt occurs. The bits are cleared when 0 is written. Writing 1 is ineffectual. Read-modify instructions can read 1.

**Table 12.5-3 ICP1, ICP0 (Output Compare Interrupt Bit) Function**

ICP1, ICP0	Funtion
0	Did not match by comparison (initial value)
1	Matched by comparison

**Note:**

ICP1: Corresponds to output compare 1/3.

ICP0: Corresponds to output compare 0/2.

**[Bits 5 and 4] ICE1, ICE0**

These are output compare interrupt enable bits. When the interrupt flag (ICP0, ICP1) is set

when these bits are 1, an output compare interrupt occurs.

**Table 12.5-4 ICE1, ICE0 (Output Compare Interrupt Enable Bit) Function**

ICE1, ICE0	Funtion
0	Output compare interrupt disabled (initial value)
1	Output compare interrupt enabled

**Note:**

ICE1: Corresponds to output compare 1/3.

ICE0: Corresponds to output compare 0/2.

**[Bits 3 and 2]**

Unused bits

**[Bits 1 and 0] CST1, CST0**

These bits enable a matching operation with the 16-bit free run timer.

The compare register value must be set before compare operation is enabled.

**Table 12.5-5 CST1, CST0 (Matching Operation with the 16-bit Free Run Timer Bit) Function**

CST1, CST0	Funtion
0	Compare operation disabled (initial value)
1	Compare operation enabled

**Note:**

CST1: Corresponds to output compare 1/3.

CST0: Corresponds to output compare 0/2.

When the 16-bit free run timer stops, the compare operation also stops because the output compare is synchronized with the clock of the 16-bit free run timer.

# 12.6 Input Capture

The input capture unit contains the following two registers:

- Input capture data register (IPCP0, IPCP1)
- Input capture control status register (ICS01)

■ Input Capture

The input capture consists of an input capture data register and control register.

Each input capture has the corresponding external input pin.

- An external input valid edge can be selected from the three types.

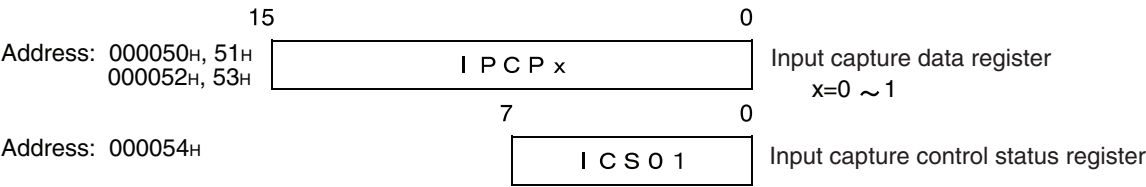
Rising edge (↑) / Falling edge (↓) / Both edges (↑ ↓)

- When an external input valid edge is detected, an interrupt occurs.

■ Entire Input Capture Registers

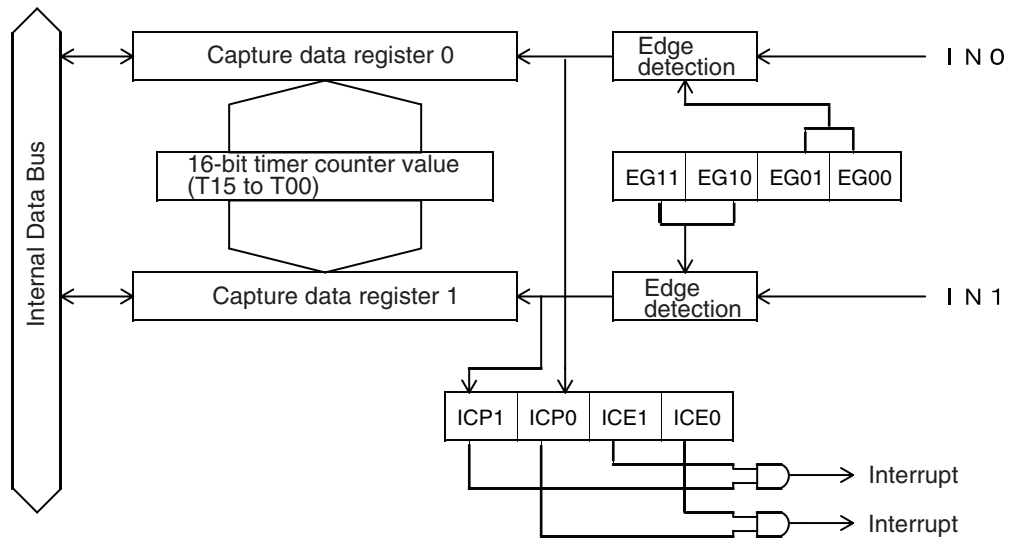
The configuration of the input capture registers is shown below.

Figure 12.6-1 Entire Input Capture Register



## ■ Block Diagram of Entire Input Capture

Figure 12.6-2 Block Diagram of Entire Input Capture



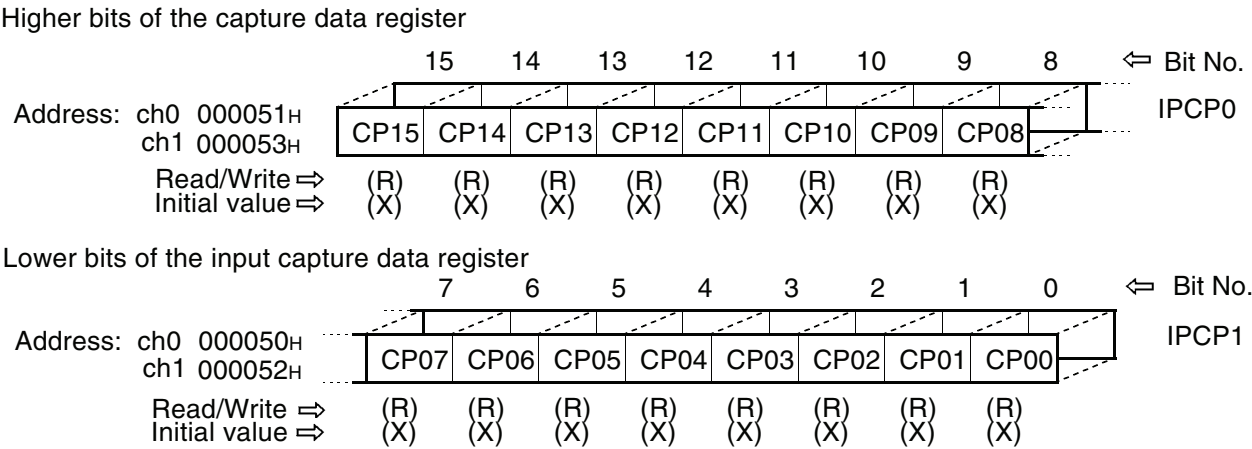


### 12.6.1 Input Capture Data Register (IPCP0, IPCP1)

When a valid edge of the corresponding external pin input waveform is detected, this register retains the value of the 16-bit free run timer.

■ Input Capture Registers (IPCP0, IPCP1)

Figure 12.6-3 Input Capture Data Register (IPCP0, IPCP1)



**Note:**

To access, word access must be used. This register cannot be written.

## 12.6.2 Input Capture Control Status Register (ICS01)

The input capture control status register (ICS01) controls the 16-bit free run timer.

### ■ Input Capture Control Status Register (ICS01)

**Figure 12.6-4 Input Capture Control Status Register (ICS01)**

Input capture control status register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000054H	ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	ICS01
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)	

**Note:**

To access the ICS01 register, byte access should be used.

**[Bits 7 and 6] ICP1, ICP0**

These bits indicate an input capture interrupt flag. When a valid edge of the external input pin is detected, these bits are set to 1. When the interrupt enabled bits (ICE0 and ICE1) are set, detecting a valid edge generates an interrupt.

Writing 0 clears these bits. Writing 1 is ineffectual.

Read-modify write instructions can read 1.

**Table 12.6-1 ICP1, ICP0 (Input Capture Interrupt Flag) Function**

ICP1, ICP0	Function
0	Valid edge is not detected (initial value)
1	Valid edge is detected

**Note:**

ICP1: Corresponds to input capture 1.

ICP0: Corresponds to input capture 0.

**[Bits 5 and 4] ICE1, ICE0**

These bits enable an input capture interrupt. When the interrupt flag (ICP0, ICP1) is set when these bits are 1, an input capture interrupt occurs.

**Table 12.6-2 ICE1, ICE0 (Input Capture Interrupt Enable Bit) Function**

ICE1, ICE0	Function
0	Interrupt disabled (initial value)
1	Interrupt enabled

**Note:**

ICE1: Corresponds to input capture 1.

ICE0: Corresponds to input capture 0.

**[Bits 3, 2, 1, and 0] EG11, EG10, EG01, EG00**

These bits specify the polarity of the external input valid edge and enable an input capture operation.

**Table 12.6-3 EGx1, EGx0 (External Input Valid Edge Polarity Specifying Bit) Function**

EG11 EG01	EG10 EG00	Polarity of edge detection	
0	0	Edges not detected (stopped status) (Initial value)	
0	1	Rising edge detected	↑
1	0	Falling edge detected	↓
1	1	Both edges detected	↑ ↓

**Note:**

EG01, EG00: Correspond to input capture 0.

EG11, EG10: Correspond to input capture 1.

## 12.7 Operation of the 16-Bit Free Run Timer

The 16-bit free run timer starts counting from the counter value 0000 after the reset is released. This counter value becomes the reference time for 16-bit output compare and 16-bit input capture.

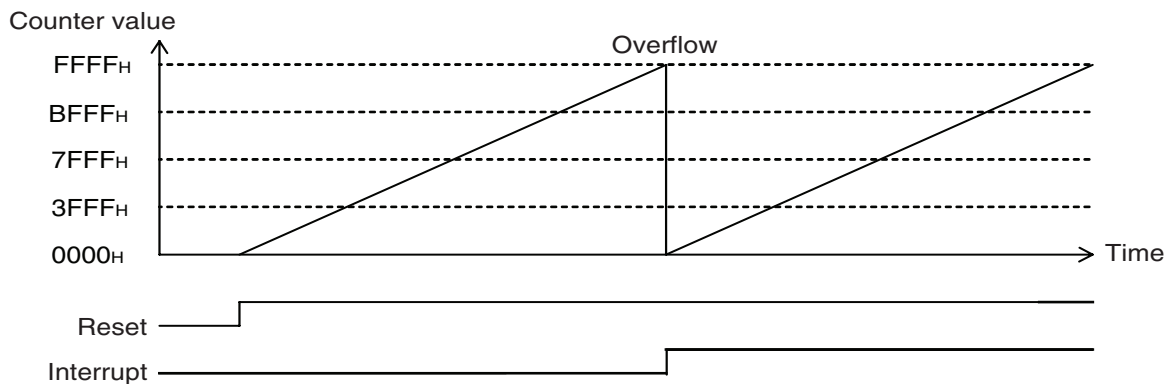
### ■ Operation of the 16-Bit Free Run Timer

The counter value is cleared when the following five conditions are met.

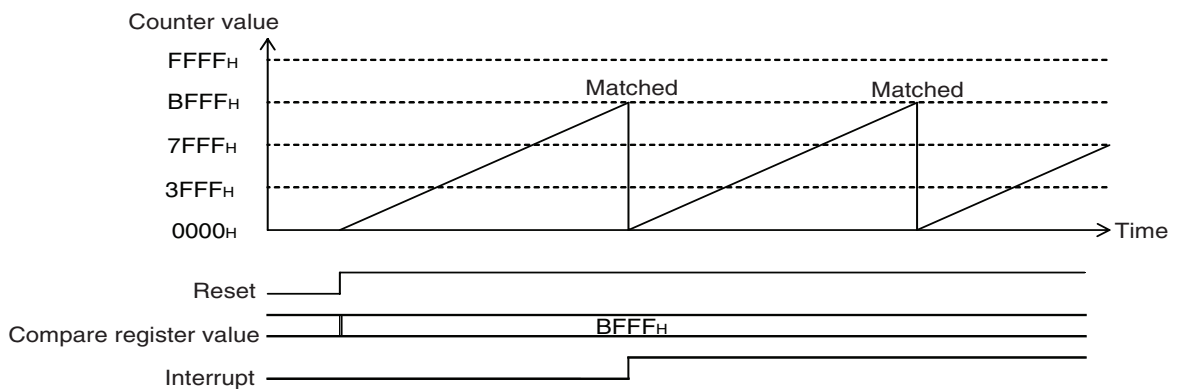
- When an overflow occurs.
- When the value of output compare register 0 matches by comparison (the mode must be set).
- When 1 is written in the CLR bit of the TCCS register during operation.
- When 0000 is written in the TCDC register during stop.
- When reset.

An interrupt can be generated when an overflow occurs and when the value of compare register 0 matches by comparison and the counter is cleared. (The mode must be set for compare match interrupt.)

**Figure 12.7-1 Counter Cleared by Overflow**



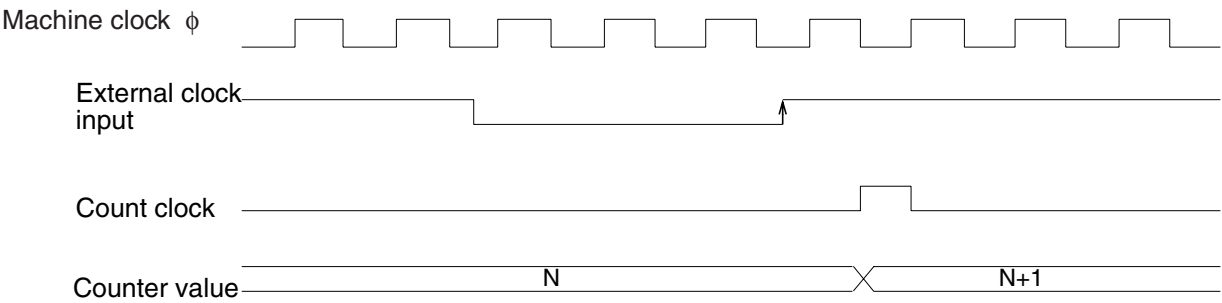
**Figure 12.7-2 Counter Cleared When the Value of Output Compare Register 0 Matches by Comparison**



■ Timing for 16-bit Free-running Timer

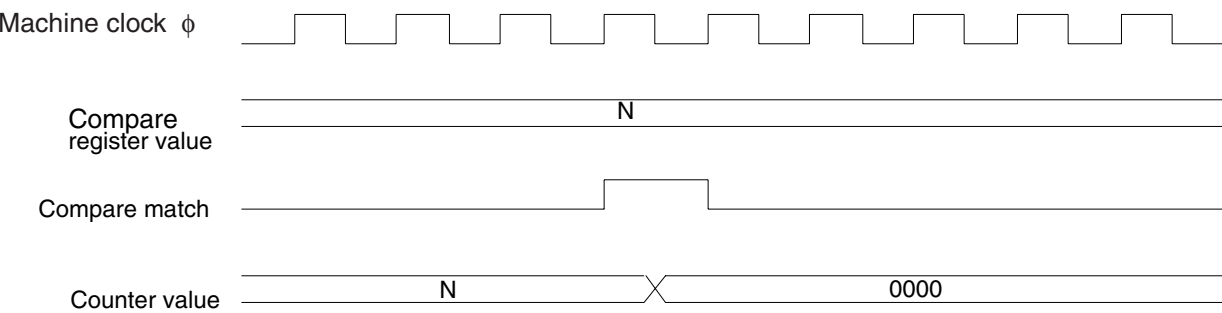
The 16-bit free-running timer is incremented based on the input clock (internal or external clock). When an external clock is selected, the 16-bit free-running timer is incremented at the rising edge.

Figure 12.7-3 16-bit free-running timer count timing



The counter can be cleared by a reset, software clear, or match with compare register 0. For a reset or software clear, the counter is cleared. For a match with compare register 0, the counter is cleared synchronously with the count timing.

Figure 12.7-4 16-bit free-running timer clear timing (match with the compare register 0)



## 12.8 Operation of the 16-Bit Output Compare

The 16-bit output compare compares the value set in the compare register with the value of the 16-bit free run timer. If the values match, it sets the interrupt request flag and reverts the output level.

### ■ Operation of the 16-Bit Output Compare

Figure 12.8-1 Example of Output Waveform When Compare Registers 0 and 1 Are Used (Initial Value of Output Is Zero)

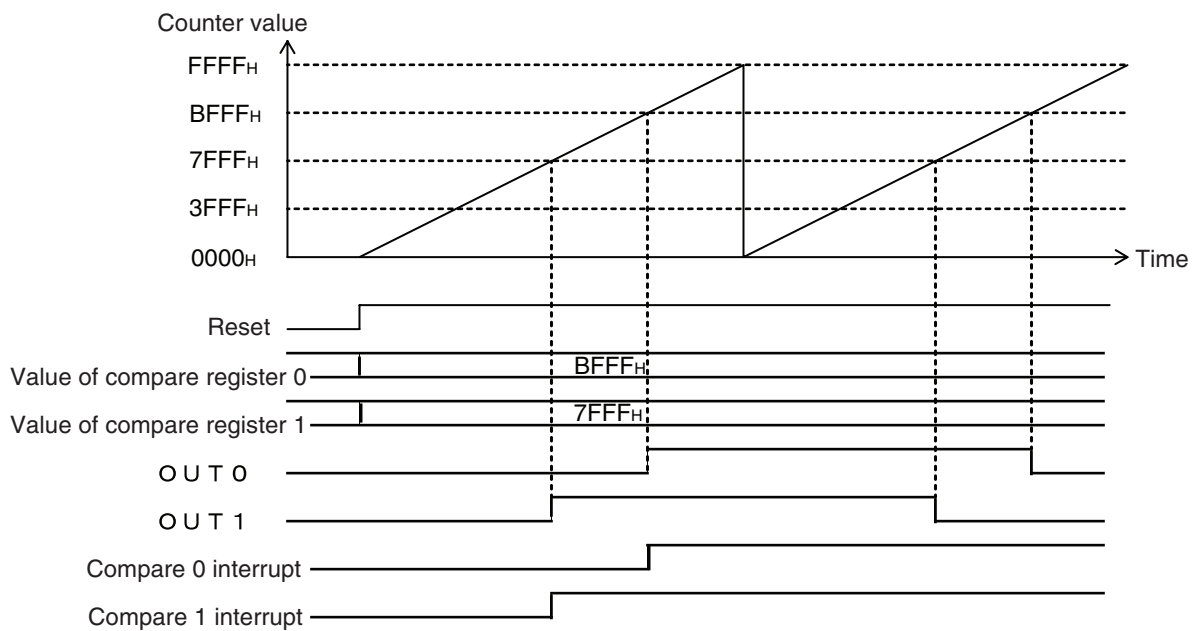
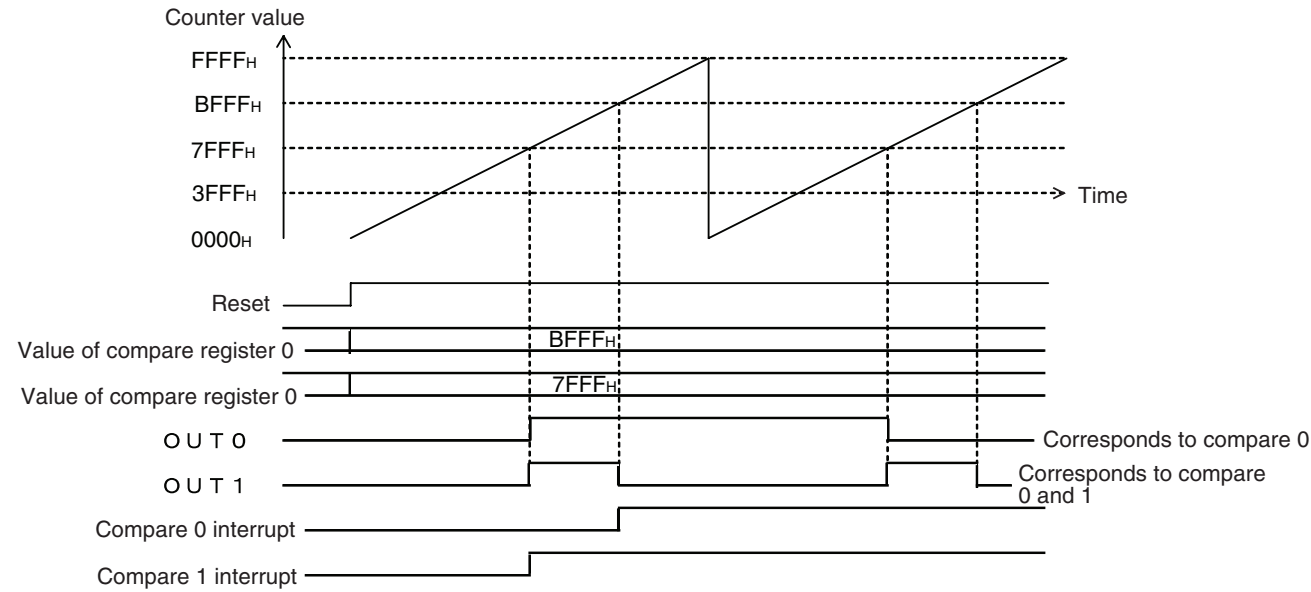


Figure 12.8-2 Example of Output Waveform from Two Pairs of Compare Registers (Initial Value of Output Is Zero)



■ Timing for 16-bit Output Compare

The output compare unit can generate a compare match signal when the free run timer value is consistent with the setting value of the compare register. It can then invert the output value and generate an interrupt.

The output invert timing when the compare match is executed synchronously with the count timing of the counter.

The counter value when the compare register is rewritten is not compared.

Figure 12.8-3 Compare Operation When the Compare Register is Rewritten

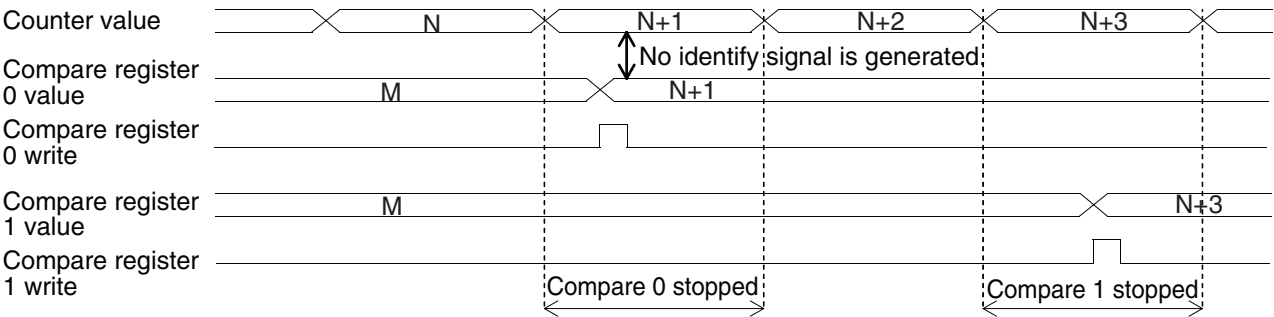


Figure 12.8-4 Output Compare Interrupt Timing

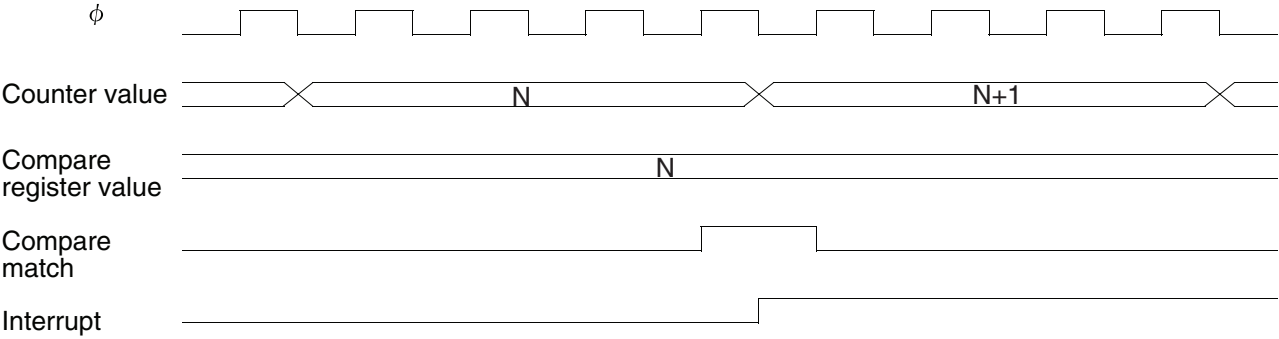
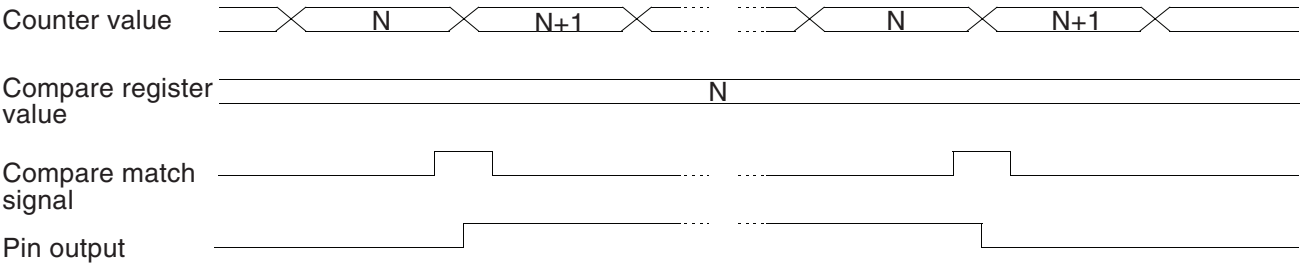


Figure 12.8-5 Output Pin Change Timing of Output Compare



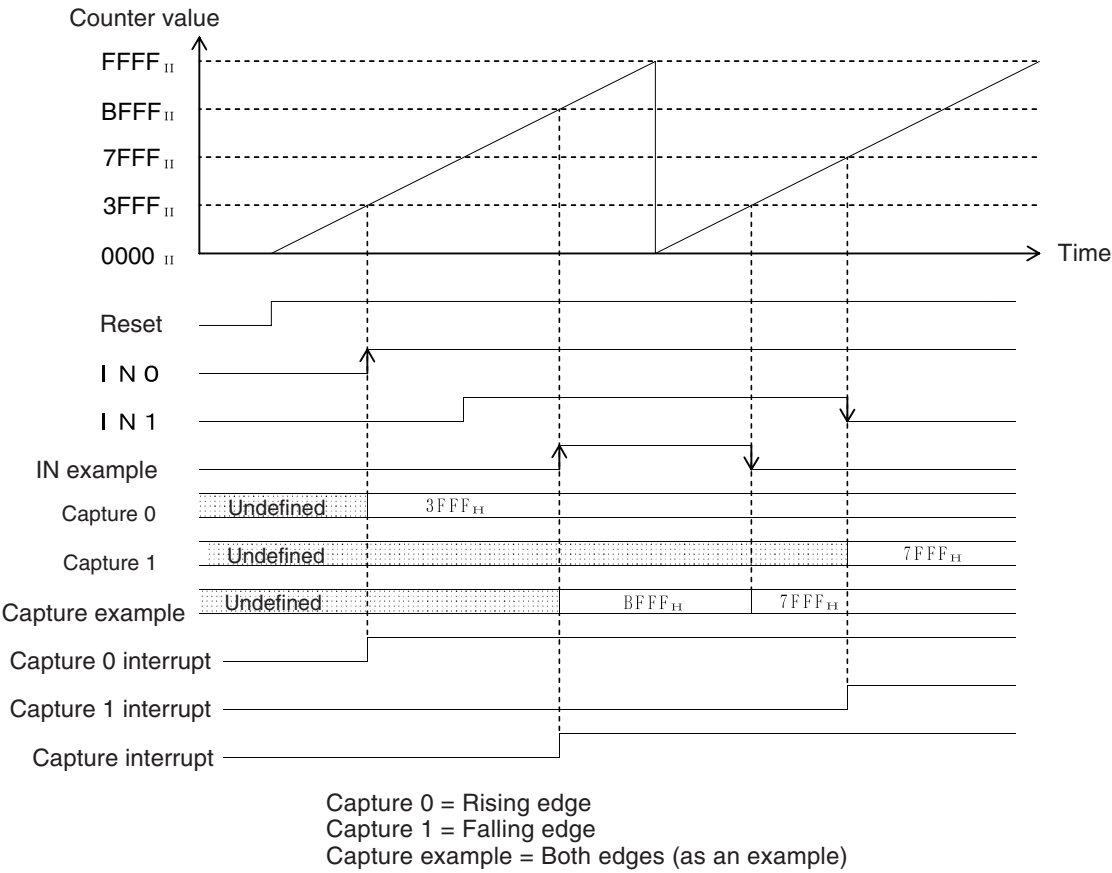


# 12.9 Operation of the 16-Bit Input Capture

The 16-bit input capture detects the set valid edge, captures the value of the 16-bit free run timer into the capture register, and generates an interrupt.

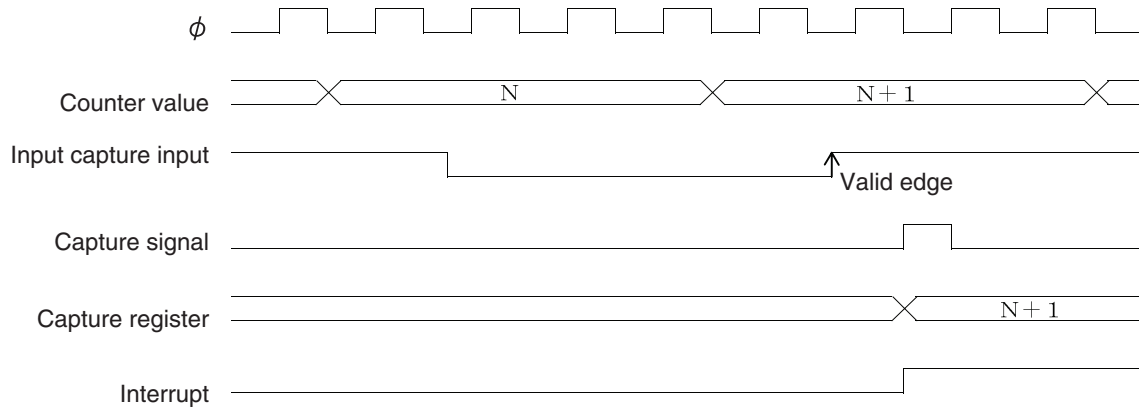
■ Operation of the 16-Bit Input Capture

Figure 12.9-1 Data Capture Timing of the Input Capture



## ■ Input Capture Input Timing

Figure 12.9-2 Input Signal Capture Timing





## CHAPTER 13 8/16-BIT PPG

---

**This chapter describes the functions and operations of the 8/16-bit PPG.**

---

13.1 "Overview of the 8/16-Bit PPG"

13.2 "Block Diagrams of the 8/16-Bit PPG"

13.3 "Registers of the 8/16-Bit PPG"

13.4 "Operation of the 8/16-Bit PPG"

## 13.1 Overview of the 8/16-Bit PPG

---

The 8/16-bit PPG is an 8-bit reload timer module that performs PPG output by pulse output control according to timer operation.

---

### ■ Overview of the 8/16-Bit PPG

The 8/16-bit PPG has two 8-bit down counters, four 8-bit reload registers, one 16-bit control register, two external pulse output pins, and two interrupt outputs as hardware and supports the following functions:

- **Two independent 8-bit output channel operating mode:**

Enables PPG output operation with two independent channels.

- **16-bit PPG output operating mode:**

Enables PPG output operation with one 16-bit channel.

- **8+8-bit PPG output operating mode:**

Uses the channel 0 output as the channel 1 clock input to enable 8-bit PPG output operation in any cycle.

- **PPG output operation:**

Outputs pulse waves with any duty cycle in any cycle and can be used as a D/A converter with an external circuit.

## 13.2 Block Diagrams of the 8/16-Bit PPG

**Figure 13.2-1 "Block Diagram of the 8/16-Bit PPG (Channel 0)" and Figure 13.2-2 "Block Diagram of the 8/16-Bit PPG (Channel 1)" show a block diagram of the 8/16-bit PPG.**

## ■ Block Diagrams of the 8/16-Bit PPG

**Figure 13.2-1 Block Diagram of the 8/16-Bit PPG (Channel 0)**

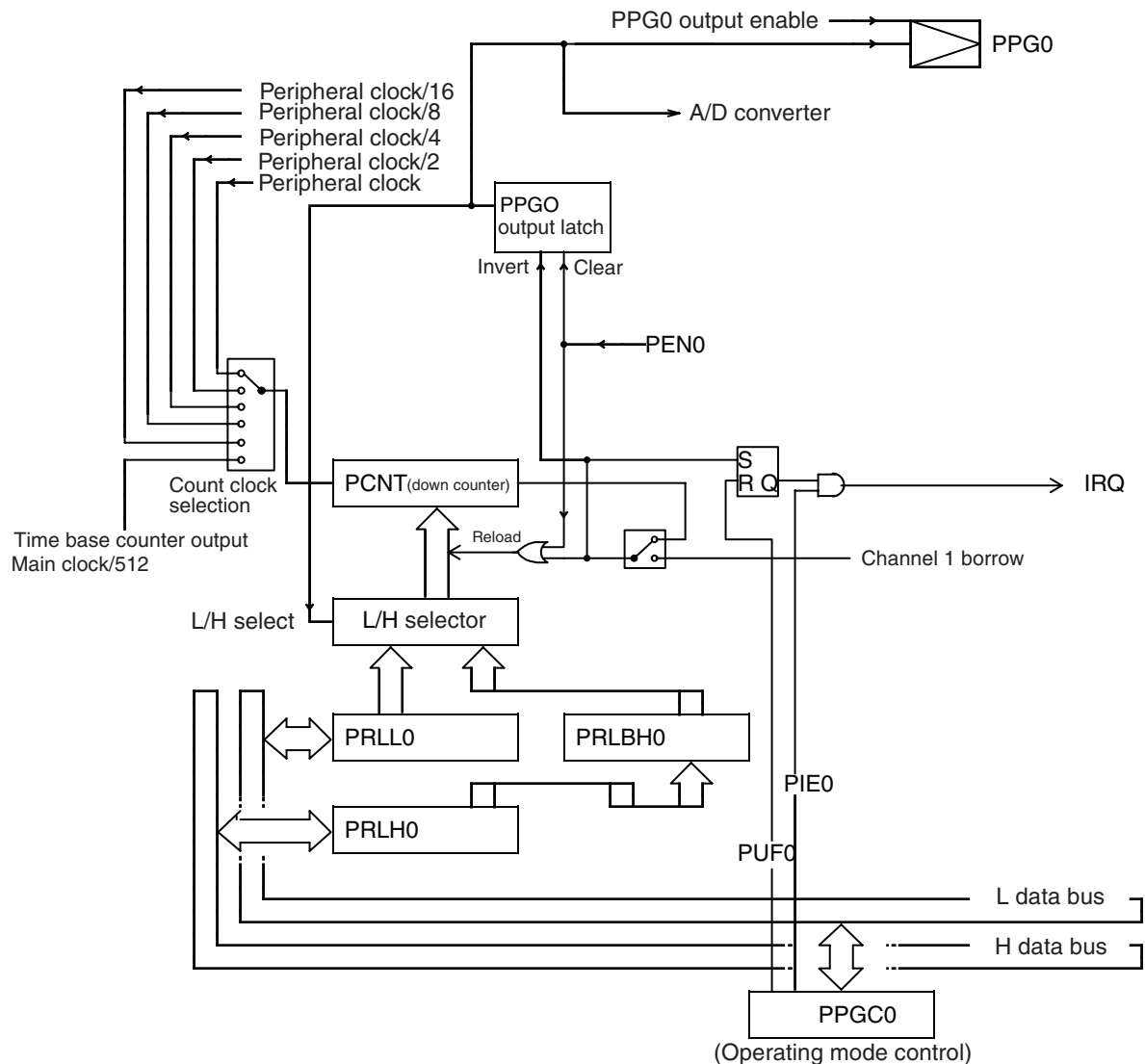
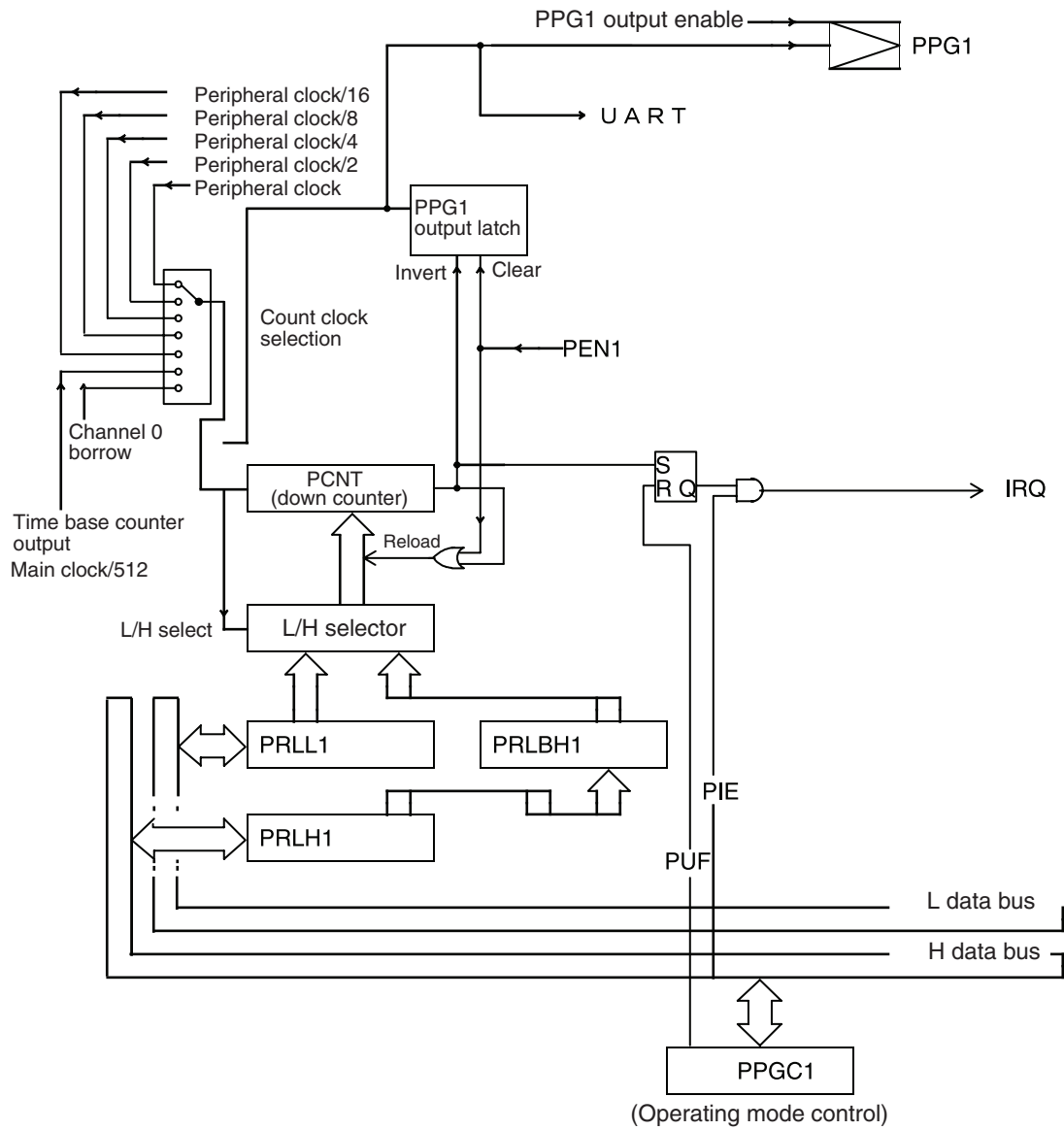


Figure 13.2-2 Block Diagram of the 8/16-Bit PPG (Channel 1)



## 13.3 Registers of the 8/16-Bit PPG

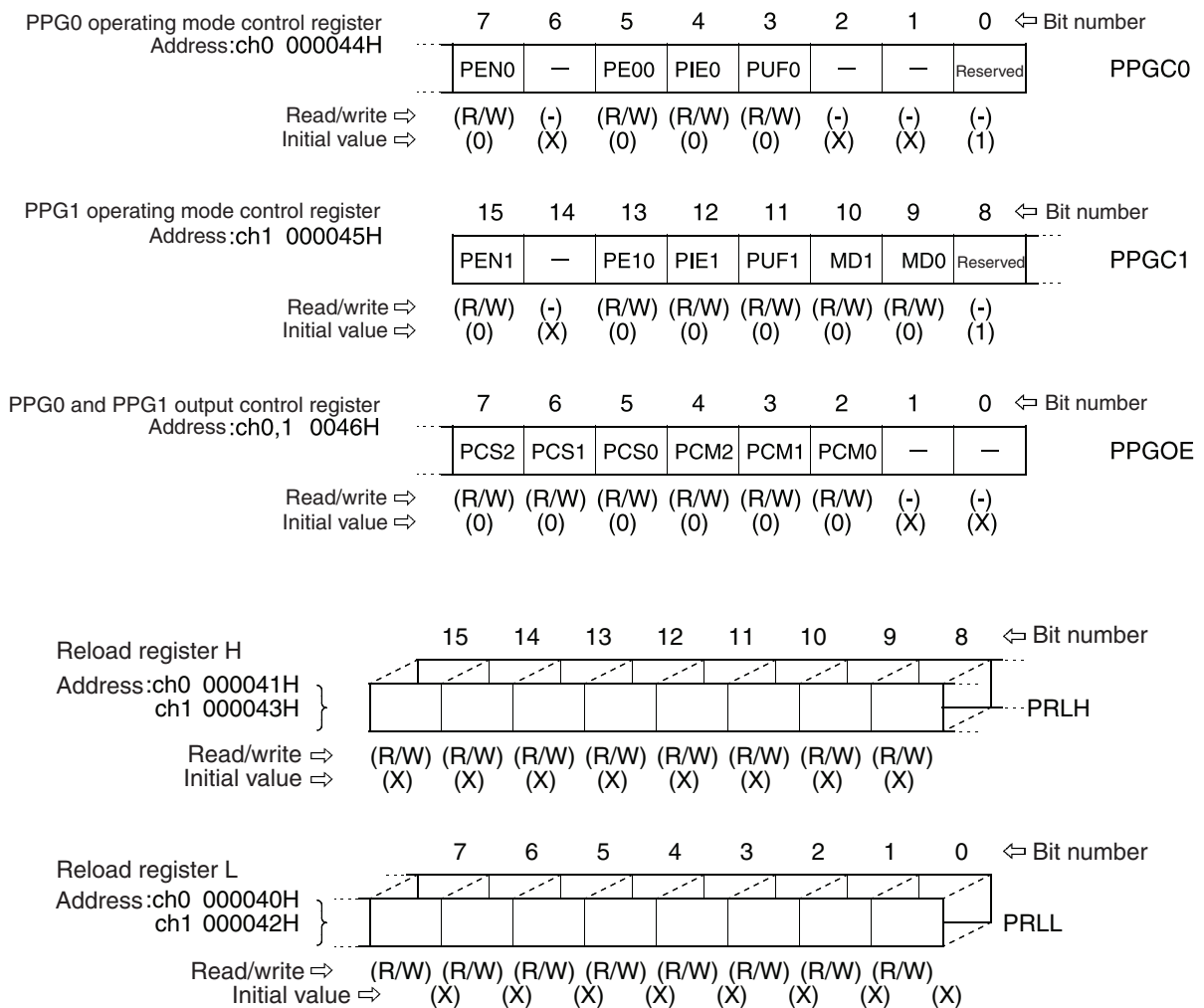
The 8/16-bit PPG register has three types as follows:

- PPG operating mode control register
- PPG output control register
- Reload register

### ■ Registers of the 8/16-Bit PPG

The register configuration of the 8/16-bit PPG is shown below.

**Figure 13.3-1 8/16-Bit PPG Registers**





### 13.3.1 PPG0 Operating Mode Control Register (PPGC0)

The PPG0 operating mode control register (PPGC0) selects the operating mode, controls pin output, selects count clock, and controls a trigger for the 8/16-bit PPG.

■ PPG0 Operating Mode Control Register (PPGC0)

The PPGC0 bit configuration is shown below.

Figure 13.3-2 PPG0 Operating Mode Control Register (PPGC0)

PPG0 operating mode control register Address:ch0 000044H	7	6	5	4	3	2	1	0	⇐ Bit number
	PEN0	—	PE00	PIE0	PUF0	—	—	Reserved	PPGC0
Read/write ⇒	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(-)	(-)	(-)	
Initial value ⇒	(0)	(-)	(0)	(0)	(0)	(-)	(-)	(1)	

[Bit 7] PPG enable bit (PEN0)

This bit selects the start of PPG operation and operating mode as follows.

Writing 1 in this bit starts PPG counting.

This bit is initialized to "0" by a reset and is a read/write bit.

Table 13.3-1 PEN0 (Operating Enable Bit) Function

PEN0	Function
0	Operation stopped ("L" level output is retained.) [Initial value]
1	PPG operation enabled

[Bit 5] PPG00 output enable bit (PE00)

This bit controls pulse output external pin PPG00 as follows.

This bit is initialized to "0" by a reset. It is a read/write bit.

Table 13.3-2 PE00 (PPG0 Pin Output Enable Bit) Function

PE00	Function
0	General-purpose port pin (pulse output disabled) [Initial value]
1	PPG0=Pulse output pin (pulse output enabled)

[Bit 4] PPG interrupt enable bit (PIE0)

This bit controls PPG interrupts as follows.

When this bit is "1", setting PUF0 to "1" generates an interrupt request. When this bit is "0", no interrupt request is generated.

This bit is initialized to "0" by a reset and is a read/write bit.

**Table 13.3-3 PIE0 (PPG Interrupt Enable Bit) Function**

PIE0	Function
0	Disables interrupts. [Initial value]
1	Enables interrupts.

**Note:**

The same interrupt vector number as for 16-bit reload timer channel 0 is assigned. When EI<sup>2</sup>OS is to be used for 16-bit reload timer channel 0, set PIE0 to "0".

**[Bit 3] PPG underflow bit (PUF0)**

The PPG underflow bit is controlled as follows.

In the two 8-bit PPG channel mode or 8-bit prescaler and 8-bit PPG mode, this bit is set to "1" by an underflow caused when the channel 0 counter value changes from 00<sub>H</sub> to FF<sub>H</sub>. In the one 16-bit PPG channel mode, this bit is set to "1" by an underflow caused when the channel 1 or 0 counter value changes from 0000<sub>H</sub> to FFFF<sub>H</sub>. Writing "0" sets this bit to "0". Writing "1" in this bit has no meaning. When this bit is read by a read-modify-write instruction, the read value is "1".

This bit is initialized to "0" by a reset and is a read/write bit.

**Table 13.3-4 PUF0 (PPG Counter Underflow Bit) Function**

PUF0	Function
0	A PPG counter underflow is not detected.
1	A PPG counter underflow was detected.

**[Bit 0]**

Reserved bit. Always set this bit to 1 when setting the PPGC0.

### 13.3.2 PPG1 Operating Mode Control Register (PPGC1)

The PPG1 operating mode control register (PPGC1) selects the operating mode, controls pin output, selects count clock, and controls the trigger for the 8/16-bit PPG.

■ PPG1 Operating Mode Control Register (PPGC1)

Figure 13.3-3 PPG1 Operating Mode Control Register (PPGC1)

PPG1 operating mode control register Address:ch1 000045H	15	14	13	12	11	10	9	8	⇐ Bit number
	PEN1	—	PE10	PIE1	PUF1	MD1	MD0	Reserved	PPGC1
Read/write ⇒	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	
Initial value ⇒	(0)	(X)	(0)	(0)	(0)	(0)	(0)	(1)	

**[Bit 15] PPG enable bit (PEN1)**

This bit selects the start of PPG operation and operating mode as follows.

Writing 1 in this bit makes the PWM start counting.

This bit is initialized to "0" by a reset and is a read/write bit.

Table 13.3-5 Operating Enable Bit (PEN1) Function

PEN1	Function
0	Operation stopped ("L" level output is retained.) [Initial value]
1	PPG operation enabled

**[Bit 13] PPG10 output enable bit (PE10)**

This bit controls pulse output external pin PPG10 as follows.

This bit is initialized to "0" by a reset and is a read/write bit.

Table 13.3-6 PE10 (PPG10 Pin Output Enable Bit) Function

PE10	Function
0	General-purpose port pin (pulse output disabled) [Initial value]
1	PPG1=Pulse output pin (pulse output enabled)

**[Bit 12] PPG interrupt enable bit (PIE1)**

This bit controls PPG interrupts as follows.

When this bit is "1", setting PUF1 to "1" generates an interrupt request. When this bit is "0", no interrupt request is generated.

This bit is initialized to "0" by a reset and is a read/write bit.

**Table 13.3-7 PIE1 (PPG Interrupt Enable Bit) Function**

PIE1	Function
0	Disables interrupts. [Initial value]
1	Enables interrupts.

**Note:**

The same interrupt vector number as for 16-bit reload timer channel 1 is assigned. When EI<sup>2</sup>OS is to be used for 16-bit reload timer channel 1, set PIE1 to "0".

**[Bit 11] PPG underflow bit (PUF1)**

The PPG underflow bit is controlled as listed below.

In the two 8-bit PPG channel mode or 8-bit prescaler and 8-bit PPG mode, this bit is set to "1" by an underflow caused when the channel 0 counter value changes from 00H to FFH. In the one 16-bit PPG channel mode, this bit is set to "1" by an underflow caused when the channel 1 or 0 counter value changes from 0000H to FFFFH. Writing "0" sets this bit to "0". Writing "1" in this bit has no meaning. When this bit is read by a read-modify-write instruction, the read value is "1".

This bit is initialized to "0" by a reset and is a read/write bit.

**Table 13.3-8 PUF1 (PPG Counter Underflow Bit) Function**

PUF1	Function
0	A PPG counter underflow is not detected. [Initial value]
1	A PPG counter underflow was detected.

**[Bits 10 and 9] PPG count mode bits (MD2 and MD1)**

These bits select the PPG timer operating mode as follows.

These bits are initialized to "00" by a reset and are read/write bits.

**Table 13.3-9 MD2, MD1 (Operating Mode Selection Bit) Function**

MD1	MD0	Operating mode
0	0	Two independent 8-bit PPG channel mode [Initial value]
0	1	8-bit prescaler and one 8-bit PPG channel mode
1	0	Reserved (setting prohibited)
1	1	One 16-bit PPG channel mode

**Note:**

- Do not set these bits to "10".
- When setting these bits to "01", do not set the PEN0 bit in the PPGC0 and the PEN1 bit in the PPGC1 to "01". Setting the PEN0 and PEN1 bits to "11" or "00" simultaneously is recommended.
- To set these bits to "11", rewrite the PPGC0 and PPGC1 by word transfer and set the PEN0 and PEN1 bits to "11" or "00" simultaneously.

**[Bit 8]**

Reserved bit. Always set this bit to 1.

### 13.3.3 PPG0 and PPG1 Output Pin Control Register (PPG0E)

PPG0 and PPG1 output pin control register (PPG0E) controls pin output for the 8/16-bit PPG.

#### ■ PPG0 and PPG1 Output Pin Control Register (PPG0E)

**Figure 13.3-4 PPG0 and PPG1 Output Pin Control Register (PPG0E)**

PPG0 and PPG1 output control register	7	6	5	4	3	2	1	0	↔ Bit number
Address : ch0,1 0046H	PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	—	—	PPG0E
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(X)	(X)	

#### [Bits 7 to 5] PPG count select bits (PCS2 to PCS0)

These bits select the operating clock of the channel 1 down counter as listed in Table 13.3-10 "PCS2 to PCS0 (Count Clock Selection Bit) Function".

These bits are initialized to "000" by a reset and are read/write bits.

**Table 13.3-10 PCS2 to PCS0 (Count Clock Selection Bit) Function**

PCS2	PCS1	PCS0	Operating mode
0	0	0	Peripheral clock (62.5 ns when the machine clock frequency is 16 MHz)
0	0	1	Peripheral clock/2 (125 ns when the machine clock frequency is 16 MHz)
0	1	0	Peripheral clock/4 (250 ns when the machine clock frequency is 16 MHz)
0	1	1	Peripheral clock/8 (500 ns when the machine clock frequency is 16 MHz)
1	0	0	Peripheral clock/16 (1 μs when the machine clock frequency is 16 MHz)
1	1	1	Clock input from the time base timer (128 μs when the OSC oscillation frequency is 4 MHz)

#### Note:

In the 8-bit prescaler and 8-bit PPG mode and 16-bit PPG mode, the channel 1 PPG operates using the count clock received from channel 0. In these modes, the PSC1 bit specification is invalidated.

#### [Bits 4 to 2] PPG count mode bits (PCM2 to PCM0)

These bits select the operating clock of the channel 0 down counter as listed in Table 13.3-11 "PCM2 to PCM0 (Count Clock Selection Bit) Function".

These bits are initialized to "000" by a reset and are read/write bits.

**Table 13.3-11 PCM2 to PCM0 (Count Clock Selection Bit) Function**

PCS2	PCS1	PCS0	Operating mode
0	0	0	Peripheral clock (62.5 ns when the machine clock frequency is 16 MHz)
0	0	1	Peripheral clock/2 (125 ns when the machine clock frequency is 16 MHz)
0	1	0	Peripheral clock/4 (250 ns when the machine clock frequency is 16 MHz)
0	1	1	Peripheral clock/8 (500 ns when the machine clock frequency is 16 MHz)
1	0	0	Peripheral clock/16 (1 $\mu$ s when the machine clock frequency is 16 MHz)
1	1	1	Clock input from the time base timer (128 $\mu$ s when the OSC oscillation frequency is 4 MHz)

**[Bit 1] PE11 (Ppg output Enable 11)**

This bit, which is a read/write bit that is initialized to "0" by a reset, controls the pulse output external pin PG11, as described in Table 13.3-12 "PE11 (PPG11 Pin Output Enable Bit)".

**Table 13.3-12 PE11 (PPG11 Pin Output Enable Bit)**

PE11	Function
0	General-purpose port pin (Pulse output disabled)
1	PG11=Pulse output pin (Pulse output enabled)

**[Bit 0] PE01 (Ppg output Enable 01)**

This bit, which is a read/write bit that is initialized to 0 by a reset, controls the pulse output external pin PG01, as described in Table 13.3-13 "PE01 (PPG01 Pin Output Enable Bit)".

**Table 13.3-13 PE01 (PPG01 Pin Output Enable Bit)**

PE01	Function
0	General-purpose port pin (Pulse output disabled) [Initial value]
1	PG01=Pulse output pin (Pulse output enabled)

### 13.3.4 Reload Registers (PRLH and PRLH)

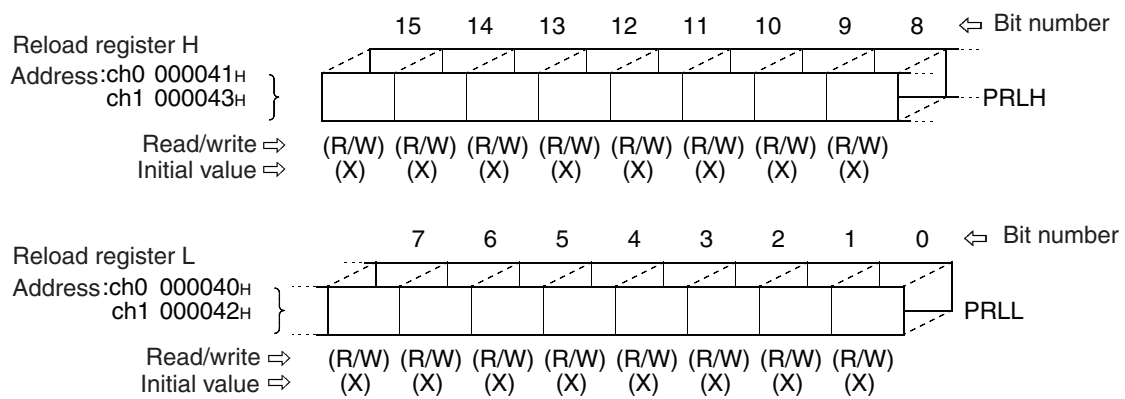
The reload registers (PRLH and PRLH) retain reload values to down counter PCNT. The registers are read/write registers and have the following functions:

- **PRLH:** Retains the H reload value.
- **PRLH:** Retains the L reload value.

#### ■ Reload Registers (PRLH and PRLH)

The PRLH and PRLH bit configurations are shown below.

**Figure 13.3-5 Reload Registers (PRLH and PRLH)**



#### Note:

In the 8-bit prescaler and 8-bit PPG mode, when different values are set in the PRLH and PRLH in channel 0, channel 1 PPG waveforms may be different in each cycle. For this reason, setting the same value in the PRLH and PRLH in channel 0 is recommended.



## 13.4 Operation of the 8/16-Bit PPG

The 8/16-bit PPG has two 8-bit PPG unit channels. The PPG operates in the two independent channel mode. By coupling these units together, the PPG can also operate in the 8-bit prescaler and 8-bit PPG mode and one 16-bit PPG channel mode. The PPG can perform a total of three types of operation.

### ■ Operation of the 8/16-Bit PPG

Each 8-bit PPG unit has two 8-bit reload registers (PRL and PRLH). Values written in H and L registers are alternately reloaded to the 8-bit down counter (PCNT). The down counter counts down for each count clock. When a counter borrow occurs at reloading, the value of the pin output (PPG) is inverted. By this operation, the pin output (PPG) is the pulse output with L and H widths corresponding to the reload register values.

Writing bits in these registers starts or restarts operation.

Table 13.4-1 "Relationship between Reload Operation and Pulse Output" shows the relationship between reload operation and pulse output.

**Table 13.4-1 Relationship between Reload Operation and Pulse Output**

Reload operation	Transition of PG0 and PG1 output pin status
PRLH --> PCNT	PG0x/1x [0 --> 1] Rise
PRL --> PCNT	PG0x/1x [1 --> 0] Fall

When bit 4 (PIE0) in the PPGC0 and bit 12 (PIE1) in the PPGC1 are 1, a borrow from 00 to FF may occur in each counter. (A borrow from 0000 to FFFF may occur in the 16-bit PPG mode.) At this time, an interrupt request is output.

### 13.4.1 8/16-Bit PPG Operating Modes

---

There are three 8/16-bit PPG operating modes:

- Two independent channel mode
  - 8-bit prescaler and an 8-bit PPG mode
  - One 16-bit PPG channel mode.
- 

#### ■ 8/16-Bit PPG Operating Modes

##### ○ Two independent channel mode

In the two independent channel mode, two channels are operated as independent 8-bit PPG units. The PPG0 pin is connected to the channel 0 PPG output. The PPG1 pin is connected to the channel 1 PPG output.

##### ○ 8-bit prescaler and 8-bit PPG mode

In the 8-bit prescaler and 8-bit PPG mode, channel 0 is operated as an 8-bit prescaler and channel 1 count signal is output in synchronization with the channel 0 borrow output. This operation enables 8-bit PPG waveforms in any cycle to be output. The PPG0 pin is connected to the channel 0 prescaler output. The PPG1 pin is connected to the channel 0 PPG output.

##### ○ One 16-bit PPG channel mode

In the one 16-bit PPG channel mode, channel 0 is coupled to channel 1 and both operate as a 16-bit PPG. The PPG0 and PPG1 pins are connected to the 16-bit PPG output.

## 13.4.2 8/16-Bit PPG Output Operation

For the 8/16-bit PPG, setting bit 7 (PEN0) in the PPG0 operating mode control register (PPGC0) to 1 activates channel 0 PPG counting. Setting bit 15 (PEN1) in the PPG1 operating mode control register (PPGC1) to 1 activates channel 1 PPG counting. When the operation starts, setting the PEN1 bit in the PPGC1 register to 0 stops the counting operation, at which time the pulse output level remains at L level.

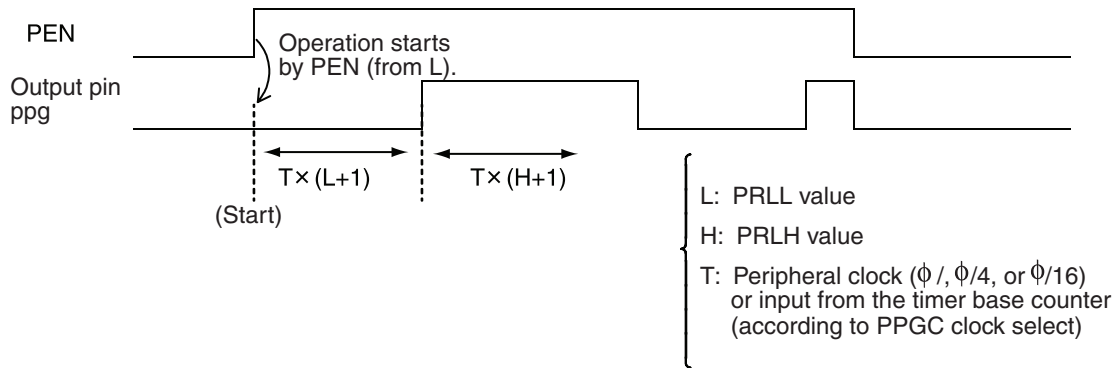
### ■ 8/16-Bit PPG Output Operation

Note the following with respect to the 8/16-bit PPG output operation:

- In the 8-bit prescaler and 8-bit PPG mode, when channel 0 stops, do not place channel 1 in the operating state.
- In the 16-bit PPG mode, simultaneously set bit 7 (PEN0) in the PPGC0 register and bit 15 (PEN1) in the PPGC1 register to control start and stop. PPG output operation is explained below. During a PPG operation, a pulse waveform with any frequency and any duty ratio (ratio of the H level period to the total period of the pulse wave) is output continuously. After starting pulse waveform output, the PPG does not stop until an operation stop is set.

Figure 13.4-1 "Output Waveform of PPG Output Operation" shows an output waveform of PPG output operation.

**Figure 13.4-1 Output Waveform of PPG Output Operation**



### ■ Relationship between Reload Values and Pulse Widths

The value obtained by multiplying the value obtained by adding 1 to a value written in a reload register by the count clock cycle is the output pulse width.

Note that when the reload register value is 00H during 8-bit PPG operation or 0000H during 16-bit PPG operation, the pulse width is equivalent to one count clock cycle. In addition, note that when the reload register value is FFH during 8-bit PPG operation, the pulse width is equivalent to 256 count clock cycles. When the reload register value is FFFFH during 16-bit PPG operation, the pulse width is equivalent to 65536 count clock cycles.

The formula for the pulse width is shown below.

P1=

Ph=

T×(L+1)

T×(H+1)

{

L: PRL value

H: PRLH value

T: Input clock cycle

Ph: High pulse width

P1: Low pulse width

}

### 13.4.3 Selecting the Count Clock for the 8/16-Bit PPG

---

The peripheral clock or time base counter input is used as the count clock for 8/16-bit PPG operation. The count clock can be selected among six types of count clock inputs.

---

#### ■ Selecting the Count Clock for the 8/16-Bit PPG

Select the channel 0 clock using bits 4 to 2 (PCM2 to PCM0) in the PPGOE register and the channel 1 clock using bits 7 to 5 (PCS2 to PCS0). The clock can be selected among peripheral clocks obtained by dividing the machine clock by 1 to 16 and the clock input from the time base counter.

In the 8-bit prescaler and 8-bit PPG mode and 16-bit PPG mode, the channel 1 PPG operates using the count clock received from channel 0. In these modes, the value of bit 14 (PCS1) in the PPGC1 register is invalidated.

When the time base counter input is used, the first count cycle at activation by a trigger or after a stop may be out of synchronization. If the time base counter is cleared during operation of this module, the cycle may be out of synchronization.

**Note:**

In the 8-bit prescaler and 8-bit PPG mode, when channel 0 is in the operating state and channel 1 is in the stopped state, channel 1 may be activated, in which case the first count cycle may be out of synchronization.

### 13.4.4 Controlling the Pulse Pin Output of the 8/16-Bit PPG

The pulse output generated by operation of this module can be output from external pins PG00, PG01, PG10, and PG11.

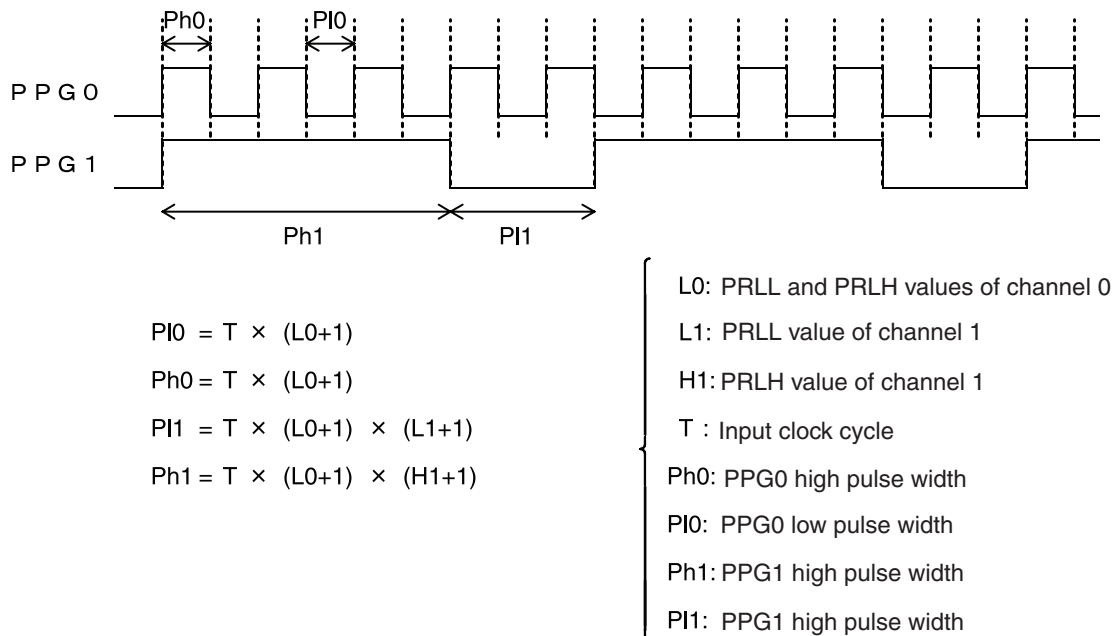
#### ■ Controlling the Pulse Pin Output of the 8/16-Bit PPG

Whether to enable the external pin outputs is determined by bit 5 (PE00) in the PPGC0 register for the PPG0 pin and bit 13 (PE10) in the PPGC1 register for the PPG1 pin. When each bit is "0" (initial value), the pulse output is not output from the corresponding external pin and the pin functions as a general-purpose port. When each bit is set to "1", the pulse output is output from the corresponding external pin

In the 16-bit PPG mode, the same waveform is output from PPG0 and PPG1. The same output can be obtained by enabling either external pin output.

In the 8-bit prescaler and 8-bit PPG mode, the toggle waveform of the 8-bit prescaler is output from PPG0 and the waveform of the 8-bit PPG is output from PPG1. Figure 13.4-2 "Output Waveforms during 8+8 PPG Output Operation" shows sample output waveforms in this mode.

**Figure 13.4-2 Output Waveforms during 8+8 PPG Output Operation**



**Note:**

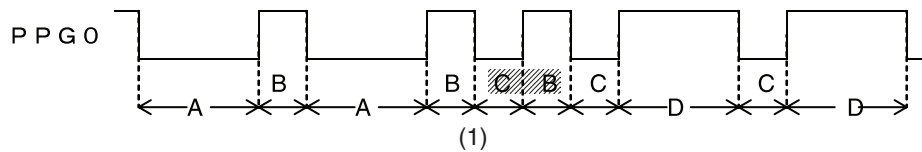
Setting the same value in the PRLH and PRLH of channel 0 is recommended.

### 13.4.5 Timing of Writing the Reload Registers in the 8/16-Bit PPG

In a mode other than the 16-bit PPG mode, use of a word transfer instruction for writing the reload registers (PRLH and PRLH) is recommended. If the reload registers are written using two byte transfer instructions, an output with an unexpected pulse width may be generated depending on the timing.

#### ■ Timing of Writing the Reload Registers in the 8/16-Bit PPG

**Figure 13.4-3 Timing Chart of Writing the Reload Registers in the 8/16-Bit PPG**

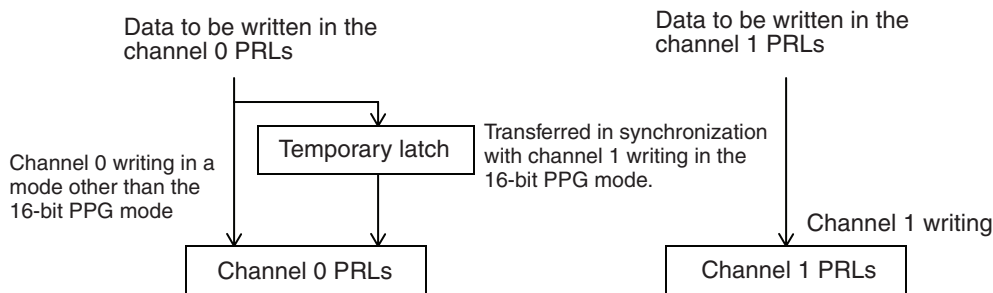


Rewriting the PRLH from A to C before timing (1) and rewriting the PRLH from B to D after (1) generates one pulse with C count cycles for L and B count cycles for H because the PRLH value is C and PRLH value is B at timing (1).

In the 16-bit PPG mode, use a long-word transfer instruction to write the PRLs for channels 0 and 1. Alternatively, use word transfer instructions to write the channel 0 PRLs and channel 1 PRLs in this order. In this mode, data to be written in the channel 0 PRLs is written temporarily. When the channel 1 PRLs are written, the data is written in the channel 0 PRLs.

In a mode other than the 16-bit PPG mode, the channel 0 PRLs and channel 1 PRLs can be written independently as shown below.

**Figure 13.4-4 Block Diagram of the PRL Write Block**



## 13.4.6 8/16-Bit PPG Interrupt

---

**The 8/16-bit PPG interrupt becomes active when the count reaches the reload value and a borrow occurs.**

---

### ■ 8/16-Bit PPG Interrupt

In the two 8-bit PPG channel mode or 8-bit prescaler and 8-bit PPG mode, an interrupt request is generated from each channel when a borrow occurs in the counter in the channel. In the 16-bit PPG mode, PUF0 and PUF1 are simultaneously set when a borrow occurs in the 16-bit counter. To use one interrupt factor only, enabling either PIE0 and PIE1 is recommended. To clear the interrupt factor, clearing PUF0 and PUF1 simultaneously is also recommended.



## 13.4.7 Initial Value of Each Hardware Component in the 8/16-Bit PPG

---

This section shows the initial value of each hardware component in the 8/16-bit PPG after a reset.

---

### ■ Initial Value of Each Hardware Component in the 8/16-Bit PPG

Each hardware component in the 8/16-bit PPG is initialized by a reset as follows.

#### ○ Registers

PPGC0 --> 0X000001<sub>B</sub>

PPGC1 --> 00000001<sub>B</sub>

PPGOE --> XXXXXX00<sub>B</sub>

#### ○ Pulse outputs

PPG0 --> "L"

PPG1 --> "L"

PE00 --> Disables the PPG0 output.

PE10 --> Disables the PPG1 output.

#### ○ Interrupt requests

IRQ0 --> "L"

IRQ1 --> "L"

Hardware components other than the above are not initialized.

# CHAPTER 14 8/16-BIT UP/DOWN COUNTER/TIMER

---

**This chapter describes the functions and operations of the 8/16-bit up/down counter/timer.**

---

- 14.1 "Overview of the 8/16-Bit Up/Down Counter/Timer"
- 14.2 "Block Diagram of the 8/16-Bit Up/Down Counter/Timer"
- 14.3 "Registers of the 8/16-Bit Up/Down Counter/Timer"
- 14.4 "Count Mode Selection for 8/16-Bit Up/Down Counter/Timer"
- 14.5 "Reload Function and Compare Function of 8/16-Bit Up/Down Counter/Timer"
- 14.6 "Simultaneous Activation of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer"
- 14.7 "Writing 8/16-Bit Up/Down Counter/Timer Data to UDCR"

## 14.1 Overview of the 8/16-Bit Up/Down Counter/Timer

---

The 8/16-bit up/down counter/timer consists of six event input terminals, two 8/16-bit up/down counters, two 8-bit reload/compare registers, and a control circuit.

---

### ■ Functions of 8/16-Bit Up/Down Counter/Timer

The main functions of the 8/16-bit up/down counter/timer are shown below.

#### ○ Counting range

An 8-bit counter register enables counting in the range from 0 to 256 (The operation mode of 16 bits x 1 enables counting in the range from 0 to 65535).

#### ○ Count mode

Four count modes by count clock selection.

- Timer mode
- Up/down counter mode
- Phase difference count mode (2 times)
- Phase difference count mode (8 times)

#### ○ Count clock

In timer mode, two internal clocks can be selected for the count clock.

- 125 ns (8 MHz: Divided into two)
- 1.0  $\mu$ s (2 MHz: Divided into eight)

#### ○ Detection edge selection

In up/down count mode, a detection edge of the external terminal input signal can be selected.

- Edge detection disabled
- Falling edge detection
- Rising edge detection
- Detection of both falling/rising edges

#### ○ Phase difference count mode

The phase difference count mode is suitable for counting an encoder such as a motor. Inputting phase A of the encoder, phase B, and phase Z output readily enables a high-precision rotation angle, number of rotations, etc. to be counted.

#### ○ ZIN terminal

For the ZIN terminal, two functions can be selected.

- Counter clear function
- Gate function

### ○ **Compare and reload function**

The 8/16-bit up/down counter/timer has a compare function and a reload function that can be used individually or in combination. Starting both functions enables up/down counting at any width.

- Compare function (interrupt output at compare)
- Compare function (interrupt output and counter clear at compare)
- Reload function (interrupt output and reload at underflow)
- Compare/reload function (interrupt output and counter clear at compare, interrupt output and reload at underflow)
- Compare/reload disabled

### ○ **Interrupt control**

At compare, at reload (underflow), and at overflow, interrupt generations can be controlled individually.

### ○ **Identifying count direction**

The count direction flag enables the preceding count direction to be identified.

### ○ **Count direction and interrupt**

When the count direction changes, an interrupt occurs.

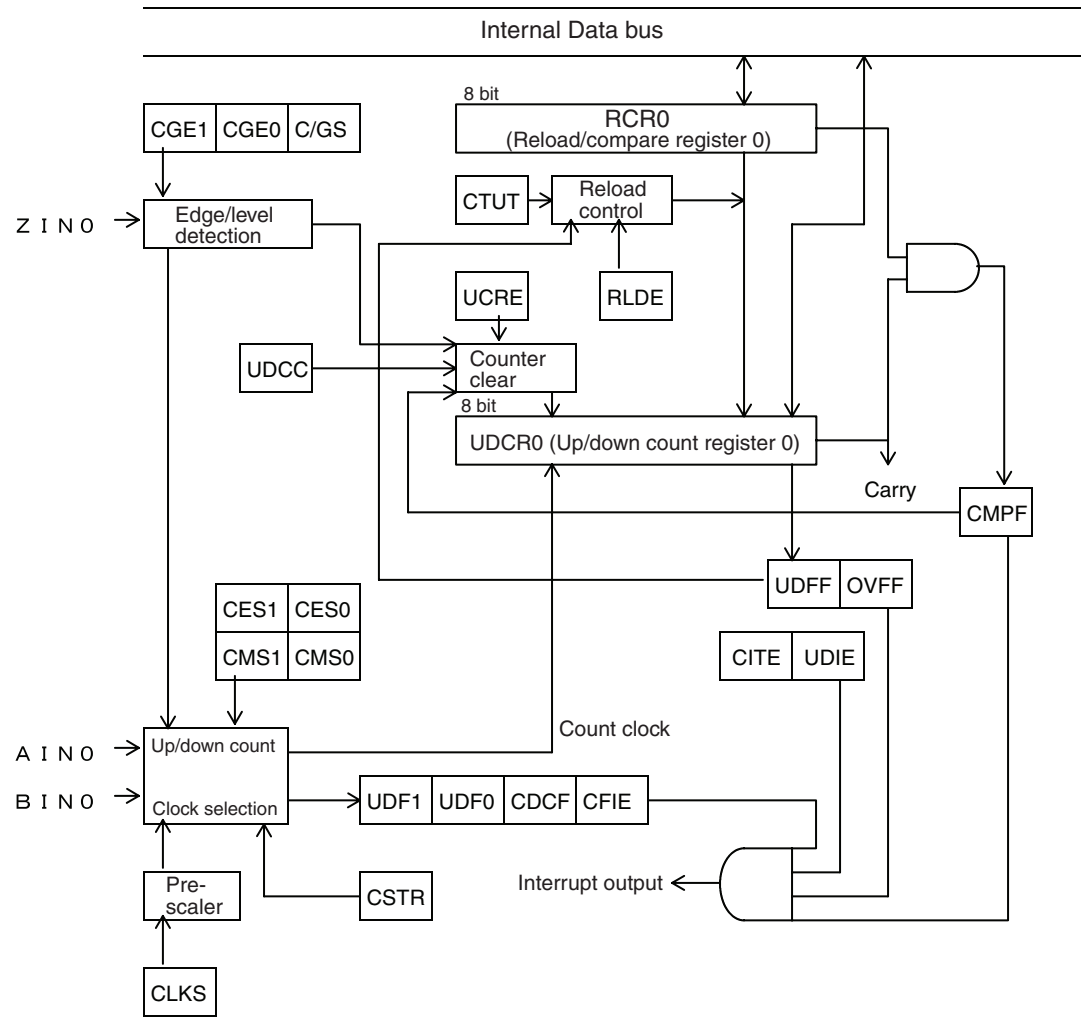
## 14.2 Block Diagram of the 8/16-Bit Up/Down Counter/Timer

Block diagram of 8/16-Bit Up/Down Counter/Timer is shown.

### ■ Block Diagram of the 8/16-Bit Up/Down Counter/Timer

Figure 14.2-1 "Block Diagram of the 8/16-Bit Up/Down Counter/Timer (Ch.0)" is a block diagram of the 8/16-bit up/down counter/timer Ch.0).

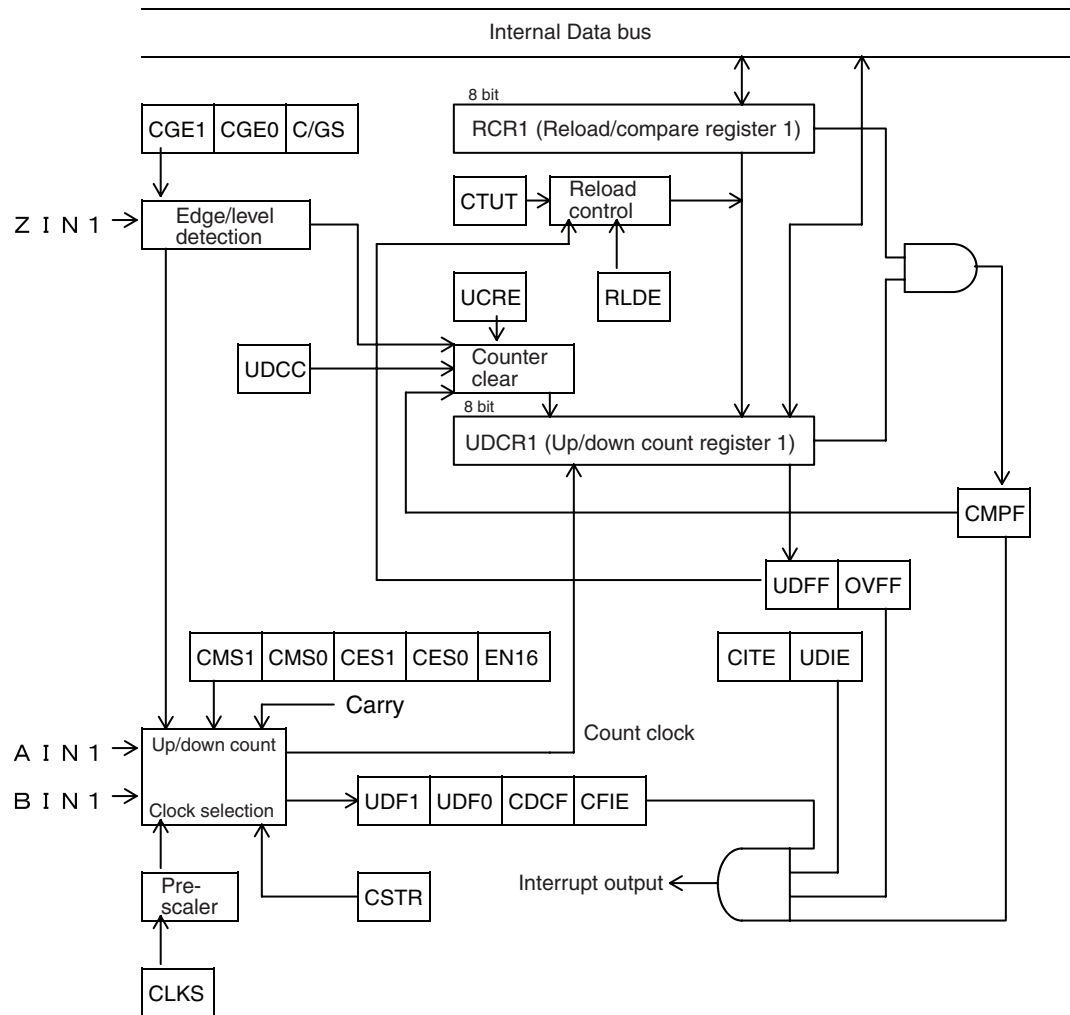
**Figure 14.2-1 Block Diagram of the 8/16-Bit Up/Down Counter/Timer (Ch.0)**



## 14.2 Block Diagram of the 8/16-Bit Up/Down Counter/Timer

Figure 14.2-2 "Block Diagram of the 8/16-Bit Up/Down Counter/Timer (Ch.1)" is a block diagram of the 8/16-bit up/down counter/timer (Ch.1).

**Figure 14.2-2 Block Diagram of the 8/16-Bit Up/Down Counter/Timer (Ch.1)**



## 14.3 Registers of the 8/16-Bit Up/Down Counter/Timer

Figure 14.3-1 "8/16-Bit Up/Down Counter/Timer Register Configuration" shows the registers of the 8/16-bit up/down counter/timer.

### ■ Registers of the 8/16-Bit Up/Down Counter/Timer

Figure 14.3-1 8/16-Bit Up/Down Counter/Timer Register Configuration

	15	8	7				0	
	UDCR 1							UDCR 0
	RCR 1							RCR 0
	Reserved area							CSR 0
	CCR H 0							CCRL 0
	Reserved area							CSR 1
	CCR H 1							CCRL 1
	8 bit							8 bit

Up/Down count register 1	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000071H	D17	D16	D15	D14	D13	D12	D11	D10	UDCR1
Read/Write ⇒	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Up/Down count register 0	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000070H	D07	D06	D05	D04	D03	D02	D01	D00	UDCR0
Read/Write ⇒	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Reload/Compare register 1	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000073H	D17	D16	D15	D14	D13	D12	D11	D10	RCR1
Read/Write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Reload/Compare register 0	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000072H	D07	D06	D05	D04	D03	D02	D01	D00	RCR0
Read/Write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Counter status registers 0/1	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000074H Address: 000078H	CSTR	CITE	UDIE	CMPF	OVFF	UDFF	UDF1	UDF0	CSR0,1
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Higher bits of the counter control register 0	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000077H	M16E	CDCF	CFIE	CLKS	CMS1	CMS0	CES1	CES0	CCR H0
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)	
Higher bits of the counter control register 1	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00007BH	—	CDCF	CFIE	CLKS	CMS1	CMS0	CES1	CES0	CCR H1
Read/Write ⇒	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(-)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Lower bits of the counter control registers 0/1	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000076H Address: 00007AH	—	CTUT	UCRE	RLDE	UDCC	CGSC	CGE1	CGE0	CCRL0,1
Read/Write ⇒	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(-)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

### 14.3.1 Up/Down Count Register Ch.0/1 (UDCR0/1)

The up/down count register is an 8-bit count register that performs up/down count operations according to the internal prescaler or input of the AIN terminal or BIN terminal.

#### ■ Up/Down Count Register Ch.0/1 (UDCR0/1)

**Figure 14.3-2 Up/Down Count Register Ch.0/1 (UDCR0/1)**

Up/Down count register 1

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000071H	D17	D16	D15	D14	D13	D12	D11	D10	UDCR1
Read/Write ⇒	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Up/Down count register 0

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000070H	D07	D06	D05	D04	D03	D02	D01	D00	UDCR0
Read/Write ⇒	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

The up/down count register is an 8-bit count register that performs up/down count operations according to the internal prescaler or input of the AIN terminal or BIN terminal. In 16-bit count mode, the up/down count register operates as a 16-bit count register. The value set in the higher 8-bit side of the control register becomes invalid during operation.

The up/down count register must be written via RCR. Write the value to be written to this register in the RCR, then write 1 in the CCRL CTUT bit, thereby transferring the value from the RCR to this register (reload by software).

To read this register that started in the 16-bit mode, read by word access.



### 14.3.2 Reload/Compare Register 0/1 (RCR0/1)

This register is an 8-bit reload/compare register. This register sets the reload value and compare value.

■ Reload/Compare Register 0/1 (RCR0/1)

Figure 14.3-3 Reload/Compare Register 0/1 (RCR0/1)

Reload/Compare register 1								
	15	14	13	12	11	10	9	8 ⇐ Bit No.
Address: 000073H	D17	D16	D15	D14	D13	D12	D11	D10 RCR1
Read/Write ⇐	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value ⇐	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0 ⇐ Bit No.
Address: 000072H	D07	D06	D05	D04	D03	D02	D01	D00 RCR0
Read/Write ⇐	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value ⇐	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

This register is an 8-bit reload/compare register that sets the reload value and compare value. The reload value and compare value are the same. Starting the reload function and compare function enables counting up/down between the 00h to RCR values (16-bit operation mode: 0000h to RCR values).

This register is write-only and cannot be read. Writing 1 in the CCR0/1 CTUT bit enables the value in this register to be transferred to the UDCR (reload by software).

To write this register, use word access.

### 14.3.3 Counter Status Register 0/1 (CSR0/1)

The counter status register 0/1 sets the event flag/interrupt operation control for Ch.0/1 in 8-bit mode.

#### ■ Counter Status Register 0/1 (CSR0/1)

The configuration of bits of the counter status register Ch.0/1 (CSR0/1) is shown below.

**Figure 14.3-4 Counter Status Register ch.0/1 (CSR0/1)**

Counter status registers 0/1

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000074 <sub>H</sub>	CSTR	CITE	UDIE	CMPF	OVFF	UDFF	UDF1	UDF0	CSR0,1
Address: 000078 <sub>H</sub>									
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

#### [Bit 7] CSTR

This bit is for control of start/stop of the UDCR count operation.

**Table 14.3-1 CSTR (Count Start Bit)**

CSTR	Function
0	Count operation stop (initial value)
1	Count operation start

#### [Bit 6] CITE

This bit is for enable/disable control of the interrupt output to the CPU when CMPF is set (compare occurred).

**Table 14.3-2 CITE (Compare Interrupt Output Control Bit)**

CITE	Function
0	Compare interrupt output disabled (initial value)
1	Compare interrupt output enabled

#### [Bit 5] UDIE

This bit is for enable/disable control of the interrupt output to the CPU when OVFF/UDFF is set (overflow/underflow occurred).

**Table 14.3-3 UDIE (Overflow/Underflow Interrupt Output Control Bit)**

UDIE	Function
0	Overflow/underflow interrupt output disabled (initial value)
1	Overflow/underflow interrupt output enabled

**[Bit 4] CMPF**

This flag indicates that the comparison results reveal that the UDCR value and RCR value are equal.

Note that this flag is set simultaneously with the start of the counter if the UDCR and RCR values match when the counter starts.

Only 0 can be written, but 1 cannot be written.

**Table 14.3-4 CMPF (Compare Detection Flag)**

<b>CMPF</b>	<b>Function</b>
0	Comparison results did not match (initial value)
1	Comparison results matched

**[Bit 3] OVFF**

This flag indicates that an overflow has occurred.

Only 0 can be written, but 1 cannot be written.

**Table 14.3-5 OVFF (Overflow Detection Flag)**

<b>OVFF</b>	<b>Function</b>
0	Overflow did not occur (initial value)
1	Overflow occurred

**[Bit 2] UDF**

This flag indicates that an underflow has occurred.

Only 0 can be written, but 1 cannot be written.

**Table 14.3-6 UDF (Underflow Detection Flag)**

<b>UDFF</b>	<b>Function</b>
0	Underflow did not occur (initial value)
1	Underflow occurred

**[Bits 1 to 0] UDF1, UDF0: Up/down flag**

This bit indicates the preceding count operation (up/down).

This bit is read-only and cannot be written.

**Table 14.3-7 UDF1, UDF0 (Up/Down Flag)**

<b>UDF1</b>	<b>UDF0</b>	<b>Detection edge</b>
0	0	No input (initial value)
0	1	Down count
1	0	Up count
1	1	Up/down generated simultaneously

### 14.3.4 Counter Control Register High Ch.0 (CCRH0)

The counter control register high Ch.0 sets the Ch.0 operation control in 8-bit mode and sets switching to the 16-bit mode.

The operation is set together with CCRL0.

#### ■ Counter Control Register High Ch.0 (CCRH0)

The configuration of bits of the counter control register high ch.0 (CCRH0) is shown below.

**Figure 14.3-5 Higher Bits of Counter Control Register 0 (CCRH0)**

Higher bits of the counter control register 0

	15	14	13	12	11	10	9	8	⇔ Bit No.
Address: 000077H	M16E	CDCF	CFIE	CLKS	CMS1	CMS0	CES1	CES0	CCRH0
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)	

#### [Bit 15] M16E

This bit is for selection (switching) of the 8 bits x 2 ch/16 bits x 1 ch operation mode.

**Table 14.3-8 M16E (16-Bit Mode Enable Setting Bit)**

M16E	16-bit mode enable setting
0	8 bits x 2 ch operation mode (initial value)
1	16 bits x 1 ch operation mode

#### [Bit 14] CDCF

This flag is set when the count direction changes from up to down or down to up during counting.

Only 0 can be written, but 1 cannot be written.

**Table 14.3-9 CDCF (Count Direction Change Flag)**

CDCF	Direction change detection
0	Direction not changed (initial value)
1	Direction changed at least once

**[Bit 13] CFIE**

This bit is for controlling an interrupt output to the CPU when the CDCF is set. An interrupt occurs if the count direction changes even once during counting.

**Table 14.3-10 CFIE (Count Direction Change Interrupt Enable Bit)**

CFIE	Direction change interrupt output
0	Direction change interrupt output disabled (initial value)
1	Direction change interrupt output enabled

**[Bit 12] CLKS**

When the timer mode is selected, this bit is for selection of the frequency of the internal prescaler.

This bit is valid only in timer mode, and the count direction is down only.

**Table 14.3-11 CLKS (Internal Prescaler Selection Bit)**

CLKS	Select internal clock
0	2 machine cycles (initial value)
1	8 machine cycles

**[Bits 11 to 10] CMS1, CMS0**

This bit is for selection of the count mode.

**Table 14.3-12 CMS1, CMS0 (Count Mode Selection Bit)**

CMS1	CMS0	Count mode
0	0	Timer mode [down count] (initial value)
0	1	Up/down count mode
1	0	Phase difference count mode (2 times)
1	1	Phase difference count mode (4 times)

**[Bits 9 to 8] CES1, CES0**

This bit is for selection of the detection edge of the external terminals AIN and BIN in up/down count mode.

This setting is invalid in modes other than the up/down count mode.

**Table 14.3-13 CES1, CES0 (Count Clock Edge Selection Bit)**

CES1	CES0	Selection edge
0	0	Edge detection disabled (initial value)
0	1	Falling edge detection
1	0	Rising edge detection
1	1	Detection of both falling/rising edges

### 14.3.5 Counter Control Register High Ch.1 (CCRH1)

The counter control register high Ch.1 sets the Ch.1 operation control in 8-bit mode. The operation is set together with CCRL1.

#### ■ Counter Control Register High Ch.1 (CCRH1)

The configuration of bits of the counter control register high Ch.1 (CCRH1) is shown below.

**Figure 14.3-6 Higher Bits of Counter Control Register ch.1 (CCRH1)**

Higher bits of the counter control register 1

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00007BH	—	CDCF	CFIE	CLKS	CMS1	CMS0	CES1	CES0	CCRH1
Read/Write ⇒	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(-)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

**[Bit 15] Unused bit**

**[Bit 14] CDCF**

This flag is set when the count direction changes from up to down or down to up during counting.

Only 0 can be written, but 1 cannot be written.

**Table 14.3-14 CDCF (Count Direction Change Flag)**

CDCF	Direction change flag
0	Direction not changed (initial value)
1	Direction changed at least once

**[Bit 13] CFIE**

This bit is for control of an interrupt output to the CPU when the CDCF is set. An interrupt occurs if the count direction changes even once during counting.

**Table 14.3-15 CFIE (Count Direction Change Interrupt Enable Bit)**

CFIE	Direction change interrupt output
0	Direction change interrupt output disabled (initial value)
1	Direction change interrupt output enabled

**[Bit 12] CLKS**

When the timer mode is selected, this bit is for selection of the internal prescaler frequency.

This bit is valid in timer mode only, and the count direction is down only.

**Table 14.3-16 CLKS (Internal Prescaler Selection Bit)**

CLKS	Select internal clock
0	2 machine cycles (initial value)
1	8 machine cycles

**[Bits 11 to 10] CMS1, CMS0**

This bit is for selection of the count mode.

**Table 14.3-17 CMS1, CMS0 (Count Mode Selection Bit)**

CMS1	CMS0	Count mode
0	0	Timer mode [down count] (initial value)
0	1	Up/down count mode
1	0	Phase difference count mode, 2 times
1	1	Phase difference count mode, 4 times

**[Bits 9 to 8] CES1, CES0**

This bit is for selection of the detection edge of the external terminals AIN and BIN in up/down count mode.

This setting is invalid in modes other than the up/down count mode.

**Table 14.3-18 CES1, CES0 (Count Clock Edge Selection Bit)**

CES1	CES0	Selection edge
0	0	Edge detection disabled (initial value)
0	1	Falling edge detection
1	0	Rising edge detection
1	1	Detection of both falling/rising edges

## 14.3.6 Counter Control Register Low Ch.0/1 (CCRL0/1)

The counter control register low Ch.0/1 sets the Ch.0/1 operation control in 8-bit mode. The operation is set together with CCRH0/1.

### ■ Counter Control Register Low Ch.0/1 (CCRL0/1)

The configuration of bits of the counter control register low Ch.0/1 (CCRL0/1) is shown below.

**Figure 14.3-7 Lower Bits of Counter Control Register ch.0/1 (CCRL0/1)**

Lower bits of the counter control registers 0/1

	7	6	5	4	3	2	1	0	⇔ Bit No.
Address: 000076H Address: 00007AH	—	CTUT	UCRE	RLDE	UDCC	CGSC	CGE1	CGE0	CCRL0,1
Read/Write ⇔	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇔	(-)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

#### [Bit 7] Unused bit

#### [Bit 6] CTUT

This bit is for data transfer from the RCR to the UDCR.

When 1 is written in this bit, data is transferred from the RCR to the UDCR.

Even if written, 0 is invalid. The read value is always 0.

1 must not be written in this bit during counting (when the CSR0 CSTR bit is 1).

#### [Bit 5] UCRE

This bit is for control of UDCR clearance by output from compare.

This bit affects clearance caused by the output from compare but does not affect the UDCR clear function (as caused by the ZIN terminal).

**Table 14.3-19 UCRE (UDCR Clear Enable Bit)**

UCRE	Count clear by compare
0	Counter clear disabled (initial value)
1	Counter clear enabled

#### [Bit 4] RLDE

This bit is for control of the reload function. The RCR value is transferred to the UDCR if the UDCR generates an underflow when the reload function is active.

**Table 14.3-20 RLDE (Reload Enable Bit)**

RLDE	Reload function
0	Reload function disabled (initial value)
1	Reload function enabled



**[Bit 3] UDCC**

This bit is for clearance of UDCR. When 0 is written in this bit, the UDCR is cleared to 0000<sub>H</sub>.

Even if written, 1 is invalid. The read value is always 1.

**[Bit 2] CGSC**

This bit is for selection of the external terminal ZIN function.

**Table 14.3-21 CGSC (Counter Clear/Gate Selection Bit)**

CGSC	ZIN function
0	Count clear function (initial value)
1	Gate function

**[Bits 1 to 0] CGE1, CGE0**

This bit is for selection of the detection edge/level of the external terminal ZIN.

**Table 14.3-22 CGE1, CGE0 (Counter Clear/Gate Edge Selection Bit)**

CGE1	CGE0	When the counter clear function is selected	When the gate function is selected
0	0	Edge detection disabled (initial value)	Level detection disabled (count disable)
0	1	Falling edge	Low level
1	0	Rising edge	High level
1	1	Setting disabled	Setting disabled

## 14.4 Count Mode Selection for 8/16-Bit Up/Down Counter/Timer

The 8/16-bit up/down counter/timer has four count modes. CCRH CMS1 and CMS0 are used for control of the count mode selection.

### ■ Count Mode Selection for the 8/16-Bit Up/Down Counter/Timer

Table 14.4-1 "Four Count Modes of the 8/16-Bit Up/Down Counter/Timer" shows the four count modes of the 8/16-bit up/down counter/timer.

**Table 14.4-1 Four Count Modes of the 8/16-Bit Up/Down Counter/Timer**

CMS1, CMS0	Count mode
00 <sub>B</sub>	Timer mode [down count]
01 <sub>B</sub>	Up/down count mode
10 <sub>B</sub>	Phase difference count mode (2 times)
11 <sub>B</sub>	Phase difference count mode (4 times)

#### ○ Timer mode [down count]

In timer mode, internal prescaler output is counted down. For the internal prescaler, 2 machine cycles/8 machine cycles can be selected through CCRH CLKS.

#### ○ Up/down count mode

In up/down count mode, inputs of external terminals AIN and BIN are counted to count up/down. AIN terminal inputs control counting up. BIN terminal inputs control counting down.

AIN terminal and BIN terminal inputs are edge detection. Detection edges can be selected through CCRH CES1 and CES0. Table 14.4-2 "Detection Edge Selection of the 8/16-Bit Up/Down Counter/Timer" shows the detection edges.

**Table 14.4-2 Detection Edge Selection of the 8/16-Bit Up/Down Counter/Timer**

CES1, CES0	Detection edge
00 <sub>B</sub>	Edge detection disabled
01 <sub>B</sub>	Falling edge detection
10 <sub>B</sub>	Rising edge detection
11 <sub>B</sub>	Detection of both falling/rising edges

#### ○ Phase difference count mode (2 times/4 times)

In phase difference count mode, when the AIN terminal input edge is detected, the BIN terminal input level is detected to count the phase difference between phase A and phase B of the encoder output signal. When the BIN terminal input edge is detected, the AIN terminal input level is detected to count the phase difference between phase A and phase B of the encoder output signal.

In 2 times/4 times mode, if AIN is earlier, the phase difference between phase A and phase B is

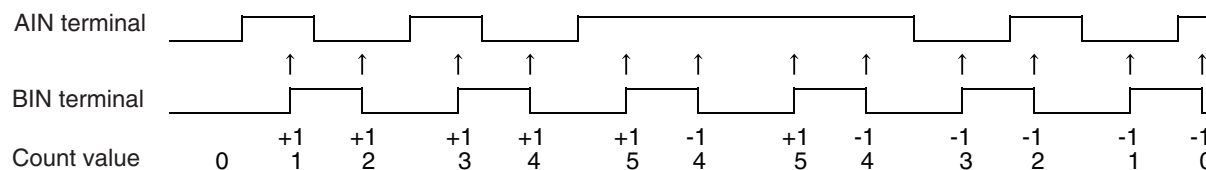
counted up. If BIN is earlier, the phase difference between phase A and phase B is counted down.

In 2 times mode, the AIN terminal value is detected to count in timings of both BIN terminal rising/falling edges. Counting is performed as follows:

- When the value of the AIN terminal detected by the BIN terminal rising edge is "H", the counter is counted up.
- When the value of the AIN terminal detected by the BIN terminal rising edge is "L", the counter is counted down.
- When the value of the AIN terminal detected by the BIN terminal falling edge is "H", the counter is counted down.
- When the value of the AIN terminal detected by the BIN terminal falling edge is "L", the counter is counted up.

Figure 14.4-1 "Overall Operation in Phase Difference Count Mode (2 Times)" shows the overall operation in phase difference count mode (2 times).

**Figure 14.4-1 Overall Operation in Phase Difference Count Mode (2 Times)**



In 4 times mode, the AIN terminal value is detected to count in timings of both BIN terminal rising/falling edges. The BIN terminal value is detected to count in timings of both AIN terminal rising/falling edges. Counting is performed as follows:

- When the value of the AIN terminal detected by the BIN terminal rising edge is "H", the counter is counted up.
- When the value of the AIN terminal detected by the BIN terminal rising edge is "L", the counter is counted down.
- When the value of the AIN terminal detected by the BIN terminal falling edge is "H", the counter is counted down.
- When the value of the AIN terminal detected by the BIN terminal falling edge is "L", the counter is counted up.
- When the value of the BIN terminal detected by the AIN terminal rising edge is "H", the counter is counted down.
- When the value of the BIN terminal detected by the AIN terminal rising edge is "L", the counter is counted up.
- When the value of the BIN terminal detected by the AIN terminal falling edge is "H", the counter is counted up.
- When the value of the BIN terminal detected by the AIN terminal falling edge is "L", the counter is counted down.

Figure 14.4-2 "Overall Operation in Phase Difference Count Mode (4 Times)" shows the overall operation in phase difference count mode (4 times).



## 14.5 Reload Function and Compare Function of 8/16-Bit Up/Down Counter/Timer

---

The 8/16-bit up/down counter/timer has a reload function and a compare function that can be combined for processing.

---

### ■ Reload Function and Compare Function of the 8/16-Bit Up/Down Counter/Timer

Table 14.5-1 "Selection of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer" shows the settings that can be selected for the reload/compare functions of the 8/16-bit up/down counter/timer.

**Table 14.5-1 Selection of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer**

RLDE, UCRE	Reload/compare functions
00 <sub>B</sub>	Reload/compare disabled (initial value)
01 <sub>B</sub>	Compare enabled
10 <sub>B</sub>	Reload enabled
11 <sub>B</sub>	Compare/reload enabled

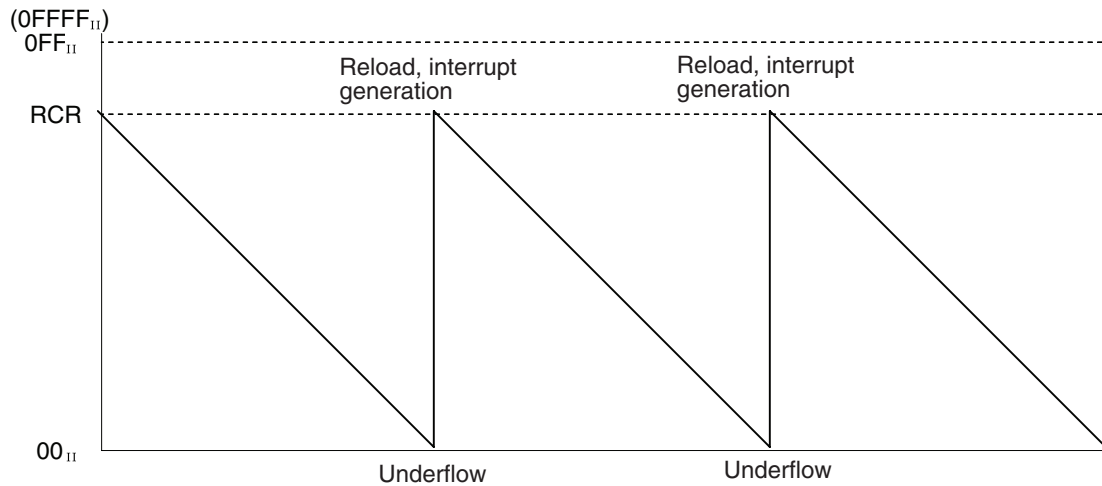
#### ○ Reload function

When the reload function is active, the RCR value is transferred to the UDCR in the next timing of the down count clock after underflow generation. UDFF is set and an interrupt request is generated.

#### **Note:**

In a mode that does not perform a counting down operation, starting this function is invalidated.

Figure 14.5-1 "Overall Operation of the Reload Function" shows the overall operation of the reload function.

**Figure 14.5-1 Overall Operation of the Reload Function**

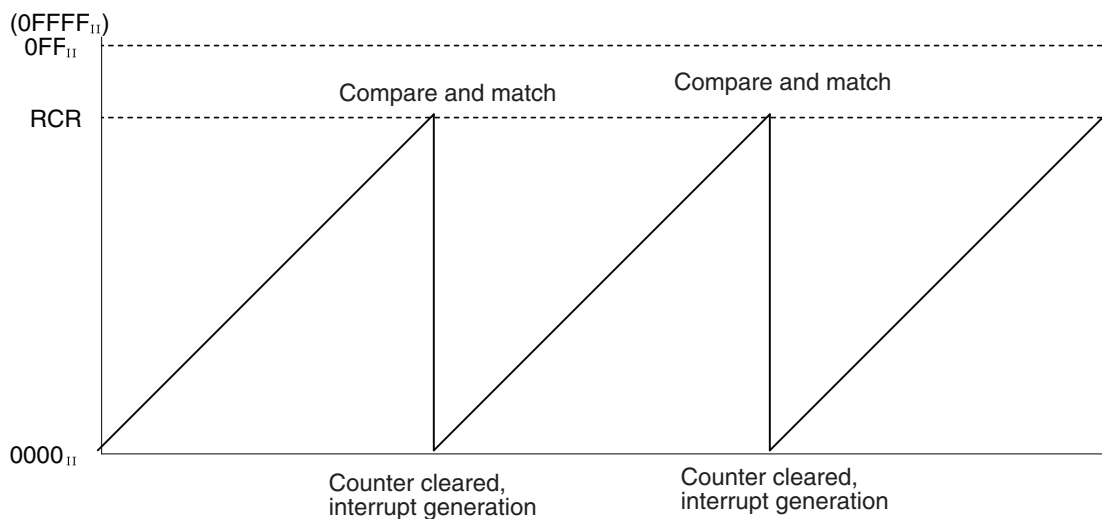
### ○ Compare function

The compare function can be used in all modes except the timer mode. With the compare function active, when the RCR and UDCR values match, the CMPF is set and an interrupt request is generated. When the compare clear function is active, the UDCR is cleared in the next timing of the up count clock.

#### **Note:**

In the mode that does not perform counting up, starting this function is invalidated.

Figure 14.5-2 "Overall Operation of the Compare Function" shows the overall operation of the compare function.

**Figure 14.5-2 Overall Operation of the Compare Function**

## 14.6 Simultaneous Activation of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer

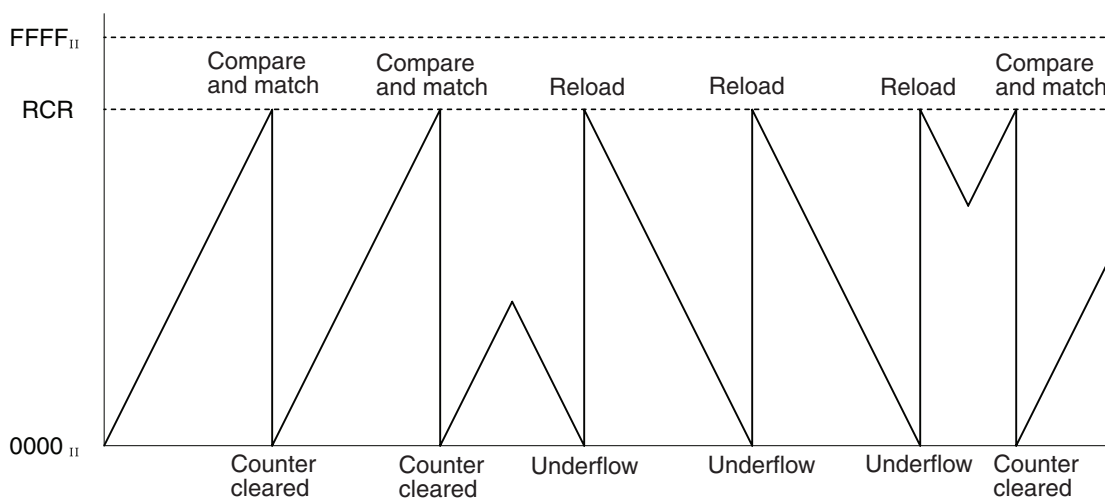
When the reload/compare functions are simultaneously activated, counting up/down is enabled in any width.

### ■ Simultaneous Activation of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer

The reload function that starts at underflow transfers the RCR value to the UDCR. The compare function that starts when the RCR and UDCR values match clears the UDCR. These two functions are used for counting up/down between the 00H to RCR values.

Figure 14.6-1 "Overall Operation when the Reload/Compare Functions are Activated Simultaneously" shows the overall operation when the reload/compare functions are activated simultaneously.

**Figure 14.6-1 Overall Operation when the Reload/Compare Functions are Activated Simultaneously**



When comparison matches or at reloading (underflow), an interrupt can be generated in the CPU. Enabling these interrupt outputs can be controlled individually.

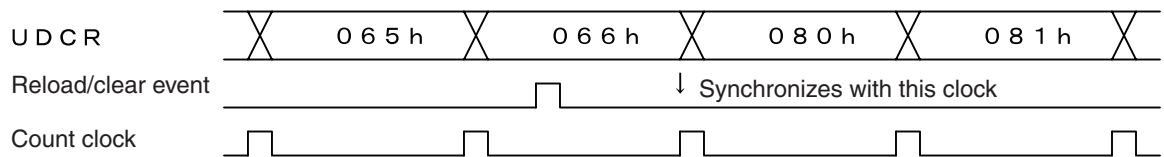
The timing for reloading to the UDCR or clearing the UDCR differs depending on whether counting is active or stopped.

#### ○ Reload/Clear timing during count operation

If a reload or clear event occurs during count operation, all events synchronize with the count clock. (Figure 14.6-2 "Reload/Clear Timing during Count Operation" is an example of reloading 80h.)

## 14.6 Simultaneous Activation of Reload/Compare Functions of 8/16-Bit Up/Down Counter/Timer

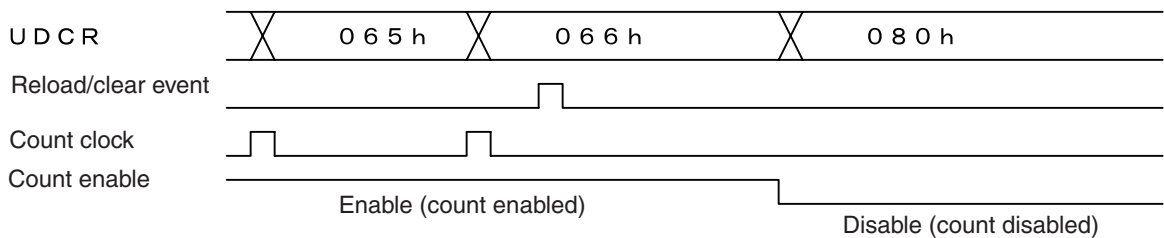
**Figure 14.6-2 Reload/Clear Timing during Count Operation**



### ○ Count value at count disable after reload clear

If reload and clear events occur during a count operation and counting stops while waiting for a count clock synchronization, reload and clear are performed when the counter stops. (Figure 14.6-3 "Count Value at Count Disable after Reload Clear" is an example of reloading 80h.)

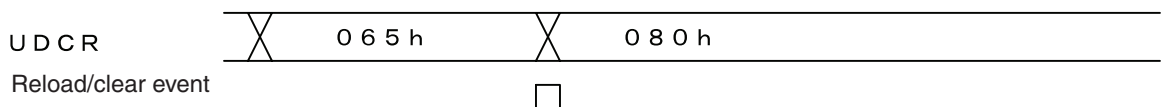
**Figure 14.6-3 Count Value at Count Disable after Reload Clear**



### ○ Reload/clear timing when counting is stopped

If reload and clear events occur when counting is stopped, reload and clear are performed when the events occur. (Figure 14.6-4 "Reload/Clear Timing when Counting is Stopped" is an example of reloading 80h.)

**Figure 14.6-4 Reload/Clear Timing when Counting is Stopped**



Regarding clear by comparison, clearance is performed when the UDCR and RCR values match and the counter is counted up. Clearance is not performed even if the UDCR and RCR values match when the counter is counted down or stopped.

The clear timing in all events except reset input conform to the above timing. The reload timing in all events conform to the above timing.

When a clear event and reload event occur synchronously, priority is given to the clear event.



## 14.7 Writing 8/16-Bit Up/Down Counter/Timer Data to UDCR

Data cannot be written to the UDCR directly from a data bus. To write data in the UDCR, proceed as follows:

1. Write data to be written to UDCR in RCR. (Note that RCR data will be lost.)
2. Writing 1 in the CCRH CTUT enables the data to be transferred from the RCR to the UDCR.

Perform the above operation when the counter is stopped (when the CSR CSTR bit is 0).

### ■ Writing Data to UDCR

To clear the counter, the following methods in addition to the above are available.

- To clear by reset input (initialize)
- To clear by edge input from the ZIN terminal
- To clear by writing 0 in CCRH UDCC
- To clear by the compare function

This writing can be performed regardless of whether counting is active/stopped.

### ■ Count Clear/Gate Function

The ZIN terminal can be selected in the CCRH CGSC and can be used as the count clear function or gate function as shown below.

**Table 14.7-1 Selection of ZIN Terminal Functions**

CGSC	ZIN terminal function
0 <sub>B</sub>	Counter clear function
1 <sub>B</sub>	Gate function

When the count clear function is active, the counter is cleared according to the edge input from the ZIN terminal. The edge of the ZIN terminal input signal to clear the counter is selected in CCRL CGE1 and CGE0. If this function inputs the encoder Z phase output to this terminal, the UDCR can be cleared at the count start point of the encoder.

When the gate function is active, counting is enabled/disabled according to the level input from the ZIN terminal. The level of the ZIN terminal input signal to enable counting is selected in CCRL CGE1 and CGE0.

This function is usable in all count modes.

Table 14.7-2 "Selection of Detection Edge by ZIN Terminal Input Signal" shows the selection of ZIN terminal functions.

**Table 14.7-2 Selection of Detection Edge by ZIN Terminal Input Signal**

CGE1,C GE0	Counter clear function	Gate function
00 <sub>B</sub>	Detection disabled	Detection disabled
01 <sub>B</sub>	Rising edge	LOW level
10 <sub>B</sub>	Falling edge	HIGH level

#### ■ Count Direction Flag, Count Direction Change Flag

The count direction flag (UDF1, UDF0) indicates whether the preceding count was counted up or down during counting up/down. The count clock generated from inputs of both AIN and BIN terminals is used for judgment and the flag is rewritten at each counting. The current rotation direction for motor control can be determined from this flag.

This function is usable in all count modes.

Table 14.7-3 "Count Direction Flag" shows the count direction flag.

**Table 14.7-3 Count Direction Flag**

UDF1, FDF0	Count direction
01 <sub>B</sub>	Count down
10 <sub>B</sub>	Count up
11 <sub>B</sub>	Up/down occur simultaneously (count operation is not performed)

The count direction change flag (CDCF) is set when the count direction changes from up to down or down to up. When this flag is set, an interrupt occurs in the CPU simultaneously. Referencing this interrupt and the count direction flag (UDF1, UDF0) enables a change in count direction. However, note that the direction indicated by the flag following a direction change may be restored to the original direction if direction changes occur continuously at short intervals.

Table 14.7-4 "Count Direction Change Flag" shows the count direction change flag.

**Table 14.7-4 Count Direction Change Flag**

CDCF	Count direction change detection
0 <sub>B</sub>	Direction not changed
1 <sub>B</sub>	Direction changed (once or more)

#### ■ Compare Detection Flag

The compare detection flag (CMPF) is set when the UDCR value and RCR value become equal during a count operation. This flag is also set when the values match by counting up, when the values match by the generation of a reload event, and when the values already match at the start of a counting.

However, even if the values match as a result of counting down (except for compare during a reload following an underflow), the results of comparing these values reveal that they are not equivalent, in which case the flag is not set.

### ■ 8 Bits × 2 Ch, 16 Bits × 1 Ch Operation

This module can be used as an 8-bit up/down counter × 2 ch or 16-bit up/down counter × 1 ch. Writing 0 in the M16E bit of the CCRH0 register sets this module to the 8 bits × 2 ch mode. Writing in the M16E bit of the CCRH0 register sets this module to the 16 bits × 1 ch mode.

In 16 bits × 1 ch mode, the CSR0, CCRL0, and CCRH0 registers become valid and the CSR1, CCRL1, and CCRH1 registers cannot be used. The AIN0, BIN0, and ZIN0 terminals become valid input terminals and AIN1, BIN1, and ZIN1 terminals are not used.

# CHAPTER 15 DTP/EXTERNAL INTERRUPT

---

**This chapter describes the functions and operations of the DTP/external interrupt.**

---

- 15.1 "Overview of the DTP/External Interrupt"
- 15.2 "Registers of the DTP/External Interrupt"
- 15.3 "Operation of the DTP/External Interrupt"
- 15.4 "Notes on Using the DTP/External Interrupt"

## 15.1 Overview of the DTP/External Interrupt

**DTP is a peripheral circuit for activating the intelligent I/O service or external interrupt processing.**

### ■ DTP and External Interrupt

The DTP is installed between a peripheral circuit outside the device and the F<sup>2</sup>MC-16LX CPU. The DTP receives a DMA request or an interrupt request from the external peripheral circuit (\*1) and posts the request to the F<sup>2</sup>MC-16LX CPU to activate the intelligent I/O service or external interrupt processing. For the intelligent I/O service, H and L can be selected to indicate the request level. For external interrupt processing, H, L, rising edge, and falling edge can be selected. Although a request indicated by a level cannot be input to ch0 and ch1, both edges can be input.

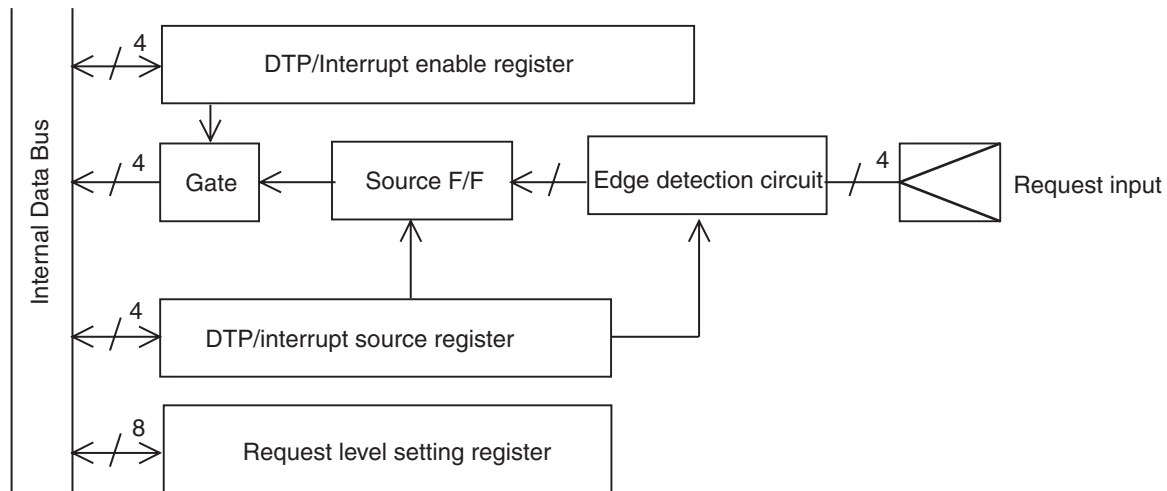
\*1 Peripheral function device to be connected outside an MB90570 series device

#### Note:

Ch0 and ch1 cannot be used for the intelligent I/O service.

### ■ Block Diagram of the DTP/External Interrupt

**Figure 15.1-1 Block Diagram of DTP/External Interrupt**



## 15.2 Registers of the DTP/External Interrupt

There are the following three types of DTP/External interrupt registers:

- DTP/interrupt enable register (ENIR)
- DTP/interrupt source register (EIRR)
- Request level setting register (ELVR)

### ■ Registers of the DTP/External Interrupt

**Figure 15.2-1 DTP/External Interrupt Register**

DTP/interrupt enable register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000030H	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	ENIR
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

DTP/interrupt source register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000031H	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	EIRR
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Request level setting register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000033H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	
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)	

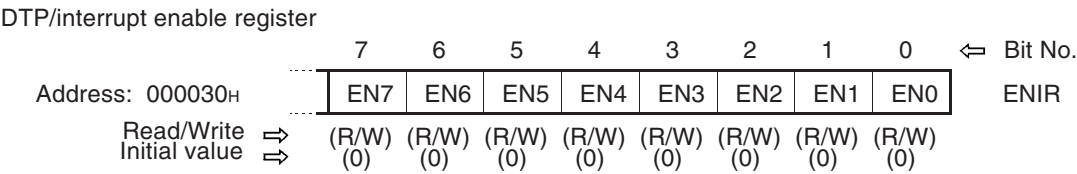
	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000032H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	ELVR
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

### 15.2.1 DTP/Interrupt Enable Register (ENIR)

This register specifies device pins to be used as DTP/external interrupt request input pins to issue a request to the interrupt controller.

■ DTP/Interrupt Enable Register (ENIR)

Figure 15.2-2 DTP/Interrupt Enable Register (ENIR)



If the pin corresponding to a bit of this register is set to 1, the pin is used as an DTP/external interrupt input pin and generates a request to the interrupt controller. If the pin corresponding to a bit of this register is set to 0, the pin holds an DTP/external interrupt request input source but does not generate a request to the interrupt controller.

## 15.2.2 DTP/Interrupt Source Register (EIRR)

This register indicates there is a corresponding DTP/external interrupt request during a read operation and clears the flip-flop contents indicating the request during a write operation.

### ■ DTP/Interrupt Source Register (EIRR)

**Figure 15.2-3 DTP/Interrupt Source Register (EIRR)**

DTP/interrupt source register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000031 <sub>H</sub>	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	EIRR
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

If this register is read and a bit of this register is set to "1", the corresponding pin indicates an DTP/external interrupt request. Writing "0" in a bit of this register clears the request flip-flop corresponding to the bit. Writing "1" does not effect an operation. During the read of a read-modify-write operation, "1" is read.



### 15.2.3 Request Level Setting Register (ELVR)

This register selects request detection conditions.

#### ■ Request Level Setting Register (ELVR)

**Figure 15.2-4 Request Level Setting Register (ELVR)**

Request level setting register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000033H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	
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)	
	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000032H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	ELVR
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)	

Two bits are assigned per pin. Table 15.2-1 "Operation of External Level Register (ELVR) (LA2 to LA7, LB2 to LB7)" describes the correspondence between detection conditions and LA2 to LA7 and LB2 to LB7.

Table 15.2-2 "Operation of External Level Register (ELVR) (LA0 to LA1, LB0 to LB1)" describes the correspondence between detection conditions and LA0 to LA1 and LB1 to LB1. If an input representing a request level is cleared but the input remains active, the condition is set again.

**Table 15.2-1 Operation of Request Level Setting Register (ELVR) (LA2 to LA7, LB2 to LB7)**

LBx	LAx	Operation
0	0	Request detection at L level
0	1	Request detection at H level
1	0	Request detection at rising edge (↑)
1	1	Request detection at falling edge (↓)

**Table 15.2-2 Operation of Request Level Setting Register (ELVR) (LA0 to LA1, LB0 to LB1)**

LBx	LAx	Operation
0	0	Request detection at both edges
0	1	Request not detected
1	0	Request detection at rising edge (↑)
1	1	Request detection at falling edge (↓)

## 15.3 Operation of the DTP/External Interrupt

The DTP /external interrupt contains the external interrupt function and the DTP function, both of which are explained in this section.

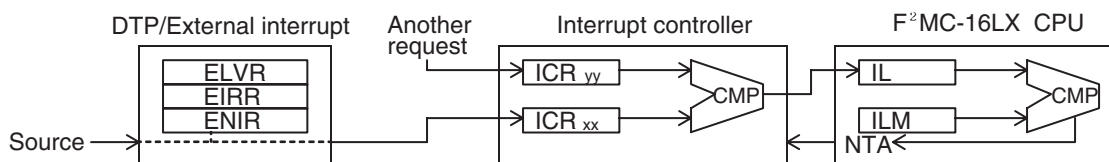
### ■ Operation of the DTP/External Interrupt

When a request set in a corresponding pin by the ELVR register is input after the external interrupt request is set, this resource generates an interrupt request signal for the interrupt controller.

The priorities of interrupts that occurred concurrently are checked in the interrupt controller. If the interrupt from this resource has the highest priority, the interrupt controller generates an interrupt request for 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 CCR register in the CPU. If the request level is higher than the ILM bit value, the CPU activates the hardware interrupt processing microprogram when the instruction being executed terminates.

Figure 15.3-1 "External Interrupt Operation" shows the operation of an external interrupt.

**Figure 15.3-1 External Interrupt Operation**



The CPU reads ISE bit information from the interrupt controller in the hardware interrupt processing microprogram to confirm that the request is interrupt processing. The CPU then branches control to the interrupt processing microprogram. The interrupt processing microprogram reads an interrupt vector area and generates interrupt acknowledge to the interrupt controller. The microprogram transfers the jump destination address of a microinstruction generated from the vector to the program counter and executes the user interrupt processing program.

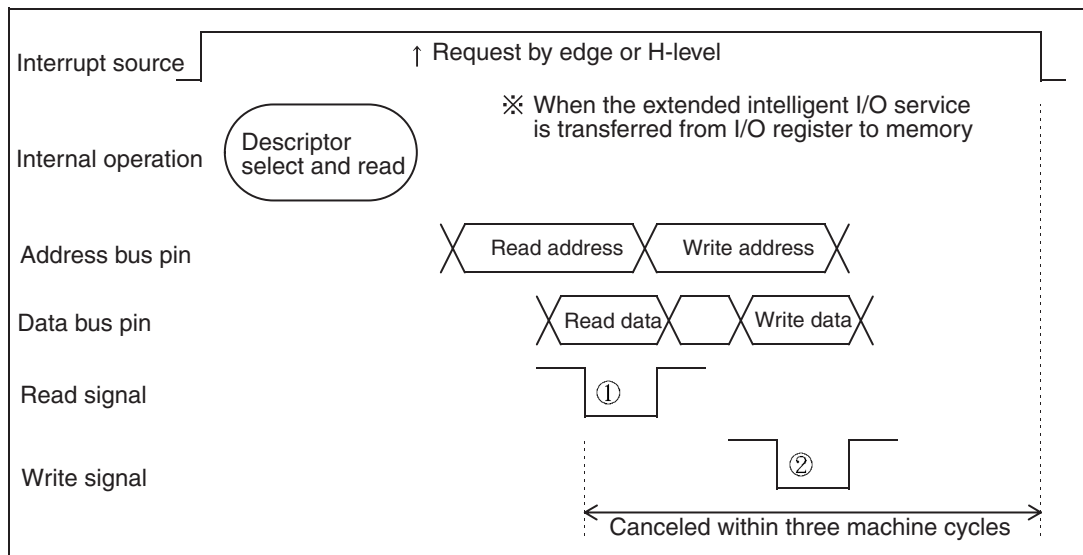
### ■ DTP Operation

As initialization, to activate the intelligent I/O service, the user program sets addresses of registers allocated to 000000<sub>H</sub> to 0000FF<sub>H</sub> in the I/O address pointer in the extended intelligent I/O service descriptor and sets the head address of the memory buffer in the buffer address pointer.

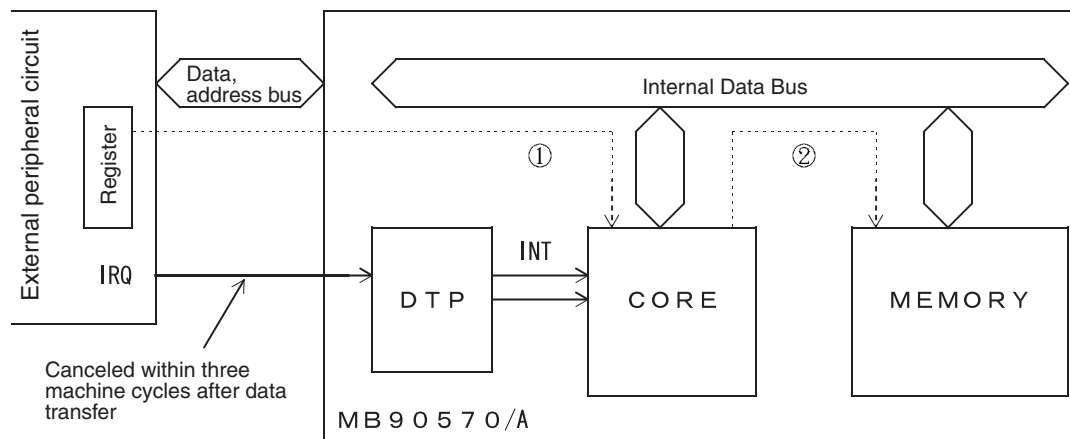
The DTP operation sequence is exactly the same as that of external interrupts up to the step in which the CPU activates the hardware interrupt processing microprogram. For the DTP, because the ISE bit that is read by the CPU during the hardware interrupt processing that is done by the microprogram indicates the DTP, control is passed to the extended intelligent I/O service processing microprogram. When the intelligent service is activated, a read or a write signal is sent to the addressed external peripheral circuit to switch data with the chip. The external peripheral circuit should cancel the interrupt request to the chip within three machine cycles after data transfer. After completion of data switching, the descriptor is updated and a signal to clear the transfer source is generated by the interrupt controller. The resource receives the signal and clears the flip-flop holding the source for the next request from a pin.

See the "F<sup>2</sup>MCR-16LX Programming Manual" for more information on extended intelligent I/O service processing.

**Figure 15.3-2 External Interrupt Cancel Timing at Termination of DTP Operation**



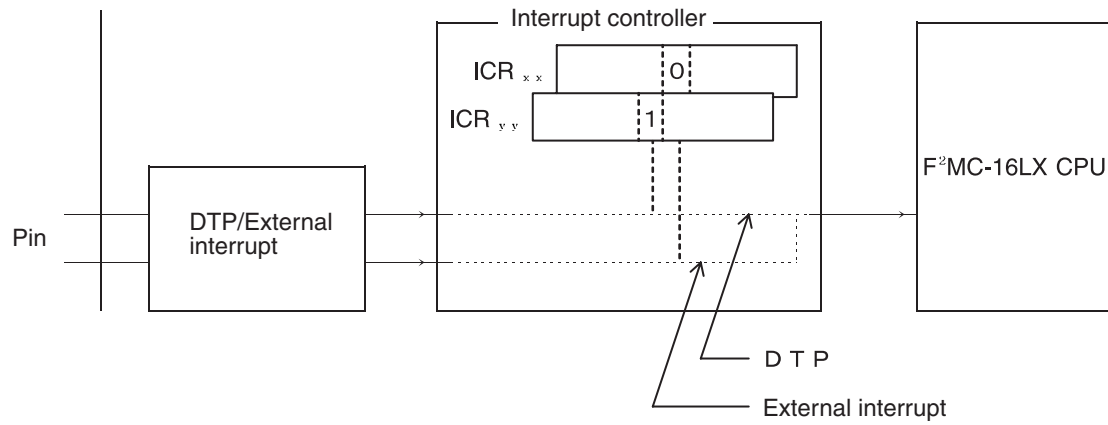
**Figure 15.3-3 Outline of an Interface between a DTP and an External Peripheral Circuit**



#### ■ Switching between DTP Request and External Interrupt Request

Switching between DTP requests and external interrupt requests is performed in accordance with corresponding ISE bit settings of the ICR register in the interrupt controller. An ICR is allocated to each pin. If the ISE bit of the corresponding ICR is set to 1, it is assumed that the pin treats a DTP request. If the bit is set to 0, it is assumed that the pin treats an external interrupt request.

Figure 15.3-4 Switching between DTP Request and External Interrupt Request



## 15.4 Notes on Using the DTP/External Interrupt

---

Note the following when using DTP/external interrupts.

- Conditions of peripheral circuits connected externally when the DTP is used
  - Return from standby state
  - Operation procedure of DTP/external interrupt
  - External interrupt request level
- 

### ■ Conditions of peripheral circuits connected externally when the DTP is used

The DTP can support external peripheral circuits that automatically clear requests after completion of transfer. The request must be canceled within three machine cycles (defined temporarily) after a transfer operation starts. Otherwise, this resource assumes the current request to be the next transfer request.

### ■ Return from Standby State

#### ○ Return from standby state (MB90V570, MB90F574, MB90573, MB90574)

If an external interrupt is used for a return from the standby state in input clock stop mode, set the request level to "H" level. If the external interrupt request is set to "L" level, a malfunction may occur.

The standby state in input clock stop mode does not return if an edge of the external interrupt request is set.

#### ○ Return from standby state (MB90V570A, MB90F574A, MB90574C)

If an external interrupt is used for a return from the standby state in input clock stop mode, select an "H" level or "L" level request for ch2 to ch7 and an edge request for ch0 and ch1.

### ■ Operation procedure of DTP/external interrupt

When using registers in DTP/external interrupts, set the registers as follows:

1. Disable the bit associated with the DTP/interrupt enable register (ENIR).
2. Set the bit associated with the external level register (ELVR).
3. Enable the bit associated with the DTP/external interrupt request register (EIRR).
4. Enable the bit associated with the enable register.

(However, for (3) and (4), concurrent writing with word specification is enabled.)

When registers in this resource are set, the ENIR register must be placed in the disable state. The ENIR register must be cleared before the ENIR register is placed in the enable state. This processing is required to prevent an interrupt source from occurring at register setting or in the interrupt enable state.

#### ○ External interrupt request level

- If the request level is set to an edge level, at least three machine cycles are required for the pulse width to detect an edge.
- When the request input level is set to the detection level, a request to the interrupt controller remains active because an internal source hold circuit is installed, as shown Figure 15.4-1

"Clear Operation of the Source Hold Circuit at Level Setting", even if an external request input is received and then canceled. To cancel the request, the external interrupt request flag bit and then the source hold circuit must be cleared as shown in Figure 15.4-2 "Interrupt Source and Interrupt Request to the Interrupt Controller When an Interrupt Is Enabled".

Figure 15.4-1 Clear Operation of the Source Hold Circuit at Level Setting

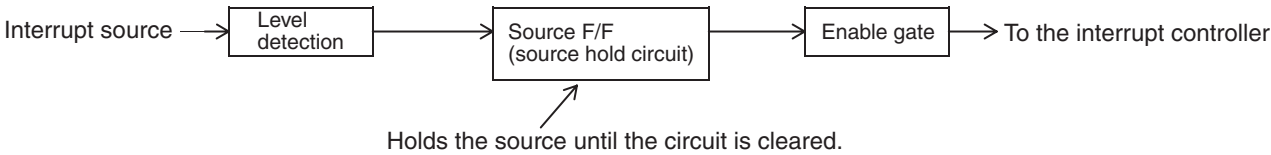
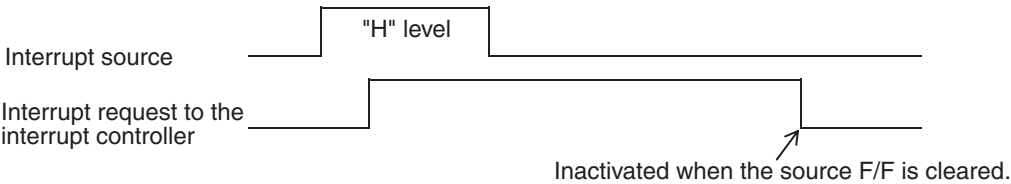


Figure 15.4-2 Interrupt Source and Interrupt Request to the Interrupt Controller When an Interrupt Is Enabled



# CHAPTER 16    DELAYED INTERRUPT REQUESTING MODULE

---

**This chapter describes the functions and operations of the delayed interrupt requesting module.**

---

16.1 "Overview of the Delayed Interrupt Requesting Module"

16.2 "Operation of the Delayed Interrupt Requesting Module"



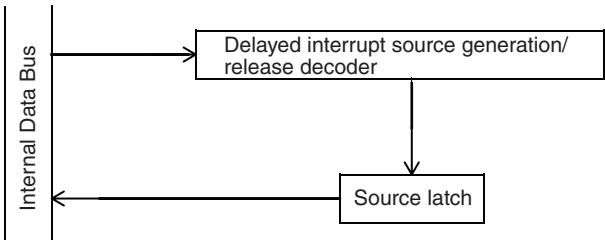
# 16.1 Overview of the Delayed Interrupt Requesting Module

The delayed interrupt requesting module is used for generating an interrupt request for task-switching. Using this module, software can be used to generate or cancel the interrupt request to the F<sup>2</sup>MC-16LX CPU.

## ■ Block Diagram of the Delayed Interrupt Requesting Module

Figure 16.1-1 "Block Diagram of the Delayed Interrupt Requesting Module" shows a block diagram of the delayed interrupt requesting module.

Figure 16.1-1 Block Diagram of the Delayed Interrupt Requesting Module



## ■ Register of the Delayed Interrupt Requesting Module

The register configuration of the delayed interrupt requesting module [delayed interrupt source generation/release register (DIRR: Delayed Interrupt Request Register)] is shown below.

The DIRR is a register that controls the generation and release of a delayed interrupt request. A delayed interrupt request is generated when 1 is written to this register, while the delayed interrupt request is released when 0 is written. The source is released at reset. Although either 0 or 1 can be written to the reserved area, it is recommended to use a set bit or clear bit command to access this register if future extension is expected.

Figure 16.1-2 Delayed Interrupt Source Generation/Release Register (DIRR)

Delayed interrupt source generation/release register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00009FH	—	—	—	—	—	—	—	R0	DIRR
Read/Write ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

## 16.2 Operation of the Delayed Interrupt Requesting Module

When the CPU writes 1 to the corresponding bit of the DIRR through software, a request latch in the delayed interrupt requesting module is set and an interrupt request is generated in the interrupt controller.

When other interrupt requests have a lower priority over this interrupt or no other interrupt request is generated, the interrupt controller generates an interrupt request for the F<sup>2</sup>MC-16LX CPU.

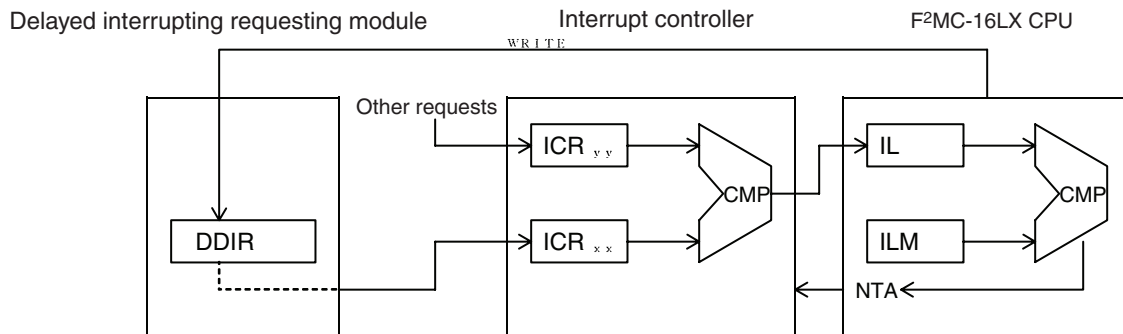
### ■ Operation of the Delayed Interrupting Requesting Module

The F<sup>2</sup>MC-16LX CPU compares the ILM bit of its internal CCR register and the interrupt request. If the interrupt level is higher than the ILM bit, the CPU starts the hardware interrupt processing microprogram upon completion of the command currently executed. As a result, the interrupt processing routine for this interrupt is executed.

This interrupt source is cleared and tasks are switched by writing 0 to the corresponding bit of the DDIR in the interrupt processing routine.

Figure 16.2-1 "Operation of the Delayed Interrupting Requesting Module" shows the operation of the delayed interrupting requesting module.

**Figure 16.2-1 Operation of the Delayed Interrupting Requesting Module**



### ■ Precaution for Using Delayed Interrupting Requesting Module

#### ○ Delayed interrupting request latch

Because this latch is set by writing 1 to the corresponding bit of the DIRR and released by writing 0 to the same bit, software must be created to clear the source in the interrupt processing routine; otherwise re-interrupt processing is started upon recovery from interrupt processing.



# CHAPTER 17    A/D CONVERTER

---

**This chapter describes the functions and operations of the A/D converter.**

---

17.1 "Overview of the A/D Converter"

17.2 "Registers of the A/D Converter"

17.3 "Operation of the A/D Converter"

17.3 "Notes on Using the A/D Converter"

17.4 "Conversion Data Protection Function"

## 17.1 Overview of the A/D Converter

---

**An A/D converter converts analog input voltage to a digital value.**

---

### ■ Overview of the A/D Converter

The A/D converter has the following characteristics:

○ **Conversion time:**

26.3  $\mu$ s

○ **Sampling time:**

64 to 4096 machine cycles per channel (4  $\mu$ s minimum to 256  $\mu$ s ) can be selected.

(The conversion time and sampling time indicate the values when the machine cycle frequency is 16 MHz.)

○ **Compare time:**

176 to 352 machine cycles per channel

(Use the 176 machine cycles of the compare time when the machine clock frequency is up to 8 MHz.)

○ **The RC successive approximation mode with a sample hold circuit is used.**

○ **A resolution of between eight and ten bits can be selected.**

○ **The analog inputs are selected among eight channels by a program.**

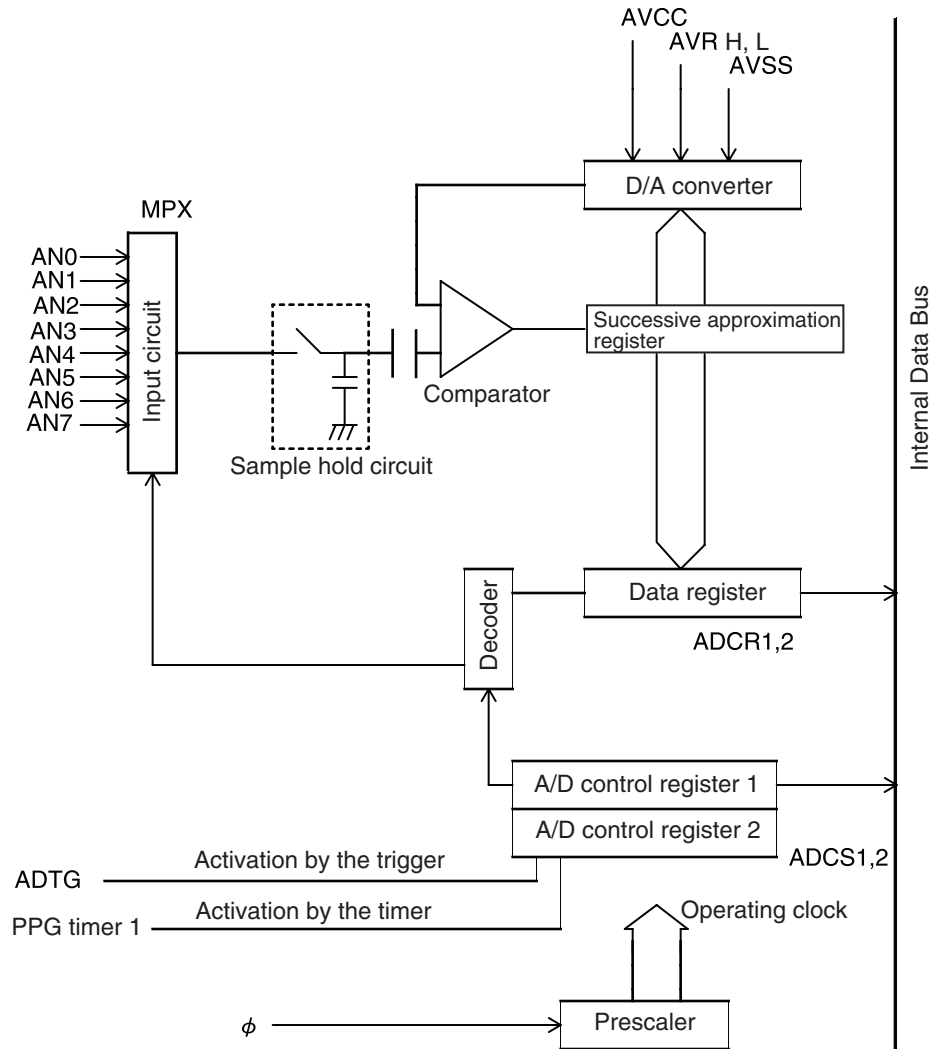
- Single conversion mode: Selects one channel and converts the analog input.
- Scan conversion mode: Converts the analog inputs from multiple contiguous channels. Up to eight channels can be programmed.
- Continuous conversion mode: Converts the analog input from the specified channel repeatedly.
- Stop and conversion mode: After converting the analog input from one channel, temporarily stops and waits for the next activation (start of conversion can be synchronized).

○ **At the end of A/D conversion, an interrupt request indicating the end of A/D conversion can be generated for CPU. EI<sup>2</sup>OS can be activated by generating this interrupt to transfer A/D conversion result data to memory. Use of interrupts is suitable for continuous processing.**

○ **The activation factor can be selected among software, the external trigger (falling edge), and the timer (rising edge).**

# ■ Block Diagram of the A/D Converter

Figure 17.1-1 Block Diagram of the A/D Converter



## 17.2 Registers of the A/D Converter

Figure 17.2-1 "Registers for A/D Converter" shows the A/D converter registers.

### ■ Registers of the A/D Converter

**Figure 17.2-1 Registers of the A/D Converter**

Higher control status register	15	14	13	12	11	10	9	8	⇔ Bit number
Address:000037 <sub>H</sub>	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS2
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)	
Lower control status register	7	6	5	4	3	2	1	0	⇔ Bit number
Address:000036 <sub>H</sub>	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS1
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)	
Higher data register	15	14	13	12	11	10	9	8	⇔ Bit number
Address:000039 <sub>H</sub>	DSEL	ST1	ST0	CT1	CT0	—	D9	D8	ADCR2
Read/write ⇔	(W)	(W)	(W)	(W)	(W)	(-)	(-)	(-)	
Initial value ⇔	(0)	(0)	(0)	(0)	(1)	(-)	(X)	(X)	
Lower data register	7	6	5	4	3	2	1	0	⇔ Bit number
Address:000038 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	ADCR1
Read/write ⇔	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇔	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

## 17.2.1 Control Status Register (ADCS1, ADCS2)

The control status register (ADCS1, ADCS2) controls the A/D converter and indicates the status.

### ■ ADCS1 and ADCS2 (Control Status Registers)

**Figure 17.2-2 ADCS1 and ADCS2 (Control Status Register)**

Higher control status register	15	14	13	12	11	10	9	8	⇐ Bit number
Address:000037 <sub>H</sub>	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS2
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)	
Lower control status register	7	6	5	4	3	2	1	0	⇐ Bit number
Address:000036 <sub>H</sub>	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS1
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)	

**Note:**

Do not rewrite ADCS1 during A/D conversion.

**[Bit 15] Busy flag and stop (BUSY)**

This bit indicates respectively controls A/D converter operations.

During the read operation: When this bit is "0", this indicates termination of A/D conversion, if it is "1", A/D conversion is executed.

During the write operation: When this bit is set to "0" during A/D operation, the A/D operation is forcibly terminated. This can be used for forcible termination in continuous or stop mode. However, this indicator bit cannot be set to "1".

Executing an RMW instruction enables the reading of "1". In single mode, this bit is cleared when A/D conversion ends. In continuous or stop mode, the bit is not cleared until the operation is stopped by setting the bit to "0".

**Note:**

Do not perform the forced stop simultaneously with the soft activation (BUSY is set to "0" while STRT is set to "1").

**[Bit 14] Interrupt (INT)**

When conversion data is written to the data register (ADCR), this bit is set to "1".

When the INTE bit is "1", setting this bit to "1" generates an interrupt request. If EI<sup>2</sup>OS activation is enabled, EI<sup>2</sup>OS is activated, and setting this bit to "1" has no effect.

This bit is cleared by writing "0" or by the EI<sup>2</sup>OS interrupt clear signal.

**Note:**

Clear this bit by writing "0" when the A/D converter stops.



**[Bit 13] Interrupt enable (INTE)**

This bit specifies whether to enable or disable an interrupt at the end of conversion.

When EI<sup>2</sup>OS is to be used, set this bit to "1".

EI<sup>2</sup>OS is designed to be activated when an interrupt request is generated.

**Table 17.2-1 Function of INTE Bit (Interrupt Enabled/Disabled Specification Bit)**

INTE	Function
0	Disables the interrupt. [Initial value]
1	Enables the interrupt.

**[Bit 12] A/D converter pause (PAUS)**

This bit is set to "1" when A/D conversion stops temporarily.

Because there is only one register for storing the A/D conversion result, if the conversion result is not transferred using EI<sup>2</sup>OS, the previous data is destroyed.

To avoid losing such data, the A/D converter is designed so that the next conversion data is not stored unless the contents of the data register are transferred using EI<sup>2</sup>OS. In this status, A/D conversion stops.

When the transfer using EI<sup>2</sup>OS is complete, the A/D converter restarts conversion.

**Note:**

This bit is valid only when EI<sup>2</sup>OS is used. See the section 17.5 "Conversion Data Protection Function".

**[Bits 11 and 10] Start source select (STS1 and STS0)**

The A/D activation factor is selected according to the setting of these bits.

**Table 17.2-2 Functions of the Bits STS1 and STS0 (A/D Activation Source Selection Bits)**

STS1	STS0	Function
0	0	Activated by software. [Initial value]
0	1	Activated by the external pin trigger and software.
1	0	Activated by the timer and software.
1	1	Activated by the external pin trigger, timer, and software.

In the mode in which there are multiple activation factors, the A/D converter is activated by the first factor that occurs. The activation factor is changed at the same time that these bits are rewritten. When these bits are required to be rewritten during A/D converter operation, rewrite the bits when no target conversion activation factor occurs.

For the external pin trigger, the falling edge is detected.

If the external trigger input level is "L", rewriting these bits to set activation by the external pin trigger may activate the A/D converter.

When the timer is selected, the output of the 16-bit reload timer 1 is used.

**[Bit 9] Start (STRT)**

Writing "1" in this bit activates the A/D converter.

To reactivate the A/D converter, write "1" in this bit again.

In the stop mode, the A/D converter is not reactivated for operational reasons.

**Note:**

Do not perform the forced stop simultaneously with the soft activation (BUSY is set to 0 while STRT is set to 1).

**[Bit 8]**

Reserved bit. Write "0" in this bit.

**[Bits 7 and 6] A/D converter mode set (MD1 and MD0)**

These bits set the operating mode.

**Table 17.2-3 MD1, MD0 Operating Mode**

MD1	MD0	Operating mode
0	0	Single mode. Reactivation during operation is always enabled. [Initial value]
0	1	Single mode. Reactivation during operation is disabled.
1	0	Continuous mode. Reactivation during operation is disabled.
1	1	Stop mode. Reactivation during operation is disabled.

Single mode: Continues A/D conversion starting with the channel set by ANS2 to ANS0 and ending with the channel set by ANE2 to ANE0, and stops when a single A/D conversion is completed.

Continuous mode: Repeats A/D conversion starting with the channel set by ANS2 to ANS0 and ending with the channel set by ANE2 to ANE0.

Stop mode: Performs A/D conversion starting with the channel set by ANS2 to ANS0 and ending with the channel set by ANE2 to ANE0 and temporarily stops when A/D conversion for each channel is completed. Conversion is restarted when an activation factor occurs.

**Note:**

When A/D conversion is activated in the continuous or stop mode, it continues until stopped by the BUSY bit.

Writing "0" in the BUSY bit stops A/D conversion.

In the mode in which reactivation during operation is disabled (single, continuous, or stop mode), the A/D converter cannot be activated by an activation factor (timer, external trigger, or software).

**[Bits 5, 4, and 3] Analog start channel set (ANS2, ANS1, and ANS0)**

Use this bit group to set the A/D conversion start channel.

When the A/D converter is activated, A/D conversion starts with the channel selected by these bits.

When these bits are read, the channel from which the analog input is being converted is indicated during A/D conversion. When the A/D converter stops in the stop mode, the read

value is the channel from which the analog input was previously converted.

**Table 17.2-4 Start Channel (ANS2, ANS1 and ANS0)**

ANS2	ANS1	ANS0	Start channel
0	0	0	AN0 [Initial value]
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

**[Bits 2, 1, and 0] Analog end channel set (ANE2, ANE1, and ANE0)**

Use this bit group to set the A/D conversion end channel.

**Table 17.2-5 End Channel (ANE2, ANE1 and ANE0)**

ANE2	ANE1	ANE0	End channel
0	0	0	AN0 [Initial value]
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

When the channel set by ANS2 to ANS0 is set, the A/D converter converts the analog input from one channel (single conversion).

In the continuous or stop mode, when the A/D converter completes conversion for the channel set by this bit group, it returns to conversion for the start channel set by ANS2 to ANS0.

When the channel number set by ANS is greater than the channel number set by ANE, conversion starts with the channel set by ANS. When the A/D converter completes conversion for channel 7, it returns to conversion for channel 0. The A/D converter then continues conversion until the analog input has been converted from the channel set by ANE.

Example:

When ANS sets channel 6 and ANE sets channel 3 in the single mode

Operation: Conversion channels 6ch → 7ch → 0ch → 1ch → 2ch → 3ch

## 17.2.2 Data Register (ADCR1, ADCR2)

The data register (ADCR1, ADCR2) stores the A/D conversion result, selects the resolution for A/D conversion, and sets the machine cycle.

### ■ ADCR2 and ADCR1 (data register)

**Figure 17.2-3 ADCR2 and ADCR1 (Data Register)**

Higher data register	15	14	13	12	11	10	9	8	⇔ Bit number
Address:000039 <sub>H</sub>	DSEL	ST1	ST0	CT1	CT0	—	D9	D8	ADCR2
Read/write ⇔	(W)	(W)	(W)	(W)	(W)	(—)	(R)	(R)	
Initial value ⇔	(0)	(0)	(0)	(0)	(1)	(—)	(X)	(X)	
Lower data register	7	6	5	4	3	2	1	0	⇔ Bit number
Address:000038 <sub>H</sub>	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	ADCR1
Read/write ⇔	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇔	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

**Note:**

For ADCR2, the read value is undefined.

**[Bit 15] DSEL**

This bit selects the 8/10-bit mode resolution.

**Table 17.2-6 SELB (Resolution Selection Bit)**

SELB	Function
0	10-bit mode
1	8-bit mode

**[Bits 14 and 13] Sampling time (ST1 and ST0)**

Use these bits to set the sampling machine cycle count.

**Table 17.2-7 ST1 and ST0 (Sampling Machine Cycle Count Setting Bit)**

ST1	ST0	Sampling machine cycle count	Sampling time
0	0	64 machine cycles	4 μs when the machine clock frequency is 16 MHz
0	1	256 machine cycles	16 μs when the machine clock frequency is 16 MHz
1	0	1024 machine cycles	64 μs when the machine clock frequency is 16 MHz
1	1	4096 machine cycles	256 μs when the machine clock frequency is 16 MHz

**[Bits 12 and 11] Compare time (CT1 and CT0)**

Use these bits to set the compare machine cycle count.

**Table 17.2-8 CT1 and CT0 (Compare Machine Cycle Count Setting Bit)**

CT1	CT0	Compare machine cycle count	Compare time
0	0	176 machine cycles	22 $\mu$ s when the machine clock frequency is 8 MHz
0	1	352 machine cycles	22 $\mu$ s when the machine clock frequency is 16 MHz
1	0	704 machine cycles	44 $\mu$ s when the machine clock frequency is 16 MHz
1	1	1408 machine cycles	88 $\mu$ s when the machine clock frequency is 16 MHz

**Note:**

Set these bits to "00" only when the machine clock frequency is up to 8 MHz.

If the machine clock is faster than 8 MHz, conversion accuracy cannot be guaranteed.

**[Bits 9 and 8] D9 and D8**

These bits are valid when DSEL is 0. The bits store the two higher bits of the conversion result value.

**[Bits 7 to 0] D7 to D0**

These bits make up an A/D conversion storage register containing the digital value indicating the conversion result.

The value of this register is updated when conversion is complete. Normally, this register contains the last conversion value. See section 17.5 "Conversion Data Protection Function".

The value of this register is undefined at a reset.

**Note:**

Do not write data in this register during A/D operation.

## 17.3 Operation of the A/D Converter

---

The A/D converter is operated in one of the following three modes:

- Single mode
  - Continuous mode
  - Stop mode
- 

### ■ Explanation of Operation

The A/D converter operates in the successive approximation mode. The resolution can be switched between 8 and 10 bits.

Because this A/D converter has only one register (8 bits) for storing the conversion result, the conversion data register (ADCR0) is updated when conversion is complete.

For this reason, the A/D converter alone cannot be used for continuous conversion processing. Conversion with conversion data transferred to memory using the EI<sup>2</sup>OS function is recommended.

### ■ Single Mode

In this mode, the A/D converter sequentially converts the analog inputs set by the ANS and ANE bits. When the A/D converter has converted the analog input from the end channel set by the ANE bit, operation stops.

When the same channel is specified for the start and end channels (when ANS is equal to ANE), the A/D converter converts only the analog input from the channel specified by ANS.

Example:

ANS=000, ANE=011

Start → AN0 → AN1 → AN2 → AN3 → End

ANS=010, ANE=010

Start → AN2 → End

### ■ Continuous Mode

In this mode, the A/D converter sequentially converts the analog inputs set by the ANS and ANE bits. When the A/D converter has converted the analog input from the end channel set by the ANE bit, it returns to the analog input set by ANS and continues A/D conversion.

When the same channel is specified for the start and end channels (when ANS is equal to ANE), the A/D converter continues converting only the analog input from the channel specified by ANS.

Example:

ANS=000, ANE=011

Start → AN0 → AN1 → AN2 → AN3 → AN0 ----→ Repeat

ANS=010, ANE=010

Start → AN2 → AN2 → AN2 ----→ Repeat

In the continuous mode, the A/D converter repeats conversion until "0" is written in the BUSY bit (writing "0" in the BUSY bit ==> stops operation).

Note that a forced operation stop also stops the analog input conversion (at a forced operation stop, the conversion register contains the previous data for which conversion is complete).

### ■ Stop Mode

In this mode, the A/D converter sequentially converts the analog inputs set by the ANS and ANE bits, temporarily stopping when the analog input from each channel is converted. To release the temporary stop state, the A/D converter must be activated again.

When the A/D converter has converted the analog input from the end channel set by the ANE bit, it returns to the analog input set by ANS and continues A/D conversion.

When the same channel is specified for the start and end channels (when ANS is equal to ANE), the A/D converter converts the analog input from the channel specified by ANS.

Example:

ANS=000, ANE=011

Start → AN0 → Stop → Activation → AN1 → Stop → Activation → AN2 → Stop → Activation → AN3 → Stop → Activation → AN0 ---→ Repeat

ANS=010, ANE=010

Start → AN2 → Stop → Activation → AN2 → Stop → Activation → AN2 ---→ Repeat

Only the activation factor(s) set by STS1 and STS0 can be used.

A conversion start can be synchronized using this mode.



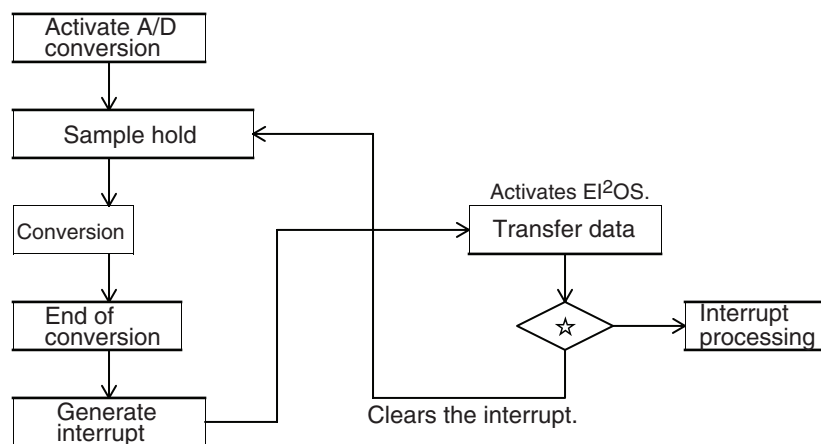
## 17.3.1 Conversion Using EI<sup>2</sup>OS

The A/D converter can transfer the A/D conversion result to memory using the extended intelligent I/O service (EI<sup>2</sup>OS).

### ■ Conversion Using EI<sup>2</sup>OS

When EI<sup>2</sup>OS is used, the conversion data protection function can safely transfer two or more data items to memory, even during continuous conversion.

**Figure 17.3-1 Sample Flow from Activation of A/D Conversion to Transfer of Conversion Data (Continuous Mode)**



Processing marked with ☆ depends on EI<sup>2</sup>OS settings.

## 17.3.2 Example of Activating EI<sup>2</sup>OS in the Single Mode

In single mode, EI<sup>2</sup>OS is activated as follows:

- Converts analog inputs AN1 to AN3 and terminates conversion.
- Sequentially transfers conversion data to addresses 200<sub>H</sub> to 206<sub>H</sub>.
- Activates conversion by software.
- Uses the highest interrupt level.

### ■ Example of Activating EI<sup>2</sup>OS in the Single Mode

Table 17.3-1 Example of Activating EI<sup>2</sup>OS in the Single Mode

Setting item	Program example	Explanation of operation
EI <sup>2</sup> OS setting	MOV ICR0, #08H	Sets the highest interrupt level, activates EI <sup>2</sup> OS when an interrupt is generated, and sets the descriptor address.
	MOV BAPL, #00H	Conversion data destination addresses.
	MOV BAPM, #02H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Transfers word data. Increments the destination address after transfer. Transfers data from I/O to memory.
	MOV IOAL, #3EH	Sets the A/D converter result registers.
	MOV IOAH, #00H	
	MOV DCTL, #03H	Performs EI <sup>2</sup> OS transfer three times (the number of conversion times).
	MOV DCTH, #00H	
A/D converter settings	MOV ADCS1, #0BH	Sets the single mode, the start channel to AN1, and the end channel to AN3.
	MOV ADCS2, #A2H	Activates A/D conversion by software.
Other processing	:	:
EI <sup>2</sup> OS end and interrupt sequence	MOV ADCS2, #80H	-
	RETI	Returns from the interrupt.

ICR3: Interrupt control register

BAPL: Lower buffer address pointer

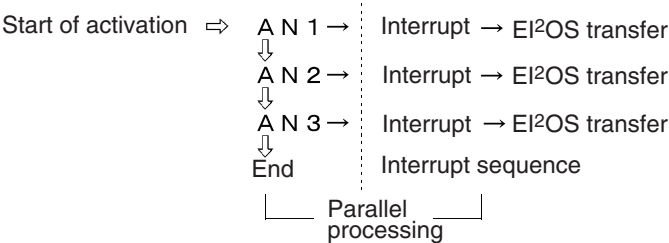
BAPM: Middle buffer address pointer

BAPH: Higher buffer address pointer

ISCS: EI<sup>2</sup>OS status register

IOAL: Lower I/O address register  
IOAH: Higher I/O address register  
DCTL: Lower data counter  
DCTH: Higher data counter

Figure 17.3-2 Example of Activating EI<sup>2</sup>OS in the Continuous Mode



### 17.3.3 Example of Activating EI<sup>2</sup>OS in the Continuous Mode

In continuous mode, EI<sup>2</sup>OS is activated as follows:

- Converts analog inputs AN3 to AN5 and obtains two conversion data items for each channel.
- Sequentially transfers conversion data to addresses 600H to 60CH.
- Activates conversion by the external edge input.
- Uses the highest interrupt level.

#### ■ Example of Activating EI<sup>2</sup>OS in the Continuous Mode

Table 17.3-2 Example of Activating EI<sup>2</sup>OS in the Continuous Mode

Setting item	Program example	Explanation of operation
EI <sup>2</sup> OS setting	MOV ICR0, #08H	Sets the highest interrupt level, activates EI <sup>2</sup> OS when an interrupt is generated, and sets the descriptor address.
	MOV BAPL, #00H	Conversion data destination addresses.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Transfers word data. Increments the destination address after transfer. Transfers data from I/O to memory. Terminates processing in response to a request by the peripheral.
	MOV IOAL, #3EH	Source address.
	MOV IOAH, #00H	
	MOV DCTL, #06H	Performs EI <sup>2</sup> OS transfer six times (twice for each of the three channels).
	MOV DCTH, #00H	
A/D converter settings	MOV ADCS1, #9DH	Sets the continuous mode, the start channel to AN3, and the end channel to AN5.
	MOV ADCS2, #A4H	Activates A/D conversion by external edge input.
Other processing	:	:
EI <sup>2</sup> OS end and interrupt sequence	MOV ADCS2, #80H	-
	RETI	Returns from the interrupt.

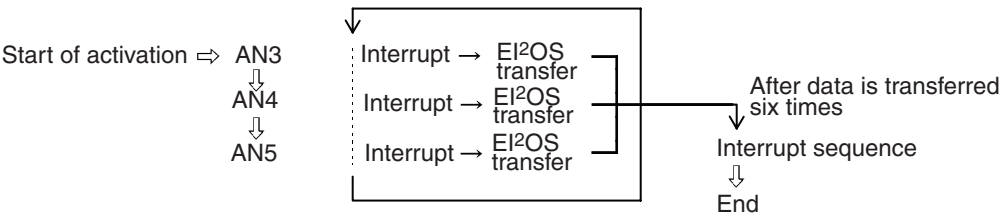
ICR3: Interrupt control register

BAPL: Lower buffer address pointer

BAPM: Middle buffer address pointer

- BAPH: Higher buffer address pointer
- ISCS: EI<sup>2</sup>OS status register
- IOAL: Lower I/O address register
- IOAH: Higher I/O address register
- DCTL: Lower data counter
- DCTH: Higher data counter

Figure 17.3-3 Example of Activating EI<sup>2</sup>OS in the Continuous Mode



## 17.3.4 Example of Activating EI<sup>2</sup>OS in the Stop Mode

In stop mode, EI<sup>2</sup>OS is activated as follows:

- Converts analog input AN3 12 times at regular intervals.
- Sequentially transfers conversion data to addresses 600H to 618H.
- Activates conversion by the external edge input.
- Uses the highest interrupt level.

### ■ Example of Activating EI<sup>2</sup>OS in the Stop Mode

Table 17.3-3 Example of Activating EI<sup>2</sup>OS in the Stop Mode

Setting item	Program example	Explanation of operation
EI <sup>2</sup> OS setting	MOV ICR0, #08H	Sets the highest interrupt level, activates EI <sup>2</sup> OS when an interrupt is generated, and sets the descriptor address.
	MOV BAPL, #00H	Conversion data destination addresses.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #19H	Transfers word data. Increments the destination address after transfer. Transfers data from I/O to memory. Terminates processing in response to a request by the peripheral.
	MOV IOAL, #3EH	Source address.
	MOV IOAH, #00H	
	MOV DCTL, #0CH	Performs EI <sup>2</sup> OS transfer 12 times.
	MOV DCTH, #00H	
A/D converter settings	MOV ADCS1, #DBH	Sets the stop mode, the start channel to AN3, and the end channel to AN3 (one channel conversion).
	MOV ADCS2, #A4H	Activates A/D conversion by external edge input.
Other processing	:	:
EI <sup>2</sup> OS end and interrupt sequence	MOV ADCS2, #80H	-
	RETI	Returns from the interrupt.

ICR3: Interrupt control register

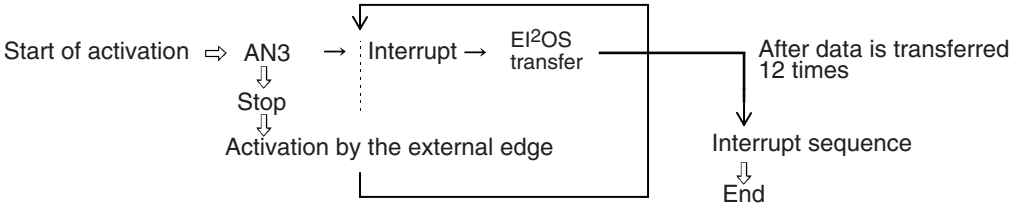
BAPL: Lower buffer address pointer

BAPM: Middle buffer address pointer

BAPH: Higher buffer address pointer

ISCS: EI<sup>2</sup>OS status register  
IOAL: Lower I/O address register  
IOAH: Higher I/O address register  
DCTL: Lower data counter  
DCTH: Higher data counter

Figure 17.3-4 Example of Activating EI<sup>2</sup>OS in the Stop Mode



## 17.4 Notes on Using the A/D Converter

---

This section describes notes on use of A/D converters.

---

### ■ Notes on Using the A/D Converter

To activate the A/D converter using the external trigger or internal timer, A/D activation factor bits STS1 and STS0 in the ADCS2 register are used. Set these bits in a status in which the input value of the external trigger or internal timer is inactive. If the input value is active, the A/D converter may start operation.

Set STS1 and STS0 in a status in which "1" is input to ADTG and "0" is output from the internal timer (16-bit reload timer).

Always set the ADER bit corresponding to a pin for use as an analog input to "1".

For details, see CHAPTER 8 "I/O PORT".



## 17.5 Conversion Data Protection Function

This A/D converter has a function to protect the converted data. It enables continuous conversion and acquisition of multiple data items using EI<sup>2</sup>OS.

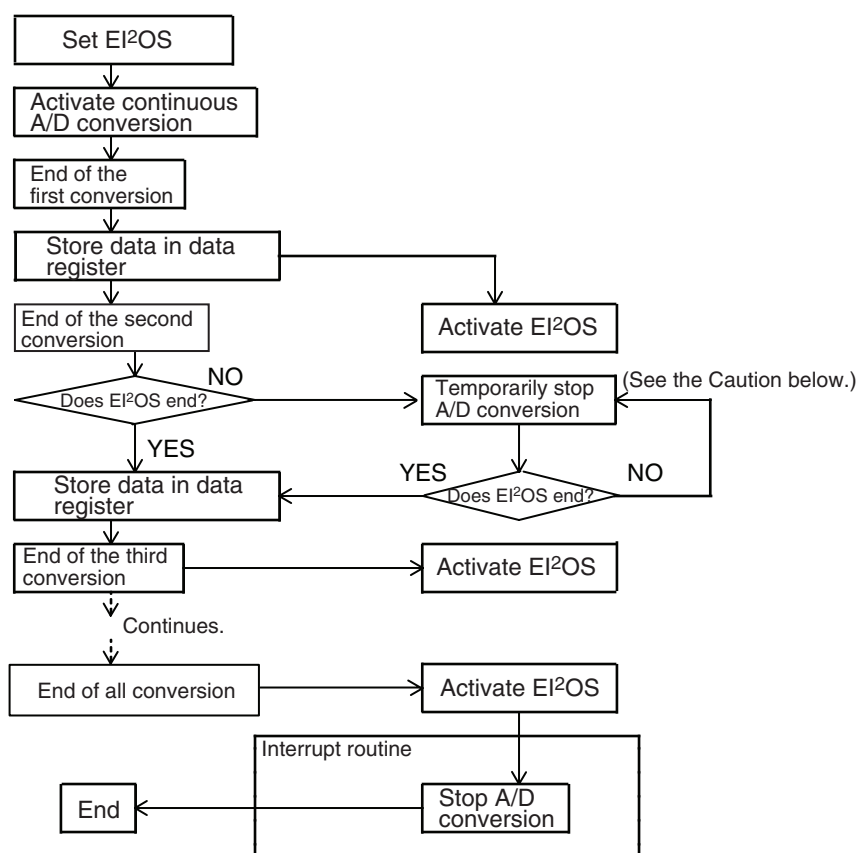
### ■ Conversion Data Protection Function

Because there is only one conversion data register, conversion data is stored at the same time one conversion is complete and previous data is lost by continuous A/D conversion. To prevent losing such data, the A/D converter has a function that does not store the next conversion data in the register unless the previous data is transferred using EI<sup>2</sup>OS and then stops A/D conversion temporarily.

The temporary stop state is released after data is transferred to memory using EI<sup>2</sup>OS.

The A/D converter converts data continuously so long as the previous data is transferred.

**Figure 17.5-1 Data Protection Function Flow (when EI<sup>2</sup>OS Is Used)**



**Note:**

- This function relates to the INT and INTE bits in ADCS2.
- The data protection function is designed to operate in the interrupt enable state only (when INTE is "1").
- In the interrupt disable state (when INTE is "0"), this function does not operate. Continuous

A/D conversion sequentially stores conversion data in the register, and old data is lost.

- When the A/D converter is stopped temporarily during EI<sup>2</sup>OS operation, the A/D converter starts if an interrupt is disabled, thereby ensuring that new data can be written before old data is transferred.
- Reactivation in the temporary stop state destroys wait data.



# CHAPTER 18 D/A CONVERTER

---

**This chapter describes the functions and operations of the D/A converter.**

---

18.1 "Overview of the D/A Converter"

18.2 "Registers of the D/A Converter"

18.3 "Operation of the D/A Converter"

## 18.1 Overview of the D/A Converter

The block described below is a D/A converter in the R-2R mode with a resolution of eight bits.

This microcontroller has two internal D/A converter channels. The outputs of the two channels can be controlled independently using corresponding D/A control registers.

### ■ D/A Converter Registers

The D/A converter register is shown below.

**Figure 18.1-1 D/A Converter Registers**

D/A converter data register 1

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00003BH	DA17	DA16	DA15	DA14	DA13	DA12	DA11	DA10	DADR1
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)	

D/A converter data register 0

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 00003AH	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00	DADR0
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)	

D/A control register 1

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00003DH	—	—	—	—	—	—	—	DAE1	DACR1
Read/Write ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

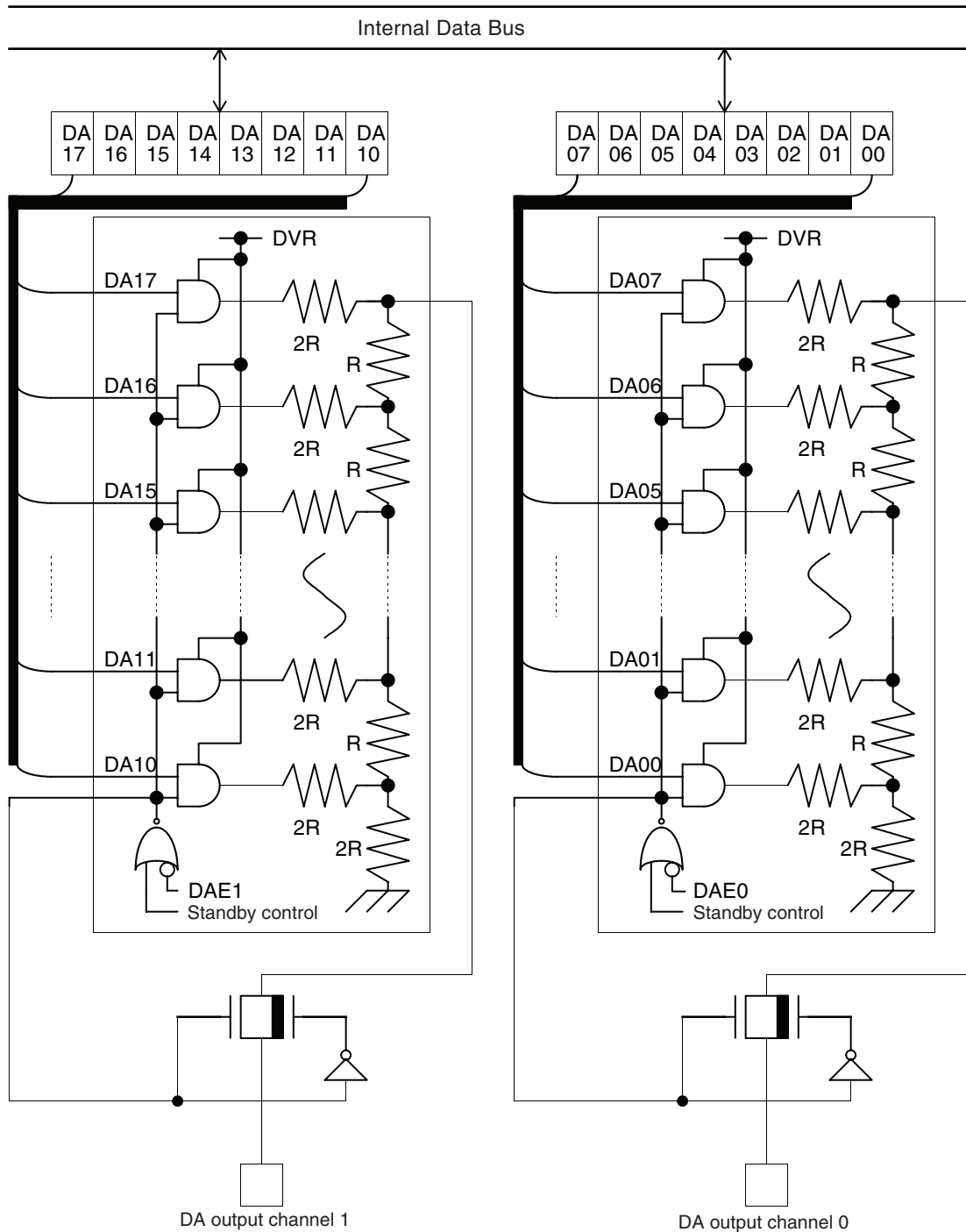
D/A control register 0

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 00003CH	—	—	—	—	—	—	—	DAE0	DACR0
Read/Write ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

### ■ Block Diagram of the D/A Converter

Figure 18.1-2 "Block Diagram of the D/A Converter" is a block diagram of the D/A converter.

**Figure 18.1-2 Block Diagram of the D/A Converter**



## 18.2 Registers of the D/A Converter

The two types of D/A converter registers are as follows:

- D/A converter registers (DACR0 and DACR1)
- D/A control registers (DADR0 and DADR1)

### ■ D/A Converter Registers (DACR0 and DACR1)

The D/A converter data register (DADR0, DADR1) configuration is shown in Figure 18.2-1 "D/A Converter Data Register (DADR)".

**Figure 18.2-1 D/A Converter Data Register (DADR)**

D/A converter data register 1

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00003B <sub>H</sub>	DA17	DA16	DA15	DA14	DA13	DA12	DA11	DA10	DADR1
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)	

D/A converter data register 0

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 00003A <sub>H</sub>	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00	DADR0
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)	

#### [Bits 15 to 8] DA17 to DA10

These bits, which are read/write bits that are not initialized by a reset, set the output voltage from channel 1 in the D/A converter.

#### [Bits 7 to 0] DA07 to DA00

These bits, which are read/write bits that are not initialized by a reset, set the output voltage from channel 0 in the D/A converter.

### ■ D/A Data Registers (DACR0 and DACR1)

The D/A control register (DACR0/1) configuration is shown in Figure 18.2-2 "D/A Control Data Register (DACR0/1)".

**Figure 18.2-2 D/A Control Data Register (DACR0/1)**

D/A control register 1

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00003D <sub>H</sub>	—	—	—	—	—	—	—	DAE1	DACR1
Read/Write ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

D/A control register 0

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 00003C <sub>H</sub>	—	—	—	—	—	—	—	DAE0	DACR0
Read/Write ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

### **[Bits 8, 0] DAE1, DAE0**

These bits enable or disable D/A converter output. DAE1 controls channel 1, and DAE0 controls channel 0.

Setting these bits to "1" enables D/A output. Setting these bits to "0" disables D/A output.

These bits are read/write bits that are initialized by a reset.



## 18.3 Operation of the D/A Converter

---

For starting D/A output, the enabling bit corresponding to the D/A output channel, which is located in the D/A control register (DACR) must be set to 1.

---

### ■ Operation of the D/A Converter

Disabling D/A output turns off the analog switch serially inserted into the output block of each D/A converter channel. The circuit in the D/A converter is also cleared to the "0" output state and the path over which direct current flows is cut off. This status also occurs in the stop mode.

The output voltage of the D/A converter falls within the range from 0V to 255 respectively 256V times DVR. Adjusting the DVR voltage externally can change the output voltage range.

No built-in buffer amplifier is mounted for D/A converter output, and because the analog switch (nearly equal to 100Ω) is inserted serially to the output, sufficient settling time is required for putting an external load on the output.

Table 18.3-1 "Theoretical Output Voltage Values of the D/A Converter" lists the theoretical output voltage values of the D/A converter.

**Table 18.3-1 Theoretical Output Voltage Values of the D/A Converter**

Setting of DA07 to DA00 and DA17 to DA10	Theoretical output voltage value
00 <sub>H</sub>	$0/256 \times \text{DVR} (= 0\text{V})$
01 <sub>H</sub>	$1/256 \times \text{DVR}$
02 <sub>H</sub>	$2/256 \times \text{DVR}$
to	to
FD <sub>H</sub>	$253/256 \times \text{DVR}$
FE <sub>H</sub>	$254/256 \times \text{DVR}$
FF <sub>H</sub>	$255/256 \times \text{DVR}$

# CHAPTER 19    UART

---

**This chapter describes the functions and operations of the UART.**

---

- 19.1 "Overview of the UART"
- 19.2 "Block Diagram of the UART"
- 19.3 "Registers of the UART"
- 19.4 "UART Baud Rates"
- 19.5 "Operation of the UART"
- 19.6 "Flags and Interrupt Sources of the UART"
- 19.7 "Applications of the UART and Precautions"

# 19.1 Overview of the UART

The UART is a serial I/O port used for asynchronous (start-stop synchronous) and CLK synchronous communications.

■ Features of the UART

Features of the UART are shown below.

- Full-duplex double buffer
- Can be used for asynchronous (start-stop synchronous) communication and CLK synchronous communication.
- Multi-processor mode is supported.
- Dedicated baud rate generator is incorporated.

Table 19.1-1 Baud Rate

Operation	Baud rate*
Asynchronous communication:	31250/ 9615/ 4808/ 2404/ 1202 bps
CLK synchronous communication:	2M/ 1M/ 500K/ 250K/ 125K/ 62.5K bps

\*: Internal machine clock: At 6, 8, 10, 12, or 16 MHz

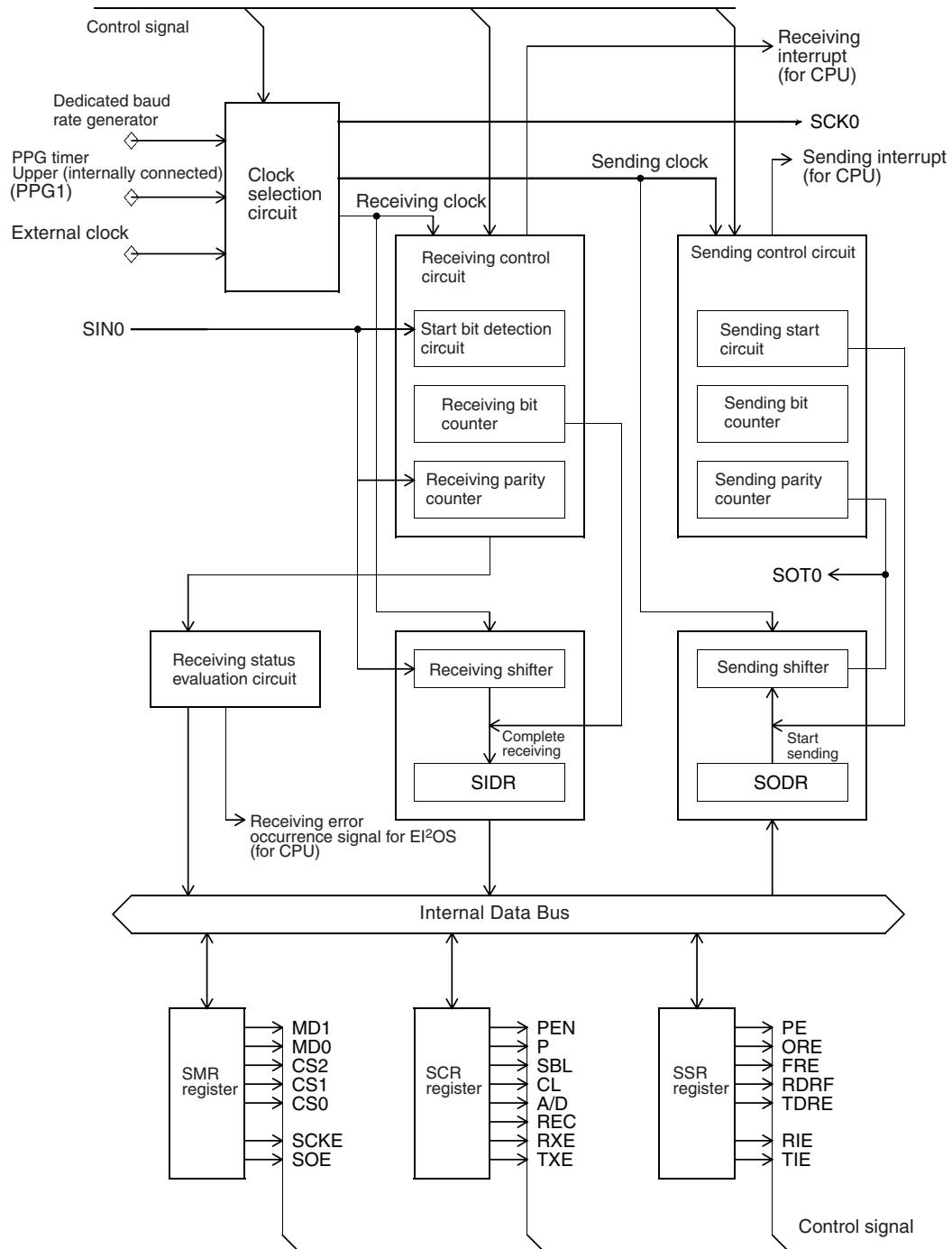
- Any baud rate can be set using an external clock.
- Error detecting function (parity, flaming, overrun)
- NRZ code is used as a transfer signal.
- Extended intelligent I/O service (EI<sup>2</sup>OS) is supported.

## 19.2 Block Diagram of the UART

Figure 19.2-1 "Block Diagram of the UART" shows the block diagram of the UART.

### ■ Block Diagram of the UART

Figure 19.2-1 Block Diagram of the UART



- **Serial mode register (SMR)**
- **Serial control register (SCR)**
- **Serial input data register (SIDR)/ Serial output data register (SODR)**
- **Serial status register (SSR)**
- **Communication prescaler control register (CDCR)**

## ■ Registers of the UART

15	8	7	0	
—		CDCR		(R/W)
SCR		SMR		(R/W)
SSR		SIDR(R)/SODR(W)		(R/W)

Address:	000020 <sub>H</sub>																
	000024 <sub>H</sub>	<table border="1" style="display: inline-table;"><tr><td>MD1</td><td>MD0</td><td>CS2</td><td>CS1</td><td>CS0</td><td>Reserved</td><td>SCKE</td><td>SOE</td></tr></table>							MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE	SMR
MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE										
Read/Wtite⇒		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)									
Initial value⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)									

Address: 000021 <sub>H</sub> 000025 <sub>H</sub>	PEN	P	SBL	CL	A/D	REC	RXE	TXE	SCR
Read/Write ⇒ Initial value ⇒	(R/W) (0)	(R/W) (0)	(R/W) (0)	(R/W) (0)	(R/W) (0)	(W) (1)	(R/W) (0)	(R/W) (0)	

Address: 000022<sub>H</sub> ..... 7 6 5 4 3 2 1 0 ⇐ Bit No.  
 000026<sub>H</sub> .....  
 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)  
 SODR(write)

Address: 000023 <sub>H</sub> 000027 <sub>H</sub>	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE	SSR
Read/Write Initial value	(R) (0)	(R) (0)	(R) (0)	(R) (0)	(R) (1)	(-) (-)	(R/W) (0)	(R/W) (0)	

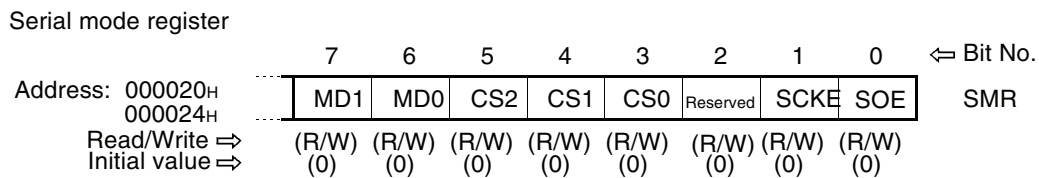
Address: 000028 <sub>H</sub> 00002A <sub>H</sub>	MD	-	-	-	DIV3	DIV2	DIV1	DIV0	CDCR
Read/Write ⇔	(R/W)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇔	(0)	(-)	(-)	(-)	(1)	(1)	(1)	(1)	

## 19.3.1 Serial Mode Register (SMR)

The serial mode register (SMR) specifies the operation mode of the UART. Set the operation mode when the register is not in operation. Also, do not write data in this register when in operation.

### ■ Serial Mode Register (SMR)

**Figure 19.3-2 Configuration of the Serial Mode Register (SMR)**



#### [bit 7, 6] MD1, MD0 (MoDe Select):

The operation mode of the UART is selected at these bits.

**Table 19.3-1 Operation Mode Select Setting**

Mode	MD1	MD0	Operation mode
0	0	0	Asynchronous (start-stop synchronous) Normal mode [Initial value]
1	0	1	Asynchronous (start-stop synchronous) Multiprocessor mode
2	1	0	CLK synchronous mode
—	1	1	Setting prohibited

#### Note:

CLK asynchronous mode (multiprocessor), represented as mode 1, is used when multiple slave CPUs are connected to the host CPU. In this resource, the data format of the receiving data cannot be identified. Thus it supports only the master in the multiprocessor mode.

Also, set 0 in the PEN of the SCR register because the parity checking function cannot be used.

#### [bit 5 to 3] CS2, CS1, CS0 (Clock Select):

Baud rate clock source is selected at these bits. When the dedicated baud rate generator is selected, a baud rate is selected at the same time.

**Table 19.3-2 Clock Input Select Setting**

CS2	CS1	CS0	Clock input
000 <sub>B</sub> to 100 <sub>B</sub>			Dedicated baud rate generator
1	0	1	Reserved
1	1	0	Internal timer (16-bit reload timer 0)
1	1	1	External clock

**Note:**

When the internal timer is selected, 16-bit reload timer 0 output is selected for MB90570 Series.

**[bit 2] Reserved bit**

0 must always be written.

**[bit 1] SCKE (SCLK Enable):**

When communicating in CLK synchronous mode (mode 2), specify whether the SCK0 pin is used as a clock input pin or clock output pin.

In CLK asynchronous mode or external clock mode, set the bit to 0.

When in CLK asynchronous or external clock mode, set the bit to 0 to use the pin as a general-purpose port pin.

**Table 19.3-3 SCKE (SCLK Enable) Bit Function**

SCKE	Function
0	Functions as a clock input pin. [Initial value]
1	Functions as a clock output pin.

**Note:**

To use the SCK0 pin as a clock input pin, the external clock source must be selected.

**[bit 0] SOE (Serial Output Enable):**

Specify whether the external pin (SOT0) that is also used as a general-purpose I/O port pin is used as a serial output pin or I/O port pin.

**Table 19.3-4 SOE (Serial Output Enable) Bit Function**

SOE	Function
0	Functions as a general-purpose I/O port pin. [Initial value]
1	Functions as a serial data output pin (SOT0).

## 19.3.2 Serial Control Register (SCR)

The Serial Control Register (SCR) controls the transfer protocol during the serial communication.

### ■ Serial Control Register (SCR)

**Figure 19.3-3 Configuration of the Serial Control Register (SCR)**

Serial control register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000021 <sub>H</sub> 000025 <sub>H</sub>	PEN	P	SBL	CL	A/D	REC	RXE	TXE	SCR
Read/Write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	

#### [bit 15] PEN (Parity Enable):

In the serial communication, specify whether to add a parity bit to data.

**Table 19.3-5 PEN (Parity Enable) Bit Function**

PEN	Function
0	Parity added [Initial value]
1	Parity not added.

#### Note:

A parity bit can be added only in the normal mode (mode 0) of asynchronous (start-stop synchronous) communication mode. A parity bit cannot be added in multiprocessor mode (mode 1) and CLK synchronous mode (mode 2).

#### [bit 14] P (Parity):

When transmitting data with a parity bit, specify whether even or odd parity is used.

**Table 19.3-6 P (Even and Odd Parity Specifying Bit)**

P	Function
0	Even parity [Initial value]
1	Odd parity

#### [bit 13] SBL (Stop Bit Length):

When communicating in asynchronous (start-stop synchronous) mode, specify the bit length of the stop bit that is used as the frame end mark.



**Table 19.3-7 SBL (Stop Bit Length Specifying Bit)**

SBL	Function
0	One stop bit
1	Two stop bits

**[bit 12] CL (Character Length):**

Specify the data length of the frame to be sent or received.

**Table 19.3-8 CL (Send or Receive Data Length Specifying Bit)**

CL	Function
0	7-bit data [Initial value]
1	8-bit data

**Note:**

7-bit data can be send or received only in the normal mode (mode 0) of asynchronous (start-stop synchronous) communications. In the multiprocessor mode (mode 1) and CLK synchronous communication (mode 2), specify 8-bit data.

**[bit 11] A/D (Address/Data):**

Specify the data format of the frame to be sent or received in multiprocessor mode (mode 1) of asynchronous (start-stop synchronous) communications.

**Table 19.3-9 A/D (Address Data) Bit Function**

A/D	Function
0	Data frame
1	Address frame

**[bit 10] REC (Receiver Error Clear):**

Clears error flags (PE, ORE, FRE) of the SSR register.

Writing 1 is invalid, and the value read is always 1.

**[bit 9] RXE (Receiver Enable):**

Controls the receiving operation of the UART.

**Table 19.3-10 RXE (Receiver Enable) Bit**

RXE	Function
0	Prohibits the receiving operation.
1	Allows the receiving operation.

**Note:**

If the receiving operation is prohibited while data is being received (data is being input to the receiving shift register), the receiving operation is stopped when the frame is received

completely and the received data is stored in the receiving data buffer SDR register.

**[bit 8] TXE (Transmitter Enable):**

Controls the sending operation of the UART.

**Table 19.3-11 Send Operation Control Bit (TXE)**

TXE	Function
0	Prohibits the sending operation.
1	Allows the sending operation.

**Note:**

If the sending operation is prohibited while data is being sent (data is being output from the sending register), the sending operation is stopped when there is no remaining data in the sending data buffer SODR register.

After data is written into the SODR, wait for the time described in the following before writing 0.

The wait time should be one sixteenth that of the baud rate when in clock asynchronous transfer mode, and equivalent to that of the baud rate when in clock synchronous transfer mode.

### 19.3.3 Serial Input Data Register (SIDR)/Serial Output Data Register (SODR)

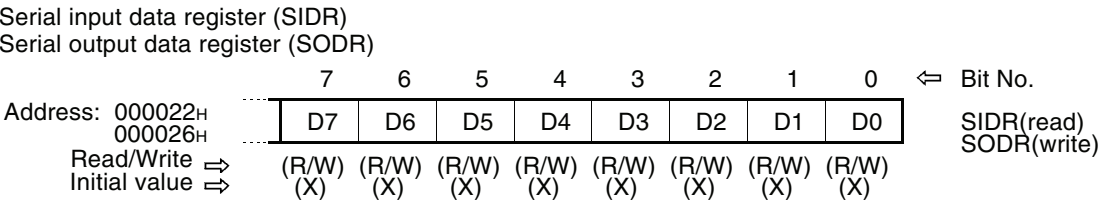
The serial input data register (SIDR) is a data buffer register used for receiving serial data.

The serial output data register (SODR) is a data buffer register used for sending serial data.

The SIDR and SODR registers are both allocated to the same address.

■ Configuration of Serial Input Data Register (SIDR)/Serial Output Data Register (SODR)

Figure 19.3-4 Configuration of Serial Input Data Register (SIDR)/Serial Output Data Register (SODR)



If the data is 7 bits in length, the high order 1 bit (D7) is invalid. Write data to the SODR register when TDRE of the SSR register is 1.

**Note:**

To write to and read from this address means that the write data is output to the SODR register and the read data is input to the SIDR register, respectively.

## 19.3.4 Serial Status Register (SSR)

The serial status register (SSR) is configured with flags that indicate the operation status of the UART.

### ■ Serial Status Register (SSR)

**Figure 19.3-5 Serial Status Register Configuration**

Serial status register									
	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 000023 <sub>H</sub> 000027 <sub>H</sub>	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE	SSR
Read/Write ⇒	(R)	(R)	(R)	(R)	(R)	(-)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(1)	(-)	(0)	(0)	

#### [bit 15] PE (Parity Error):

This is an interrupt request flag that is set when a parity error occurs while data is being received.

To clear the flag that is set, write 0 to the REC bit (bit 10) of the SCR register.

When this bit is set, data in the SIDR becomes invalid.

**Table 19.3-12 PE (Parity Error) Bit Function**

PE	Function
0	Parity error not occurred [Initial value]
1	Parity error occurred

#### [bit 14] ORE (Over Run Error):

This is an interrupt request flag that is set when an overrun error occurs while data is being received.

To clear the flag that is set, write 0 to the REC bit (bit 10) of the SCR register.

When this bit is set, data in the SIDR becomes invalid.

**Table 19.3-13 ORE (Over Run Error) Function**

ORE	Function
0	Overrun error not occurred [Initial value]
1	Overrun error occurred

#### [bit 13] FRE (Framing Error):

This is an interrupt request flag that is set when a flaming error occurs while data is being received.

To clear the flag that is set, write 0 to the REC bit (bit 10) of the SCR register.

When this bit is set, data in the SDR becomes invalid.

**Table 19.3-14 FRE (Framing Error) Function**

<b>FRE</b>	<b>Function</b>
0	Flaming error not occurred [Initial value]
1	Flaming error occurred

**[bit 12] RDRF (Receiver Data Register Full):**

This is an interrupt request flag that shows there is receiving data in the SDR register.

It is set when receiving data is loaded to the SDR register and is cleared automatically when data is read from the SDR register.

**Table 19.3-15 RDRF (Receiver Data Register Full) Function**

<b>SDR</b>	<b>Function</b>
0	No receiving data.
1	Receiving data.

**[bit 11] TDRE (Transmitter Data Register Empty):**

This is an interrupt request flag that shows that sending data can be written to the SDR register.

It is cleared when sending data is written to the SDR register and is set again when the written data is loaded to the sending shifter and transfer has started, thereby indicating that next sending data can be written.

**Table 19.3-16 TDRE (Transmitter Data Register Empty) Function**

<b>TDRE</b>	<b>Setting</b>
0	Writing sending data prohibited
1	Writing sending data allowed

**[bit 9] RIE (Receiver Interrupt Enable):**

This flag controls the receiving interrupt.

**Table 19.3-17 RIE (Receiver Interrupt Enable) Function**

<b>RIE</b>	<b>Function</b>
0	Prohibits interrupt.
1	Allows interrupt.

**Note:**

Receiving interrupt sources include error occurrence by PE, ORE, and FRE as well as normal reception by RDRF.

**[bit 8] TIE (Transmitter Interrupt Enable):**

This flag controls the sending interrupt.

**Table 19.3-18 TIE (Transmitter Interrupt Enable) Function**

<b>TIE</b>	<b>Function</b>
0	Prohibits interrupt
1	Allows interrupt

**Note:**

The sending interrupt source is a sending request by TDRE.

### 19.3.5 Communication Prescaler Control Register (CDCR)

The communication prescaler control register (CDCR) controls the machine clock frequency divide ratio.

■ Communication Prescaler Control Register (CDCR)

The operation clock of the UART is obtained by the frequency divided machine clock. This communication prescaler is designed to obtain a constant baud rate for various machine clocks.

This communication prescaler output is used for the operation clock of the extended I/O serial interface.

The bit configuration of the CDCR is shown below.

**Figure 19.3-6 Configuration of the CDCR (Communication Prescaler Control Register)**

Communication prescaler control register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000028 <sub>H</sub> 00002A <sub>H</sub>	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	CDCR
Read/Write ⇒	(R/W)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(-)	(-)	(-)	(1)	(1)	(1)	(1)	

**Note:**

The CDCR0 (address: 000028<sub>H</sub>) is a communication prescaler register responsible for channel 0 of the UART and for channels 2, 3, and 4 of the extended serial I/O interface. The use of the common register requires that a common division ratio be selected for channel 0 of the UART and for channels 2, 3, and 4 of the extended serial I/O interface. Note that use of more than one division ratio for these channels is prohibited. The CDCR1 (address: 00002A<sub>H</sub>) is responsible for channel 1 of the UART.

**[bit 7] MD (Machine clock divide moDe select)**

This is the operation allowance bit of the communication prescaler.

**Table 19.3-19 MD (Machine Clock Divide Mode Select) Bit Function**

CDCR	Function
0	Stops the communication prescaler.
1	Operates the communication prescaler.

**[bit 3,2,1,0] DIV3 to 0 (DIVide 3 to 0):**

Determines the division ratio of the machine clock.

Table 19.3-20 DIV3 to 0 (Divide 3 to 0) Bit Function

DIV3	DIV2	DIV1	DIV0	Division ratio (div)
1	1	1	1	Disabled (initial value)
1	1	1	0	Two
1	1	0	1	Three
1	1	0	0	Four
1	0	1	1	Five
1	0	1	0	Six
1	0	0	1	Seven
1	0	0	0	Eight

**Note:**

- For actual operation, set this value as a bit sequence other than "1111".
- If the division ratio is altered, wait for the time equivalent of two cycles for the clock to stabilize before starting communication.

**■ Setting of the Communication Prescaler Register**

Depending on the machine clock frequency  $\phi$  used, the communication prescaler register is set as shown below.

Table 19.3-21 Setting of the Communication Prescaler Register

Machine clock frequency $\phi$	div	DIV3	DIV2	DIV1	DIV0	$\phi/\text{div}$
4 MHz	4	1	1	0	0	1 MHz
6 MHz	6	1	0	1	0	
8 MHz	8	1	0	0	0	
6 MHz	3	1	1	0	1	2 MHz
8 MHz	4	1	1	0	0	
10 MHz	5	1	0	1	1	
12 MHz	6	1	0	1	0	
14 MHz	7	1	0	0	1	
16 MHz	8	1	0	0	0	
8 MHz	2	1	1	1	0	4 MHz
12 MHz	3	1	1	0	1	
16 MHz	4	1	1	0	0	

When a combination of machine clock and div value is to be used other than the combinations in the above table, a value not exceeding the maximum value of 4.25 MHz is selected for  $\phi/\text{div}$ .



## 19.4 UART Baud Rates

The UART clock can be selected from among the following sources.

- Dedicated baud rate generator
- Internal timer
- External clock

### ■ Dedicated baud rate generator

Tables 19.4-1 "Baud Rates (Asynchronous)" and 19.4-2 "Baud Rates (CLK Synchronous)" show the baud rates when the dedicated baud rate generator is selected. These baud rates are calculated using an assumed machine clock  $\phi$  value of 16 MHz and a div value (machine clock division ratio) of 8. Table 19.4-3 "Communication Prescaler Settings" shows the communication prescaler settings.

**Table 19.4-1 Baud Rates (Asynchronous)**

CS2	CS1	CS0	Asynchronous (start-stop synchronous)	Expression
0	0	0	9615 bps	$(\phi/\text{div}) / (8 \times 13 \times 2)$
0	0	1	4808 bps	$(\phi/\text{div}) / (8 \times 13 \times 2^2)$
0	1	0	2404 bps	$(\phi/\text{div}) / (8 \times 13 \times 2^3)$
0	1	1	1202 bps	$(\phi/\text{div}) / (8 \times 13 \times 2^4)$
1	0	0	31250 bps	$(\phi/\text{div}) / 2^6$

**Table 19.4-2 Baud Rates (CLK Synchronous)**

CS2	CS1	CS0	CLK synchronous When $\phi / \text{div} = 2 \text{ MHz}$ :	Expression
0	0	0	1 M bps	$(\phi/\text{div}) / 2$
0	0	1	500 K bps	$(\phi/\text{div}) / 2^2$
0	1	0	250 K bps	$(\phi/\text{div}) / 2^3$
0	1	1	125 K bps	$(\phi/\text{div}) / 2^4$
1	0	0	62.5 K bps	$(\phi/\text{div}) / 2^5$

**Table 19.4-3 Communication Prescaler Settings**

MD	DIV3	DIV2	DIV1	DIV0	div	Recommended machine clock
1	1	1	0	1	3	6 MHz
1	1	1	0	0	4	8 MHz
1	1	0	1	1	5	10 MHz
1	1	0	1	0	6	12 MHz
1	1	0	0	0	8	16 MHz

### ■ Internal timer

When the bits CS2 to CS0 in the serial mode register (SMR) are set to a sequence of 110 for an internal timer setting, the 16-bit timer (timer 0) operates in reload mode. To obtain the baud rate, use the following formulas.

Asynchronous (start-stop synchronous):	$(\phi) / (16 \times 2 \times (n+1))$
CLK synchronous:	$(\phi/N) / (2 \times (n+1))$

$\phi$ : Machine clock

N: Count clock source of the timer

n: Reload value of the timer

Table 19.4-4 "Baud Rates and Reload Values (Asynchronous)" shows baud rates and reload values (in decimal) when the machine clock is set at 7.3728 MHz.

**Table 19.4-4 Baud Rates and Reload Values (Asynchronous)**

Baud rate	Reload value	
	N=2 <sup>1</sup> Machine clock frequency divided by two	N=2 <sup>3</sup> Machine clock frequency divided by eight
38400	2	—
19200	5	—
9600	11	2
4800	23	5
2400	47	11
1200	95	23
600	191	47
300	383	95

When the internal timer (16-bit reload timer 0) is selected as a baud rate clock source, the output (T00) of the 16-bit timer 0 is already connected in this controller. Therefore, the external

## CHAPTER 19 UART

pin (T00) of the 16-bit timer 0 and the external clock input pin SCK0 do not require external connection. Also, the output pin of the timer 0 can be used as an I/O port pin if not used otherwise.

### ■ External clock

When the external clock is selected by setting CS2 to CS0 to 111, baud rates are calculated as follows by expressing the frequency of the external clock as  $f$ .

Asynchronous (start-stop synchronous):	$f/16$
CLK synchronous:	$f$ (up to 1 MHz)

Note that the maximum value for  $f$  is 1 MHz.

## 19.5 Operation of the UART

The UART has two different operation modes: the asynchronous mode and the CLK synchronous mode. The operation mode can be switched by specifying the value for the serial mode register (SMR) or the serial control register (SCR).

### ■ UART Operation Modes

Table 19.5-1 UART Operation Modes

Mode	Parity	Data length	Operation mode	Stop bit length
0	Yes/No	7	Asynchronous (start-stop synchronous)	1 bit or 2 bits
	Yes/No	8	Normal mode	
1	No	8+1	Asynchronous (start-stop synchronous) Multiprocessor mode	
2	No	8	CLK synchronous mode	No

**Note:**

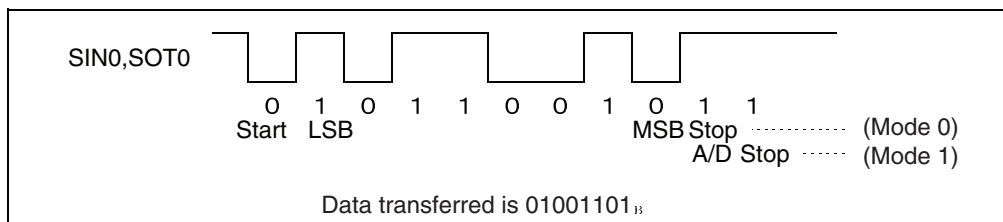
The stop bit length in asynchronous (start-stop synchronous) mode is specified only for a sending operation. A one-bit length is always specified for a receiving operation. Because the register cannot be operated in a mode other than above, one of above-shown modes must be set.

## 19.5.1 Asynchronous (Start-stop Synchronous) Mode

When the UART operates in operation mode 0 (normal mode) or operation mode 1 (multi-processor mode), the asynchronous mode is used for data transfer. Also, the UART manages data in NRZ (Non Return to Zero) format only.

### ■ Transfer Data Format

Figure 19.5-1 Transfer Data Format (Mode 0, 1)



As shown in Figure 19.5-1 "Transfer Data Format (Mode 0, 1)", transfer data always starts with the start bit (L level data), followed by data bits in the LSB first method, and ends with the stop bit (H level data). When the external clock is selected, the clock must be entered.

In normal mode (mode 0), data length can be set to either 7 or 8 bits. In multiprocessor mode (mode 1), however, 8 bits must be set. In addition, parity cannot be added in the multiprocessor mode. Instead of a parity, the A/D bit is always added.

### ■ Receiving Operation

If 1 is set in the RXE bit of the SCR register, the receiving operation is performed.

When the start bit appears in the receiving line, 1-frame data is received according to the data format specified in the SCR register. When the reception of 1-frame data is completed, (after the error flag is set in the case of an error) the RDRF flag (of the SSR register) is set. If 1 is set in the RIE bit of the SSR register, receiving interrupt occurs to the CPU. Check the flags in the SSR register and read the SDR register if the receiving operation is completed normally. If an error occurs, respond as required.

The RDRF flag is cleared when the SDR register is read.

### ■ Sending Operation

When the TDRE flag of the SSR register is 1, the sending data is written in the SODR register. If the TXE bit of the SCR register is 1, the data is sent.

When the data set in the SODR register is loaded to the sending shift register and sending begins, the TDRE flag is set again and the next sending data can be set. If 1 is set in the TIE bit of the SSR register, a sending interrupt occurs to the CPU and the CPU is required to set the sending data in the SODR register.

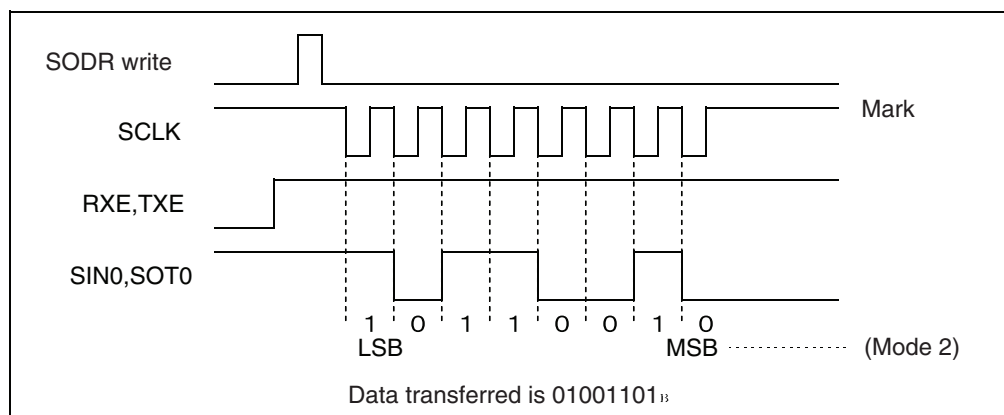
The TDRE flag is cleared temporarily if the data is set in the SODR register.

## 19.5.2 CLK Synchronous Mode

When the UART operates in the operation mode 2 (normal mode), the CLK synchronous mode is used for data transfer. Also, the UART manages data in NRZ (Non Return to Zero) format only.

### ■ Transfer Data Format of CLK Synchronous Mode

Figure 19.5-2 Transfer Data Format



When an internal clock (dedicated baud rate generator or internal timer) is selected, a data receiving synchronous clock is automatically generated when the data is sent.

When an external clock is selected, confirm that the data exists in the sending data buffer SODR register when sending the UART side (the TDRE flag is 0) and then provide clock pulses that cover a duration of one byte. Also, confirm that a mark level "H" is set before and after the sending operation.

All data must be 8-bits long and parity cannot be added, and because there are no start/stop bits, errors other than overrun errors cannot be detected.

### ■ Settings for the Control Registers When CLK Synchronous Mode Is Used

The settings for the control registers when CLK synchronous mode is used are as follows:

**Table 19.5-2 Setting for the Control Register when CLK Synchronous Mode is Used**

Register name	Bit	Setting
SMR register	MD1, MD0	"10"
	CS2, CS1, CS0	Specifies the clock input.
	SCKE	When the dedicated baud rate generator or internal timer is used: "1", when the external clock is used: "0"
	SOE	To send: 1, to receive: 0
SCR register	PEN	"0"
	P, SBL, A/D	These bits have no meaning.
	CL	"1"
	REC	"0" (for initialization)
	RXE, TXE	At least one must be "1".
SSR register	RIE	When an interrupt is used: "1", when an interrupt is not used: "0"
	TIE	"0"

### ■ Starting the Communication in CLK Synchronous Mode

Communication can be started by writing in the SODR register. Even when receiving data, temporary sending data must be written in the SODR register.

### ■ Terminating the Communication in CLK Synchronous Mode

The user can confirm termination when the RDRF flag of the SSR register has changed to 1. Determine if the communication was completed normally by checking the ORE bit of the SSR register.

## 19.6 Flags and Interrupt Sources of the UART

---

The UART has five flags, PE, ORE, FRE, RDRF, and TDRE. Interrupt sources are categorized for reception and transmission.

---

### ■ Flags of the UART

- **PE (Parity Error), ORE (Overrun Error), and FRE (Flaming Error)**

These flags are set when a receiving error occurs and cleared when 0 is written to REC of the SCR register.

- **RDRF**

RDRF is set when a receiving data is loaded in the SDR register and cleared when the SDR register is read. However, the parity detect function is not supported in mode 1, while the parity detect and flaming error detect functions are not supported in mode 2.

- **TDRE**

TDRE is set when the SODR register becomes empty and ready to be written and cleared when written in the SODR register.

### ■ Interrupt Sources of the UART

Interrupt sources of the UART are used for either receiving or sending data. When receiving data, an interrupt is requested by PE, ORE, FRE, or RDRF. When sending data, an interrupt is requested by TDRE.

For the timing to set an interrupt and flag in each operation mode, see Section 19.6.1 "Timing to Set an Interrupt and Flag of the UART".



### 19.6.1 Timing to Set an Interrupt and Flag of the UART

This section describes the timing to set an interrupt and a flag in each operation mode.

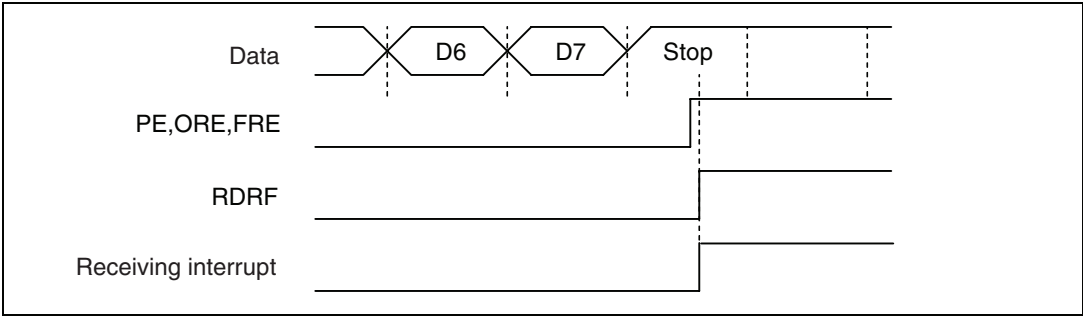
■ Timing to Set an Interrupt and Flag of the UART

○ In receiving operation of mode 0

PE, ORE, FRE, or RDRF is set when the receiving transfer is completed and the last stop bit is detected. Thereafter, an interrupt request to the CPU is generated. When PE, ORE, or FRE is active, data in SISR is invalid.

Figure 19.6-1 "Timing Chart to Set PE, ORE, FRE, and RDRF (Mode 0)" shows a timing chart to set PE, ORE, FRE, and RDRF (mode 0).

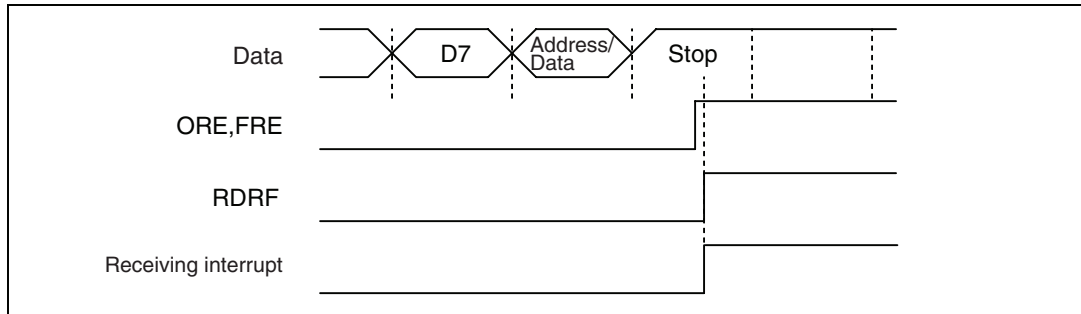
Figure 19.6-1 Timing Chart to Set PE, ORE, FRE, and RDRF (Mode 0)



○ In receiving operation of mode 1

ORE, FRE, or RDRF is set when the receiving transfer is completed and the last stop bit is detected. Thereafter, an interrupt request to the CPU is generated. Since the receivable data length is 8 bits, the data in the ninth bit representing address/data is invalid. When ORE or FRE is active, data in SISR is invalid.

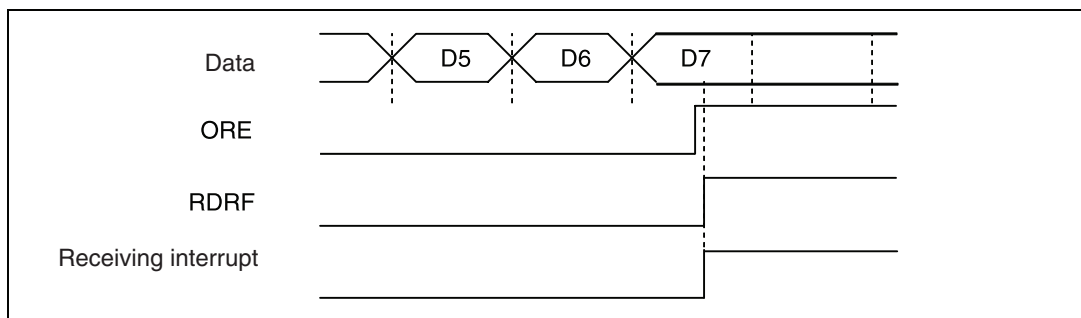
Figure 19.6-2 "Timing Chart to Set ORE, FRE, and RDRF (Mode 1) " shows a timing chart to set ORE, FRE, and RDRF (mode 1).

**Figure 19.6-2 Timing Chart to Set ORE, FRE, and RDRF (Mode 1)**

○ **In receiving operation of mode 2**

ORE or RDRF is set when the receiving transfer is completed and the last data (D7) is detected. Thereafter, an interrupt request to the CPU is generated. When ORE is active, data in SDR is invalid.

Figure 19.6-3 "Timing Chart to Set ORE and RDRF (Mode 2)" shows a timing chart to set ORE and RDRF (mode 2).

**Figure 19.6-3 Timing Chart to Set ORE and RDRF (Mode 2)**

○ **In sending operation of modes 0, 1, and 2.**

The TDRE flag is cleared when the sending data is written in the SODR register. Also, the SODR register is ready to be written when the SODR register value is transferred to the internal shift register, so the TDRE flag is set. When this flag is set, an interrupt request to the CPU is generated. If 0 is written to TXE (RXE as well in mode 2) of the SCR register during the sending operation, TDRE of the SSR register turns to 1. As a result, the sending shifter is stopped and the sending operation of the UART is prohibited. Even though 0 is written to TXE (RXE as well in mode 2) of the SCR register during the sending operation, the data is sent if written to the SODR register before the sending operation is stopped.

Figure 19.6-4 "Timing Chart to Set TDRE (Mode 0, 1)" shows the timing to set TDRE (mode 0, 1).

Also, Figure 19.6-5 "Timing to Set TDRE (Mode 2)" shows a timing chart to set TDRE (mode 2).

Figure 19.6-4 Timing Chart to Set TDRE (Mode 0, 1)

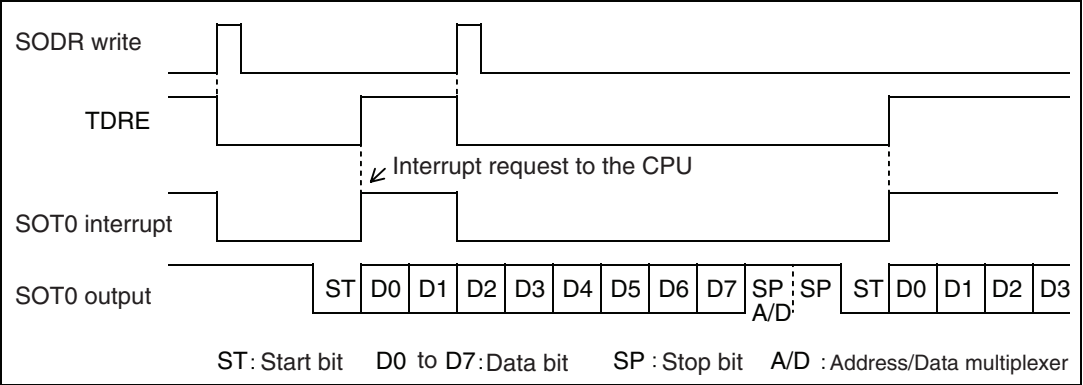
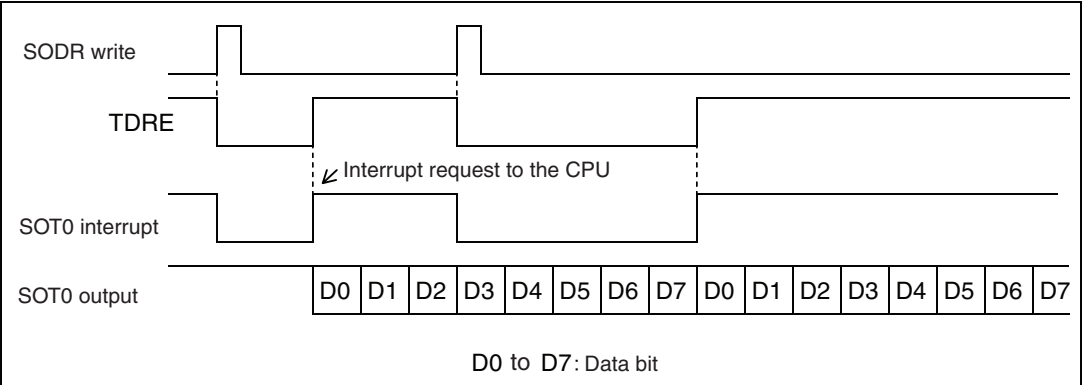


Figure 19.6-5 Timing to Set TDRE (Mode 2)



## 19.7 Applications of the UART and Precautions

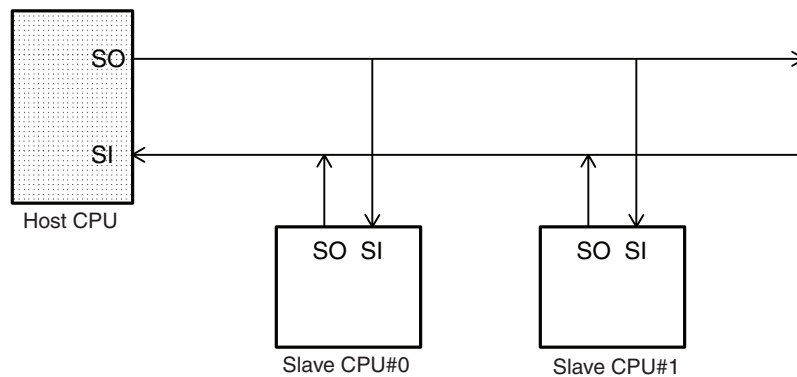
As an application of the UART, this section shows an example of system configuration in mode 1 and communication flow chart.

### ■ Application of the UART (Example of System Configuration in Mode 1)

Mode 1 is used when multiple slave CPUs are connected to one host CPU (see Figure 19.7-1 "Example of System Configuration in Mode 1"). In this resource, only the communication interface of the host is supported.

Figure 19.7-1 "Example of System Configuration in Mode 1" shows an example of system configuration in mode 1.

**Figure 19.7-1 Example of System Configuration in Mode 1**

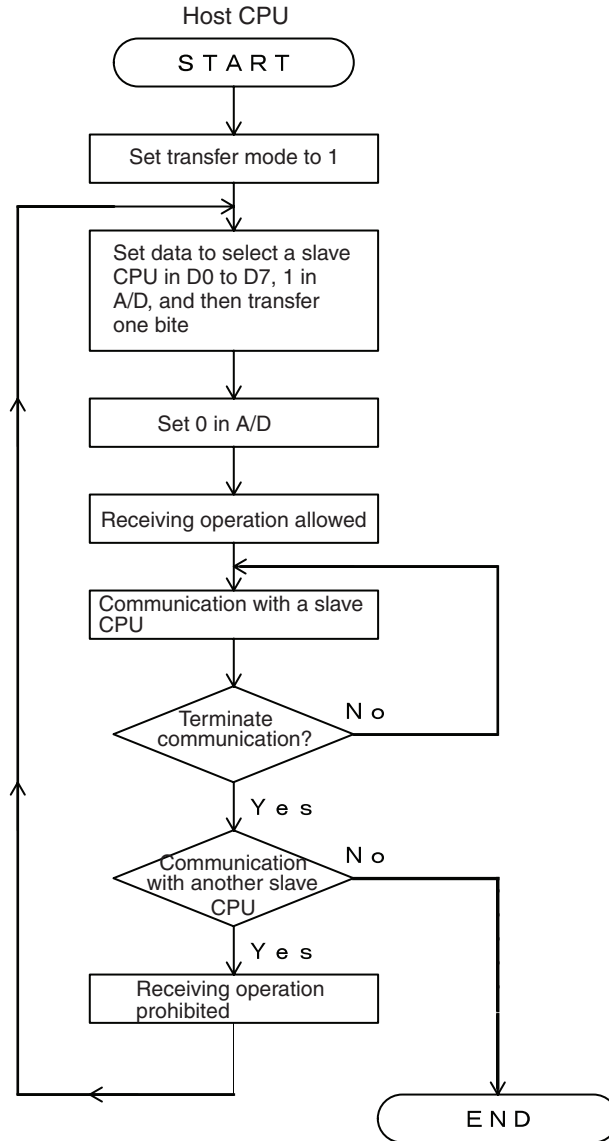


### ■ Communication Flow Chart of the UART

Communication starts when the host CPU transfers the address data. Address data is data when A/D of the SCR register is 1. A slave CPU is selected according to this address data and communication with the host CPU is enabled. The usual data is data when A/D of the SCR register is 0.

The parity checking function is not used in this mode, so set 0 in PEN of the SCR register.

Figure 19.7-2 Communication Flow Chart in Mode 1



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

For EI<sup>2</sup>OS, see Section 3.6 "Extended Intelligent I/O Service (EI<sup>2</sup>OS)".

### ■ Precautions for Using the UART

Select a communication mode when the operation is stopped to guarantee data sent or received during mode setting.

# CHAPTER 20    EXTENDED SERIAL I/O INTERFACE

---

**This chapter describes the functions and operations of the extended serial I/O interface.**

---

20.1 "Overview of the Extended Serial I/O Interface"

20.2 "Registers of the Extended Serial I/O Interface"

20.3 "Operation of the Extended Serial I/O Interface"

## 20.1 Overview of the Extended Serial I/O Interface

---

The extended serial I/O interface is a serial type I/O interface that can transfer data with an 8-bit and 3-channel structure in clock synchronous mode. Also, in data transfer, selection between LSB-first transfer and MSB-first transfer is possible.

---

### ■ Overview of Extended Serial I/O Interface

The two types of extended serial I/O operating modes are as follows:

- **Internal shift clock mode:**

Transfers data in synchronization with the internal clock (communication prescaler).

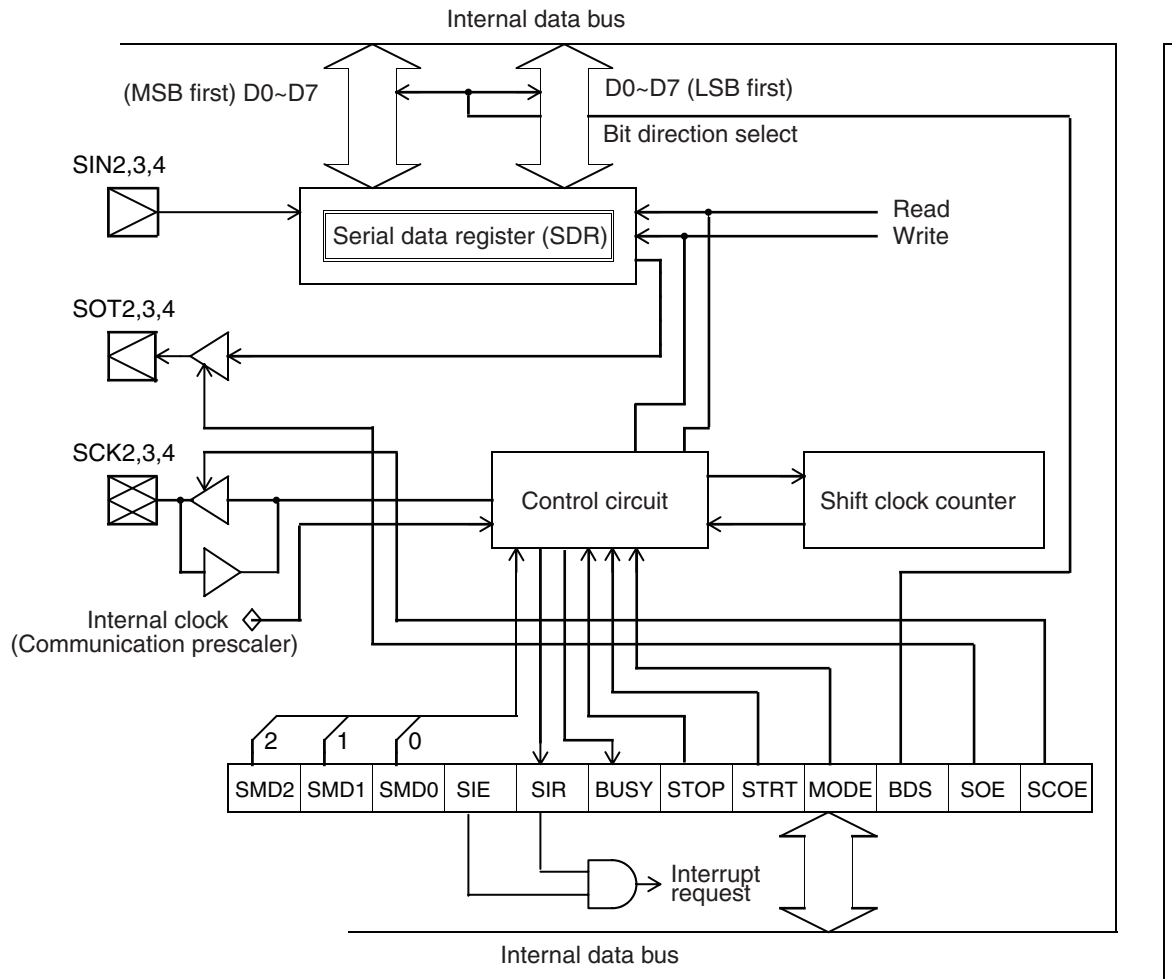
- **External shift clock mode:**

Transfers data in synchronization with the clock input from an external pin (SCK). In this mode, general-purpose ports that share the external pin (SCK) can perform transfer operations using CPU instructions.

The unit of this series contains three channels of the extended serial I/O interface.

# ■ Block Diagram of the Extended Serial I/O Interface

Figure 20.1-1 Block Diagram of the Extended Serial I/O Interface





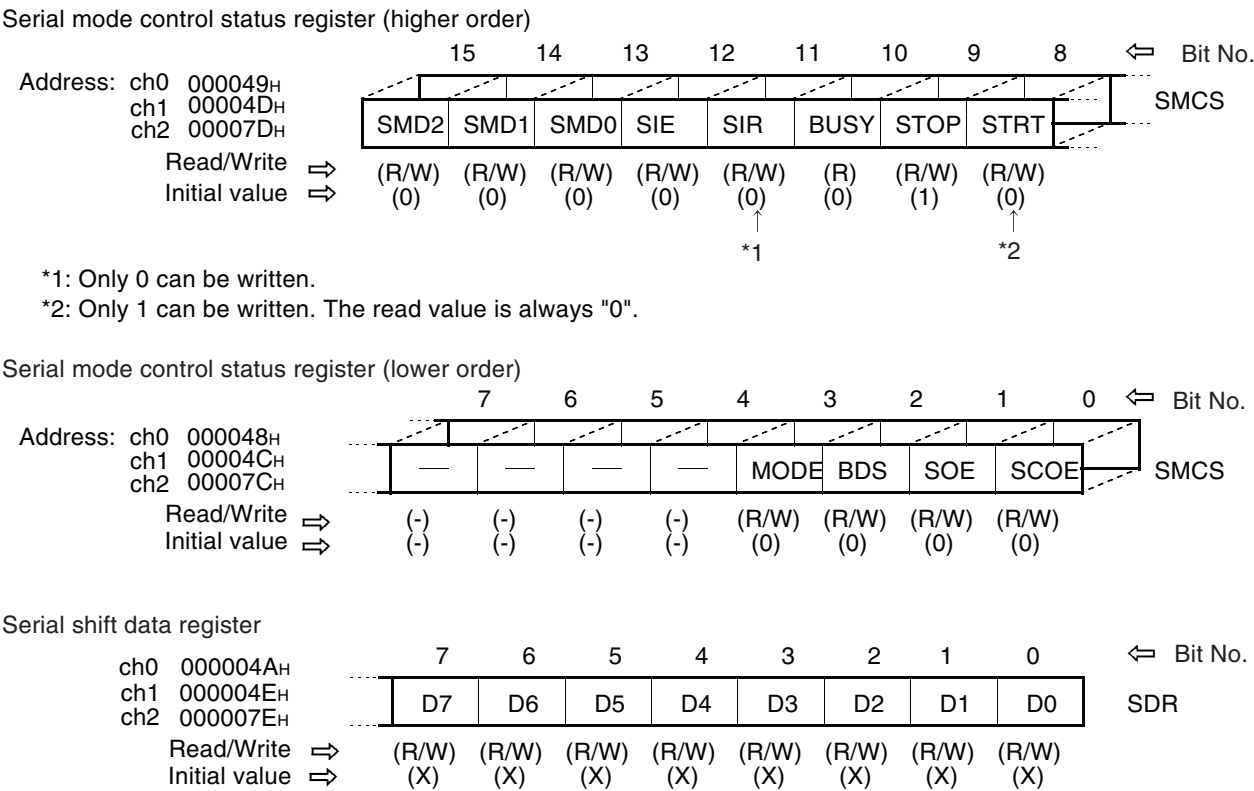
## 20.2 Registers in the Extended Serial I/O Interface

The extended serial I/O interface has three registers as follows.

- Serial mode control status register (higher order)
- Serial mode control status register (lower order)
- Serial data register

### ■ Registers of the Extended Serial I/O Interface

Figure 20.2-1 Registers in the Extended Serial I/O Interface



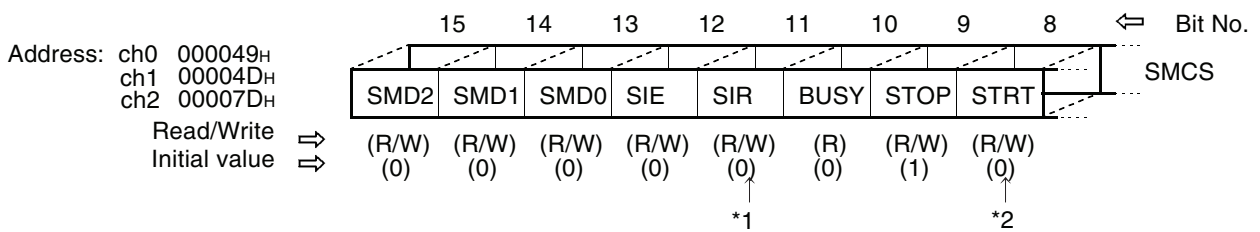
## 20.2.1 Serial Mode Control Status Register (SMCS)

The serial mode control status register (SMCS) controls the serial I/O transfer operating mode.

### ■ Serial Mode Control Status Register (SMCS)

**Figure 20.2-2 Serial Mode Control Status Register (SMCS)**

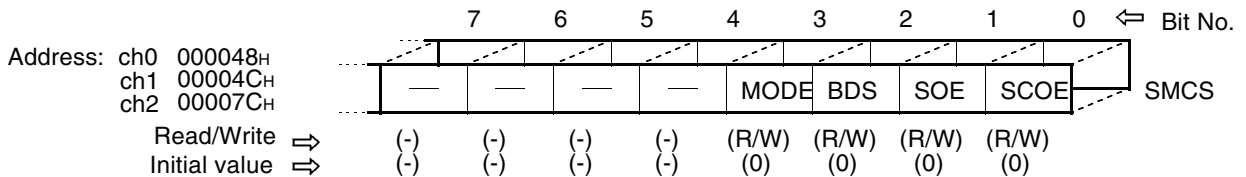
Serial mode control status register (higher order)



\*1: Only "0" can be written.

\*2: Only "1" can be written. The read value is always "0".

Serial mode control status register (lower order)



Notes: Only "0" can be used in write operation for the SIR (bit No. 11).

Only "1" can be used in write operation for the STRT (bit No. 8). The read value is always "0".

The function of each bit is explained below.

#### [Bit 3] Serial mode select bit (MODE)

Use this bit to select the activation condition in the stopped state. A write operation is prohibited during operation.

This bit is initialized to "0" by a reset and is a read/write bit. To activate the intelligent I/O service, set this bit to "1".

**Table 20.2-1 MODE Function (Activation Condition Select Bit)**

MODE	Function
0	Activated by setting STRT to 1. [Initial value]
1	Activated by reading or writing the serial data register.

**[Bit 2] Bit direction select bit (BDS)**

This bit selects the bit direction during I/O of serial data as listed in Table 20.2-2 "Settings of the Bit Direction Select Bit". Data can be transferred from the least significant bit (LSB first) or the most significant bit (MSB first).

This bit is initialized to "0" and is a read/write bit.

**Table 20.2-2 Settings of the Bit Direction Select Bit**

BDS	Function
0	LSB first [Initial value]
1	MSB first

This bit is initialized to "0" and is a read/write bit.

**Note:**

Set the bit direction select bit before writing data in the SDR.

**[Bit 1] Serial output enable bit (SOE)**

This bit controls the outputs of the serial I/O output external pins (SOT2, SOT3, and SOT4) as listed in Table 20.2-3 "Functions of the Serial Out Enable Bit (SOE)".

**Table 20.2-3 Functions of the Serial Out Enable Bit (SOE)**

SOE	Function
0	General-purpose port pins [Initial value]
1	Serial data outputs

This bit is initialized to "0" by a reset and is a read/write bit.

**[Bit 0] SCK1 output enable bit (SCOE)**

This bit controls the outputs of the shift clock external pins (SCK2, SCK3, and SCK4) as listed in Table 20.2-4 "Functions of the SCK1 (Output Enable) Bit (SCOE)".

Set this bit to "0" to transfer data for each instruction in the external shift clock mode.

This bit is initialized to "0" by a reset and is a read/write bit.

**Table 20.2-4 Functions of the SCK1 (Output Enable) Bit (SCOE)**

SCOE	Function
0	General-purpose pins, transfer for each instruction [Initial value]
1	Shift clock output pins

**[Bits 15, 14, and 13] Serial shift clock mode bits (SMD2, SMD1, and SMD0)**

These bits select the serial shift clock mode as listed in Table 20.2-5 "Functions of the SMD0 to SMD2 (Serial Shift Clock Mode Selection Bit)".

These bits are initialized to "000" by a reset. Writing these bits is prohibited during transfer.

A shift clock can be selected among five internal shift clocks and an external shift clock. Do not set SMD2, SMD1, and SMD0 to "110" or "111" because these values are reserved.

When SCOE is 0 for clock selection, ports that share the SCK1 or SCK2 pin can perform

shift operations for each instruction.

**Table 20.2-5 Functions of the SMD0 to SMD2 (Serial Shift Clock Mode Selection Bit)**

SMD2	SMD1	SMD0	Divide factor A	$\phi = 16\text{MHz}$ div=8	$\phi = 8\text{MHz}$ div=4	$\phi = 4\text{MHz}$ div=4
0	0	0	2	1 MHz	1 MHz	500 KHz
0	0	1	4	500 KHz	500 KHz	250 KHz
0	1	0	16	125 KHz	125 KHz	62.5 KHz
0	1	1	32	62.5 KHz	62.5 KHz	31.25 KHz
1	0	0	64	31.25 KHz	31.25 KHz	15.625 KHz
1	0	1	1	External shift clock mode		
1	1	0	—	Reserved		
1	1	1	—	Reserved		

**Table 20.2-6 Recommended Machine Cycles by Communication Prescaler (CDCR) Settings**

div*	Machine clock					Recommended machine cycle
	MD	D3	D2	D1	D0	
3	1	1	1	0	1	6 MHz
4	1	1	1	0	0	8 MHz
5	1	1	0	1	1	10 MHz
6	1	1	0	1	0	12 MHz
7	1	1	0	0	1	14 MHz
8	1	1	0	0	0	16 MHz

**Note:**

- For details on the communication prescaler (CDCR), see Section "Communication Prescaler Control Register (CDCR)".
- D3 to D0 are abbreviations for DIV0 to DIV3.

**[Bit 12] Serial I/O interrupt enable bit (SIE)**

This bit controls serial I/O interrupt requests as listed in Table 20.2-7 "Functions of the Serial I/O Interrupt Enable Bit".

This bit is initialized to "0" by a reset and is a read/write bit.

**Table 20.2-7 Functions of the Serial I/O Interrupt Enable Bit**

SIE	Function
0	Disables serial I/O interrupts. [Initial value]
1	Enables serial I/O interrupts.

**[Bit 11] Serial I/O interrupt request bit (SIR)**

When serial data transfer terminates, this bit is set to "1". When this bit is set to "1" in the interrupt enable state (when SIE is "1"), an interrupt request is issued to the CPU. The clear condition is dependent on the setting of the MODE bit. When the MODE bit is "0", the SIR bit is cleared by writing "0" in this bit. When the MODE bit is "1", the SIR bit is cleared by reading or writing the SDR. The SIR bit is also cleared by a reset or by writing "1" in the STOP bit regardless of which value is set in the MODE bit.

Writing "1" in the SIR bit has no meaning. When a read-modify-write instruction reads the SIR bit, the read value is always "1".

**[Bit 10] Transfer status bit (BUSY)**

This bit indicates whether serial transfer is being executed.

This bit is initialized to "0" by a reset and is a read-only bit.

**Table 20.2-8 Functions of the Transfer Status Bit (BUSY)**

<b>BUSY</b>	<b>Function</b>
0	Stopped or serial data register R/W wait state [Initial value]
1	Serial transfer state

**[Bit 9] Stop bit (STOP)**

This bit forcibly stops serial transfer. Setting this bit to "1" places serial transfer in the stopped state with STOP set to 1.

This bit is initialized to "1" by a reset and is a read/write bit.

**Table 20.2-9 Stop Bit Function**

<b>STOP</b>	<b>Function</b>
0	Normal operation
1	Transfer stopped with STOP set to 1 [Initial value]

**[Bit 8] Start bit (STRT)**

This bit starts serial transfer. Writing "1" in this bit in the stopped state starts transfer. Writing "1" is ignored and writing "0" has no meaning during serial transfer or in the serial shift register R/W wait state.

The read value is always "0".

### 20.2.2 Serial Shift Data Register (SDR)

The serial shift data register (SDR) retains serial I/O transfer data. Writing and reading the SDR is prohibited during transfer.

■ Serial Shift Data Register (SDR)

Figure 20.2-3 Serial Shift Data Register (SDR)

Serial shift data register										
ch0	000004A <sub>H</sub>	7	6	5	4	3	2	1	0	↵ Bit No.
ch1	000004E <sub>H</sub>	<div>D7</div>	<div>D6</div>	<div>D5</div>	<div>D4</div>	<div>D3</div>	<div>D2</div>	<div>D1</div>	<div>D0</div>	SDR
ch2	000007E <sub>H</sub>									
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)	

## 20.3 Operation of the Extended Serial I/O Interface

---

The extended serial I/O interface consists of the serial mode control status register (SMCS) and shift register (SDR).

It is used to input and output 8-bit serial data.

---

### ■ Operation of the Extended Serial I/O Interface

The contents of the shift register are output to the serial output pin (SOT1 pin) in the bit serial mode in synchronization with the falling edge of the serial shift clock (external or internal clock).

Data is input from the serial input pin (SIN1 pin) to the shift register (SDR) in the bit serial mode in synchronization with the rising edge of the clock.

The shift direction (transfer from the MSB or LSB) can be selected using the bit direction bit (BDS) in the serial mode control status register (SMCS).

When transfer terminates, the extended serial I/O interface enters the stopped state or data register R/W wait state according to the setting of the MODE bit in the serial mode control status register (SMCS). To create a transition from each state to the transfer state, proceed as follows:

- To return from the stopped state, write "0" in the STOP bit and "1" in the STRT bit (STOP and STRT can be set simultaneously).
- To return from the serial shift data register R/W wait state, read or write the data register.

## 20.3.1 Shift Clock

---

The two modes for the shift clock are as follows: internal shift clock mode and external shift clock mode. The mode is specified by SMCS settings. Switch the mode in the serial I/O stopped state. The stopped state can be checked by reading the BUSY bit.

---

### ■ Internal Shift Clock Mode

By using the output of the communication prescaler, a shift clock with a 50% duty cycle can be delivered as synchronous timing output from the SCK pin. One-bit data is transferred for each clock. The transfer rate can be calculated as follows:

$$\text{transfer-rate (s)} = \frac{A}{\text{internal-clock-machine-cycle (Hz)}}$$

A is the divide factor set by the SMD bits in the SMCS, 2, 4, 16, 32, or 64.

### ■ External Shift Clock Mode

One-bit data is transferred for each clock in synchronization with the external shift clock input from the SCK pin.

A transfer rate with a frequency from DC to 1 divided by five machine cycles is available. For example, when one machine cycle is 0.1 μs, a transfer rate of up to 2 MHz is available.

Data can also be transferred for each instruction. To transfer data for each operation, effect the following settings:

1. Select the external shift clock mode and set the SCOE bit in the SMCS to "0",
2. Then write "1" in the direction register for one of the ports that share the SCK pin to set the port to the output mode.

After effecting the above settings, write "1", then "0" in the port data register (PDR). The port value output to the SCK pin is fetched as the external clock and data is transferred. Start the shift clock from "H".

#### **Note:**

Writing the SMCS and SDR is prohibited during serial I/O operation.



## 20.3.2 Operating States of the Extended Serial I/O Interface

---

The four operating states of the extended serial I/O interface are as follows:

- STOP
  - Stopped
  - SDR R/W wait
  - Transfer
- 

### ■ STOP State

When a reset occurs or "1" is written in the STOP bit in the SMCS, the shift counter is initialized and SIR is set to "0".

The extended serial I/O interface can return from the STOP state by setting "0" in STOP and "1" in STRT (can be set simultaneously). When STOP is 1, setting STRT to 1 cannot start transfer because the STOP bit has a higher priority than the STRT bit.

### ■ Stopped state

When the MODE bit is "0" and transfer terminates, BUSY is set to "0" and SIR is set to "1" in the SMCS. The counter is initialized and the extended serial I/O interface enters the stopped state. By setting STRT to "1", the extended serial I/O interface returns from the stopped state and restarts transfer.

### ■ Serial data register R/W wait state

When the MODE bit in the SMCS is "1" and serial transfer terminates, BUSY is set to "0" and SIR is set to "1". The extended serial I/O interface enters the serial data register R/W wait state. When the interrupt enable register indicates the enable state, an interrupt signal is output from this block.

When the serial data register is read or written, BUSY is set to "1" and the extended serial I/O interface returns from the R/W wait state and restarts transfer.

### ■ Transfer state

BUSY is "1" and the extended serial I/O interface is transferring serial data. A transition to the stopped or R/W wait state occurs depending on the MODE bit setting.

Figure 20.3-1 "Diagram of Operation Transition of the Extended Serial I/O Interface" shows a diagram of operation transition from each state. Figure 20.3-2 "Conceptual Diagram of Read from and Write to the Serial Data Register" shows a conceptual diagram of read from and write to the serial data register.

Figure 20.3-1 Diagram of Operation Transition of the Extended Serial I/O Interface

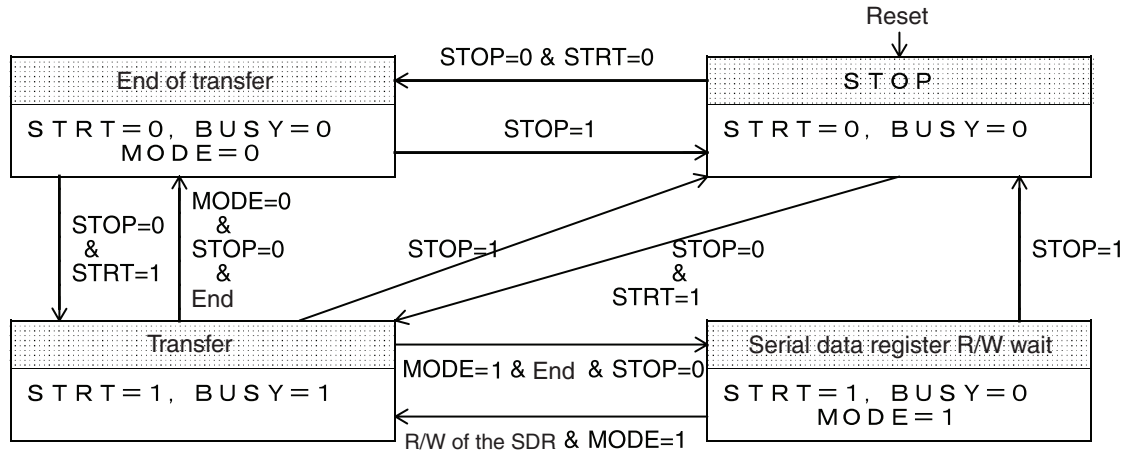
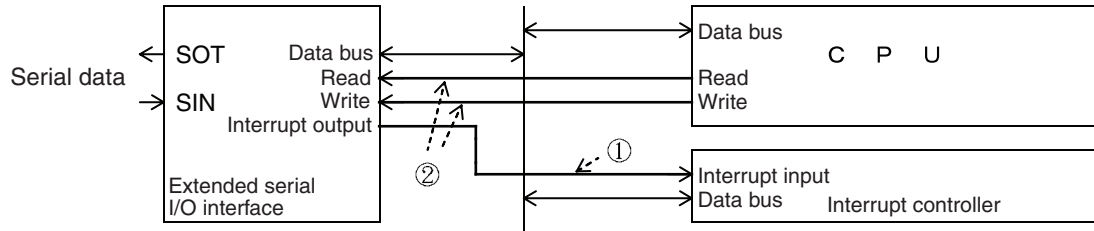


Figure 20.3-2 Conceptual Diagram of Read from and Write to the Serial Data Register



① and ② in Figure 20.3-2 are explained below.

- ① When MODE is 1, transfer terminates according to the shift clock counter. SIR is set to 1 and the extended serial I/O interface enters the read/write wait state. When the SIE bit is "1," the extended serial I/O interface generates an interrupt signal. When SIE indicates that the inactive state or transfer is stopped by writing "1" in STOP, the interface does not generate an interrupt signal.
- ② When the serial data register is read or written, the interrupt request is cleared and the interface starts serial transfer.

### 20.3.3 Operation Timings of the Extended Serial I/O Interface

To start and stop shift operation, effect the following settings:

**Start:**

Set the STOP bit to "0" and STRT bit to "1" in the SMCS.

**Stop:**

Shift operation stops when transfer terminates or when STOP is set to 1.

[Stop by setting STOP to 1 ] Shift operation stops and SIR remains set to 0 regardless of the value set in the MODE bit.

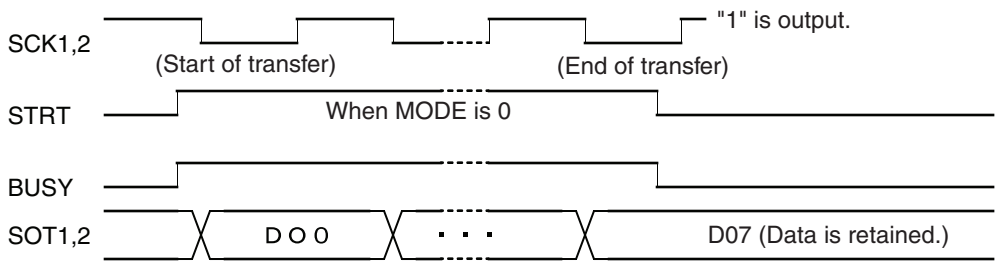
[Stop at end of transfer] SIR is set to 1 and the shift operation stops regardless of the value set in the MODE bit.

#### ■ Shift Operation Start/Stop Timings

The BUSY bit is "1" in the serial transfer state or "0" in the stopped or R/W wait state regardless of the value set in the MODE bit. To check the transfer state, read this bit.

**Figure 20.3-3 Shift Operation Start/Stop Timings (Internal Clock)**

○ Internal shift clock mode (LSB first)

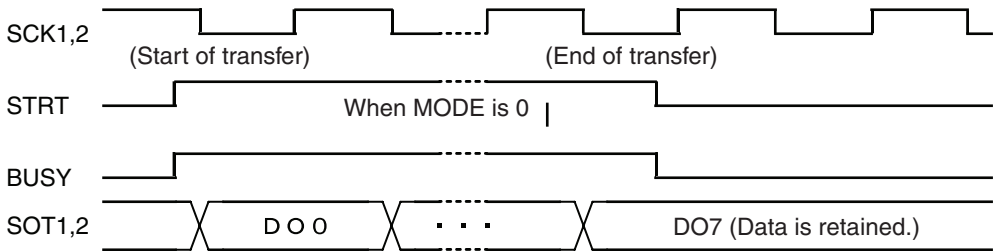


**Note:**

DO7 to DO0 indicate output data.

**Figure 20.3-4 Shift Operation Start/Stop Timings (External Clock)**

○ External shift clock mode (LSB first)

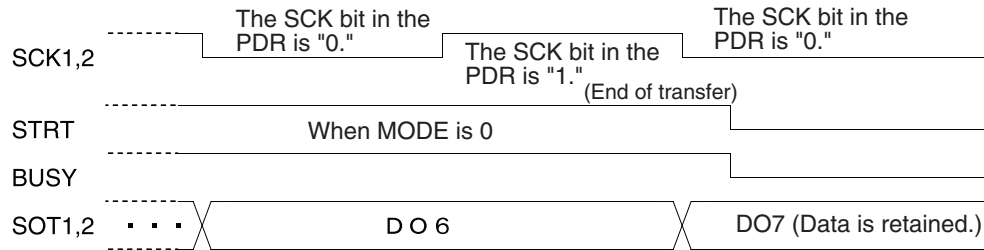


**Note:**

DO7 to DO0 indicate output data.

**Figure 20.3-5 Shift Operation Start/Stop Timings (When a Shift Operation is Performed in Accordance with Instructions in the External Shift Clock Mode)**

- When a shift operation is performed in accordance with instructions in the external shift clock mode (LSB first)

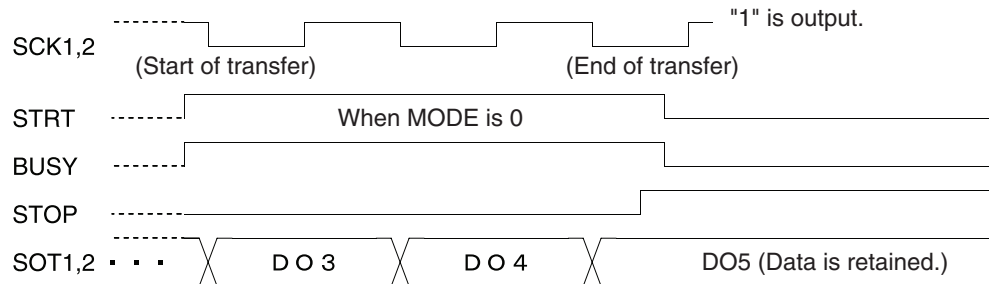
**Note:**

DO7 to DO0 indicate output data.

During a shift operation in accordance with instructions, when "1" is written in the bit corresponding to SCK in the PDR, "H" is output. When "0" is written, "L" is output (only when the external shift clock mode is selected and SCOE is 0).

**Figure 20.3-6 Stop Timing when the STOP Bit Is Set to "1"**

- Stop by setting STOP to 1 (LSB first, internal clock mode)

**Note:**

DO7 to DO0 indicate output data.

### 20.3.4 Serial Data I/O Shift Timings

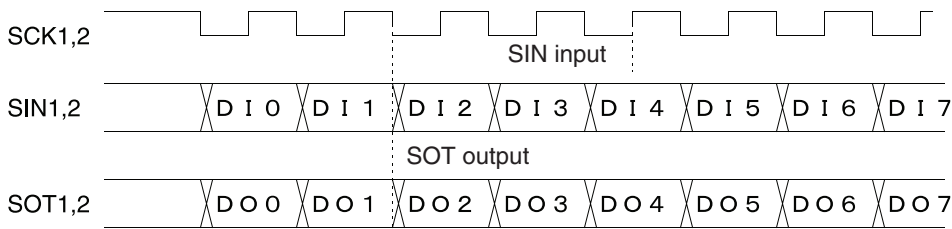
During serial data transfer, data from the serial output pin (SOT2) is output at the falling edge of the shift clock and data from the serial input pin (SIN) is input at the rising edge.

■ Serial Data I/O Shift Timings

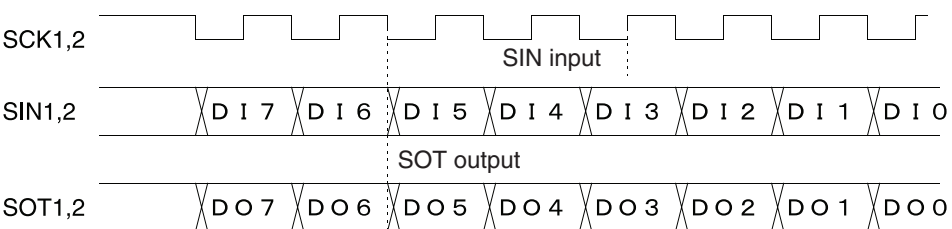
Figure 20.3-7 "Serial Data I/O Shift Timings" shows the serial data I/O shift timings in the LSB and MSB first modes.

Figure 20.3-7 Serial Data I/O Shift Timings

○ LSB first (when the BDS bit is "0")



○ MSB first (when the BDS bit is "1")



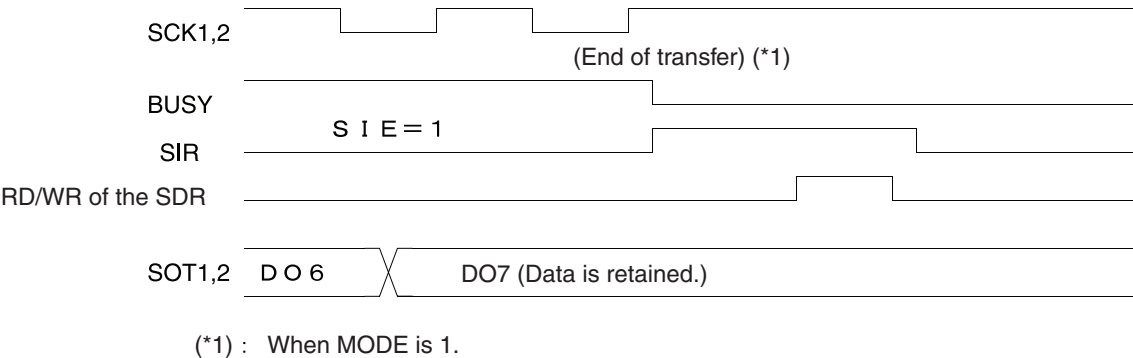
### 20.3.5 Interrupt Function of the Extended Serial I/O Interface

The extended serial I/O interface can issue interrupt requests to the CPU. At the end of data transfer, when the SIR bit is set and the SIE bit in the SMCS is "1", the extended serial I/O interface outputs an interrupt request to the CPU.

■ Interrupt Function of the Extended Serial I/O Interface

Figure 20.3-8 "Interrupt Signal Output Timing of the Extended Serial I/O Interface" shows the interrupt signal output timing of the extended serial I/O interface.

Figure 20.3-8 Interrupt Signal Output Timing of the Extended Serial I/O Interface





# CHAPTER 21 I<sup>2</sup>C INTERFACE

---

**This chapter describes the functions and operations of the I<sup>2</sup>C interface.**

---

21.1 "Overview of the I<sup>2</sup>C Interface"

21.2 "Block Diagram of the I<sup>2</sup>C Interface"

21.3 "I<sup>2</sup>C Interface Registers"

21.4 "Operation of the I<sup>2</sup>C Interface"



## 21.1 Overview of the I<sup>2</sup>C Interface

---

The I<sup>2</sup>C interface operates as a master or slave device on the I<sup>2</sup>C bus at the serial I/O port that supports an inter IC bus.

---

### ■ Features of the I<sup>2</sup>C Interface

MB90570 series micro controllers have one channel of I<sup>2</sup>C built-in interface.

The features of the I<sup>2</sup>C interface are:

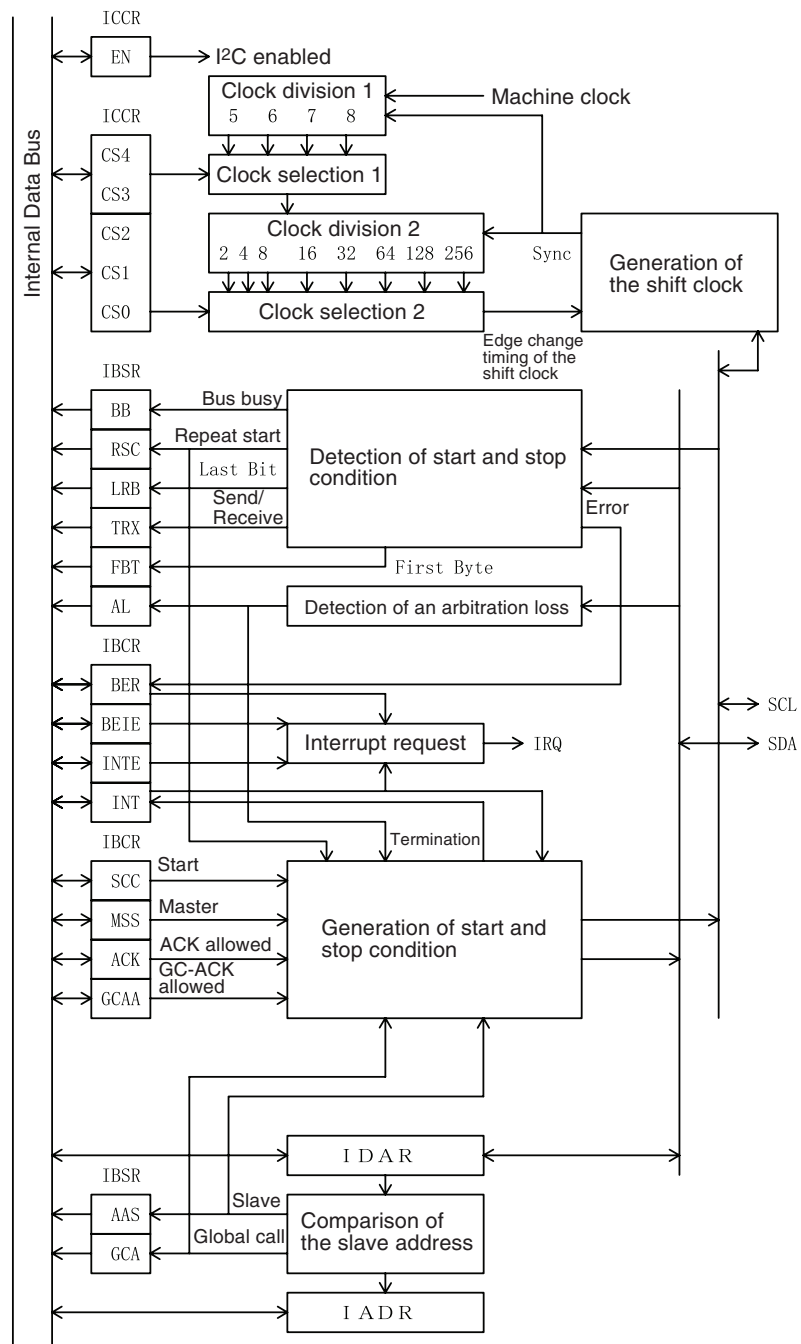
- Master/slave transmission
- Arbitration function
- Clock synchronization function
- Function to detect the slave address and general call address
- Function to detect the transfer direction
- Function to repeat or detect the start condition
- Detection function of bus errors
- The I<sup>2</sup>C corresponds to the standard mode (serial clock frequency: maximum 100 kHz) for the I<sup>2</sup>C.

## 21.2 Block Diagram of the I<sup>2</sup>C Interface

Figure 21.2-1 "Block Diagram of the I<sup>2</sup>C Interface" is a block diagram of the I<sup>2</sup>C Interface.

### ■ Block Diagram of the I<sup>2</sup>C Interface

Figure 21.2-1 Block Diagram of the I<sup>2</sup>C Interface



## 21.3 I<sup>2</sup>C Interface Registers

The I<sup>2</sup>C interface has 5 types of register as follows:

- Bus status register
- Bus control register
- Clock control register
- Address register
- Data register

### ■ Registers of the I<sup>2</sup>C Interface

**Figure 21.3-1 Registers of the I<sup>2</sup>C Interface**

Bus status register Address: 000068H	7	6	5	4	3	2	1	0	⇔ Bit No.
	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 Address: 000069H	15	14	13	12	11	10	9	8	⇔ Bit No.
	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 Address: 00006AH	7	6	5	4	3	2	1	0	⇔ Bit No.
	—	—	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 Address: 00006BH	15	14	13	12	11	10	9	8	⇔ Bit No.
	—	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 Address: 00006CH	7	6	5	4	3	2	1	0	⇔ Bit No.
	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)	

## 21.3.1 Bus Status Register (IBSR)

The bus status register (IBSR) has the following functions:

- Indicates the status of the I<sup>2</sup>C
- Detection of a repeated start condition
- Detection of arbitration losses
- Storing acknowledgements
- Detection of the first byte
- Detection of addressing
- Detection of the general call address
- Data transfer

### ■ Bus Status Register (IBSR)

**Figure 21.3-2 Bus Status Register (IBSR)**

Bus status register Address: 000068 <sub>11</sub>	7	6	5	4	3	2	1	0	⇐ Bit No.
	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)	

#### [bit 7]: BB (Bus Busy)

This bit shows the status of the I<sup>2</sup>C bus.

**Table 21.3-1 BB (Bus Busy) Bit Function**

BB	Function
0	Stop condition is detected. [Initial value]
1	Start condition is detected (indicating that the bus is in use).

#### [bit 6]: RSC (Repeated Start Condition)

This bit shows whether the repeated start condition is detected.

This bit is cleared when (1) "0" is written to the INT bit, (2) no addressing is made in slave mode, (3) the start condition is detected while the bus stops, or (4) the stop condition is detected.

**Table 21.3-2 RSC (Repeated Start Condition) Bit Function**

RSC	Function
0	Repeated start condition is not detected. [Initial value]
1	Start condition is detected again when the bus is in use.

**[bit 5]: AL (Arbitration Lost)**

This bit shows whether an arbitration loss is detected.

This bit is cleared by writing "0" to the INT bit.

**Table 21.3-3 AL (Arbitration Lost) Bit Function**

AL	Function
0	No arbitration loss is detected. [Initial value]
1	Either an arbitration loss has occurred during transmission by the master, or "1" has been written to the MSS bit when another system is using the bus.

**[bit 4]: LRB (Last Received Bit)**

This bit is used to store the acknowledgement.

This bit stores the acknowledgement from the receiving side.

This bit is cleared upon the detection of the start or stop condition.

**[bit 3]: TRX (Transfer/Receive)**

This bit shows the status of data transfer.

**Table 21.3-4 TRX (Transfer/Receive) Bit Function**

TRX	Function
0	Receiving [Initial value]
1	Sending

**[bit 2]: AAS (Addressed As Slave)**

This bit is used to detect addressing.

This bit is cleared by the detection of the start or stop condition.

**Table 21.3-5 ASS (Addressed As Slave) Bit Function**

AAS	Function
0	No addressing is made in slave mode. [Initial value]
1	Addressing is made in slave mode.

**[bit 1]: GCA (General Call Address)**

This bit is used to detect the general call address (00<sub>H</sub>).

This bit is cleared by the detection of the start or stop condition.

**Table 21.3-6 GCA (General Call Address) Bit Function**

GCA	Function
0	The general call address is not received in slave mode.
1	The general call address is received in slave mode.

**[bit 0]: FBT (First Byte Transfer)**

This bit is used to detect the first byte.

If this bit is set to "1" by the detection of the start condition, it is cleared when "0" is written to the INT bit or when no addressing is made in slave mode.

**Table 21.3-7 FBT (First Byte Transfer) Bit Function**

<b>FBT</b>	<b>Function</b>
0	The received data is not the first byte.
1	The received data is the first byte (the address data).

### 21.3.2 Bus Control Register (IBCR)

The bus control register has the following functions:

- Interrupt request and interrupt enable
- Generation of the start condition
- Selection between the master and slave
- Enable to generate acknowledgements

■ Bus Control Register (IBCR)

Figure 21.3-3 IBCR (Bus Control Register)

Bus control register Address: 000069 <sub>11</sub>	15	14	13	12	11	10	9	8	↔ Bit No.
	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)	

[bit 15]: BER (Bus Error)

This bit is for the bus error interrupt request flag.

When this bit is set, the EN bit of the CCR register is cleared, the I<sup>2</sup>C interface is suspended, and data transmission is aborted.

Table 21.3-8 BER (Bus Error) Bit Function

BER		Function
For writing	0	Clears the bus error interrupt request flag. [Initial value]
	1	No meaning
For reading	0	No bus error is detected. [Initial value]
	1	An invalid start or stop condition is detected during the data transmission.

[bit 14]: BEIE (Bus Error Interrupt Enable)

This bit is the interrupt enable bit for bus errors.

When this bit is "1" and the BER bit is "1", the interrupt occurs.

Table 21.3-9 BEIE (Bus Error Interrupt Enable) Bit Function

BEIE	Function
0	The bus error interrupt is disabled. [Initial value]
1	The bus error interrupt is enabled.

**[bit 13]: SCC (Start Condition Continue)**

This bit generates the start condition.

The read value of this bit is always "0".

**Table 21.3-10 SCC (Start Condition Continue) Bit Function when Writing**

SCC	Function
0	No meaning [Initial value]
1	Generates the start condition again when the master is transmitting.

**[bit 12]: MSS (Master Slave Select)**

This bit is used for selection between the master and slave.

This bit is cleared when an arbitration loss occurs during transmission by the master, thereby enabling slave mode.

**Table 21.3-11 MSS (Master Slave Select) Function**

MSS	Function
0	Slave mode is turned on after the generation of a stop condition and termination of the transmission.
1	Master mode is turned on, the start condition is generated, and transmission starts

**[bit 11]: ACK (ACKnowledge)**

This bit enables the generation of an acknowledgement upon receipt of data.

This bit is invalid when address data is received in slave mode.

**Table 21.3-12 ACK (ACKnowledge) Function**

ACK	Function
0	No acknowledgement is generated. [Initial value]
1	An acknowledgement is generated.

**[bit 10]: GCAA (General Call Address Acknowledge)**

This bit enables generation of an acknowledgement upon receipt of the general call address.

**Table 21.3-13 GCAA (General Call Address Acknowledge) Function**

GCAA	Function
0	No acknowledgement is generated. [Initial value]
1	An acknowledgement is generated.



**[bit 9]: INTE (INTerrupt Enable)**

This is the interrupt enable bit.

When this bit is "1" and the INT bit is "1", an interrupt occurs.

**Table 21.3-14 INTE (INTerrupt Enable) Function**

INTE	Function
0	Interrupts are disabled.
1	Interrupts are enabled.

**[bit 8]: INT (INTerrupt)**

This bit is for the interrupt request flag for transmission termination.

When this bit is "1", the SCL line stays at the "L" level. The next byte is transmitted after this bit is cleared by writing "0" and the SCL line is freed. In master mode, generation of the start or stop condition resets this bit to "0".

**Table 21.3-15 INT (INTerrupt) Function**

INT		Function
For writing	0	Clears the interrupt request flag for transmission termination. [Initial value]
	1	No meaning
For reading	0	The transmission is not terminated. [Initial value]
	1	This is set when transmission of one byte, including the acknowledgement bit, is terminated and any of the following conditions occur: <ul style="list-style-type: none"> <li>• The device is the bus master.</li> <li>• The device is the addressed slave.</li> <li>• The general call address is received.</li> <li>• An arbitration loss has occurred.</li> <li>• An attempt was made to generate the start condition while another system was using the bus.</li> </ul>

### ■ Competition among the SCC, MSS, and INT Bits

When the SCC, MSS, and INT bits are written to at the same time, there is competition for the transmission of the next byte, the start condition generation, and the stop condition generation. Prioritization is as follows:

1. Transmission of the next byte and stop condition generation
  - When "0" is written to the INT bit and MSS bit, the MSS bit takes precedence and the stop condition is generated.
2. Transmission of the next byte and start condition generation
  - When "0" is written to the INT bit and "1" to the SCC bit, the SCC bit takes precedence and the start condition is generated.
3. Start condition generation and stop condition generation
  - Writing "1" to the SCC bit and "0" to the MSS bit at the same time is prohibited.

### 21.3.3 Clock Control Register (ICCR)

The clock control register has the following functions:

- Enabling I<sup>2</sup>C interface operation
- Setting the frequency for the serial clock

#### ■ Clock Control Register (ICCR)

**Figure 21.3-4 Clock Control Register (ICCR)**

Clock control register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 00006A <sub>11</sub>	—	—	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)	

**[bit 7], [bit 6]:**

Unused bits

**[bit 5]: EN (ENable)**

This bit enables the I<sup>2</sup>C interface to operate.

When this bit is "0", the bits of the BSR register and the BCR register (except for the BER and BEIE bits) are cleared. When the BER bit is set, this bit is cleared.

**Table 21.3-16 EN (Enable) Bit Function**

EN	Function
0	Operation disabled
1	Operation enabled

**[bit 4] to [bit 0]: CS4-0 (Clock Period Select 4-0)**

This bit is used to set the frequency of the serial clock.

The frequency of the shift clock (fsck) is set according to the following calculation:

$$fsck = \frac{\phi}{m \times n + 4} \quad \phi : \text{Machine clock}$$

**Note:**

- Do not set the serial clock frequency to 100 kHz or more.
- Four extra (+4) cycles are the minimum overhead for checking the changes of the output level of the SCL pin. When the SCL pin starts up with a long delay, or when the slave device prolongs the clock, the value increases.

Table 21.3-17 "Settings of Serial Clock Frequency (CS4 and CS3)" and Table 21.3-18 "Settings of Serial Clock Frequency (CS2 to CS0)" show the values of m and n for CS4 to CS0.

**Table 21.3-17 Settings of Serial Clock Frequency (CS4 and CS3)**

m	CS4	CS3
5	0	0
6	0	1
7	1	0
8	1	1

**Table 21.3-18 Settings of Serial Clock Frequency (CS2 to CS0)**

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

When the values (m and n) are set, for example, to m=5 and n=32 with ( $\phi$ =16 MHz, the serial clock frequency is determined to be 97.561 kHz.

**Notes:**

According to the setting for the I<sup>2</sup>C operation enable bit (EN bit), output at the I<sup>2</sup>C common port pin is determined as described below.

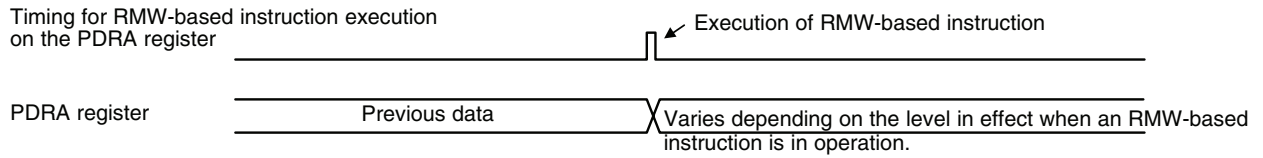
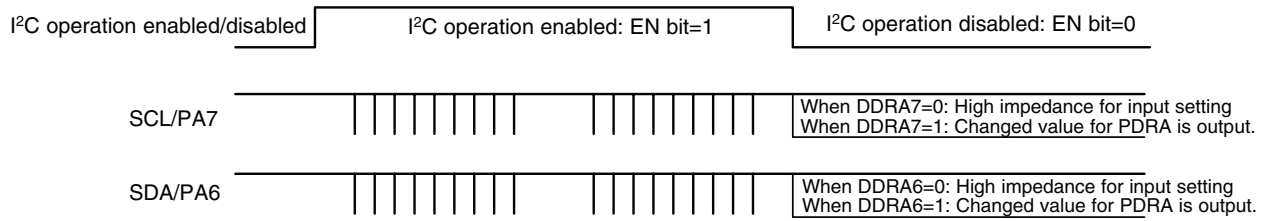
- When the operation is enabled (EN bit=1): An I<sup>2</sup>C output signal is output at the SDA/PA6 pin and the SCL/PA7 pin, irrespective of the input and output setting values for the DDRA (bit No.6) and the DDRA (bit No.7).
- When the operation is disabled (EN bit=0): If the output setting is such that the DDRA (bit No.6) equals 1 and the DDRA (bit No.7) equals 1, the setting values of PA6 and PA7 in the PDRA register are output at the SDA/PA6 pin and the SCL/PA7 pin, respectively.

**Notes:**

When the I<sup>2</sup>C is in operation and an RMW-based instruction is executed on the port data register (PDRA) in the same series as the I<sup>2</sup>C pin, the pin levels of PDRA bit No.6 and PDRA bit No.7 are loaded with a reading operation. Therefore, note that the values for PDRA bit No.6 and PDRA bit No.7 vary depending on the levels at the PA7/SCL pin and the PA6/SDA pin.

The following chart shows the change timing for the I<sup>2</sup>C common port.

- Operation timing for the I<sup>2</sup>C common port



See Appendix B.8 "F<sup>2</sup>MC-16LX Instruction List", for details on RMW-based instructions.

### 21.3.4 Address Register (IADR)

The address register (IADR) is used for specifying the slave address.

■ Address Register (IADR)

Figure 21.3-5 IADR (Address Register)

Address register  
Address:00006B<sub>11</sub>

15	14	13	12	11	10	9	8	⇔ Bit No.
—	A6	A5	A4	A3	A2	A1	A0	IADR

Read/write ⇔

Initial value ⇔

(-)

(R/W)

(R/W)

(R/W)

(R/W)

(R/W)

(R/W)

(R/W)

(-)

(X)

(X)

(X)

(X)

(X)

(X)

(X)

[bit 14] to [bit 8]: Slave address bits(A6-A0)

This register is used for specifying the slave address. In slave mode, the received address data is compared with the DAR register and if a match occurs an acknowledgement is sent to the master.

## 21.3.5 Data Register(IDAR)

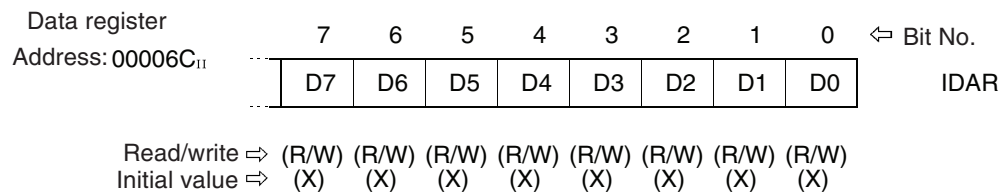
---

The data register (IDAR) is used for serial transmission.

---

### ■ Data Register (IDAR)

**Figure 21.3-6 Data Register (IDAR)**



#### [bit 7] to [bit 0]: Data bits (D7-D0)

The data register is used for serial transmission. Data is transferred, starting from the MSB. The data output value is "1" when data is being received (TRX=0).

The writing side of this register is double-buffered. When the bus is in use (BB=1), the written data is loaded in the register for serial transmission whenever one-byte data is transmitted. The data is read directly from the register for serial transmission, indicating the received data is available only when the INT bit is set.

## 21.4 Operation of the I<sup>2</sup>C Interface

---

The I<sup>2</sup>C bus is used for communication with two bi-directional bus lines, a serial data line (SDA) and a serial clock line (SCL). The I<sup>2</sup>C Interface has two corresponding open drain input-output pins (SDA and SCL) to support the wired logic.

---

### ■ Start Condition

If "1" is written to the MSS bit when the bus is free (BB=0 and MSS=0), the I<sup>2</sup>C interface is in master mode and the start condition is generated at the same time. In master mode, the start condition can be regenerated by writing "1" to the SCC bit even if the bus is in use (BB=1).

The start condition can be generated as follows:

- Write "1" to the MSS bit when the bus is not in use (MSS=0\*BB=0\*INT=0\*AL=0).
- Write "1" to the SCC bit in the interrupt state in bus master mode (MSS=1\*BB=1\*INT=1\*AL=0).

If "1" is written to the MSS bit when another system (being idle) is using the bus, the AL bit is set to "1". Under other conditions, writing "1" to the MSS bit or the SCC bit is ignored.

### ■ Stop condition

Writing "0" to the MSS bit in master mode (MSS=1) generates the stop condition and the mode switches to slave mode.

The condition necessary to generate the stop condition is as follows:

- Write "0" to the MSS bit in the interrupt state in bus master mode (MSS=1\*BB=1\*INT=1\*AL=0).

Under other conditions, writing "0" to the MSS bit is ignored.

### ■ Addressing

In master mode, after the start condition is generated, the BB and TRX bits are set to "1" and the contents of the IDAR register are output starting from the MSB. When the acknowledgement is received from the slave after the address data is sent, bit 0 of the send data (bit 0 of the sent IDAR register) is reversed and stored in the TRX bit.

In slave mode, after the start condition is generated, the BB bit is set to "1", the TRX bit is set to "0", and the send data from the master is received by the IDAR register. After the address data is received, the IDAR register is compared with the IADR register. If a match occurs, the AAS bit is set to "1" and the acknowledgement is sent to the master. Bit 0 of the received data (bit 0 of the received IDAR register) is then stored in the TRX bit.

### ■ Arbitration

Arbitration occurs if another master is sending data at the same time as the master. When the send data is "1" and the data on the SDA line is at "L" level, the sender assumes arbitration is lost and the AL bit is set to "1". As previously described, the AL bit is also set to "1" when the start condition is generated while the bus is in use. When the AL bit is set to "1", the MSS bit and TRX bit are set to "0" and the mode switches to slave receive mode.

### ■ Acknowledgement

An acknowledgement is sent from the receiver to the sender. When data is received, the use of acknowledgement can be selected by setting the ACK bit. When data is sent, the

acknowledgement from the receiver is stored in the LRB bit.

If the slave does not receive the acknowledgement from the master when sending data, the TRX bit is set to "0" and the mode switches to slave receive mode, thereby enabling the master to generate the stop condition when the slave frees the SCL line.

#### ■ Bus Error

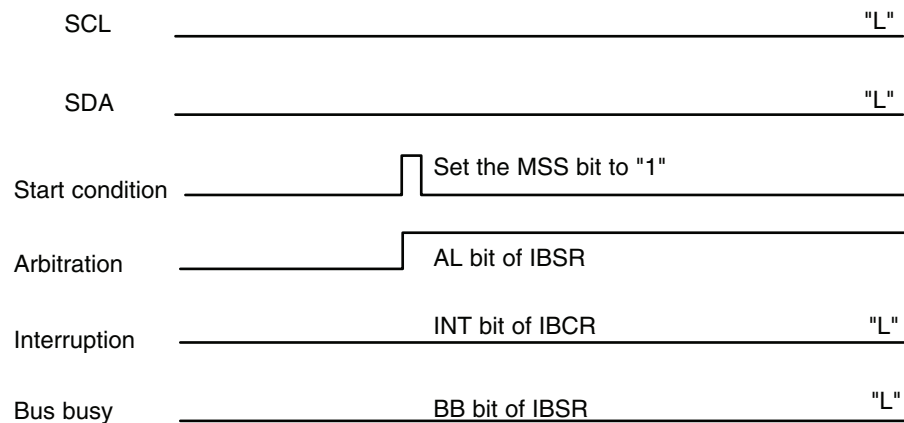
If any of the following occur, a bus error is determined and the I<sup>2</sup>C interface is halted:

- A basic specifications error is detected on the I<sup>2</sup>C bus during data transmission (including the ACK bit).
- The stop condition is detected in master mode.
- A basic specifications error is detected on the I<sup>2</sup>C bus when the bus is idle.

#### ■ Execution of Start Condition Generation Instruction When SDA="L" and SCL="L"

When a start condition generation instruction (for writing 1 in the MSS bit) is executed under the conditions SDA="L" and SCL="L", BB="0" and AL="1" are established. In this case, the transfer is not completed and so the transfer complete interrupt request flag (INT bit) is not set. This condition is detected by monitoring the BB and AL bits via the program.

**Figure 21.4-1 Change Timing for Each Flag when a Start Condition Generation Instruction is Executed Under SDA="L" and SCL="L"**



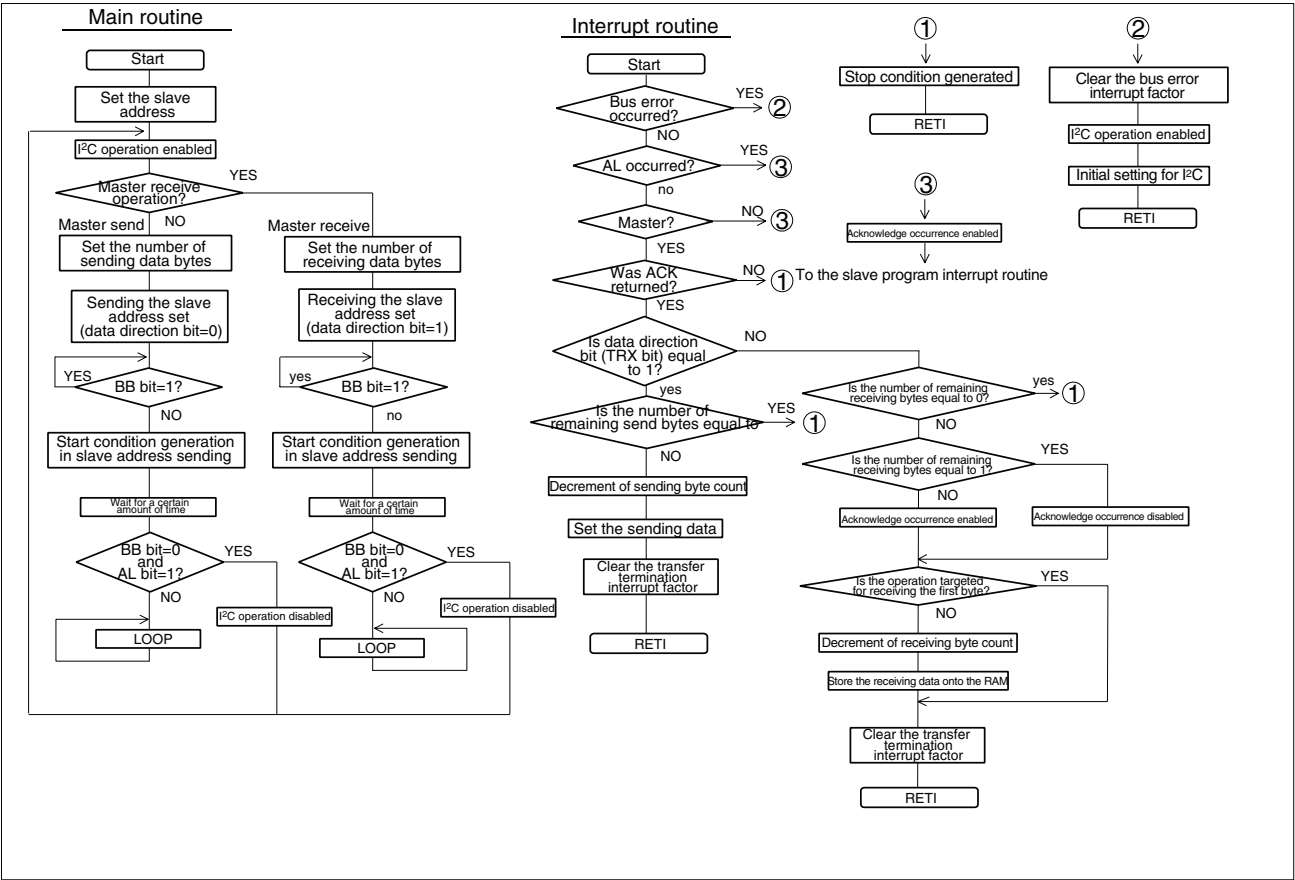


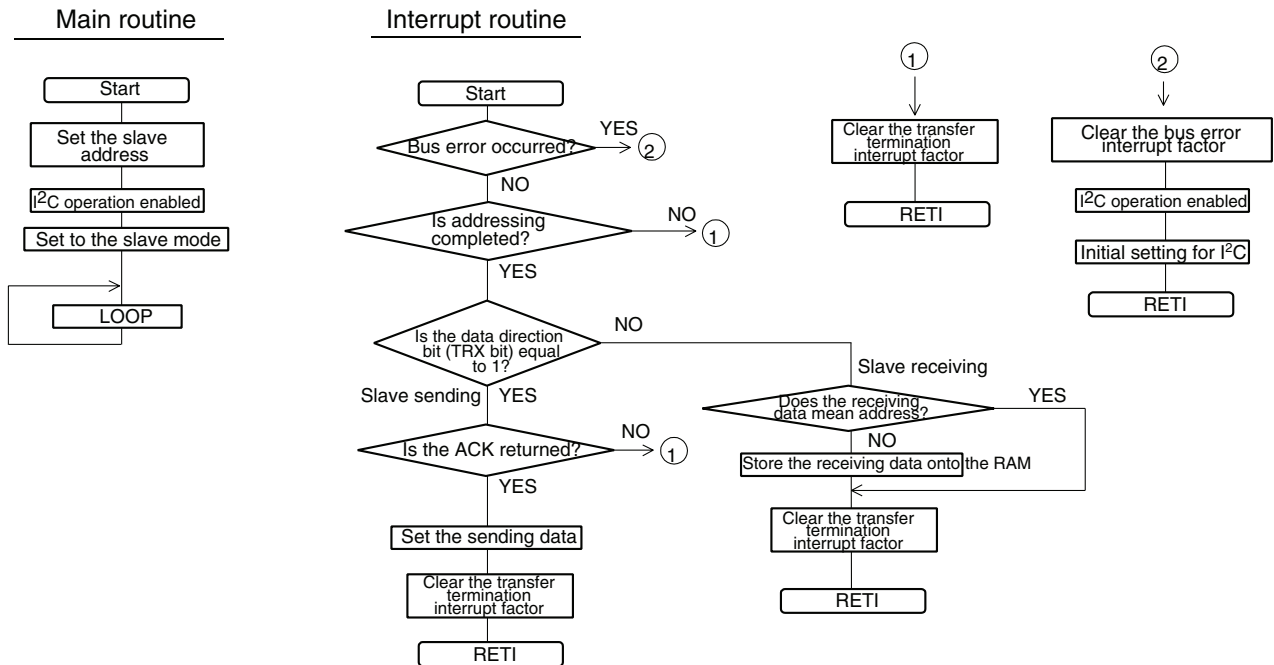
21.4.1 Operation Flow of the I<sup>2</sup>C Interface

Figure 21.4-2 "Examples: Operation Flow of Master Send/Receive Program (with Interruption) for I<sup>2</sup>C Interface" shows example of the operation flow of the master send/receive program (with interruption) for the I<sup>2</sup>C Interface. Figure 21.4-3 "Examples: Operation Flow of Slave Program (with Interruption) for I<sup>2</sup>C Interface" shows example of the operation flow of the slave program (with interruption) for the I<sup>2</sup>C Interface.

■ Operation Flow of the I<sup>2</sup>C Interface

Figure 21.4-2 Examples: Operation Flow of Master Send/Receive Program (with Interruption) for I<sup>2</sup>C Interface



**Figure 21.4-3 Examples: Operation Flow of Slave Program (with Interruption) for I<sup>2</sup>C Interface**



# CHAPTER 22    CHIP SELECT FUNCTION

---

**This chapter describes the functions and operations of the chip select function.**

---

22.1 "Overview of the Chip Select Function"

22.2 "Register of the Chip Select Function"

22.3 "Operation of the Chip Select Function"

22.4 "Decode Address Space of the Chip Select Function"

## 22.1 Overview of the Chip Select Function

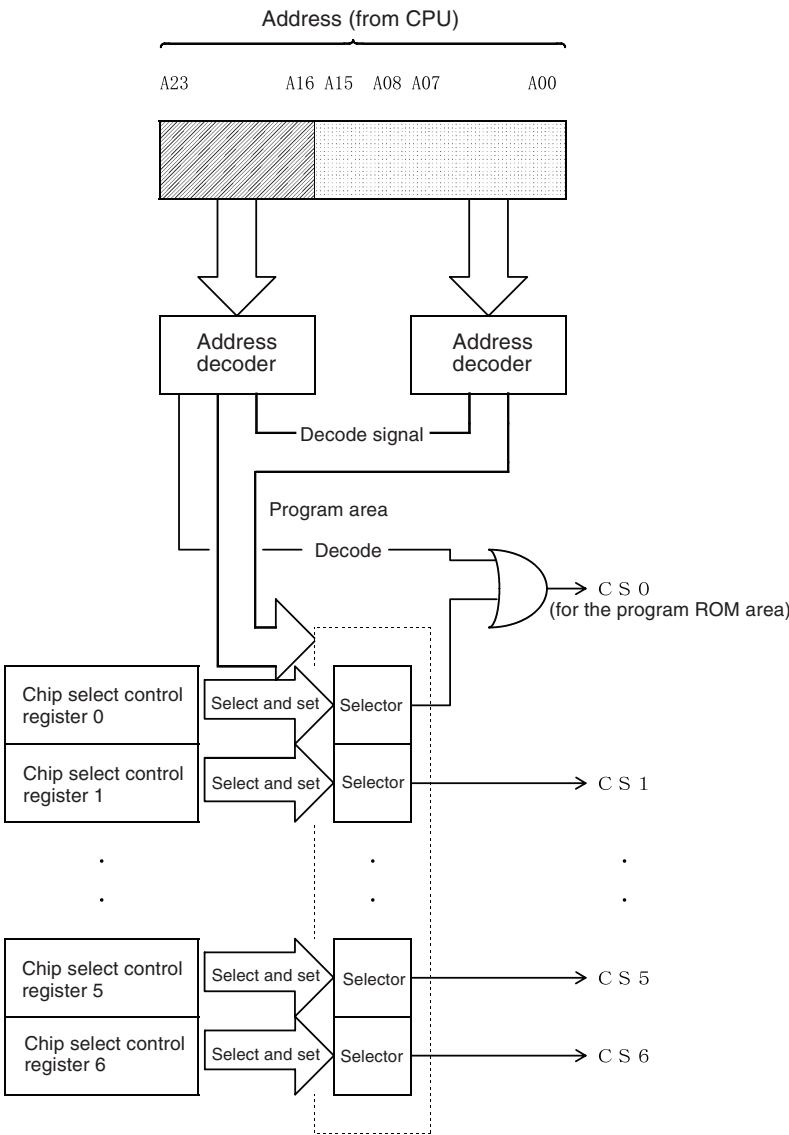
This chip select function module generates chip select signals to facilitate connection to memory or an I/O device.

### ■ Overview of the Chip Select Function

There are eight chip select output pins. The area specified by the hardware is set to each pin register, and a select signal is output from a pin upon the detection of access to the address.

### ■ Block Diagram of the Chip Select Function

Figure 22.1-1 Block Diagram of the Chip Select Function



## 22.2 Register of the Chip Select Function

The following is the only register of the chip select function:

- Chip select control register (CSCR0 to CSCR6)

### ■ Chip Select Control Register (CSCR0 to CSCR6)

**Figure 22.2-1 Chip Select Control Register (CSCR0 to CSCR6)**

	15	14	13	12	11	10	9	8	
Address: 000081 <sub>H</sub> : 000083 <sub>H</sub> : 000085 <sub>H</sub>	—	—	—	—	ACTL	OPEL	CSA1	CSA0	Chip select control register (Odd numbers: CSCR1/3/5)
	7	6	5	4	3	2	1	0	
Address: 000080 <sub>H</sub> : 000082 <sub>H</sub> : 000084 <sub>H</sub> : 000086 <sub>H</sub>	—	—	—	—	ACTL	OPEL	CSA1	CSA0	Chip select control register (Even numbers: CSCR0/2/4/6)

#### [bit 15/07 to 12/04]: Unused bits

These bits are not used, and the bit read values are not defined.

#### [bit 11/03]: ACTL

This bit is used to set the active level of the CS0 to CS6 pins. Operation is set as follows:

- "0": The CS0 to CS6 pins output "L" at decoding.
- "1": The CS0 to CS6 pins output "H" at decoding.

#### [bit 10/02]: OPEL

This bit is used to determine whether to enable the external output of the CS0 to CS6 pins. The settings are as follows:

- "0": Decode output from the CS0 to CS6 pins is prohibited.
- "1": Decode output from the CS0 to CS6 pins is allowed.

#### [bit 09/01 to 08/00] CSA1, CSA0

These bits render the address decode range (ROM/RAM/external circuit) of each chip select pins selectable. See Table 22.4-1 "Decode Address Spaces" for sizes and ranges.

## 22.3 Operation of the Chip Select Function

Chip select is activated according to the settings of the output setting register by decoding the highest and lowest bytes of the address of the data or program to which the CPU obtains access.

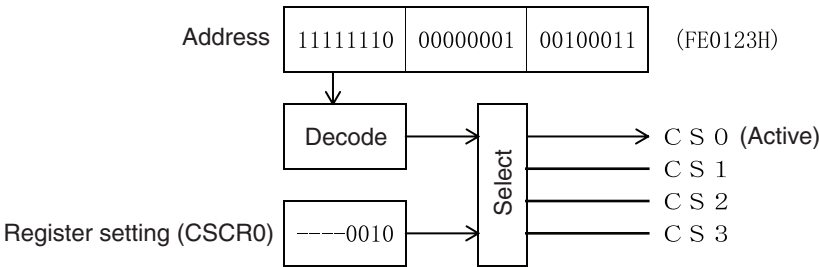
### ■ Operation of the Chip Select Function

The chip select function of this module is activating the signals of the CS0 to CS7 pins when the CPU obtains access to the area set in the CSA1 to 0 bits of the output setting register according to Table 22.4-1 "Decode Address Spaces". The output can be masked by the OPEL bit of the output setting register.

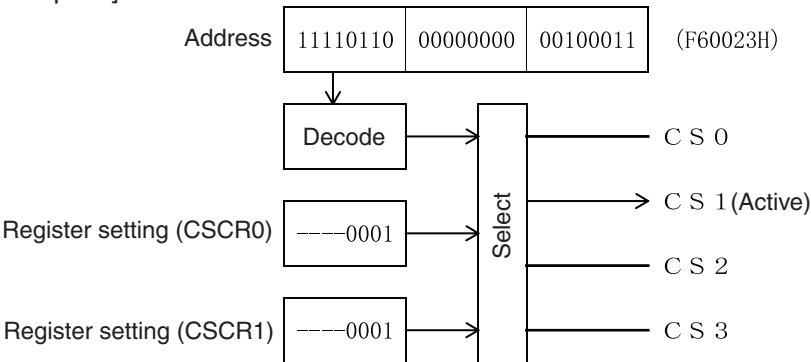
The active level of the CS0 to 7 pins can be altered using the ACTL bit of the output setting register.

Figure 22.3-1 Examples: Operation of the Chip Select Function

[Example 1]



[Example 2]



## 22.4 Decode Address Space of the Chip Select Function

Table 22.4-1 "Decode Address Spaces" shows decode address spaces of the chip select function. Figure 22.4-1 "Map of the Decode Address Spaces" shows a map of decode address spaces. Figure 22.4-2 "CS0 Output after a reset is cleared" shows CS0 output after a reset is cleared.

### ■ Decode Address Spaces of the Chip Select Function

Table 22.4-1 Decode Address Spaces

Pin	CSA		Decode space	Bytes of the area	Remark
	1	0			
CS0	0	0	F00000h to FFFFFFFh	1 Mbyte	This decode address space selection is activated when the program area or the program vector is to be fetched.
	0	1	F80000h to FFFFFFFh	512 Kbyte	
	1	0	FE0000h to FFFFFFFh	128 Kbyte	
	1	1	—	Prohibited	
CS1	0	0	E00000h to EFFFFFFh	1 Mbyte	This decode address space selection is used for the ROM or RAM area connection, or the external memory circuit connection.
	0	1	F00000h to F7FFFFh	512 Kbyte	
	1	0	FC0000h to FFFFFFFh	128 Kbyte	
	1	1	68FF80h to 68FFFFh	128 byte	
CS2	0	0	003000h to 003FFFh	4 Kbyte	This decode address space selection is used for the ROM or RAM area connection, or the external memory circuit connection.
	0	1	FA0000h to FBFFFFh	128 Kbyte	
	1	0	68FF80h to 68FFFFh	128 byte	
	1	1	68FF00h to 68FF7Fh	128 byte	
CS3	0	0	F80000h to F9FFFFh	128 Kbyte	This decode address space selection is used for the ROM or RAM area connection, or the external memory circuit connection.
	0	1	68FF00h to 68FF7Fh	128 byte	
	1	0	68FE80h to 68FEFFh	128 byte	
	1	1	—	Prohibited	
CS4	0	0	002800h to 002FFFh	2 Kbyte	This decode address space selection is used for the ROM or RAM area connection, or the external memory circuit connection.
	0	1	68FE80h to 68FEFFh	128 byte	
	1	0	—	Prohibited	
	1	1	—	Prohibited	



Table 22.4-1 Decode Address Spaces (Continued)

Pin	CSA		Decode space	Bytes of the area	Remark
	1	0			
CS5	0	0	68FF80h to 68FFFFh	128 byte	This decode address space selection is used for the ROM or RAM area connection, or the external memory circuit connection.
	0	1	—	Prohibited	
	1	0	—	Prohibited	
	1	1	—	Prohibited	
CS6	0	0	68FF00h to 68FF7h	128 byte	This decode address space selection is used for the ROM or RAM area connection, or the external memory circuit connection.
	0	1	—	Prohibited	
	1	0	—	Prohibited	
	1	1	—	Prohibited	
CS7	*	*	—	Prohibited	Prohibited

**Note:**

When the mode pin is set to external ROM, the CS0 pin outputs a decode signal in the space of addresses F00000<sub>H</sub> to FFFFFFF<sub>H</sub> (one megabyte), the area for program ROM, to fetch program vector immediately after reset (the initial status). In such a case, the CS0 pin must be dedicated to be used for the program ROM selection. The active level of this pin is "L" output.

In the initial status, the chip select decode signal pin is set to an input pin by the port function. Rewrite setting registers before using.

Figure 22.4-1 Map of the Decode Address Spaces

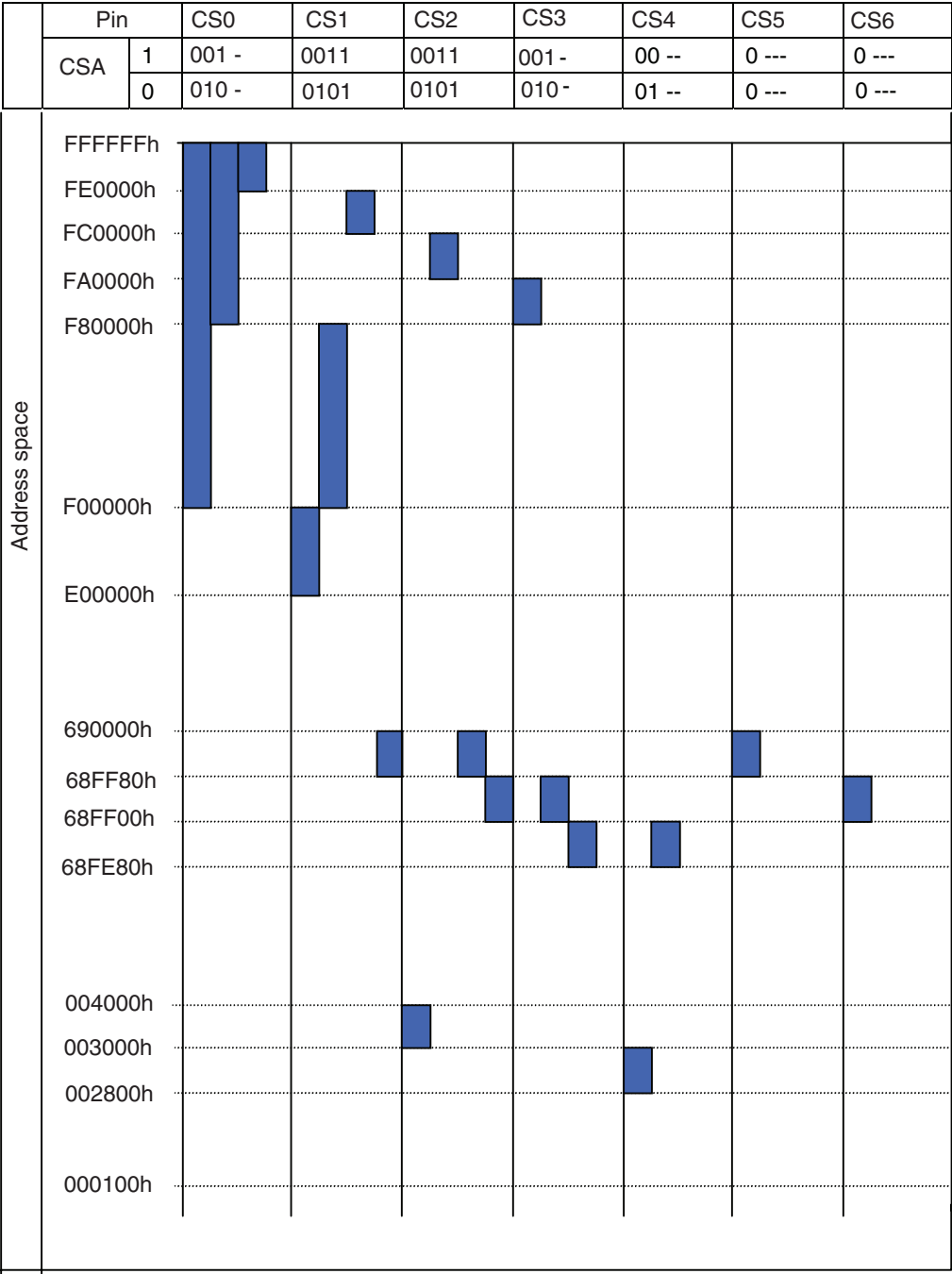
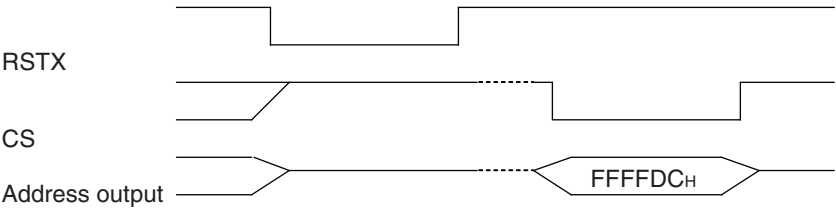


Figure 22.4-2 CS0 Output after a reset is cleared





## CHAPTER 23    CLOCK MONITOR FUNCTION

---

**This chapter describes the functions and operations of the clock monitor function.**

---

23.1 "Overview of the Clock Monitor Function"

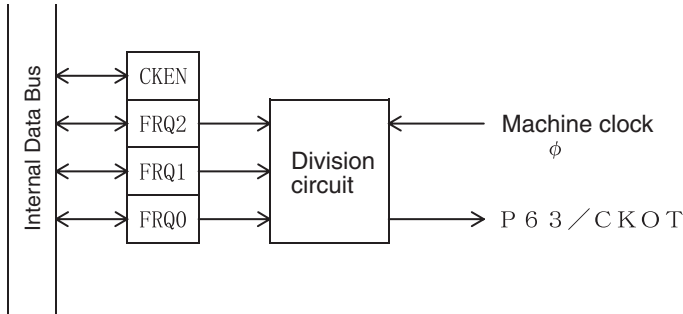
23.2 "Clock Output Enable Register (CLKR)"

## 23.1 Clock Monitor Function

The clock monitor function is output a division clock (a clock for monitoring) from the clock monitor CKOT pin.

■ Block Diagram of the Clock Monitor

Figure 23.1-1 Block Diagram of the Clock Monitor



## 23.2 Clock Output Enable Register (CLKR)

The clock output enable register (CLKR) selects the CKOT output enable and the clock output frequency.

### ■ Clock Output Enable Register (CLKR)

**Figure 23.2-1 Clock Output Enable Register (CLKR)**

Clock output enable register	7	6	5	4	3	2	1	0	↔ Bit No.
Address: 0003EH	—	—	—	—	CKEN	FRQ2	FRQ1	FRQ0	CLKR
Read/write ↔	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↔	(-)	(-)	(-)	(-)	(0)	(0)	(0)	(0)	

**[bit 7, 6, and 5] Unused bit**

**[bit 3] CKEN**

This is the CKOT output enable bit.

**Table 23.2-1 CKEN Bit Function**

CKEN	Function
0	Normal port
1	CKOT output

**[bit 2, 1, 0] FRQ2, FRQ1, FRQ0**

This bit is used to select the clock output frequency.

**Table 23.2-2 FRQ2, FRQ1, FRQ0 (Clock Output Frequency Selection Bit) Function**

FRQ2	FRQ1	FRQ0	Output clock	$\phi=16\text{ MHz}$	$\phi=8\text{ MHz}$	$\phi=4\text{ MHz}$
0	0	0	$2^1/\phi$	125 ns	250 ns	500 ns
0	0	1	$2^1/\phi$	250 ns	500 ns	1 $\mu\text{s}$
0	1	0	$2^3/\phi$	500 ns	1 $\mu\text{s}$	2 $\mu\text{s}$
0	1	1	$2^4/\phi$	1 $\mu\text{s}$	2 $\mu\text{s}$	4 $\mu\text{s}$
1	0	0	$2^5/\phi$	2 $\mu\text{s}$	4 $\mu\text{s}$	8 $\mu\text{s}$
1	0	1	$2^6/\phi$	4 $\mu\text{s}$	8 $\mu\text{s}$	16 $\mu\text{s}$
1	1	0	$2^7/\phi$	8 $\mu\text{s}$	16 $\mu\text{s}$	32 $\mu\text{s}$
1	1	1	$2^8/\phi$	16 $\mu\text{s}$	32 $\mu\text{s}$	64 $\mu\text{s}$



# CHAPTER 24 ADDRESS MATCH DETECTION FUNCTION

---

**This chapter describes the functions and operation of the address match detection function.**

---

24.1 "Overview of the Address Match Detection Function"

24.2 "Registers of the Address Match Detection Function"

24.3 "Operation of the Address Match Detection Function"

24.4 "Example of the Address Match Detection Function"

24.5 "Example and Flow of Program Patch Processing"

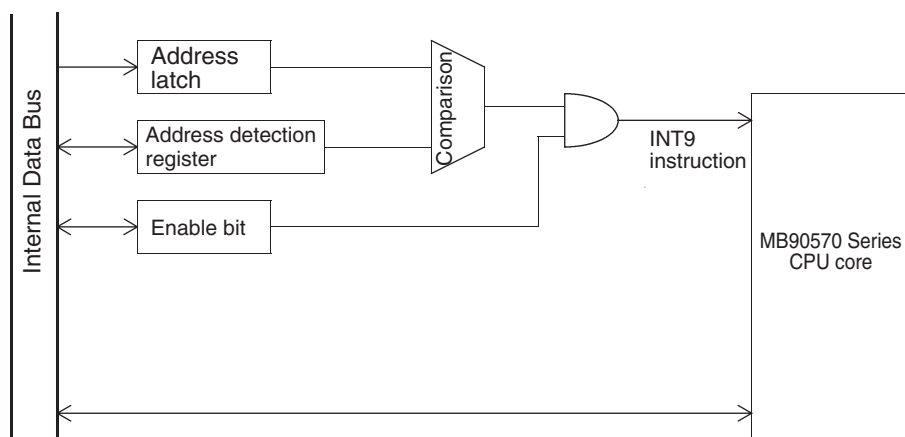


## 24.1 Overview of the Address Match Detection Function

When an address matches the value set in the address detection register, the instruction code to be read by the CPU is replaced with the INT9 instruction code (01<sub>H</sub>). The CPU then executes the INT9 instruction when executing a specified instruction. Program patching is enabled by processing the INT9 interrupt routine. Each of the two address detection registers has an interrupt enable bit. When an address matches the value set in the address detection register and the interrupt enable bit is "1", instruction code to be read by the CPU is replaced with the INT9 instruction code.

### ■ Block Diagram of the Address Match Detection Function

Figure 24.1-1 Block Diagram of the Address Match Detection Function



#### Note:

If the address detection register is set with a value other than the address of the first byte of the instruction, this function does not work normally, which causes the data of the set address to change to "01<sub>H</sub>", which in turn results in the execution of an incorrect instruction or access to an incorrect address.

Confirm that the interrupt enable bit is "0" before changing the settings of the address detection registers. If the setting is changed when the interrupt enable bit is "1", an incorrect address detection is performed during writing and an illegal operation may occur.

## 24.2 Registers of the Address Match Detection Function

There are two registers for the address match detection function.

- Program address detection register (PADR0 and PADR1).
- Program address detection control status register (PACSR).

### ■ Registers of the Address Match Detection Function

**Figure 24.2-1 Registers of the Address Match Detection Function**

- Program address detection register

	bit23	bit16	bit15	bit8	bit7	bit0	Access	Initial value
PADR0 Address: 1FF2H/ 1FF1H/ 1FF0H							R/W	Not defined
PADR1 Address: 1FF5H/ 1FF4H/ 1FF3H							R/W	Not defined

- Program address detection control status register

	7	6	5	4	3	2	1	0 ⇐	Bit No.
Address: 009EH	Reserved	Reserved	Reserved	Reserved	AD1E	Reserved	AD0E	Reserved	PACSR
Read/Write	⇒ (-)	⇒ (-)	⇒ (-)	⇒ (-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value	⇒ (0)	⇒ (0)	⇒ (0)	⇒ (0)	(0)	(0)	(0)	(0)	

### 24.2.1 Program Address Detection Registers (PADR0 and PADR1)

These registers compare the address with the value written in each register. If a match occurs when the interrupt enable bit corresponding to PACSR register is "1", and the CPU is requested to issue the INT9 instruction. When the corresponding interrupt enable bit is "0", an event does not occur.

■ Program Address Detection Registers (PADR0 and PADR1)

Figure 24.2-2 Program Address Detection Registers (PADR0 and PADR1)

- Program address detection register

	bit23	bit16	bit15	bit8	bit7	bit0	Access	Initial value
PADR0 Address: 1FF2H/ 1FF1H/ 1FF0H							R/W	Not defined
	bit23	bit16	bit15	bit8	bit7	bit0		
PADR1 Address: 1FF5H/ 1FF4H/ 1FF3H							R/W	Not defined

The correspondence of PADR0/1 Register and PACSR Register is as shown below:

Table 24.2-1 Correspondence of PADR0/1 Register and PACSR Register

Address detection register	Program address detection control status register (PACSR)
	Interrupt enable bit
PADR0	AD0E
PADR1	AD1E

## 24.2.2 Program Address Detection Control Status Register (PACSR)

---

This register controls the operation of the address detection function.

---

### ■ Program Address Detection Control Status Register (PACSR)

**Figure 24.2-3 Program Address Detection Control Status Register (PACSR)**

Program address detection control status register

	7	6	5	4	3	2	1	0	Bit No.
Address: 009EH	Reserved	Reserved	Reserved	Reserved	AD1E	Reserved	AD0E	Reserved	PACSR
Read/Write ⇒	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

#### [bit 7 to 4] Reserved bit

These bits are reserved. Always write "0".

#### [bit 3] AD1E (Address Detect register 1 Enable)

This is the interrupt enable bit of PADR1.

When this bit is "1", the address is compared with the PADR1 register. If a match occurs, the INT9 instruction is issued.

#### [bit 2] Reserved bit

These bits are reserved. Always write "0".

#### [bit 1] AD0E (Address Detect register 0 Enable)

This is the interrupt enable bit of PADR0.

When this bit is "1", the address and the PADR0 register are compared. If a match occurs, the INT9 instruction is issued.

#### [bit 0] Reserved bit

These bits are reserved. Always write "0".

## 24.3 Operation of the Address Match Detection Function

---

If the address is equal to the value specified in the address detection register, the address match detection function replaces the instruction code to be loaded into the CPU with the INT9 instruction code, "01<sub>H</sub>". As a result, when the CPU executes a specified instruction, the INT9 instruction is executed. Performing interrupt processing with the INT #9 interrupt routine will enable the patch application function.

---

### ■ Operation of the Address Match Detection Function

Each of the two address detection registers has an interruption enable bit. When the interruption enable bit is set to "1", the value specified in the address detection register is compared with the address. If the address comparison indicates matching, the instruction code to be loaded into the CPU is replaced with the INT9 instruction code.

### ■ Notes on the Address Match Detection Function

If the address specified for the address detection register does not match the address at the first byte of the instruction, the function does not operate correctly. Because the data at the specified address is changed to "01<sub>H</sub>", execution of an unwanted instruction or access to an unwanted address will occur.

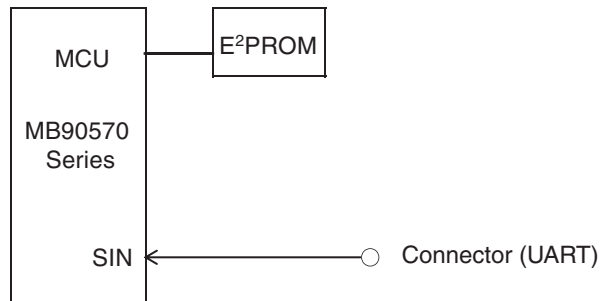
Also, changes to the address detection register should be made after the interruption enable bit is set to "0". If a write operation is performed when the interruption enable bit has the value "1", erroneous address detection will occur and the wrong operation may be executed.

## 24.4 Example of the Address Match Detection Function

The configuration of a system is shown as an example of the address match detection function.

### ■ System Configuration of the Address Match Detection Function

Figure 24.4-1 System Configuration Example of the Address Match Detection Function



### ■ E²PROM Memory Map

Table 24.4-1 E²PROM Memory Map

Address	Description
0000 <sub>H</sub>	Number of bytes of patch program No.0 (If 0, no program error exists.)
0001 <sub>H</sub>	Program Address No.0 bits 7 to 0
0002 <sub>H</sub>	Program Address No.0 bits 15 to 8
0003 <sub>H</sub>	Program Address No.0 bits 24 to 16
0004 <sub>H</sub>	Number of bytes of patch program No.1 (If 0, no program error exists.)
0005 <sub>H</sub>	Program Address No.1 bits 7 to 0
0006 <sub>H</sub>	Program Address No.1 bits 15 to 8
0007 <sub>H</sub>	Program Address No.1 bits 24 to 16
Up to 0010 <sub>H</sub> plus the number of bytes of patch program No. 0	Main body of patch program No. 0

### ■ Initial Status

E²PROM is set to all "0"s.

## CHAPTER 24 ADDRESS MATCH DETECTION FUNCTION

### ■ When a program error occurs:

The main body of the patch program and program address are transferred to the MCU through the connector (UART). The MCU writes the information to E<sup>2</sup>PROM.

### ■ Reset Sequence

The MCU reads the data of E<sup>2</sup>PROM after a reset. If the number of bytes of the patch program is not "0", the MCU reads the main body of the patch program and writes it to RAM. The program address is set to either PADDR0 or ADDR1, and the operation is allowed. The first address of the program written in RAM is held in the specified RAM of each address detection register.

### ■ INT9 Interrupt

The interrupt routine determines the address detection interrupted using the program counter (PC) value saved in stack and makes a branch to the corresponding program. The information stuck by the interrupt is discarded.

## 24.5 Example and Flow of Program Patch Processing

Figure 24.5-1 "Program Patch Processing" is an example of program patch processing, and Figure 24.5-2 "Flow of the Program Patch Processing" shows the flow of program patch processing.

### ■ Example and Flow of Program Patch Processing

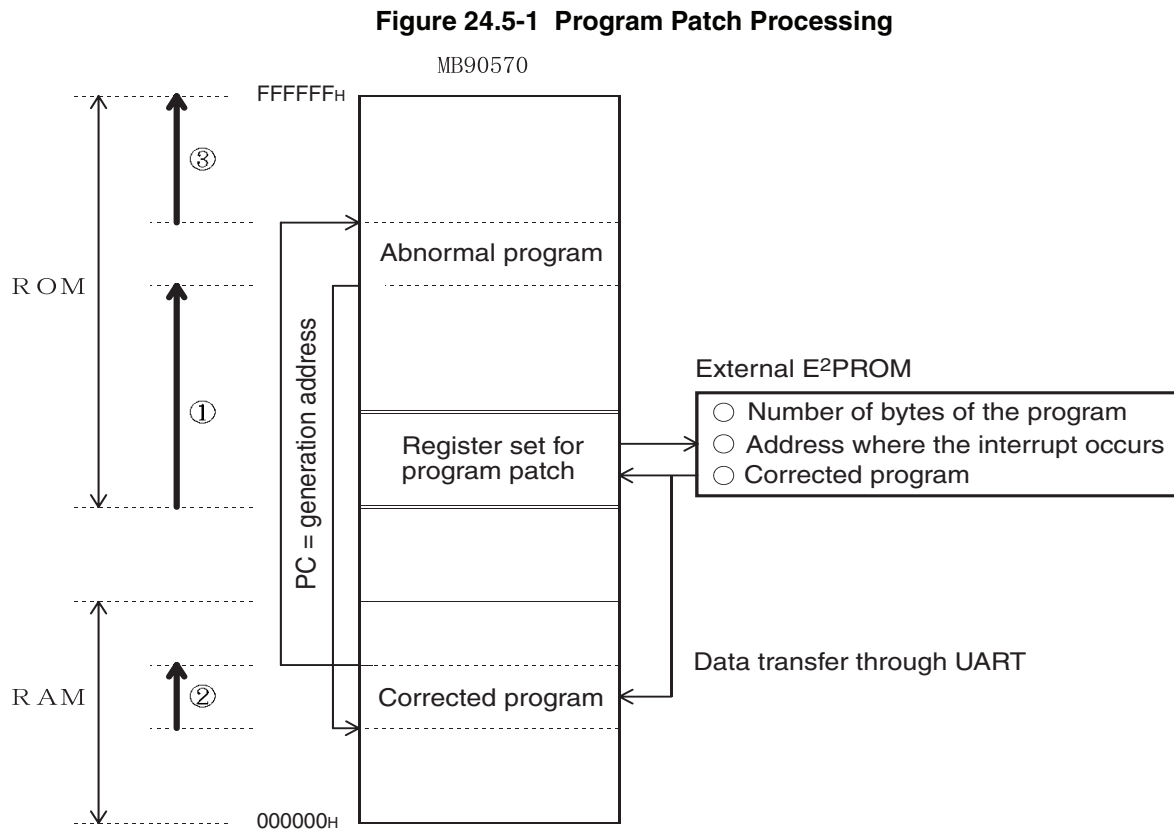
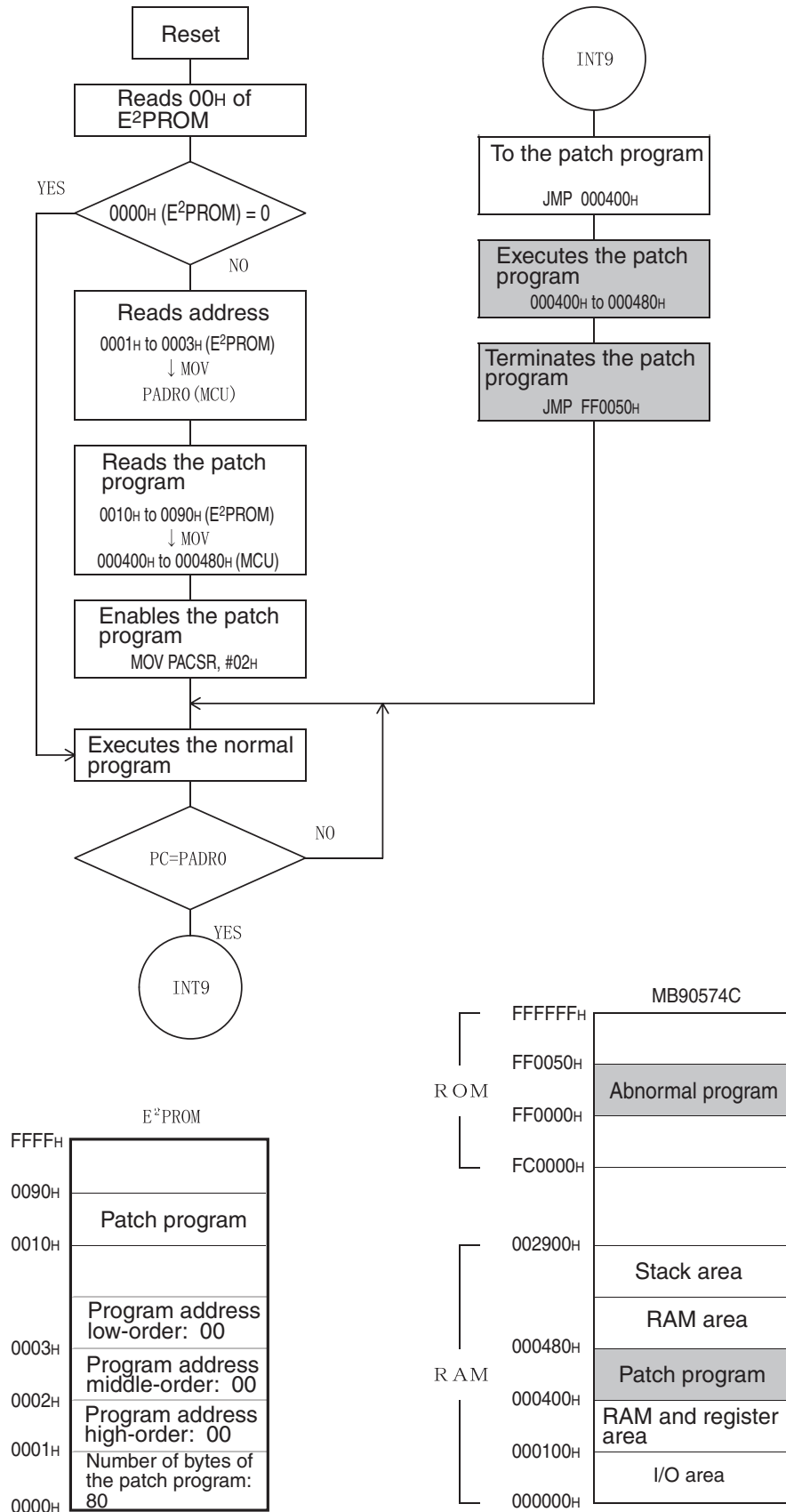




Figure 24.5-2 Flow of the Program Patch Processing



# CHAPTER 25 ROM MIRROR FUNCTION SELECTION MODULE

---

**This chapter describes the functions and operations of the ROM mirror function selection module.**

---

25.1 "Overview of the ROM Mirror Function Selection Module"

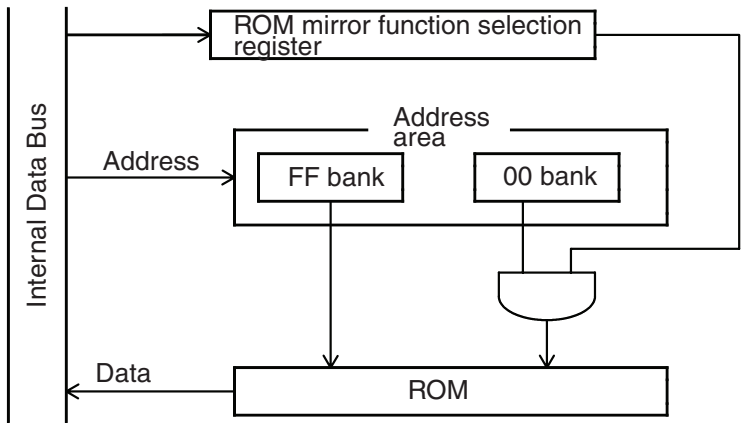
25.2 "Register of the ROM Mirror Function Selection Module"

# 25.1 Overview of the ROM Mirror Function Selection Module

The ROM mirror function selection module makes it possible to select whether to see the ROM data in the FF bank on the 00 bank side by setting the register.

■ Block Diagram of the ROM Mirror Function Selection Module

Figure 25.1-1 Block Diagram of the ROM Mirror Function Selection Module



## 25.2 Register of the ROM Mirror Function Selection Module

This section describes the register of the ROM mirror function selection module (ROMM).

### ■ Registers of the ROM Mirror Function Selection Module (ROMM)

**Figure 25.2-1 Registers of the ROM Mirror Function Selection Module (ROMM)**

Address:	15	14	13	12	11	10	9	8	↔ Bit No.
00006F <sub>H</sub>	—	—	—	—	—	—	—	MI	ROMM
Read/write ↔	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(W)	
Initial value ↔	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(1)	

**Note:**

Do not obtain access to this register during operation between 004000<sub>H</sub> and 00FFFF<sub>H</sub>.

**[bit 15 to 9] Unused bit**

**[bit 8] MI**

When "1" is written in this bit, the ROM data in the FF bank can be read from the 00 bank. When this bit is "0", this function of the 00 bank does not work.

This bit is for write only.

Figure 25.2-2 "Registers of the ROM Mirror Function Selection Module (ROMM)" shows the memory space in single chip mode shows the memory space in single chip mode.

**Note:**

Because the 00 bank is allocated at addresses 004000 to 00FFFF only and is mirrored at addresses FF4000 to FFFFFF, addresses FFF000 to FF3FFF are not mirrored regardless of the selection of the mirror function.

**Table 25.2-1 Memory Space Address**

	MB90573	MB90574/C	MB90F574/A	MB90V570/A
Address 1	FE0000 <sub>H</sub>	FC0000 <sub>H</sub>	FC0000 <sub>H</sub>	(FC0000 <sub>H</sub> )
Address 2	001800 <sub>H</sub>	002900 <sub>H</sub>	002900 <sub>H</sub>	002900 <sub>H</sub>

Figure 25.2-2 Memory Space in Single Chip Mode

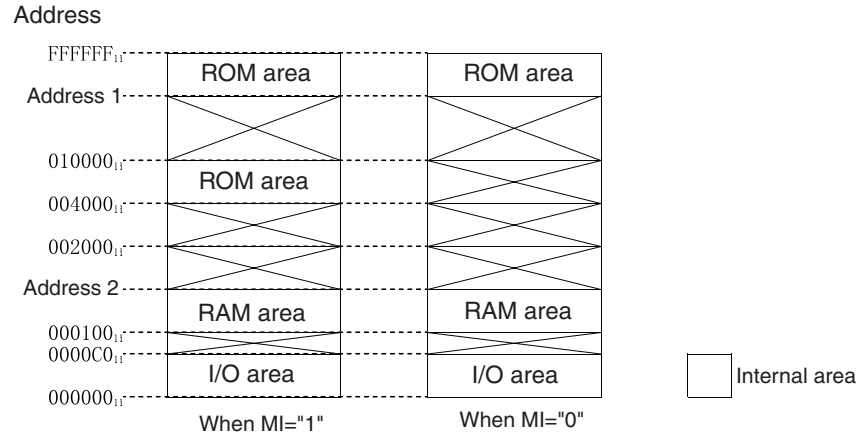
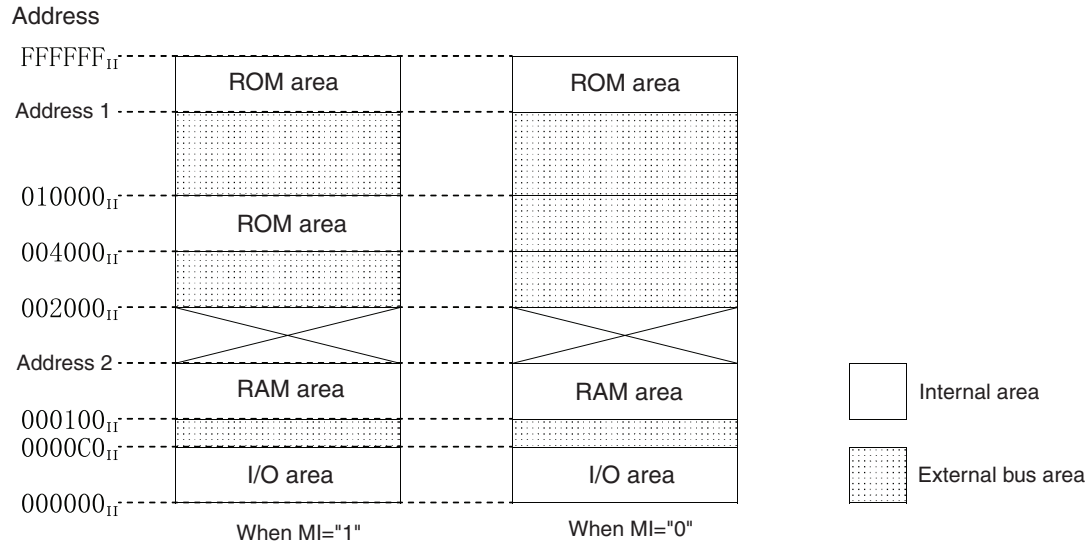


Figure 25.2-3 Memory Space in Internal ROM External Bus Mode



# CHAPTER 26 2M-BIT FLASH MEMORY

---

**This chapter describes the functions and operations of 2M-bit flash memory.**

**The following three methods are available for writing data to and erasing data from the flash memory:**

- 1. Parallel programmer**
- 2. Dedicated serial programmer**
- 3. Executing programs to write/erase data**

**This chapter describes "Executing programs to write/erase data".**

---

26.1 "Overview of the 2M-Bit Flash Memory"

26.2 "Sector Configuration of the 2M-Bit Flash Memory"

26.3 "Flash Memory Control Status Register (FMCS)"

26.4 "Method for Activating Flash Memory Automatic Algorithm"

26.5 "Checking Automatic Algorithm Execution Status"

26.6 "Detailed Explanation of Flash Memory Write/Erase"

26.7 "Example of Flash Memory Program"

# 26.1 Overview of the 2M-Bit Flash Memory

The 2M-bit flash memory is mapped to the FC<sub>H</sub> to FF<sub>H</sub> bank in the CPU memory map. The functions of the flash memory interface circuit enable read-access and program-access from the CPU in the same way as mask ROM. Instructions from the CPU can be used via the flash memory interface circuit to write data to and erase data from the flash memory. Internal CPU control therefore enables rewriting of the flash memory while it is mounted. As a result, improvements in programs and data can be performed efficiently.

## ■ Features of the 2M-Bit Flash Memory

- Sector configuration of "256K words x 8/128K words x 16 bits (16K + 512 x 2 + 7K + 8K + 32K + 64K + 64K + 64K)"
- Use of automatic program algorithm (Embedded Algorithm: Equivalent to MBM29F400TA)
- Erase pause/restart functions provided
- Detection of completion of writing/erasing using data polling or toggle bit functions
- Detection of completion of writing/erasing using CPU interrupts
- Compatible with JEDEC standard commands
- Sector erase function (any combination of sectors)
- Minimum of 10,000 write/erase operations

Embedded Algorithm is a trademark of Advanced Micro Device Corporation.

## ■ Flash Memory Write/Erase Method

Data write and read cannot be performed simultaneously in flash memory. That is, for data write and erase to and from flash memory, only a write operation without program access from flash memory is enabled by copying the program temporarily from flash memory to RAM and executing RAM.

## ■ Flash Memory Register

### ○ Flash memory control status register (FMCS)

Figure 26.1-1 Flash memory control status register (FMCS)

FMCS		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	↔	Bit No.
Address : 0000AE <sub>H</sub>		INTE	RDYINT	WE	RDY	Reserved	—	—	LPM		FMCS
Read/write	⇒	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(W)	(W)	(R/W)		
Initial value	⇒	(0)	(0)	(0)	(1)	(0)	(-)	(-)	(0)		

## 26.2 Sector Configuration of the 2M-Bit Flash Memory

Figure 26.2-1 "Sector Configuration of the 2M-Bit Flash Memory" shows the sector configuration of 2M-bit flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

### ■ Sector Configuration of the 2M-Bit Flash Memory

For access from the CPU, SA0 is allocated to the FC bank register, SA1 is allocated to the FD bank register, SA2 is allocated to the FE bank register, and SA3 to SA8 are allocated to the FF bank register.

**Figure 26.2-1 Sector Configuration of the 2M-Bit Flash Memory**

Flash memory	CPU address	Programmer address(*1)
SA8 (16K bytes)	FFFFFFH	7FFFFH
	FFC000H	7C000H
SA7 (512 bytes)	FFBFFFH	7BFFFH
	FFBE00H	7BE00H
SA6 (512 bytes)	FFBDFFH	7BDFFH
	FFBC00H	7BC00H
SA5 (7K bytes)	FFBBFFH	7BBFFH
	FFA000H	7A000H
SA4 (8K bytes)	FF9FFFH	79FFFH
	FF8000H	78000H
SA3 (32K bytes)	FF7FFFH	77FFFH
	FF0000H	70000H
SA2 (64K bytes)	FEFFFFH	6FFFFH
	FE0000H	60000H
SA1 (64K bytes)	FDFFFFH	5FFFFH
	FD0000H	50000H
SA0 (64K bytes)	FCFFFFH	4FFFFH
	FC0000H	40000H

#### \*1 Programmer address

When data is written to flash memory by the parallel programmer, the addresses corresponding to CPU addresses are programmer addresses. The general-purpose programmer uses programmer addresses to write and erase data.



### 26.3 Flash Memory Control Status Register (FMCS)

The FMCS, which exists in the flash memory interface circuit, is used when data is written to or erased from flash memory.

■ Flash Memory Control Status Register (FMCS)

Figure 26.3-1 Flash Memory Control Status Register (FMCS)

FMCS		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	⇐ Bit No.
Address : 0000AE <sub>H</sub>		INTE	RDYINT	WE	RDY	Reserved	—	—	LPM	FMCS
Read/write	⇒	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(W)	(W)	(R/W)	
Initial value	⇒	(0)	(0)	(0)	(1)	(0)	(-)	(-)	(0)	

○ Description of bits

[bit 7] INTE (INTerrupt Enable)

Bit 7 interrupts the CPU when flash memory read/erase terminates.

When this bit and the RDYINT bit are "1", the CPU is interrupted.

When this bit is "0", the CPU is not interrupted.

0	Interrupts are enabled when write/erase terminates.
1	Interrupts are disabled when write/erase terminates.

[Bit 6] RDYINT (ReaDY INTerrupt)

Bit 6 represents the operating status of flash memory.

When flash memory write/erase terminates, this bit is set to "1". When this bit is "0" after flash memory read/erase has terminated, data cannot be written to and erased from flash memory. After setting this bit to "1" (i.e., after flash memory write/erase has terminated), data write to and erase from flash memory is enabled. This bit is cleared to "0" when "0" is written. Writing "1" is ignored. This bit is set to "1" when flash memory automatic algorithm (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") terminates.

When the read modify write (RMW) instruction is used, "1" is always read.

0	Write/erase operation is being executed.
1	Write/erase operation terminated (an interrupt request was issued).

[Bit 5] WE (Write Enable)

Write enable bit to the flash memory area.

When this bit is 1, writing after the command sequence (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") is issued to the FC to FF bank writes to the flash memory area. When this bit is "0", write/erase signal is not issued.

### 26.3 Flash Memory Control Status Register (FMCS)

This bit is used to activate the write/erase command for flash memory. If write/erase is not performed, this bit should always be set to "0" so that data is not inadvertently written to flash memory.

0	Flash memory write/erase is prohibited.
1	Flash memory write/erase is allowed.

#### [Bit 4] RDY (ReaDY)

The RDY bit allows flash memory write/erase.

When this bit is "0", flash memory write/erase cannot be performed. Even if this bit is "0", the read/reset command and suspend commands (e.g., sector erase temporary stop) are accepted.

0	Write/erase operation is being executed.
1	Write/erase operation terminated (write/erase of subsequent data was allowed).

#### [Bit 3] Reserved bit

Bit 3 is reserved for tests. For normal use, set this bit to "0".

#### [Bits 2 and 1] Free bits

For normal use, set these bits to "0".

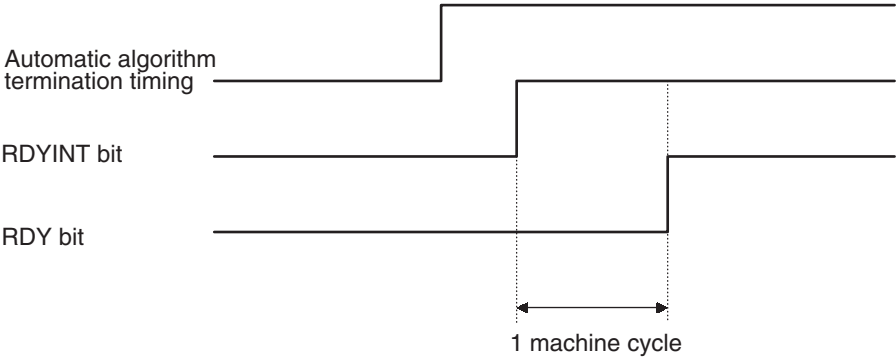
#### [Bit 0] LPM (Low Power Mode)

Setting LPM bit to "1" minimizes the select signal to flash memory when flash memory is accessed, thereby suppressing the power consumption of the main unit of flash memory. However, because the access time when this bit is 1 is considerably longer than the access time when this bit is 0, flash memory access is impossible when the CPU operates at high speed. Operate the CPU at a frequency of 4 MHz or lower when using this mode. When the low power mode switches to the sub-block mode, setting by software is unnecessary because this bit is set to "1" automatically.

0	Normal power consumption mode
1	Low-power consumption mode (The CPU operates at an internal operating frequency of 4 MHz or lower.)

#### Note:

The RDYINT bit and RDY bit do not change at the same time. Code the program so that a decision is made in accordance with one of these bits.



## 26.4 Method for Activating Flash Memory Automatic Algorithm

There are four commands (read/reset, write program, chip erase, and sector erase) for activating flash memory automatic algorithm. The sector erase command is further divided into a sector erase temporary stop command and a sector erase restart command.

### ■ Command Sequence Table

Table 26.4-1 "Command Sequence Table" lists the commands used when data is written to and erased from flash memory. All data to be written to the command register is represented in bytes but the data must be written by using word access. In this case, high-order byte data is ignored.

**Table 26.4-1 Command Sequence Table**

Command sequence	Bus write cycle	1st bus cycle		2nd bus cycle		3rd bus cycle		4th bus cycle		5th bus cycle		6th bus cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/ reset(*1)	1	FxXXXX	XXF0	—	—	—	—	—	—	—	—	—	—
Read/ reset(*1)	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	—	—	—	—
Write program	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	—	—	—	—
Chip erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector erase temporary stop		When Address "FxXXXX" Data (xxB0 <sub>H</sub> ) is entered, sector erase is temporarily stopped.											
Sector erase restart		When Address "FxXXXX" Data (xx30 <sub>H</sub> ) is entered, sector erase is temporarily stopped and then restarted.											

#### Note:

Address Fx in the table indicates FF, FE, FD, or FC. When specifying these addresses, use the relevant value that indicates the bank to be accessed.

Addresses in the table are values on the CPU memory map. All addresses and data are represented as hexadecimal numbers. "X", however, is an arbitrary value.

RA: Read address

PA: Only write addresses and even-number addresses can be specified.

SA: Sector address. See Section 26.2 "Sector Configuration of the 2M-Bit Flash Memory".

RD: Read data

PD: Only write data and word data can be specified.

\*1: Both of these read/reset commands can reset flash memory in the read mode.

## 26.5 Checking Automatic Algorithm Execution Status

Flash memory has hardware for reporting the operating status in flash memory and the completion of operation because write/erase is executed by automatic algorithm. This algorithm can use the hardware sequence flag explained below to check the operating status of flash memory.

### ■ Hardware Sequence Flag

The hardware sequence flag consists of the outputs of four bits (DQ7, DQ6, DQ5, and DQ3). The DQ7 bit has a data polling flag function. The DQ6 bit has a toggle bit flag function. The DQ5 bit has a timing limit excess flag function. The DQ3 bit has a sector erase timer flag function. These functions enable the user to check whether write/chip, sector erase termination, and erase code write are valid.

The hardware sequence flag can be referenced by read-accessing the target sector address in flash memory after setting the command sequence (see Table 26.5-1 "Bit Assignment of Hardware Sequence Flag" in Section 26.4 "Method for Activating Flash Memory Automatic Algorithm").

Table 26.5-1 "Bit Assignment of Hardware Sequence Flag" shows the bit assignment of the hardware sequence flag.

**Table 26.5-1 Bit Assignment of Hardware Sequence Flag**

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	—	DQ3	—	—	—

It can be determined whether an automatic write/chip or sector erase is being executed or a write is being terminated by checking the hardware sequence flag or the RDY bit of the flash memory control register (FMCS). When the automatic write/chip or sector erase has terminated, the read/reset status is restored.

When creating a program, determine whether the automatic write/chip or sector erase terminated with a flag and then execute the next processing, such as data read. The validity of the second and subsequent sector erase code write operations can also be determined by the hardware sequence flag. Table 26.5-2 "Hardware Sequence Flag Functions" lists hardware sequence flag functions.

Table 26.5-2 Hardware Sequence Flag Functions

Status		DQ7	DQ6	DQ5	DQ3
Status change at normal operation	Write operation ---> write completion (Valid when a write address is specified)	$\overline{\text{DQ7}}$ ---> DATA:7	Toggle ---> DATA:6	0 ---> DATA:5	0 ---> DATA:3
	Chip or sector erase operation ---> erase completion	0 ---> 1	Toggle ---> DATA:6	0 ---> 1	1
	Sector erase wait --->erase start	0	Toggle	0	0 ---> 1
	Erase operation ---> sector erase temporary stop (Sector being erased)	0 ---> 1	Toggle ---> 1	0	1 ---> 0
	Sector erase temporary stop ---> erase restart (Sector being erased)	1 ---> 0	1 ---> Toggle	0	0 ---> 1
	Sector erase temporary stop in progress (Sector not being erased)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Write operation	$\overline{\text{DQ7}}$	Toggle	1	0
	Chip or sector erase operation	0	Toggle	1	0

The hardware sequence flag is explained in the next and subsequent sections.

## 26.5.1 Data Polling Flag (DQ7)

The data polling flag indicates whether an automatic algorithm is being executed or is in the termination state.

### ■ At Write Operation

If read access is attempted when automatic write algorithm is being executed, flash memory outputs the reverse data of bit 7 of the most recent written data without reference to the indicated address. If read access is attempted when automatic write algorithm terminates, flash memory outputs bit 7 of the read value of the indicated address.

### ■ At Chip Erase or Sector Erase Operation

If read access is attempted when chip erase algorithm is being executed, flash memory outputs "0" without reference to the indicated address. If read access is attempted when sector erase algorithm is being executed, flash memory outputs "0" from the sector being erased, and when chip or sector erase operation terminates, flash memory outputs "1".

### ■ At Sector Erase Temporary Stop

If read access is attempted when sector erase is stopped temporarily, flash memory outputs "1" when the indicated address is being erased. If the sector is not being erased, flash memory outputs bit 7 (DATA:7) of the read value of the indicated address. A temporary stop or an erase of a sector can be detected by referencing this flag with the toggle bit flag (DQ6).

### ■ Transition of Data Polling Flag States

Table 26.5-3 "Transition of Data Polling Flag States at Normal Operation" shows the transition of data polling flag states at normal operation. Table 26.5-4 "Transition of Data Polling Flag States at Abnormal Operation" shows the transition of data polling flag states at abnormal operation.

**Table 26.5-3 Transition of Data Polling Flag States at Normal Operation**

Operating state	Write operation ---> completion	Chip or sector erase - --> completion	Sector erase wait ---> start	Sector erase ---> erase temporary stop (sector being erased)	Sector erase temporary stop ---> restart (sector being erased)	Sector erase temporary stop in progress (sector not being erased)
DQ7	$\overline{\text{DQ7}} \text{ ---> DATA:7}$	0 ---> 1	0	0 ---> 1	1 ---> 0	DATA:7

Table 26.5-4 Transition of Data Polling Flag States at Abnormal Operation

Operating state	Write operation	Chip or sector erase operation
DQ7	$\overline{\text{DQ7}}$	0

**Note:**

When automatic algorithm is activated, read access to the specified address is ignored. For data read, output of other bits is enabled when the data polling flag (DQ7) terminates. For this reason, a data read after automatic algorithm termination must be executed following the read access that checked data polling.



## 26.5.2 Toggle Bit Flag (DQ6)

As with the data polling flag, the toggle bit flag (function) indicates whether automatic algorithm is being executed or is in the termination state.

### ■ At Write/Chip or Sector Erase Operation

If read access is continuously attempted when automatic write algorithm or chip or sector erase algorithm is being executed, flash memory outputs the toggle state in which "1" and "0" are alternately output for each read. Flash memory performs this output without reference to the indicated address. If read access is continuously attempted after automatic write algorithm or chip or sector erase algorithm has terminated, flash memory stops the toggle operation of bit 6 and outputs bit 6 (DATA:6) of the read value of the indicated address.

### ■ At Sector Erase Temporary Stop

If read access is attempted when sector erase is temporarily stopped, flash memory outputs "1" when the indicated address belongs to the sector being erased. If the indicated address does not belong to the sector being erased, flash memory outputs bit 6 (DATA:6) of the read value of the indicated address.

#### Tip:

If the sector to be written is write-protected, the toggle bit performs a toggle operation for about 2  $\mu$ s and then terminates the operation without rewriting data.

If all of the selected sectors are write-protected, the toggle bit performs a toggle operation for about 100  $\mu$ s and then returns to the read/reset state without rewriting data.

### ■ Transition of Toggle Bit Flag States

Table 26.5-5 "Transition of Toggle Bit Flag States at Normal Operation" shows the transition of toggle bit flag states at normal operation. Table 26.5-6 "Transition of Toggle Bit Flag States at Abnormal Operation" shows the transition of toggle bit flag states at abnormal operation.

**Table 26.5-5 Transition of Toggle Bit Flag States at Normal Operation**

Operating state	Write operation ---> completion	Chip or sector erase ---> completion	Sector erase wait ---> start	Sector erase ---> erase temporary stop (sector being erased)	Sector erase temporary stop ---> restart (sector being erased)	Sector erase temporary stop in progress (sector not being erased)
DQ6	Toggle ---> DATA:6	Toggle ---> Stop	Toggle	Toggle ---> 1	1 ---> Toggle	DATA:6

**Table 26.5-6 Transition of Toggle Bit Flag States at Abnormal Operation**

Operating state	Write operation	Chip or sector erase operation
DQ6	Toggle	Toggle

### 26.5.3 Timing Limit Excess Flag (DQ5)

The timing limit excess flag indicates that automatic algorithm execution exceeds the specified time (internal pulse count) in flash memory.

#### ■ At Write/Chip or Sector Erase

If read access is attempted after write automatic algorithm or chip or sector erase automatic algorithm has been activated, flash memory outputs "0" when automatic algorithm execution does not exceed the specified time (time required for write or erase). If automatic algorithm execution exceeds the specified time, flash memory outputs "1". Because this output processing is performed without reference to whether automatic algorithm is being executed or is in the termination state, it can be determined whether write or erase is successful or unsuccessful. If automatic algorithm is being executed by the data polling function or toggle bit function when "1" is output, it can be determined that write is unsuccessful.

If an attempt is made to write "1" to the flash memory address to which "0" is already written, for example, a fail occurs. In this case, flash memory is locked but automatic algorithm is not terminated, and so valid data is not output from the data polling flag (DQ7). The time limit is also exceeded because the toggle bit flag (DQ6) does not stop toggle operation. In this case, the timing limit excess flag (DQ5) outputs "1". This state indicates that flash memory was not used correctly, but does not indicate that flash memory is defective. If this state occurs, execute the reset command.

#### ■ Transition of Timing Limit Excess Flag States

Table 26.5-7 "Transition of Timing Limit Excess Flag States at Normal Operation" shows the transition of timing limit excess flag states at normal operation. Table 26.5-8 "Transition of Timing Limit Excess Flag States at Abnormal Operation" shows the transition of timing limit excess flag at abnormal operation.

**Table 26.5-7 Transition of Timing Limit Excess Flag States at Normal Operation**

Operating state	Write operation ---> completion	Chip or sector erase ---> completion	Sector erase wait ---> start	Sector erase ---> erase temporary stop (sector being erased)	Sector erase temporary stoop ---> restart (sector being erased)	Sector erase temporary stop in progress (sector not being erased)
DQ5	0 ---> DATA:5	0 ---> 1	0	0	0	DATA:5

**Table 26.5-8 Transition of Timing Limit Excess Flag States at Abnormal Operation**

Operating state	Write operation	Chip or sector erase operation
DQ5	1	1

## 26.5.4 Sector Erase Timer Flag (DQ3)

The sector erase timer flag indicates whether the sector erase wait period is exceeded after the sector erase command has been activated.

### ■ At Sector Erase Operation

If read access is attempted after the sector erase command has been activated, flash memory outputs "0" when the sector erase wait period is not exceeded. If the sector erase wait period is exceeded, flash memory outputs "1". Flash memory performs this output processing without reference to the address indicated by the address signal of the sector issuing the sector erase command.

If this flag is "1" when erase algorithm is being executed by the data polling function or toggle bit function, erase has already started in flash memory. In this case, the subsequent writing of sector erase codes and commands other than the erase temporary stop command are ignored.

When this flag is "0", flash memory accepts the writing of additional sector erase codes. To check this acceptance, the state of this flag should be checked before the subsequent sector erase codes are written. If this flag is found to be "1" as a result of the second check, the erase codes of additional sectors may not be accepted.

### ■ At Sector Erase Operation

If read access is attempted when sector erase is temporarily stopped, flash memory outputs "1" when the indicated address belongs to the sector being erased. If the indicated address does not belong to the sector being erased, flash memory outputs bit 3 (DATA:3) of the read value of the indicated address.

### ■ Transition of Sector Erase Timer Flag States

Table 26.5-9 "Transition of Sector Erase Timer Flag States at Normal Operation" shows the transition of sector erase timer flag states at normal operation. Table 26.5-10 "Transition of Sector Erase Timer Flag States at Abnormal Operation" shows the transition of sector erase timer flag states at abnormal operation.

**Table 26.5-9 Transition of Sector Erase Timer Flag States at Normal Operation**

Operating state	Write operation ---> completion	Chip or sector erase ---> completion	Sector erase wait ---> start	Sector erase ---> erase temporary stop (sector being erased)	Sector erase temporary stoop ---> restart (sector being erased)	Sector erase temporary stop in progress (sector not being erased)
DQ3	0 ---> DATA:3	1	0 ---> 1	1 ---> 0	0 ---> 1	DATA:3

**Table 26.5-10 Transition of Sector Erase Timer Flag States at Abnormal Operation**

Operating state	Write operation	Chip or sector erase operation
DQ3	0	1

## 26.6 Detailed Explanation of Flash Memory Write/Erase

---

This section describes how to perform read/reset, data write, chip erase, sector erase, sector erase temporary stop, and sector erase restart for flash memory by issuing the commands for activating automatic algorithm.

---

### ■ Detailed Explanation of Flash Memory Write/Erase

Flash memory can execute automatic algorithm by executing the write cycles to the buses of command sequences (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") for read/reset, data write, chip erase, sector erase, sector erase temporary stop, and sector erase restart. The write cycles to these buses must always be executed continuously. Automatic algorithm can also use the data polling function to determine if a termination occurs. Automatic algorithm returns to the read/reset state after terminating.

Flash memory write/erase is explained as follows:

1. Read/reset (See Section 26.6.1 "Read/Reset".)
2. Data write (See Section 26.6.2 "Data Write".)
3. Data erase (all chip erase) (See Section 26.6.3 "Data Erase (All Chip Erase)".)
4. Data erase (sector erase) (See Section 26.6.4 "Data Erase (Sector Erase)".)
5. Sector erase temporary stop (See Section 26.6.5 "Sector Erase Temporary Stop".)
6. Sector erase restart (See Section 26.6.6 "Sector Erase Restart".)

## 26.6.1 Read/Reset

---

**This section describes how to put flash memory into the read/reset state by issuing the read/reset command.**

---

### ■ Putting Flash Memory into Read/Reset State

Flash memory can be put into the read/reset state by continuously transmitting the read/reset command shown in the command sequence table (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") to the target sector in flash memory.

The read/reset command provides the two command sequences that perform one and three bus operations. These sequences are essentially the same.

The read/reset state indicates that flash memory is in the initial state. When power is turned on or when a command terminates normally, flash memory enters the read/reset state. When flash memory is in the read/reset state, other commands are in the input wait state.

In the read/reset state, normal read access can be used to read data. Like mask ROM, program access from the CPU is possible. The read/reset command is unnecessary for normal read access. This command is used primarily to initialize automatic algorithm when a command does not terminate normally.

## 26.6.2 Data Write

---

**This section describes how to write data to flash memory by issuing the write command.**

---

### ■ Writing Data to Flash Memory

Automatic algorithm for flash memory can be activated by continuously transmitting the write command shown in the command sequence table (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") to the target sector in flash memory. When data write to the target address terminates at the 4th cycle, automatic algorithm is activated to start automatic writing.

### ■ Addressing

In the write data cycle, only an even-number address can be specified as a write address. If an odd-number address is specified, data cannot be written to flash memory correctly. That is, data must be written to even-number addresses in words.

Data can be written to flash memory in any address order. Data can also be written beyond a sector boundary, but only one word can be written by one write command.

### ■ Notes on Data Write

Data "0" cannot be restored to data "1" by the write command. If data "1" is written to data "0", the flash memory element is determined to be faulty because the data polling algorithm (DQ7) or toggle operation (DQ6) does not terminate. The timing limit excess flag (DQ6) is determined to be an error because the specified write time was exceeded or data "1" was written. However, even if data is read in the read/reset state, it remains "0". Only an erase operation enables data to be changed from "0" to "1".

All commands are ignored during automatic write execution. Note that if hardware reset is activated during automatic write execution, data at the written address is not guaranteed.

### ■ Flash Memory Write Procedure

The hardware sequence flag (see Section 26.5 "Checking Automatic Algorithm Execution Status") enables the determination of the automatic algorithm state in flash memory.

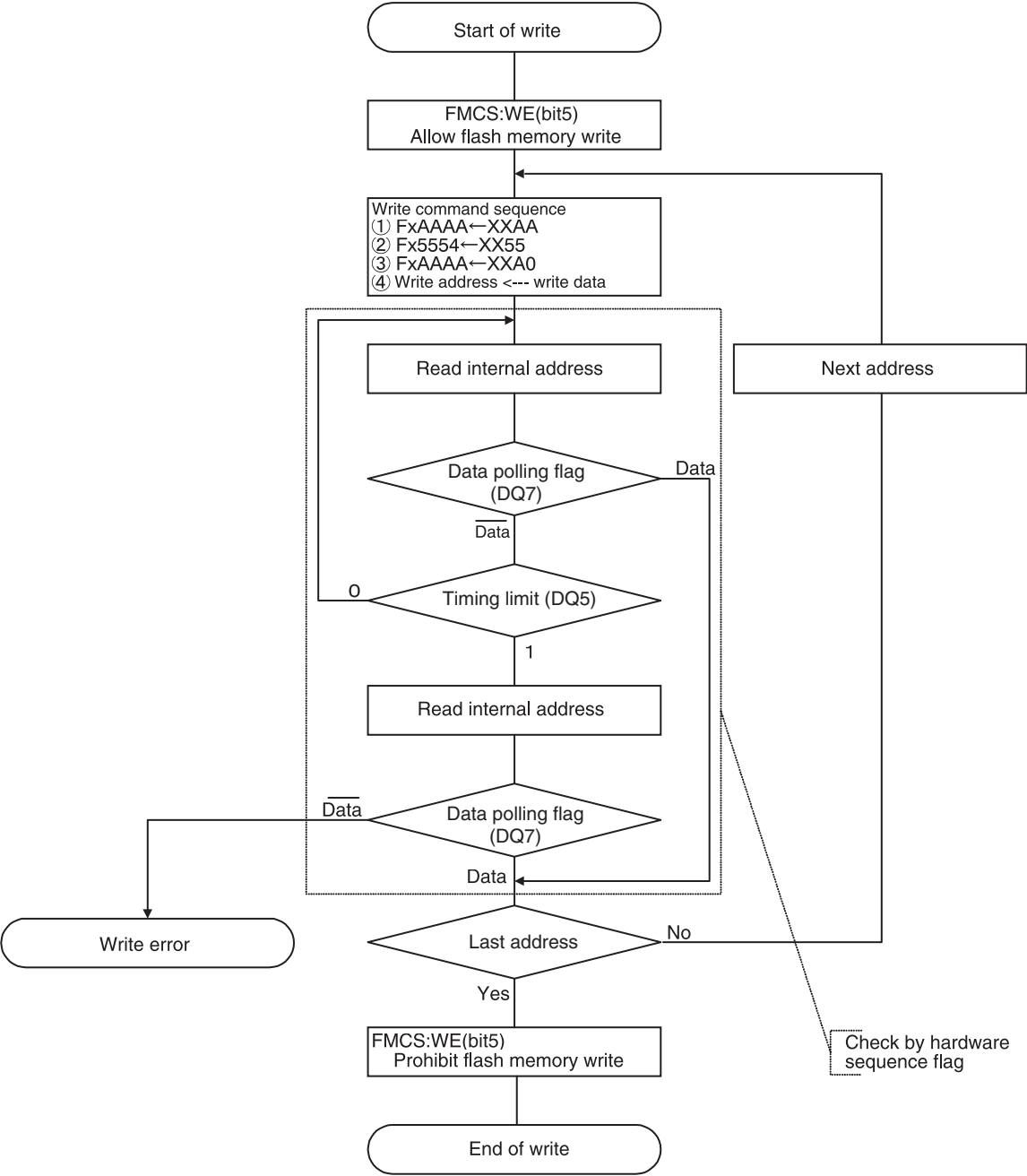
Figure 26.6-1 "Example of Flash Memory Write Procedure" is an example of the flash memory write procedure. In this example, the data polling flag (DQ7) is used to check data write termination.

Data for flag check is read into the most recent written address.

The data polling flag (DQ7) and timing limit excess flag (DQ5) change at the same time. For this reason, even if DQ5 is "1", DQ7 must be re-checked.

When DQ5 changes to "1", the toggle bit flag (DQ6) also stops toggle operation. DQ6 must also be re-checked.

Figure 26.6-1 Example of Flash Memory Write Procedure



### 26.6.3 Data Erase (All Chip Erase)

---

**This section describes how to erase all data in flash memory by issuing the chip erase command.**

---

#### ■ Erasing Data (Erasing All Chips)

All data can be erased from flash memory by continuously issuing the chip erase command shown in the command sequence table (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") to the target sector in flash memory.

The chip erase command performs a bus operation six times. When a write operation in the 6th cycle is completed, a chip erase operation is started. In chip erase, the user does not have to write data to flash memory before erase. Flash memory writes "0" for verification before all cells are erased automatically.



## 26.6.4 Data Erase (Sector Erase)

---

This section describes how to erase an arbitrary sector from flash memory by issuing the sector erase command. Data can be erased per sector and several sectors can be specified at the same time.

---

### ■ Erasing Data (Erasing Sector)

An arbitrary sector can be erased from flash memory by continuously transmitting the sector erase command shown in the command sequence table (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") to the target sector in flash memory.

### ■ Sector Specification Method

The sector erase command performs bus operation six times. When the sector erase code (30<sub>H</sub>) is written to an accessible even-number address in the target sector at the 6th cycle, a 50- $\mu$ s sector erase wait is started. To erase several sectors, write the erase code (30<sub>H</sub>) to an address in the sector to be erased as explained in the above processing.

### ■ Notes on Specification of Several Sectors

Data erase is started when the 50  $\mu$ s sector erase wait period from the write of the last sector erase code terminates. That is, to erase several sectors at the same time, the address and erase code (6th cycle of the command sequence) of the next erase sector must be entered within 50  $\mu$ s. If these address and erase code are not entered within 50  $\mu$ s, they may not be accepted. The validity of a write of the subsequent sector erase codes can be determined by the sector erase timer (hardware sequence flag DQ3). In this case, the address from which the sector erase timer is to be read must indicate the sector to be erased.

### ■ Sector Erase Procedure

The hardware sequence flag (see Section 26.5 "Checking Automatic Algorithm Execution Status") enables the determination of the automatic algorithm state in flash memory.

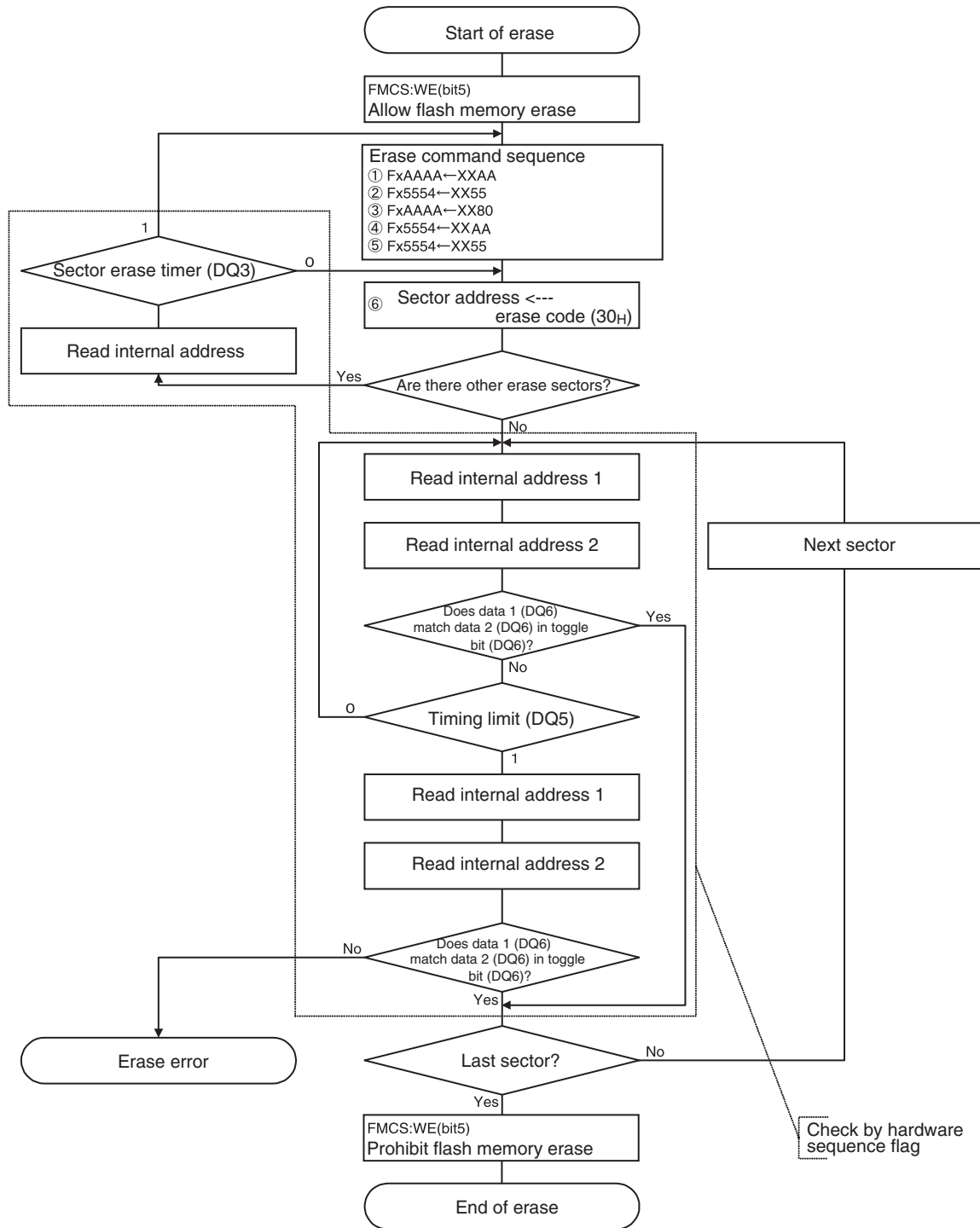
Figure 26.6-2 "Example of Sector Erase Procedure" is an example of the flash memory sector erase procedure. In this example, the toggle bit flag (DQ6) is used to check sector erase termination.

Note that data for flag check is read into the sector to be erased.

The toggle bit flag (DQ6) stops toggle operation when the timing limit excess flag (DQ5) changes to "1", and so even if DQ5 is "1", DQ6 must be re-checked.

When DQ5 changes, the data polling flag (DQ7) also changes. DQ7 must also be re-checked.

Figure 26.6-2 Example of Sector Erase Procedure



## 26.6.5 Sector Erase Temporary Stop

---

**This section describes how to stop flash memory sector erase temporarily by issuing the sector erase temporary stop command. Data can also be read from the sector not being erased.**

---

### ■ Stopping Sector Erase Temporarily

Flash memory sector erase can be stopped temporarily by continuously transmitting the sector erase temporary stop command shown in the command sequence table (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") to flash memory.

The sector erase temporary stop command stops sector erase temporarily to enable data to be read from the sector not being erased. In this state, a read is possible, but a write is not possible. This command can be used only when a sector is being erased (sector erase time including the erase wait time). The command is ignored when a chip is being erased or when data is being written to flash memory.

This command is executed by writing the erase temporary stop code (B0<sub>H</sub>). In this case, however, the address must indicate an address in flash memory. In sector erase temporary stop, re-issuance of the sector erase temporary stop command is ignored.

If the sector erase temporary stop command is entered during the sector erase wait period, flash memory immediately terminates sector erase wait, interrupts erase operation, and enters the erase top state. If the sector erase temporary stop command is entered when sector erase operation is in progress after the sector erase wait period has elapsed, flash memory enters the erase temporary stop state within 15  $\mu$ s.

## 26.6.6 Sector Erase Restart

---

**This section describes how to restart the temporarily stopped flash memory sector erase operation by issuing the sector erase restart command.**

---

### ■ Restarting Sector Erase Operation

The temporarily stopped sector erase operation can be restarted by transmitting the sector erase restart command shown in the command sequence table (see Section 26.4 "Method for Activating Flash Memory Automatic Algorithm") continuously to flash memory.

The sector erase restart command restarts the sector erase operation temporarily stopped by the sector erase temporary stop command. This command is executed by writing the erase restart code (30<sub>H</sub>). In this case, however, the address must indicate any address in flash memory.

In sector erase operation, issuance of the sector erase restart command is ignored.

## 26.7 Example of Flash Memory Program

This section provides an example of a flash memory program.

### ■ Example of Flash Memory Program

```

NAME    FLASHWE
TITLE   FLASHWE

;-----
;MB90F574/A-FLASH test program
;
;1: Transmits the program (address: FFBC00H, sector: SA6) from FLASH to RAM
;   (address: 001500H).
;2: Executes the program on RAM.
;3: Writes the PDR1 value to FLASH (address: FD0000H, sector: SA1).
;4: Reads the written value (address: FD0000H, sector: SA1) and outputs it to PDR2.
;5: Erases the written sector (SA1).
;6: Checks and outputs erase data.
;Conditions
; - Number of bytes transmitted to RAM: 100H (256B)
; - Write/erase termination judgment
;   Judgment according to DQ5 (timing limit excess flag)
;   Judgment according to DQ6 (toggle bit flag)
;   Judgment according to RDY (FMCS)
; - Error handling
;   Hi output to P00 to P07
;   Reset command issuance
;-----
;
RESOUS   IOSEG   ABS=00           ;"RESOUS" I/O segment definition
        ORG     0000H
PDR0     RB      1
PDR1     RB      1
PDR2     RB      1
PDR3     RB      1
        ORG     0010H
DDR0     RB      1
DDR1     RB      1
DDR2     RB      1
DDR3     RB      1
        ORG     00A1H
CKSCR    RB      1
        ORG     00AEH
FMCS     RB      1
        ORG     006FH
ROMM     RB      1
RESOUS   ENDS
;
SSTA     SSEG
        RW      0127H
STA_T    RW      1
SSTA     ENDS
;
DATA     DSEG    ABS=0FFH        ;FLASH command address
        ORG     5554H
COMADR2   RW      1
        ORG     0AAAAH
COMADR1   RW      1
DATA     ENDS

```

```

;////////////////////////////////////
;Main program (SA3)
;////////////////////////////////////
CODE    CSEG
START:
;
;    Initialization
;
;    MOV    CKSCR,#0BAH    ;3-multiple setting
;    MOV    RP,#0
;    MOV    A,#!STA_T
;    MOV    SSB,A
;    MOVW   A,#STA_T
;    MOVW   SP,A
;    MOV    ROMM,#00H      ;Mirror OFF
;    MOV    PDR0,#00H      ;For error check
;    MOV    DDR0,#0FFH
;    MOV    PDR1,#00H      ;Port for data input
;    MOV    DDR1,#00H
;    MOV    PDR2,#00H      ;Port for data output
;    MOV    DDR2,#0FFH
;
;    Transfer of "FLASH write erase program (FFBC00H)" to RAM (1500H address)
;
;    MOVW   A,#1500H      ;Transfer destination RAM area
;    MOVW   A,#0BC00H      ;Transfer source address (program position)
;    MOVW   RW0,#100H      ;Number of bytes to be transferred
;    MOVS   ADB,PCB        ;Transfer of 100H from FFBC00H to 001500H
;    CALLP  001500H        ;Jump to the address containing the transferred
;                           program
;
;    Data output
;
;    OUT    MOV    A,#0FDH
;           MOV    ADB,A
;           MOVW   RW2,#0000H
;           MOVW   A,@RW2+00
;           MOV    PDR2,A
;    END    JMP    *
;    CODE    ENDS
;
;FLASH write erase program (SA6)
;////////////////////////////////////
RAMPRG  CSEG    ABS=0FFH
;           ORG    0BC00H
;
;    Initialization
;
;    MOVW   RW0,#0500H      ;RW0:RAM space for input data acquisition  00:0500~
;    MOVW   RW2,#0000H      ;RW2:Flash memory write address            FD:0000~
;    MOV    A,#00H          ;DTB modification
;    MOV    DTB,A           ;Bank specification for @RW0
;    MOV    A,#0FDH         ;ADB modification 1
;    MOV    ADB,A           ;Bank specification for write mode specification
;                           address
;
;    MOV    PDR3,#00H        ;Switch initialization
;    MOV    DDR3,#00H
;
;    WAIT1  BBC    PDR3:0,WAIT1 ;PDR3: 0(write start at high level)
;
;    Write (SA1)
;
;    MOV    A,PDR1
;    MOVW   @RW0+00,A        ;PDR1 data allocation to RAM
;    MOV    FMCS,#20H        ;Write mode setting
;    MOVW   ADB:COMADR1,#00AAH ;Flash write command 1
;    MOVW   ADB:COMADR2,#0055H ;Flash write command 2
;    MOVW   ADB:COMADR1,#00A0H ;Flash write command 3

```

## CHAPTER 26 2M-BIT FLASH MEMORY

```

;
        MOVW    A,@RW0+00                ;Input data (RW0) write to flash memory (RW2)
        MOVW    @RW2+00,A
WRITE   ;Wait time check
;
;      ///////////////////////////////////////////////////
;      ERROR when the time limit excess check flag is set and toggle operation is
;      in progress
;      ///////////////////////////////////////////////////
;
        MOVW    A,@RW2+00
        AND     A,#20H                    ;DQ5 time limit check
        BZ      NTOW                      ;Time limit over
        MOVW    A,@RW2+00                ;AH
        MOVW    A,@RW2+00                ;AL
        XORW    A                        ;XOR of AH and AL (1 when the values differ)
        AND     A,#40H                    ;Is the DQ6 toggle bit different?
        BNZ     ERROR                     ;To ERROR when the DQ6 toggle bit is different
;
;      ///////////////////////////////////////////////////
;      Write termination check (FMCS-RDY)
;      ///////////////////////////////////////////////////
;      ///////////////////////////////////////////////////
NTOW    MOVW    A,FMCS
        AND     A,#10H                    ;Extraction of FMCS RDY bit (bit 4)
        BZ      WRITE                     ;End of write?
        MOV     FMCS,#00H                  ;Write mode release
;
;      ///////////////////////////////////////////////////
;      Write data output
;      ///////////////////////////////////////////////////
        MOVW    RW2,#0000H                ;Write data output
        MOVW    A,@RW2+00
        MOV     PDR2,A
;
WAIT2   BBC     PDR3:1,WAIT2               ;PDR3: 1(sector erase start at high level)
;
;      ///////////////////////////////////////////////////
;      Sector erase (SA1)
;      ///////////////////////////////////////////////////
        MOV     @RW2+00,#0000H            ;Address initialization
        MOV     FMCS,#20H                  ;Erase mode setting
        MOVW    ADB:COMADR1,#00AAH        ;Flash erase command 1
        MOVW    ADB:COMADR2,#0055H        ;Flash erase command 2
        MOVW    ADB:COMADR1,#0080H        ;Flash erase command 3
        MOVW    ADB:COMADR1,#00AAH        ;Flash erase command 4
        MOVW    ADB:COMADR2,#0055H        ;Flash erase command 5
        MOV     @RW2+00,#0030H            ;Issuance of erase command 6 to the sector
                                         ;to be erased
ELS     ;Wait time check
;
;      ///////////////////////////////////////////////////
;      ERROR when the time limit excess check flag is set and toggle operation is
;      in progress
;      ///////////////////////////////////////////////////
;
        MOVW    A,@RW2+00
        AND     A,#20H                    ;DQ5 time limit check
        BZ      NTOE                      ;Time limit over
        MOVW    A,@RW2+00                ;AH High and Low are alternately output from
        MOVW    A,@RW2+00                ;AL DQ6 per read during write operation.
        XORW    A                        ;XOR of AH and AL (If the DQ6 value differs,
;                                         ;write operation is in progress (1)).
        AND     A,#40H                    ;Is the DQ6 toggle bit High?
        BNZ     ERROR                     ;ERROR when the DQ6 toggle bit is High
;
;      ///////////////////////////////////////////////////
;      Erase termination check (FMCS-RDY)
;      ///////////////////////////////////////////////////
;
NTOE    MOVW    A,FMCS                    ;
        AND     A,#10H                    ;Extraction of FMCS RDY bit (bit 4)
        BZ      ELS                       ;End of sector erase?
        MOV     FMCS,#00H                  ;FLASH erase mode release
        RETP                                ;Return to the main program

```

```

;////////////////////////////////////
;Error
;////////////////////////////////////
ERROR    MOV     ADB:COMADR1,#0F0H      ;Reset command (read is enabled)
          MOV     PDR0,#0FFH           ;Error handling check
          MOV     FMCS,#00H            ;FLASH mode release
          RETP                          ;Return to the main program
RAMPRG   ENDS
;////////////////////////////////////
VECT     CSEG     ABS=0FFH
          ORG      0FFDCH
          DSL      START
          DB        00H
VECT     ENDS
;

```





# CHAPTER 27    EXAMPLES OF MB90F574/A SERIAL PROGRAMMING CONNECTION

---

**This chapter describes examples of serial programming connection with the AF220/AF210/AF120/AF110 flash microcomputer programmer manufactured by YDC Corporation.**

---

- 27.1 "Basic Configuration of MB90F574/A Serial Programming Connection"
- 27.2 "Example of Serial Programming Connection (User Power Supply Used)"
- 27.3 "Example of Serial Programming Connection (Power Supplied from the Programmer)"
- 27.4 "Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)"
- 27.5 "Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)"

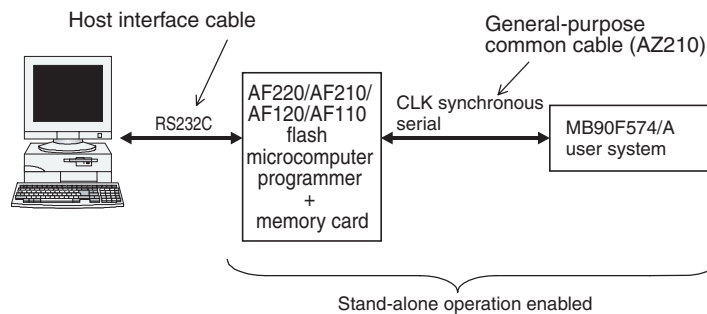
## 27.1 Basic Configuration of MB90F574/A Serial Programming Connection

The MB90F574/A supports the serial on-board programming (Fujitsu standard) of flash ROM. This section describes the specifications.

### ■ Basic Configuration of MB90F574/A Serial Programming Connection

The AF220/AF210/AF120/AF110 flash microcomputer programmer of YDC Corporation is used as Fujitsu standard serial on-board programming. Figure 27.1-1 "Basic Configuration of MB90F574/A Serial Programming Connection" shows the basic configuration of MB90F574/A serial programming connection.

**Figure 27.1-1 Basic Configuration of MB90F574/A Serial Programming Connection**



**Note:**

Ask YDC Corporation for information about the functions and operations of the AF220/AF210/AF120/AF110 flash microcomputer programmer, general-purpose common cable (AZ210) for connection, and connectors.

**Table 27.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming**

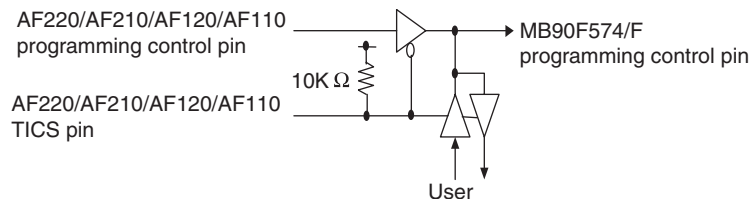
Pin	Function	Additional information
MD2, MD1, MD0	Mode pin	Controls programming mode from the flash microcomputer programmer.
X0, X1	Oscillation pins	In programming mode, the CPU internal operation clock signal is one multiple of the PLL clock signal frequency. Therefore, because the oscillation clock frequency becomes the internal operation clock signal, the resonator used for serial reprogramming is 1 MHz to 16 MHz.
P00, P01	Programming activation pins	-
RSTX	Reset pin	-
SIN0	Serial data input pin	The UART0 is used in CLK synchronous mode.
SOT0	Serial data output pin	
SCK0	Serial clock input pin	

**Table 27.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming (Continued)**

Pin	Function	Additional information
C	C pin	This external capacitor pin is used to stabilize the power supply. Connect a ceramic capacitor of approximately 0.1 $\mu$ F to the outside.
VCC	Power voltage supply pin	If the programming voltage (5 V $\pm$ 10%) is supplied from the user system, the flash microcomputer programmer need not be connected. Connect so that the power supply of the user side is not short-circuited.
VSS	GND pin	Common to the ground of the flash microcomputer programmer.
HSTX	Hardware standby pin	Input high level during serial programming mode.

Even if the P00, SIN0, SOT0, and SCK0 pins are used for the user system, the control circuit shown in the figure below is required. (The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial Programming).

**Figure 27.1-2 Control Circuit**



Section 27.2 "Example of Serial Programming Connection (User Power Supply Used)" to 27.5 "Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)" present examples the following four types of serial programming connection. See each Section as required.

- Internal vector mode (single chip mode and internal ROM external bus mode) [user power supply used]
- Internal vector mode (single chip mode and internal ROM external bus mode) [power supplied from the programmer]
- Example of minimum connection to the flash microcomputer programmer [user power supply used]
- Example of minimum connection to the flash microcomputer programmer [power supplied from the programmer]

**Table 27.1-2 System Configuration of AF220/AF210/AF120/AF110 Flash Microcomputer Programmer (YDC Corporation)**

Model		Function
Mainframe	AF220/AC4P	Built-in Ethernet interface model (100 to 220 V power adapter)
	AF210/AC4P	Standard model (100 to 220 V power adapter)
	AF120/AC4P	Single-key Ethernet interface model (100 to 220 V power adapter)
	AF110/AC4P	Single-key model (100 to 220 V power adapter)
AZ221		PC/AT RS232C cable only for programmer
AZ210		Standard target probe (a), length: 1 m
FF201		Fujitsu F <sup>2</sup> MC-16LX flash microcomputer control model
AZ290		Remote controller
/P2		2 MB PC card (option) for flash memory sizes of up to 128 KB
/P4		4 MB PC card (option) for flash memory sizes of up to 512 KB

Inquiries: YDC Corporation

Telephone number: (81) 42-333-6224

**Note:**

The AF200 flash microcomputer programmer, which is not supported now, can be used by using control module FF201. Section 27.2 "Example of Serial Programming Connection (User Power Supply Used)". to 27.5 "Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the programmer)" present examples the following four types of serial Programming connection.

■ **Oscillation Clock Frequency and Serial Clock Input Frequency**

The formula shown below can be used to calculate the maximum serial clock frequency that can be input to the MB90F574/A.

Maximum serial clock frequency that can be input = 0.125 x oscillation clock frequency

Consequently, change the serial clock input frequency by setting the serial clock frequency of the flash microcomputer programmer according to the current oscillation clock frequency.

**Table 27.1-3 Examples of the Maximum Serial Clock Frequency That Can Be Input**

Oscillation clock frequency	Maximum serial clock frequency that can be input for the microcomputer	Maximum serial clock frequency that can be set with AF220/AF210/AF120/AF110	Maximum serial clock frequency that can be set with AF200
4 MHz	500 kHz	500 kHz	500 kHz
8 MHz	1 MHz	850 kHz	500 kHz
16 MHz	2 MHz	1.25 MHz	500 kHz

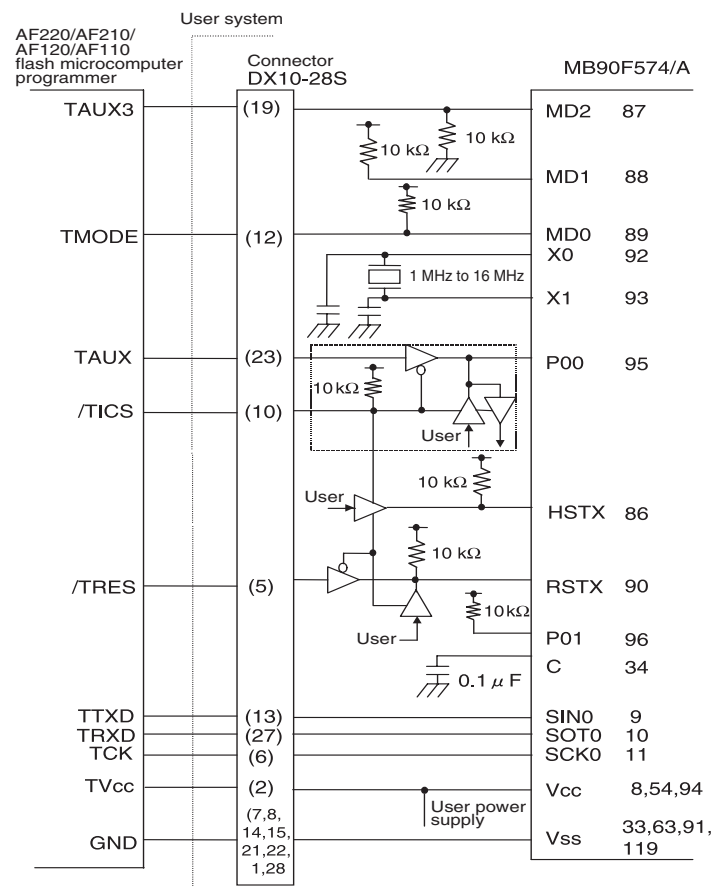
## 27.2 Example of Serial Programming Connection (User Power Supply Used)

Figure 27.2-1 "Example of Serial Programming Connection in MB90F574/A Internal Vector Mode (User Power Supply Used)" is an example of a serial programming connection (when the power for the microcomputer is supplied from the user power supply.) Note that mode pins MD2 and MD0 receive input signals MD2=1 and MD0=0, respectively, from the TAUX3 and TMODE of the AF220/AF210/AF120/AF110 programmer.

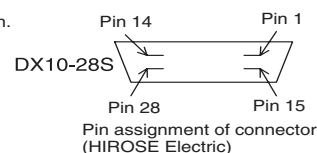
Serial reprogramming mode pins MD2, MD1, and MD0 are set to 110.

### ■ Example of Serial Programming Connection (User Power Supply Used)

Figure 27.2-1 Example of Serial Programming Connection in MB90F574/A Internal Vector Mode (User Power Supply Used)

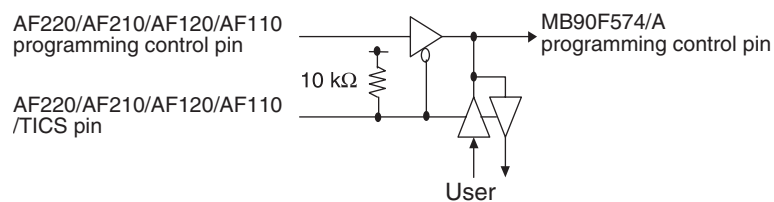


- Pins 3, 4, 9, 11, 16, 17, 18, 20, 24, 25, and 26 are open.
- DX10-28S: Right angle type
- The Pin Numbers of the written microcomputer correspond to those of the FTP-120P-M05, FTP-120P-M13, and FTP-120P-M21 packages.



## CHAPTER 27 EXAMPLES OF MB90F574/A SERIAL PROGRAMMING CONNECTION

- When the SIN0, SOT0, and SCK0 pins are used in the user system, the control circuit shown in the figure below is necessary, as with P00. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



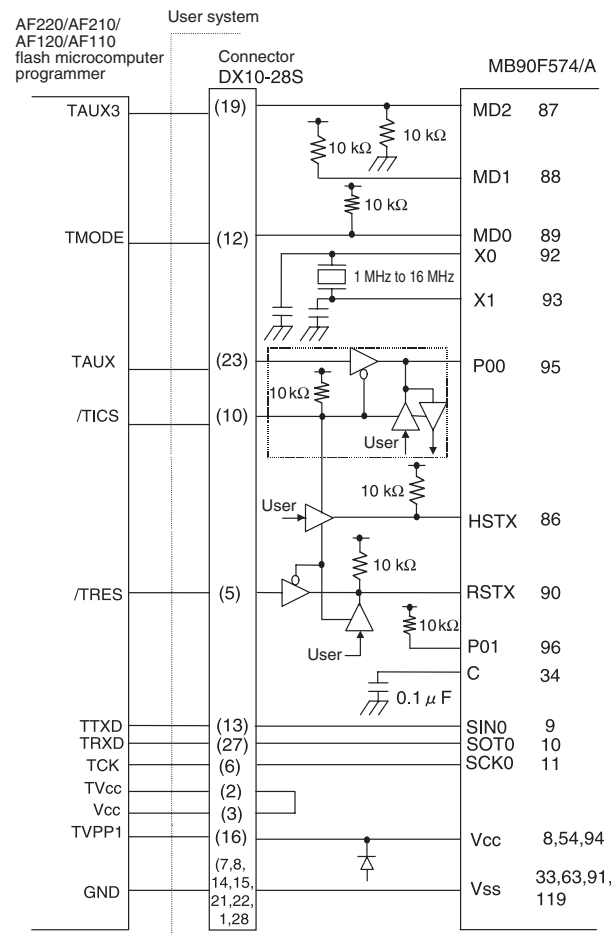
- Connect pins to AF220/AF210/AF120/AF110 when the user power supply is off.

## 27.3 Example of Serial Programming Connection (Power Supplied from the Programmer)

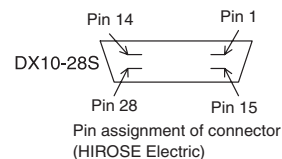
Figure 27.3-1 "Example of Serial Programming Connection in MB90F574/A Internal Vector Mode (Power Supplied from the Programmer)" is an example of serial programming connection (when the power for the microcomputer is supplied from the programmer.) Note that mode pins MD2 and MD0 receive input signals MD2=1 and MD0=0, respectively, from the TAUX3 and TMODE of the AF220/AF210/AF120/AF110. Serial programming mode pins MD2, MD1, and MD0 are set to 110.

### ■ Example of Serial Programming Connection (Power Supplied from the Programmer)

Figure 27.3-1 Example of Serial Programming Connection in MB90F574/A Internal Vector Mode (Power Supplied from the Programmer)



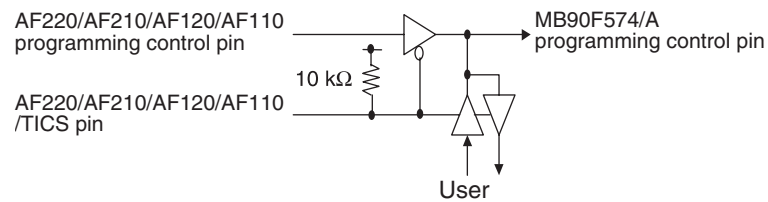
- Pins 4, 9, 11, 17, 18, 20, 24, 25, and 26 are open.
- DX10-28S: Right angle type
- The Pin Numbers of the written microcomputer correspond to those of the FTP-120P-M05, FTP-120P-M13, and FTP-120P-M21 packages.





## CHAPTER 27 EXAMPLES OF MB90F574/A SERIAL PROGRAMMING CONNECTION

- When the SIN0, SOT0, and SCK0 pins are used in the user system, the control circuit shown in the figure below is necessary, as with P00. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect pins to AF220/AF210/AF120/AF110 when the user power supply is off.
- When programming power is supplied from AF220/AF210/AF120/AF110, do not connect the circuit between the programming power and user power supplies.

## 27.4 Example of Minimum Connection to Flash Microcomputer Programmer (User Power Supply Used)

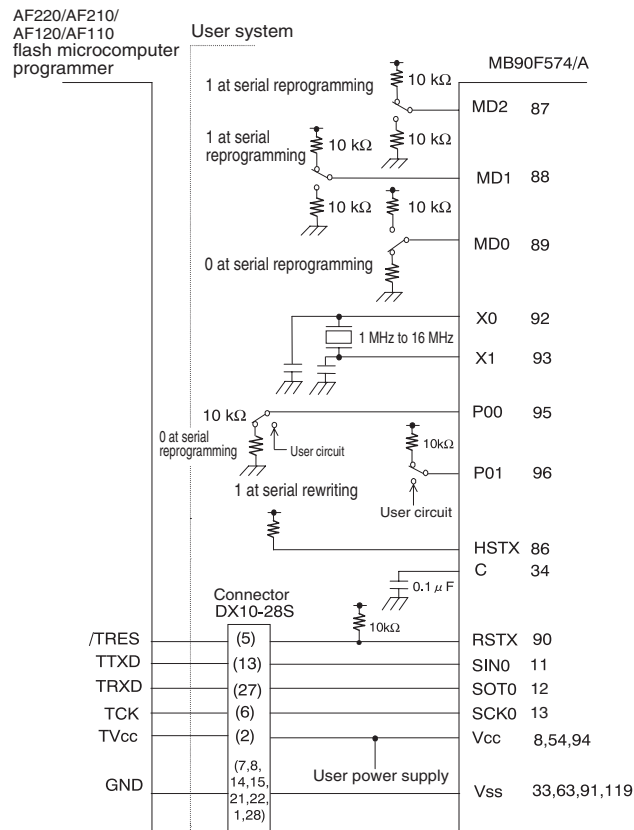
Figure 27.4-1 "Example of Minimum Connection to Flash Microcomputer Programmer (User Power Supply Used)" is an example of minimum connection to the flash microcomputer programmer (when the power for the microcomputer is supplied from the user power supply.)

Serial programming mode pins MD2, MD1, and MD0 are set to 110.

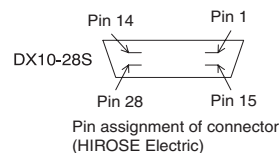
### ■ Example of Minimum Connection to Flash Microcomputer Programmer (User Power Supply Used)

Setting each pin as shown in Figure 27.4-1 "Example of Minimum Connection to Flash Microcomputer Programmer (User Power Supply Used)" when flash memory is written eliminates the need to connect MD2, MD1, MD0, and P00 to the flash microcomputer programmer.

**Figure 27.4-1 Example of Minimum Connection to Flash Microcomputer Programmer (User Power Supply Used)**

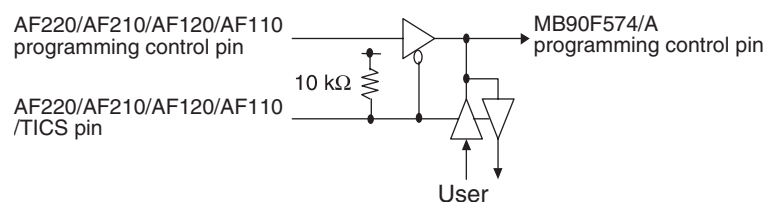


- Pins 3, 4, 9, 10, 11, 12, 16, 17, 18, 19, 20, 23, 24, 25, and 26 are open.
- DX10-28S: Right angle type
- The pin Numbers of the written microcomputer correspond to those of the FTP-120P-M05, FTP-120P-M13, and FTP-120P-M21 packages.



## CHAPTER 27 EXAMPLES OF MB90F574/A SERIAL PROGRAMMING CONNECTION

- When the SIN0, SOT0, and SCK0 pins are used in the user system, the control circuit shown in the figure below is necessary, as with P00. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect pins to AF220/AF210/AF120/AF110 when the user power supply is off.

## 27.5 Example of Minimum Connection to Flash Microcomputer Programmer (Power Supplied from the Programmer)

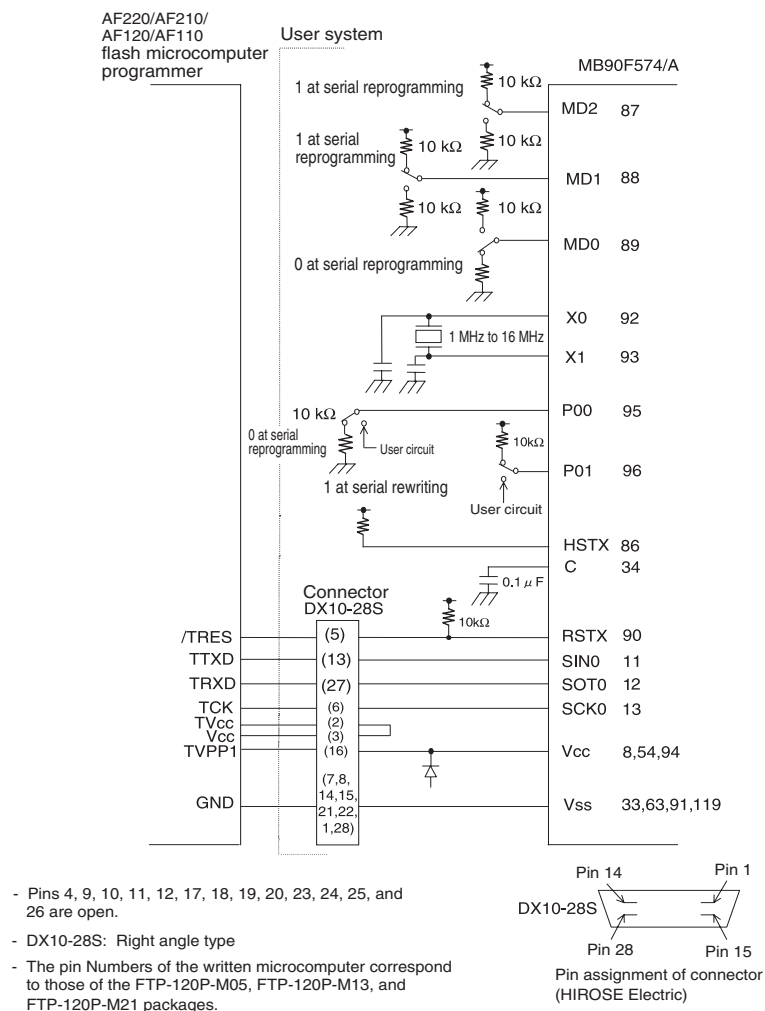
Figure 27.5-1 "Example of Minimum Connection to Flash Microcomputer Programmer (Power Supplied from the Programmer)" is an example of minimum connection to the flash microcomputer programmer (when the power for the microcomputer is supplied from the programmer.)

Serial programming mode pins MD2, MD1, and MD0 are set to 110.

### ■ Example of Minimum Connection to Flash Microcomputer Programmer (Power Supplied from the Programmer)

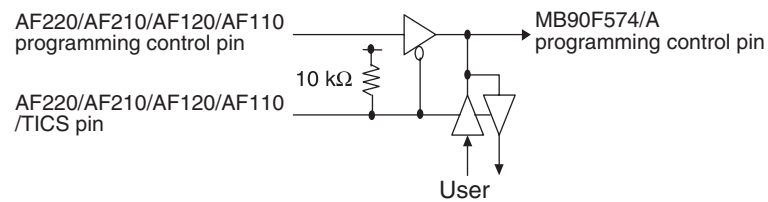
Setting each pin as shown in Figure 27.5-1 "Example of Minimum Connection to Flash Microcomputer Programmer (Power Supplied from the Programmer)" when flash memory is written eliminates the need to connect MD2, MD1, MD0, and P00 to the flash microcomputer programmer.

Figure 27.5-1 Example of Minimum Connection to Flash Microcomputer Programmer (Power Supplied from the Programmer)



## CHAPTER 27 EXAMPLES OF MB90F574/A SERIAL PROGRAMMING CONNECTION

- When the SIN0, SOT0, and SCK0 pins are used in the user system, the control circuit shown in the figure below is necessary, as with P00. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect pins to AF220/AF210/AF120/AF110 when the user power supply is off.
- When programming power is supplied from AF220/AF210/AF120/AF110, do not connect the circuit between the programming power and user power supplies.

# APPENDIX

---

**This appendix describes an I/O map, an instructions list, and other information.**

---

APPENDIX A "I/O Map"

APPENDIX B "Instructions"

## APPENDIX A I/O Map

Individual registers for the peripheral functions incorporated in the MB90570 Series are assigned addresses as shown in Table A-1 "I/O Map".

### ■ I/O Map

Table A-1 I/O Map

Address	Register	Name	Access	Peripheral	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>
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	XXXXXXXX <sub>B</sub>
0B <sub>H</sub>	Port B data register	PDRB	R/W	Port B	XXXXXXXX <sub>B</sub>
0C <sub>H</sub>	Port C data register	PDRC	R/W	Port C	XXXXXXXX <sub>B</sub>
0D <sub>H</sub> to 0F <sub>H</sub>	Not available				
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>
17 <sub>H</sub>	Port 7 direction register	DDR7	R/W	Port 7	----000 <sub>B</sub>
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>

Table A-1 I/O Map (Continued)

Address	Register	Name	Access	Peripheral	Initial value
1A <sub>H</sub>	Port A direction register	DDRA	R/W	Port A	--000000 <sub>B</sub>
1B <sub>H</sub>	Port B direction register	DDRB	R/W	Port B	00000000 <sub>B</sub>
1C <sub>H</sub>	Port C direction register	DDRC	R/W	Port C	----0000 <sub>B</sub>
1D <sub>H</sub>	Port 4 output pin register	ODR4	R/W	Port 4	00000000 <sub>B</sub>
1E <sub>H</sub>	Analog input enable register	ADER	R/W	Port 8, A/D	11111111 <sub>B</sub>
1F <sub>H</sub>	Not available				
20 <sub>H</sub>	Serial mode register 0	SMR0	R/W	UART0	00000000 <sub>B</sub>
21 <sub>H</sub>	Serial control register 0	SCR0	R/W	UART0	00000100 <sub>B</sub>
22 <sub>H</sub>	Serial input data register 0/serial output data register 0	SIDR0/ SODR0	R/W		XXXXXXXX <sub>B</sub>
23 <sub>H</sub>	Serial status register 0	SSR0	R/W		00001-00 <sub>B</sub>
24 <sub>H</sub>	Serial mode register 1	SMR1	R/W	UART1	00000000 <sub>B</sub>
25 <sub>H</sub>	Serial control register 1	SCR1	R/W		00000100 <sub>B</sub>
26 <sub>H</sub>	Serial input data register 1/serial output data register 1	SIDR1/ SODR1	R/W		XXXXXXXX <sub>B</sub>
27 <sub>H</sub>	Serial status register 1	SSR1	R/W		00001-00 <sub>B</sub>
28 <sub>H</sub>	Communication prescaler control register 0	CDCR0	R/W	UART0	0---1111 <sub>B</sub>
29 <sub>H</sub>	Not available				
2A <sub>H</sub>	Communication prescaler control register 1	CDCR1	R/W	UART1	0---1111 <sub>B</sub>
2B <sub>H</sub> to 2F <sub>H</sub>	Not available				
30 <sub>H</sub>	DTP/interrupt enable register	ENIR	R/W	DTP/ external interrupt	00000000 <sub>B</sub>
31 <sub>H</sub>	DTP/interrupt source register	EIRR	R/W		XXXXXXXX <sub>B</sub>
32 <sub>H</sub>	Request level setting register	ELVR	R/W		00000000 <sub>B</sub>
33 <sub>H</sub>					00000000 <sub>B</sub>
34 <sub>H</sub> to 35 <sub>H</sub>	Not available				
36 <sub>H</sub>	Control status register	ADCS1 ADCS2	R/W	A/D converter	00000000 <sub>B</sub>
37 <sub>H</sub>					00000000 <sub>B</sub>
38 <sub>H</sub>	Data register	ADCR1 ADCR2	R		XXXXXXXX <sub>B</sub>
39 <sub>H</sub>					00001-XX <sub>B</sub>



## APPENDIX A I/O Map

**Table A-1 I/O Map (Continued)**

Address	Register	Name	Access	Peripheral	Initial value
3A <sub>H</sub>	D/A converter data register 0	DADR0	R/W	D/A converter	XXXXXXXX <sub>B</sub>
3B <sub>H</sub>	D/A converter data register 1	DADR1	R/W		XXXXXXXX <sub>B</sub>
3C <sub>H</sub>	D/A control register 0	DACR0	R/W		-----0 <sub>B</sub>
3D <sub>H</sub>	D/A control register 1	DACR1	R/W		-----0 <sub>B</sub>
3E <sub>H</sub>	Clock output enable register	CLKR	R/W	Clock monitor function	----0000 <sub>B</sub>
3F <sub>H</sub>	Not available				
40 <sub>H</sub>	Reload register L (channel 0)	PRLLO	R/W	8/16bit PPG0/1	XXXXXXXX <sub>B</sub>
41 <sub>H</sub>	Reload register H (channel 0)	PRLH0	R/W		XXXXXXXX <sub>B</sub>
42 <sub>H</sub>	Reload register L (channel 1)	PRLLO1	R/W		XXXXXXXX <sub>B</sub>
43 <sub>H</sub>	Reload register H (channel 1)	PRLH1	R/W		XXXXXXXX <sub>B</sub>
44 <sub>H</sub>	PPG0 operating mode control register	PPGC0	R/W		0X000XX1 <sub>B</sub>
45 <sub>H</sub>	PPG1 operating mode control register	PPGC1	R/W		0X000001 <sub>B</sub>
46 <sub>H</sub>	PPG0 and PPG1 output pin control register	PPGOE	R/W		000000XX <sub>B</sub>
47 <sub>H</sub>	Not available				
48 <sub>H</sub>	Serial mode control status register 0	SMCS0	R/W	Extended serial I/O interface 0	----0000 <sub>B</sub>
49 <sub>H</sub>					00000010 <sub>B</sub>
4A <sub>H</sub>	Serial shift data register 0	SDR0	R/W		XXXXXXXX <sub>B</sub>
4B <sub>H</sub>	Not available				
4C <sub>H</sub>	Serial mode control status register 1	SMCS1	R/W	Extended serial I/O interface 1	----0000 <sub>B</sub>
4D <sub>H</sub>					00000010 <sub>B</sub>
4E <sub>H</sub>	Serial shift data register 1	SDR1	R/W		XXXXXXXX <sub>B</sub>
4F <sub>H</sub>	Not available				
50 <sub>H</sub>	Input capture register (channel 0)	IPCP0	R	16-bit I/O timer (input capture block)	XXXXXXXX <sub>B</sub>
51 <sub>H</sub>					XXXXXXXX <sub>B</sub>
52 <sub>H</sub>	Input capture register (channel 1)	IPCP1	R		XXXXXXXX <sub>B</sub>
53 <sub>H</sub>					XXXXXXXX <sub>B</sub>
54 <sub>H</sub>	Input capture control status register	ICS01	R/W		00000000 <sub>B</sub>
55 <sub>H</sub>	Not available				

Table A-1 I/O Map (Continued)

Address	Register	Name	Access	Peripheral	Initial value
56 <sub>H</sub>	Timer counter data register	TCDT	R/W	16-bit I/O timer (free run timer block)	00000000 <sub>B</sub>
57 <sub>H</sub>					00000000 <sub>B</sub>
58 <sub>H</sub>	Timer counter control status register	TCCS	R/W		00000000 <sub>B</sub>
59 <sub>H</sub>	Not available				
5A <sub>H</sub>	Output compare register (channel 0)	OCCP0	R/W	16-bit I/O timer (output compare block)	XXXXXXXX <sub>B</sub>
5B <sub>H</sub>					XXXXXXXX <sub>B</sub>
5C <sub>H</sub>	Output compare register (channel 1)	OCCP1	R/W		XXXXXXXX <sub>B</sub>
5D <sub>H</sub>					XXXXXXXX <sub>B</sub>
5E <sub>H</sub>	Output compare register (channel 2)	OCCP2	R/W		XXXXXXXX <sub>B</sub>
5F <sub>H</sub>					XXXXXXXX <sub>B</sub>
60 <sub>H</sub>	Output compare register (channel 3)	OCCP3	R/W	16-bit I/O timer (output compare block)	XXXXXXXX <sub>B</sub>
61 <sub>H</sub>					XXXXXXXX <sub>B</sub>
62 <sub>H</sub>	Output compare control status register (channel 0)	OCS0	R/W		0000--00 <sub>B</sub>
63 <sub>H</sub>	Output compare control status register (channel 1)	OCS1	R/W		---00000 <sub>B</sub>
64 <sub>H</sub>	Output compare control status register (channel 2)	OCS2	R/W		0000--00 <sub>B</sub>
65 <sub>H</sub>	Output compare control status register (channel 3)	OCS3	R/W		---00000 <sub>B</sub>
66 <sub>H</sub> to 67 <sub>H</sub>	Not available				
68 <sub>H</sub>	Bus status register	IBSR	R/W	I <sup>2</sup> C interface	00000000 <sub>B</sub>
69 <sub>H</sub>	Bus control register	IBCR	R/W		00000000 <sub>B</sub>
6A <sub>H</sub>	Clock control register	ICCR	R/W		--0XXXXX <sub>B</sub>
6B <sub>H</sub>	Address register	IADR	R/W		-XXXXXXXX <sub>B</sub>
6C <sub>H</sub>	Data register	IDAR	R/W		XXXXXXXX <sub>B</sub>
6D <sub>H</sub> to 6E <sub>H</sub>	Not available				
6F <sub>H</sub>	ROM mirror function selection register	ROMM	W	ROM mirror function	-----1 <sub>B</sub>
70 <sub>H</sub>	Up/Down count register 0	UDCR0	R	8/16-bit up/down timer counter	00000000 <sub>B</sub>
71 <sub>H</sub>	Up/Down count register 1	UDCR1			00000000 <sub>B</sub>

## APPENDIX A I/O Map

**Table A-1 I/O Map (Continued)**

Address	Register	Name	Access	Peripheral	Initial value
72 <sub>H</sub>	Reload/Compare register 0	RCR0	W	8/16-bit up/ down timer counter	00000000 <sub>B</sub>
73 <sub>H</sub>	Reload/Compare register 1	RCR1			00000000 <sub>B</sub>
74 <sub>H</sub>	Counter status register 0	CSR0	R/W		00000000 <sub>B</sub>
75 <sub>H</sub>	Reserved area	-	-		-
76 <sub>H</sub>	Counter control register 0	CCRL0	R/W		-0000000 <sub>B</sub>
77 <sub>H</sub>		CCRH0			00000000
78 <sub>H</sub>	Counter status register 1	CSR1	R/W		00000000
79 <sub>H</sub>	Reserved area	-	-		-
7A <sub>H</sub>	Counter control register 1	CCRL1	R/W		-0000000 <sub>B</sub>
7B <sub>H</sub>		CCRH1			-0000000 <sub>B</sub>
7C <sub>H</sub>	Serial mode control status register 2	SMCS2	R/W	Extended serial I/O interface 2	----0000 <sub>B</sub>
7D <sub>H</sub>					00000010 <sub>B</sub>
7E <sub>H</sub>	Serial shift data register 2	SDR2	R/W		XXXXXXXX <sub>B</sub>
7F <sub>H</sub>	Not available				
80 <sub>H</sub>	Chip select control register 0	CSCR0	R/W	Chip select	----0000 <sub>B</sub>
81 <sub>H</sub>	Chip select control register 1	CSCR1	R/W		----0000 <sub>B</sub>
82 <sub>H</sub>	Chip select control register 2	CSCR2	R/W		----0000 <sub>B</sub>
83 <sub>H</sub>	Chip select control register 3	CSCR3	R/W		----0000 <sub>B</sub>
84 <sub>H</sub>	Chip select control register 4	CSCR4	R/W		----0000 <sub>B</sub>
85 <sub>H</sub>	Chip select control register 5	CSCR5	R/W		----0000 <sub>B</sub>
86 <sub>H</sub>	Chip select control register 6	CSCR6	R/W		----0000 <sub>B</sub>
87 <sub>H</sub> to 8B <sub>H</sub>	Not available				
8C <sub>H</sub>	Port 0 resistor register	RDR0	R/W	Port 0	00000000 <sub>B</sub>
8D <sub>H</sub>	Port 1 resistor register	RDR1	R/W	Port 1	00000000 <sub>B</sub>
8E <sub>H</sub>	Port 6 resistor register	RDR6	R/W	Port 6	00000000 <sub>B</sub>
8F <sub>H</sub> to 9D <sub>H</sub>	Not available				
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>	Delayed interrupt source generation/release register	DIRR	R/W	Delayed interrupt requesting module	-----0 <sub>B</sub>

Table A-1 I/O Map (Continued)

Address	Register	Name	Access	Peripheral	Initial value
A0 <sub>H</sub>	Low power consumption mode control register	LPMCR	R/W	Low-power consumption control circuit	00011000 <sub>B</sub>
A1 <sub>H</sub>	Clock selection register	CKSCR	R/W		11111100 <sub>B</sub>
A2 <sub>H</sub> to A4 <sub>H</sub>	Not available				
A5 <sub>H</sub>	Automatic ready function selection register	ARSR	W	External bus pin control circuit	0011--00 <sub>B</sub>
A6 <sub>H</sub>	External address output control register	HACR	W		00000000 <sub>B</sub>
A7 <sub>H</sub>	Bus control signal selection register	ECSR	W		0000000- <sub>B</sub>
A8 <sub>H</sub>	Watchdog timer control register	WDTC	R/W	Watchdog timer	XXXXXXXX <sub>B</sub>
A9 <sub>H</sub>	Timebase timer control register	TBTC	R/W	Time-base timer	1--00100 <sub>B</sub>
AA <sub>H</sub>	Watch timer control register	WTC	R/W	Watch timer	1X000000 <sub>B</sub>
AB <sub>H</sub> to AD <sub>H</sub>	Not available				
AE <sub>H</sub>	Flash memory control status register	FMCS	R/W	Flash memory	00010--0 <sub>B</sub>
AF <sub>H</sub>	Not available				
B0 <sub>H</sub>	Interrupt control register 00	ICR00	R/W	Interrupt controller	00000111 <sub>B</sub>
B1 <sub>H</sub>	Interrupt control register 01	ICR01	R/W		00000111 <sub>B</sub>
B2 <sub>H</sub>	Interrupt control register 02	ICR02	R/W		00000111 <sub>B</sub>
B3 <sub>H</sub>	Interrupt control register 03	ICR03	R/W		00000111 <sub>B</sub>
B4 <sub>H</sub>	Interrupt control register 04	ICR04	R/W		00000111 <sub>B</sub>
B5 <sub>H</sub>	Interrupt control register 05	ICR05	R/W		00000111 <sub>B</sub>
B6 <sub>H</sub>	Interrupt control register 06	ICR06	R/W		00000111 <sub>B</sub>
B7 <sub>H</sub>	Interrupt control register 07	ICR07	R/W		00000111 <sub>B</sub>
B8 <sub>H</sub>	Interrupt control register 08	ICR08	R/W		00000111 <sub>B</sub>
B9 <sub>H</sub>	Interrupt control register 09	ICR09	R/W		00000111 <sub>B</sub>
BA <sub>H</sub>	Interrupt control register 10	ICR10	R/W		00000111 <sub>B</sub>
BB <sub>H</sub>	Interrupt control register 11	ICR11	R/W		00000111 <sub>B</sub>
BC <sub>H</sub>	Interrupt control register 12	ICR12	R/W		00000111 <sub>B</sub>
BD <sub>H</sub>	Interrupt control register 13	ICR13	R/W		00000111 <sub>B</sub>
BE <sub>H</sub>	Interrupt control register 14	ICR14	R/W		00000111 <sub>B</sub>
BF <sub>H</sub>	Interrupt control register 15	ICR15	R/W		00000111 <sub>B</sub>

## APPENDIX A I/O Map

Table A-1 I/O Map (Continued)

Address	Register	Name	Access	Peripheral	Initial value
C0 <sub>H</sub> to FF <sub>H</sub>	External area				
100 <sub>H</sub> to # <sub>H</sub>	RAM area				
# <sub>H</sub> to 1FEF <sub>H</sub>	Reserved area				
1FF0 <sub>H</sub>	Program address detection register 0	PADR0	R/W	Address match detection function	XXXXXXXX <sub>B</sub>
1FF1 <sub>H</sub>			R/W		XXXXXXXX <sub>B</sub>
1FF2 <sub>H</sub>			R/W		XXXXXXXX <sub>B</sub>
1FF3 <sub>H</sub>	Program address detection register 1	PADR1	R/W		XXXXXXXX <sub>B</sub>
1FF4 <sub>H</sub>			R/W		XXXXXXXX <sub>B</sub>
1FF5 <sub>H</sub>			R/W		XXXXXXXX <sub>B</sub>
1FF6 <sub>H</sub> to 1FFF <sub>H</sub>	Reserved area				

### Note:

Note: for programmable bits, initial individual values show values at initialization by reset, not at read-out.

In addition, parts or all of LPMCR, CKSCR, and WDTC are not always initialized, depending on the type of reset. The initial values shown above indicate values at initialization.

- Addresses 00FF<sub>H</sub> and later are reserved. External bus access signals are not generated.
- The boundary address #<sub>H</sub> between the RAM area and reserved area is 1900H for the MB90573 and 1FEF<sub>H</sub> for the MB90574 and MB90574C. In the latter case, no reserved area exists.
- Explanation on read/write
  - R/W: Read and write allowed
  - R: Read only
  - W: Write only
- Explanation on the 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 unpredictable.
  - \*: This bit is unused. The initial value is undefined.

## APPENDIX B Instructions

---

**APPENDIX B describes the instructions used by the F<sup>2</sup>MC-16LX.**

---

- B.1 Instruction Types
- B.2 Addressing
- B.3 Direct Addressing
- B.4 Indirect Addressing
- B.5 Execution Cycle Count
- B.6 Effective address field
- B.7 How to Read the Instruction List
- B.8 F<sup>2</sup>MC-16LX Instruction List
- B.9 Instruction Map

## B.1 Instruction Types

---

**The F<sup>2</sup>MC-16LX supports 351 types of instructions. Addressing is enabled by using an effective address field of each instruction or using the instruction code itself.**

---

### ■ Instruction Types

The F<sup>2</sup>MC-16LX supports the following 351 types of instructions:

- 41 transfer instructions (byte)
- 38 transfer instructions (word or long word)
- 42 addition/subtraction instructions (byte, word, or long word)
- 12 increment/decrement instructions (byte, word, or long word)
- 11 comparison instructions (byte, word, or long word)
- 11 unsigned multiplication/division instructions (word or long word)
- 11 signed multiplication/division instructions (word or long word)
- 39 logic instructions (byte or word)
- 6 logic instructions (long word)
- 6 sign inversion instructions (byte or word)
- 1 normalization instruction (long word)
- 18 shift instructions (byte, word, or long word)
- 50 branch instructions
- 6 accumulator operation instructions (byte or word)
- 28 other control instructions (byte, word, or long word)
- 21 bit operation instructions
- 10 string instructions

## B.2 Addressing

---

With the F<sup>2</sup>MC-16LX, the address format is determined by the instruction effective address field or the instruction code itself (implied). When the address format is determined by the instruction code itself, specify an address in accordance with the instruction code used. Some instructions permit the user to select several types of addressing.

---

### ■ Addressing

The F<sup>2</sup>MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)



## Effective Address Field

Table B.2-1 lists the address formats specified by the effective address field.

**Table B.2-1 Effective Address Field**

Code	Representation			Address format	Default bank
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	DTB
09	@RW1				DTB
0A	@RW2				ADB
0B	@RW3				SPB
0C	@RW0+			Register indirect with post increment	DTB
0D	@RW1+				DTB
0E	@RW2+				ADB
0F	@RW3+				SPB
10	@RW0+disp8			Register indirect with 8-bit displacement	DTB
11	@RW1+disp8				DTB
12	@RW2+disp8				ADB
13	@RW3+disp8				SPB
14	@RW4+disp8			Register indirect with 8-bit displacement	DTB
15	@RW5+disp8				DTB
16	@RW6+disp8				ADB
17	@RW7+disp8				SPB
18	@RW0+disp16			Register indirect with 16-bit displacement	DTB
19	@RW1+disp16				DTB
1A	@RW2+disp16				ADB
1B	@RW3+disp16				SPB
1C	@RW0+RW7			Register indirect with index	DTB
1D	@RW1+RW7			Register indirect with index	DTB
1E	@PC+disp16			PC indirect with 16-bit displacement	PCB
1F	addr16			Direct address	DTB

## B.3 Direct Addressing

An operand value, register, or address is specified explicitly in direct addressing mode.

### ■ Direct Addressing

- Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

**Figure B.3-1 Example of Immediate Addressing (#imm)**

MOVW A, #01212H (This instruction stores the operand value in A.)		
Before execution	A	2 2 3 3 : 4 4 5 5
After execution	A	4 4 5 5 : 1 2 1 2 (Some instructions transfer AL to AH.)

- Register direct addressing

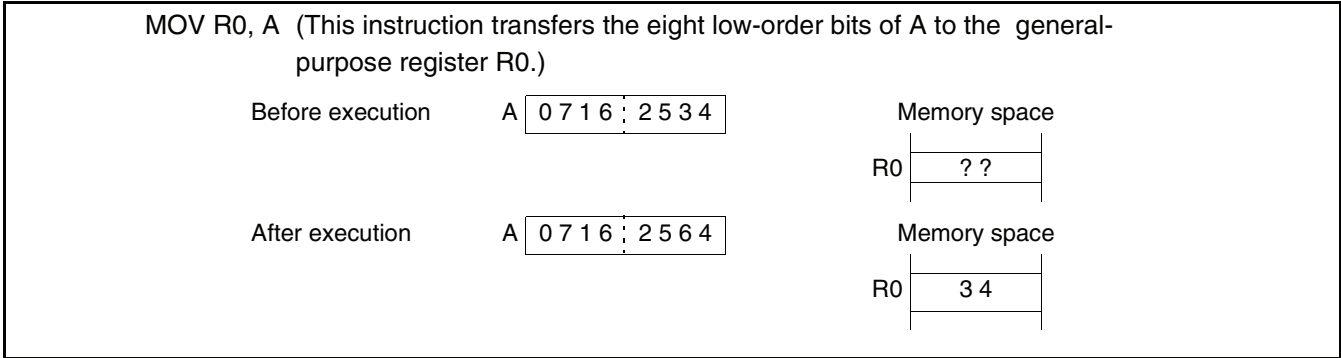
Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

**Table B.3-1 Direct Addressing Registers**

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

\*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

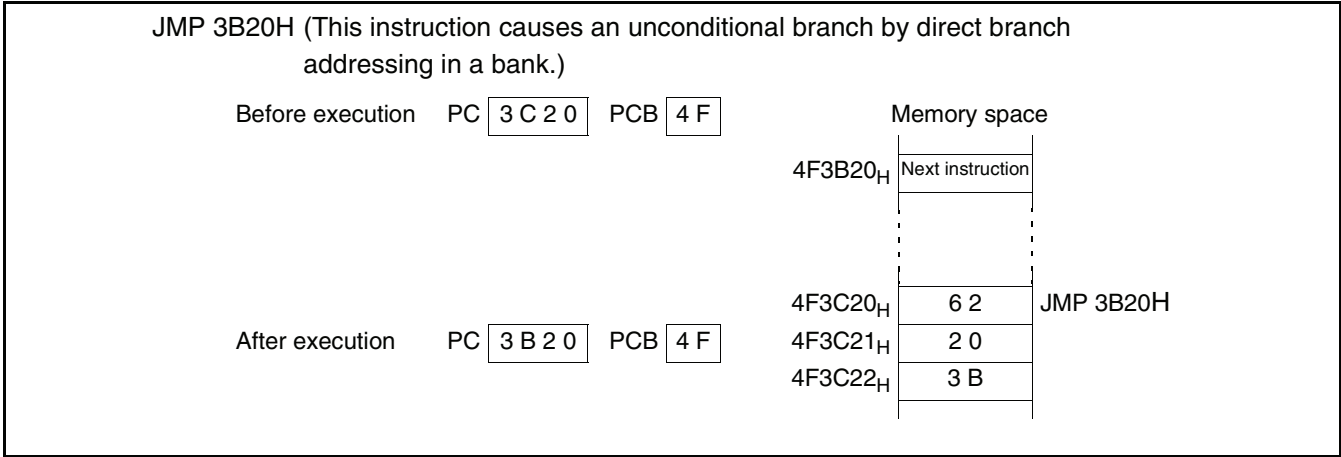
Figure B.3-2 Example of Register Direct Addressing



● Direct branch addressing (addr16)

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bit23 to bit16 of the address are specified by the program counter bank register (PCB).

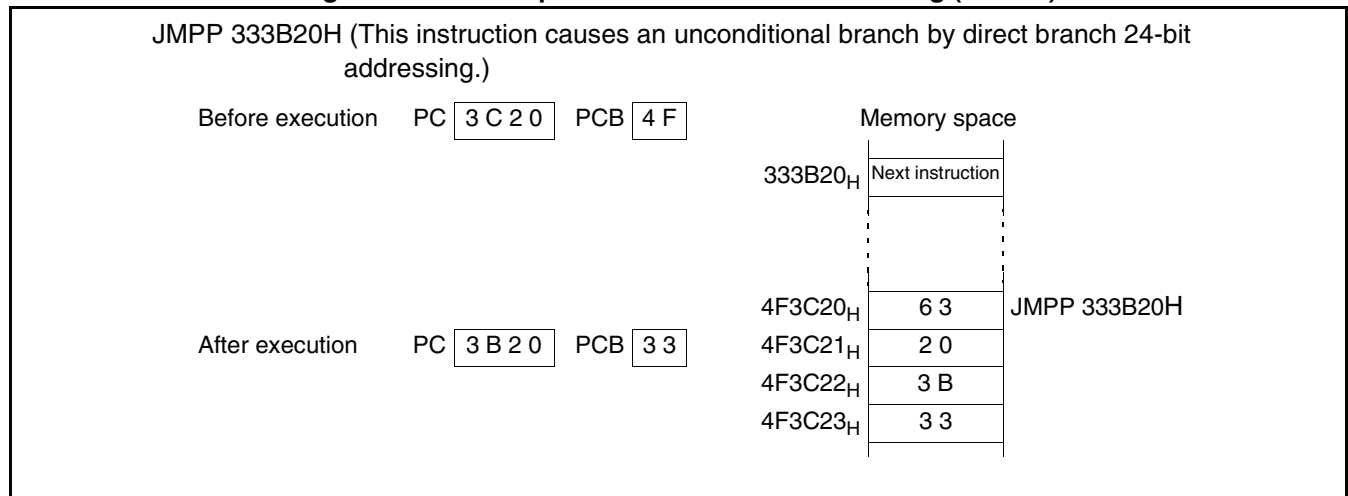
Figure B.3-3 Example of Direct Branch Addressing (addr16)



- Physical direct branch addressing (addr24)

Specify an offset explicitly for the branch destination address. The size of the offset is 24 bits. Physical direct branch addressing is used for unconditional branch, subroutine call, or software interrupt instruction.

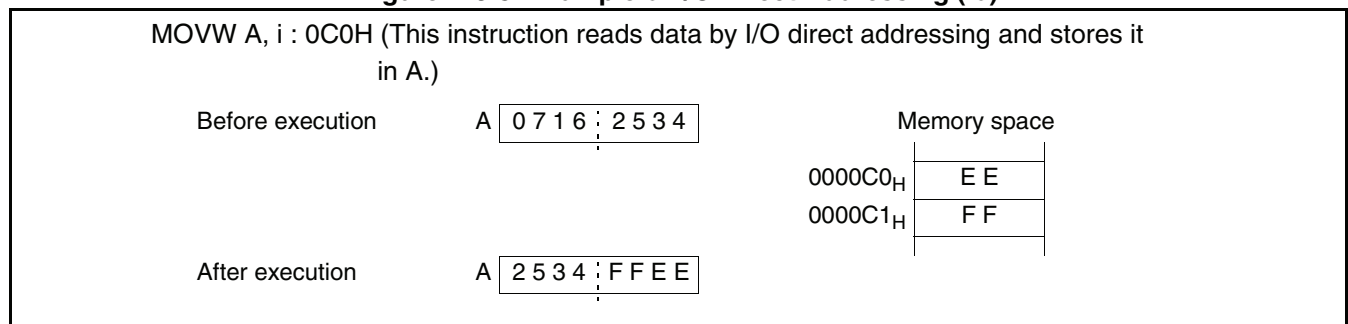
**Figure B.3-4 Example of Direct Branch Addressing (addr24)**



- I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000<sub>H</sub> to 0000FF<sub>H</sub> is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

**Figure B.3-5 Example of I/O Direct Addressing (io)**

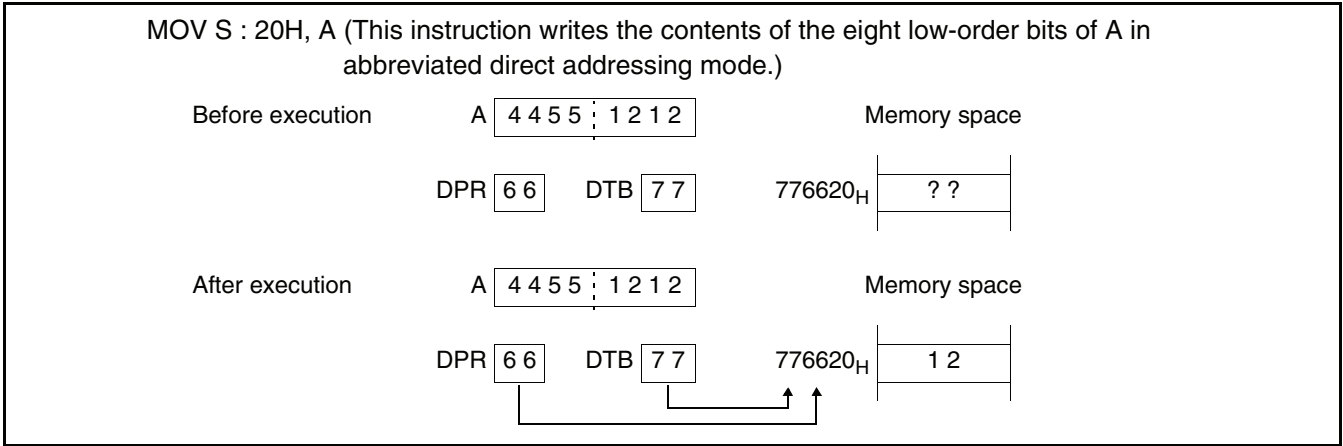


APPENDIX B Instructions

● Abbreviated direct addressing (dir)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB).

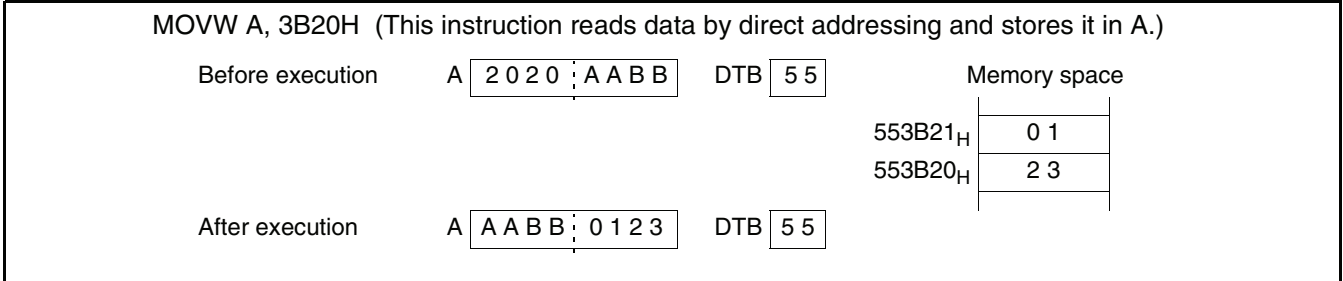
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)



● Direct addressing (addr16)

Specify the 16 low-order bits of a memory address explicitly in an operand. Address bits 16 to 23 are specified by the data bank register (DTB). A prefix instruction for access space addressing is invalid for this mode of addressing.

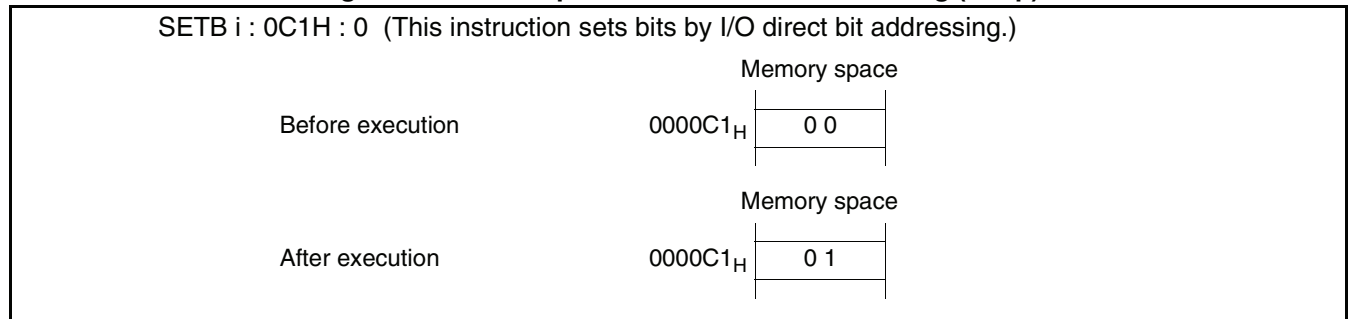
Figure B.3-7 Example of Direct Addressing (addr16)



- I/O direct bit addressing (io:bp)

Specify bits in physical addresses 000000<sub>H</sub> to 0000FF<sub>H</sub> explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

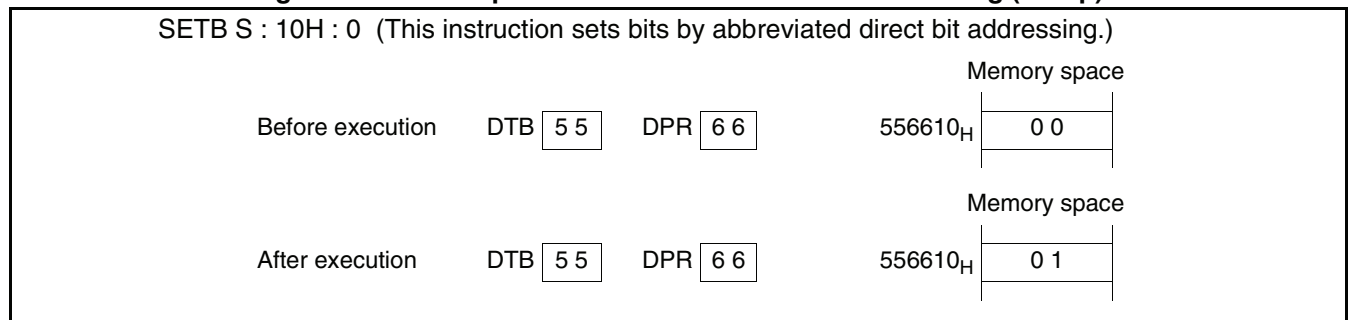
**Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)**



- Abbreviated direct bit addressing (dir:bp)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

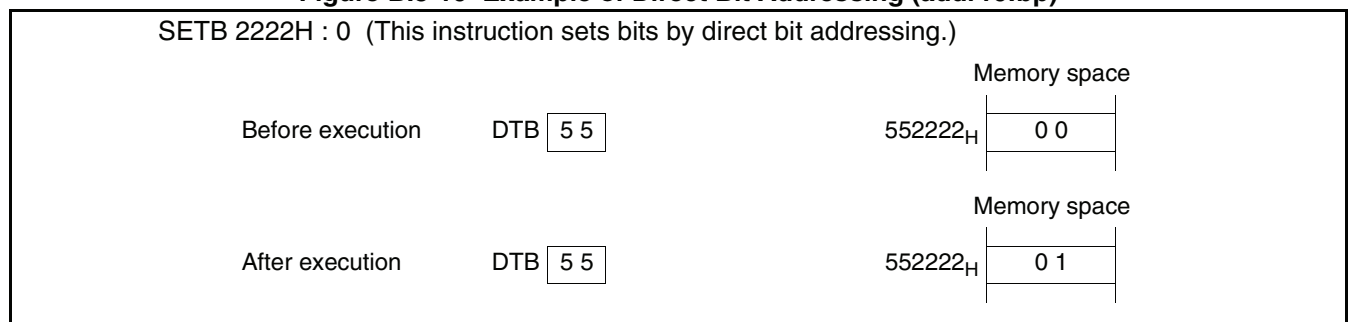
**Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)**



- Direct bit addressing (addr16:bp)

Specify arbitrary bits in 64 kilobytes explicitly. Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

**Figure B.3-10 Example of Direct Bit Addressing (addr16:bp)**



APPENDIX B Instructions

● Vector Addressing (#vct)

Specify vector data in an operand to indicate the branch destination address. There are two sizes for vector numbers: 4 bits and 8 bits. Vector addressing is used for a subroutine call or software interrupt instruction.

Figure B.3-11 Example of Vector Addressing (#vct)

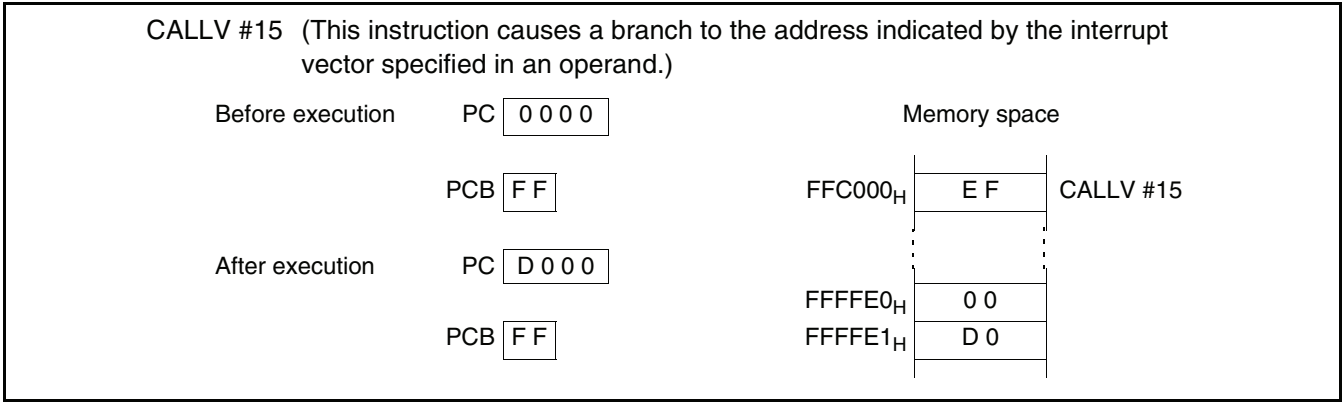


Table B.3-2 CALLV Vector List

Instruction	Vector address L	Vector address H
CALLV #0	XXFFFE <sub>H</sub>	XXFFFF <sub>H</sub>
CALLV #1	XXFFFC <sub>H</sub>	XXFFFD <sub>H</sub>
CALLV #2	XXFFFA <sub>H</sub>	XXFFFB <sub>H</sub>
CALLV #3	XXFFF8 <sub>H</sub>	XXFFF9 <sub>H</sub>
CALLV #4	XXFFF6 <sub>H</sub>	XXFFF7 <sub>H</sub>
CALLV #5	XXFFF4 <sub>H</sub>	XXFFF5 <sub>H</sub>
CALLV #6	XXFFF2 <sub>H</sub>	XXFFF3 <sub>H</sub>
CALLV #7	XXFFF0 <sub>H</sub>	XXFFF1 <sub>H</sub>
CALLV #8	XXFFEE <sub>H</sub>	XXFFEF <sub>H</sub>
CALLV #9	XXFFEC <sub>H</sub>	XXFFED <sub>H</sub>
CALLV #10	XXFFEA <sub>H</sub>	XXFFEB <sub>H</sub>
CALLV #11	XXFFE8 <sub>H</sub>	XXFFE9 <sub>H</sub>
CALLV #12	XXFFE6 <sub>H</sub>	XXFFE7 <sub>H</sub>
CALLV #13	XXFFE4 <sub>H</sub>	XXFFE5 <sub>H</sub>
CALLV #14	XXFFE2 <sub>H</sub>	XXFFE3 <sub>H</sub>
CALLV #15	XXFFE0 <sub>H</sub>	XXFFE1 <sub>H</sub>

Note: A PCB register value is set in XX.

Note:

When the program counter bank register (PCB) is FF<sub>H</sub>, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2).

## B.4 Indirect Addressing

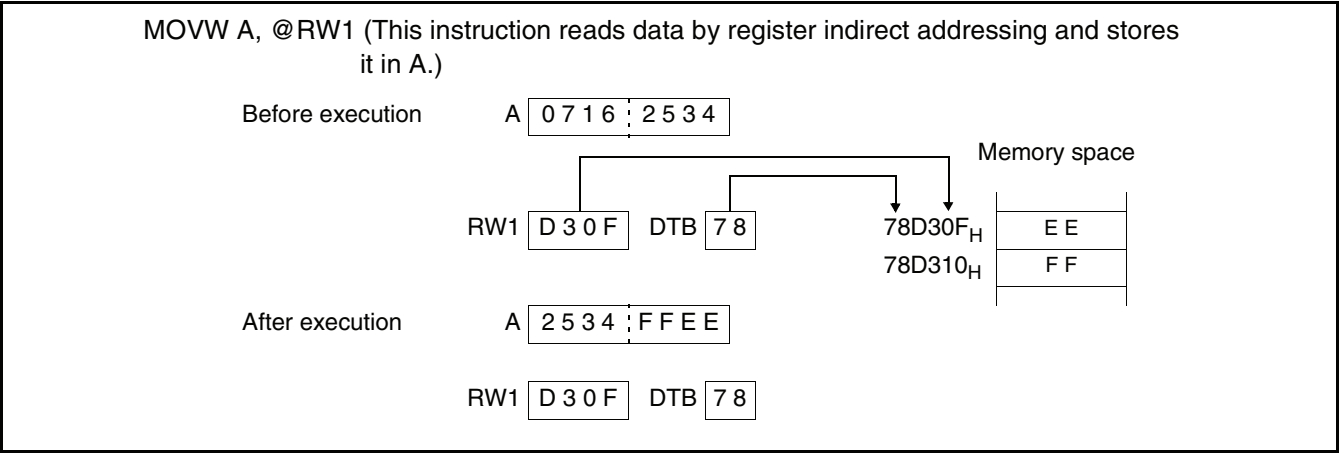
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

### ■ Indirect Addressing

- Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)



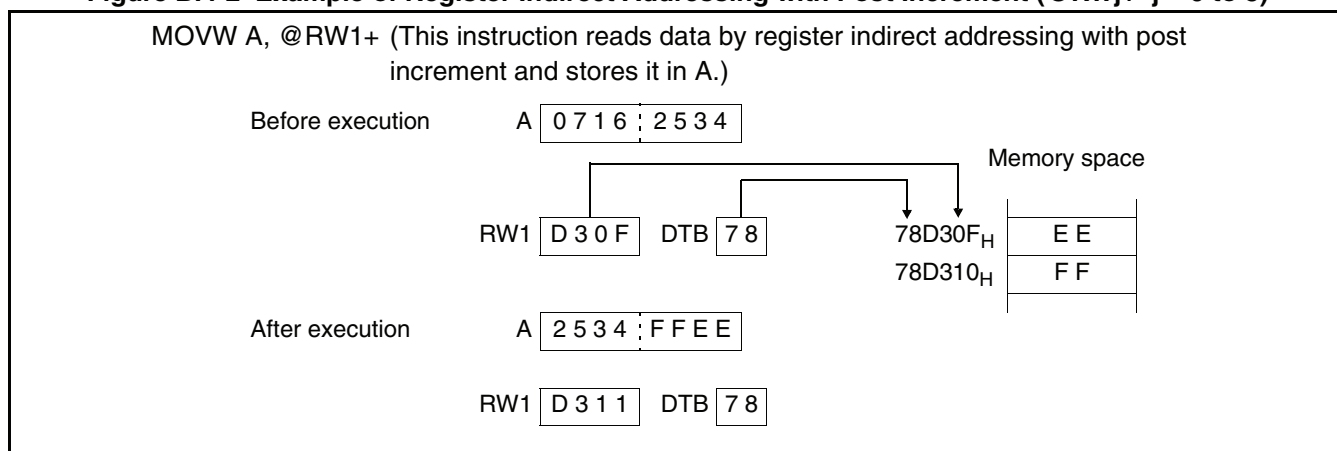
- Register indirect addressing with post increment (@RWj+ j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.



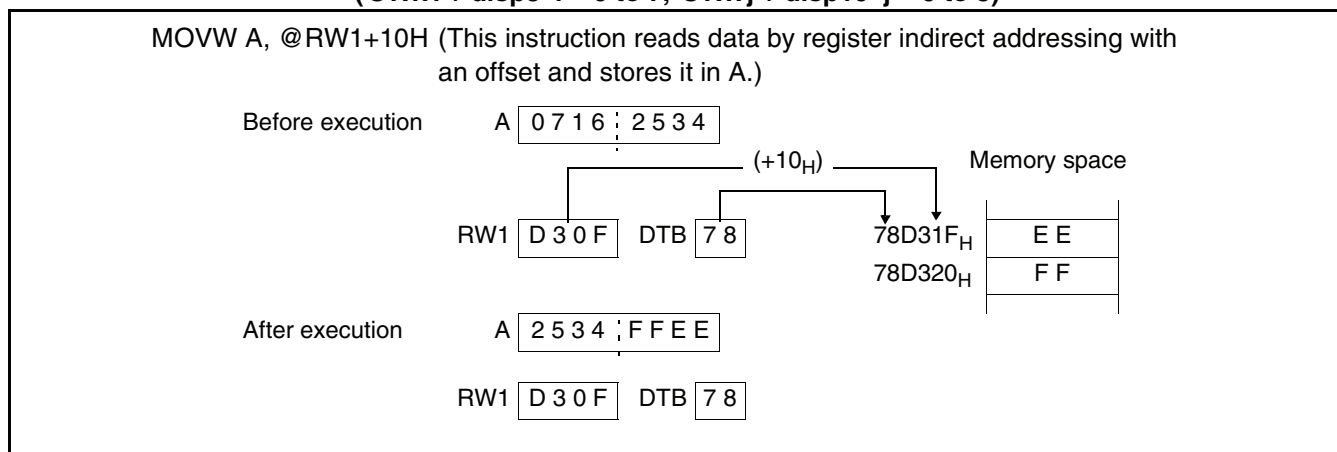
**Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj+ j = 0 to 3)**



- Register indirect addressing with offset ( $@RW_i + \text{disp8}$   $i = 0$  to  $7$ ,  $@RW_j + \text{disp16}$   $j = 0$  to  $3$ )

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

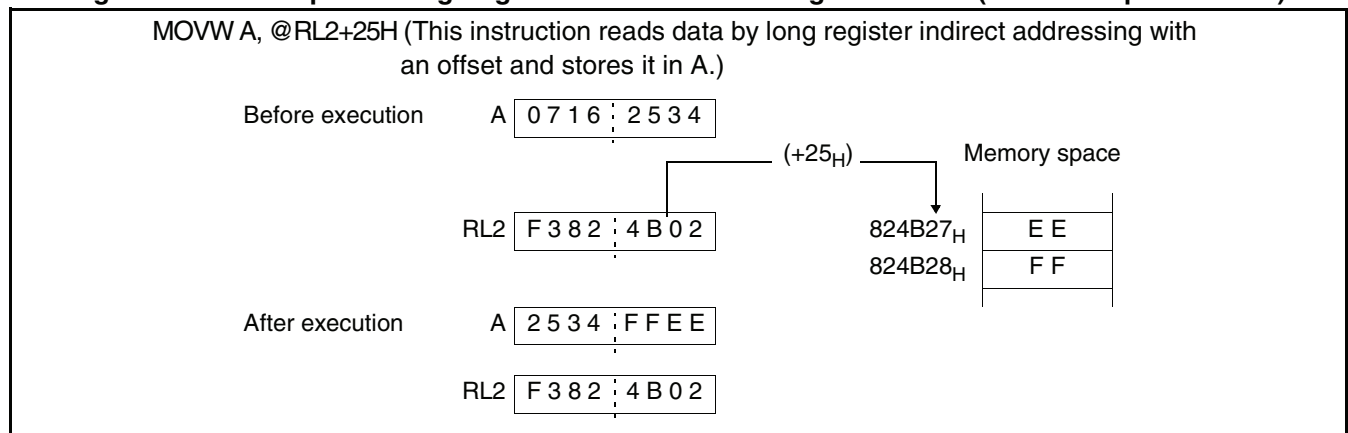
**Figure B.4-3 Example of Register Indirect Addressing with Offset**  
 (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)



● Long register indirect addressing with offset ( $@RLi + \text{disp8}$   $i = 0$  to  $3$ )

Memory is accessed using the address that is the 24 low-order bits obtained by adding an offset to the contents of general-purpose register  $RLi$ . The offset is 8-bits long and is added as a signed numeric value.

**Figure B.4-4 Example of Long Register Indirect Addressing with Offset ( $@RLi + \text{disp8}$   $i = 0$  to  $3$ )**

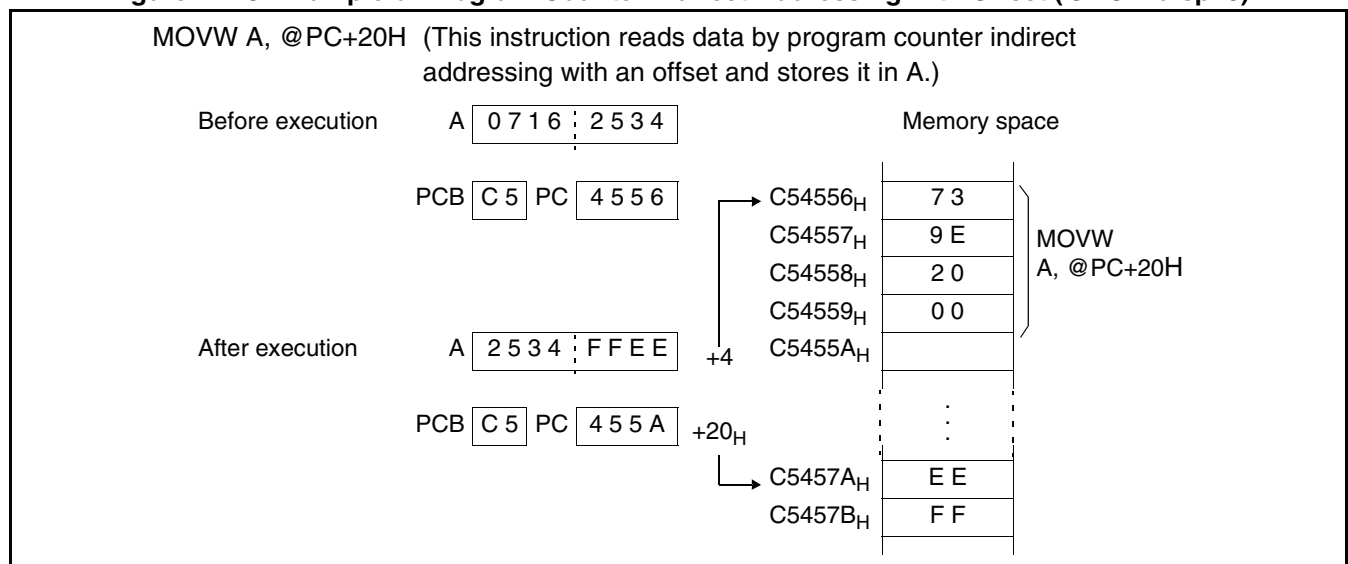


● Program counter indirect addressing with offset ( $@PC + \text{disp16}$ )

Memory is accessed using the address indicated by (instruction address + 4 +  $\text{disp16}$ ). The offset is one word long. Address bits 16 to 23 are specified by the program counter bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address +  $\text{disp16}$ ):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8
- MOVW eam, #imm16

**Figure B.4-5 Example of Program Counter Indirect Addressing with Offset ( $@PC + \text{disp16}$ )**

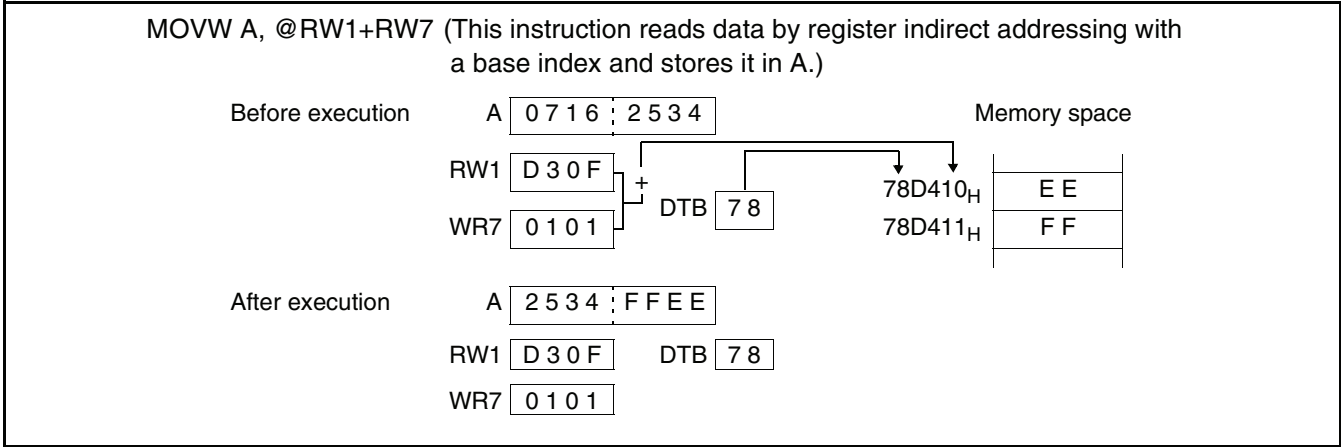


APPENDIX B Instructions

- Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)

Memory is accessed using the address determined by adding RW0 or RW1 to the contents of general-purpose register RW7. Address bits 16 to 23 are indicated by the data bank register (DTB).

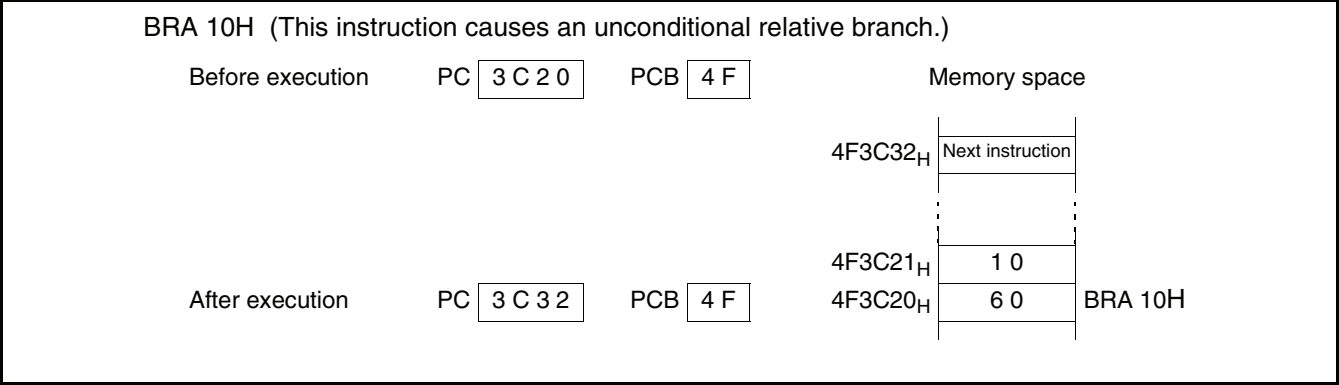
Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)



● Program counter relative branch addressing (rel)

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-kilobyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program counter bank register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



● Register list (rlst)

Specify a register to be pushed onto or popped from a stack.

Figure B.4-8 Configuration of the Register List

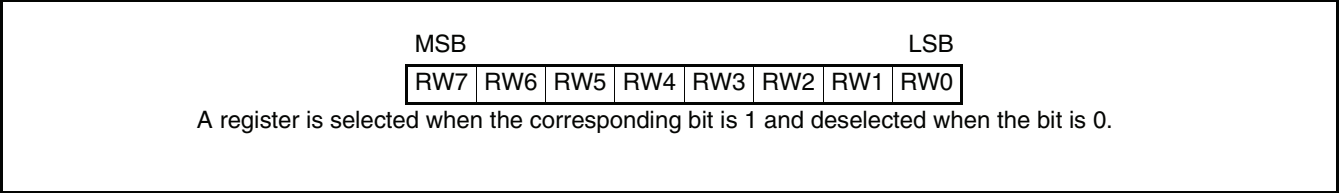
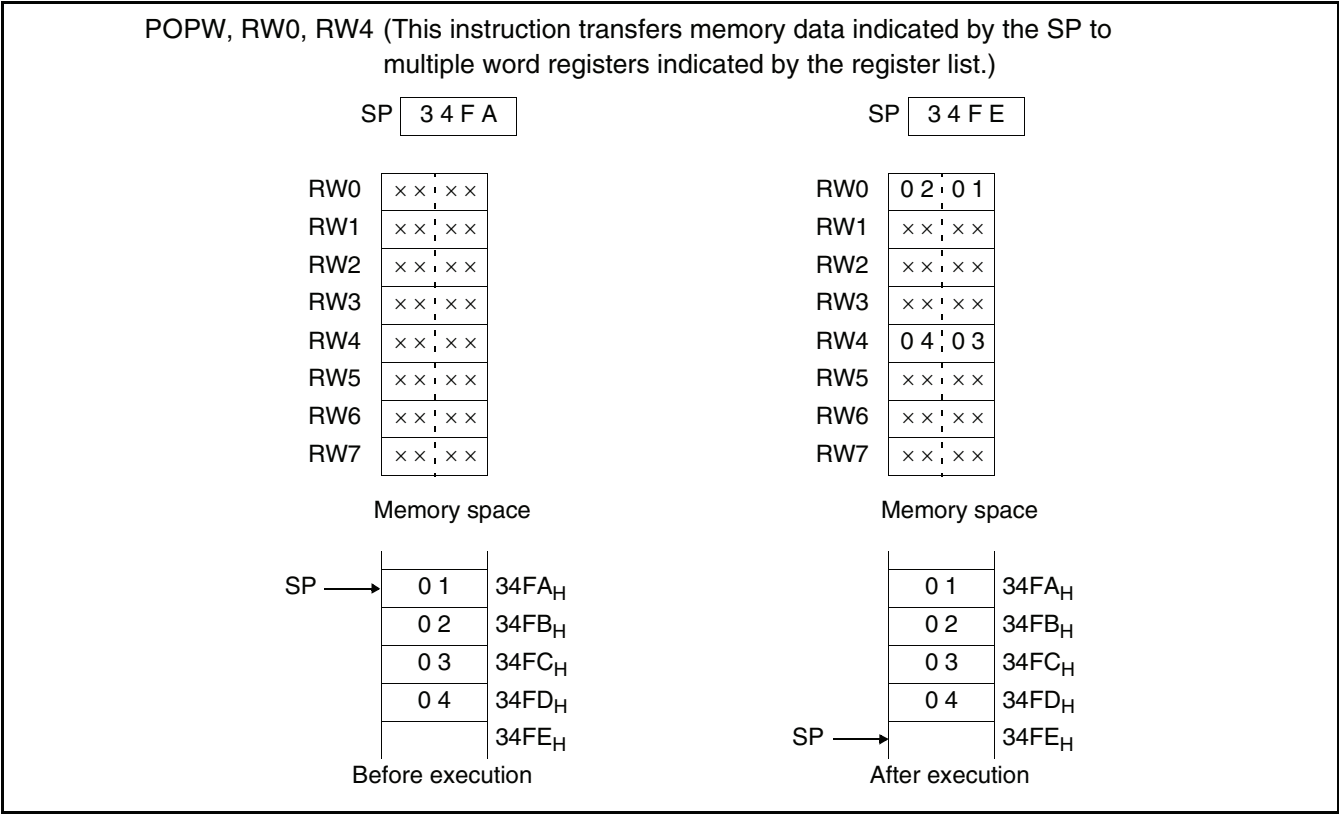


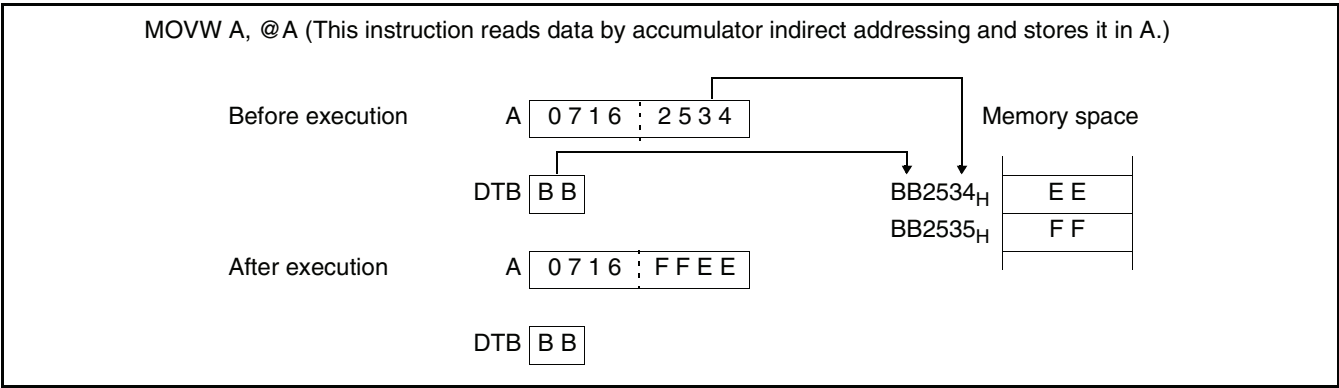
Figure B.4-9 Example of Register List (rlist)



● Accumulator indirect addressing (@A)

Memory is accessed using the address indicated by the contents of the low-order bytes (16 bits) of the accumulator (AL). Address bits 16 to 23 are specified by a mnemonic in the data bank register (DTB).

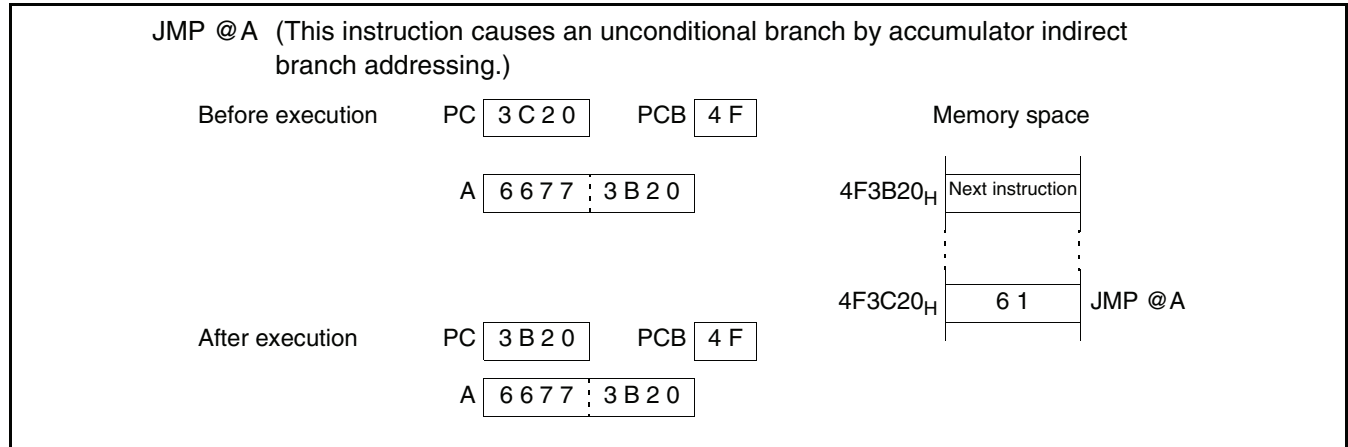
Figure B.4-10 Example of Accumulator Indirect Addressing (@A)



- Accumulator indirect branch addressing (@A)

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program counter bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

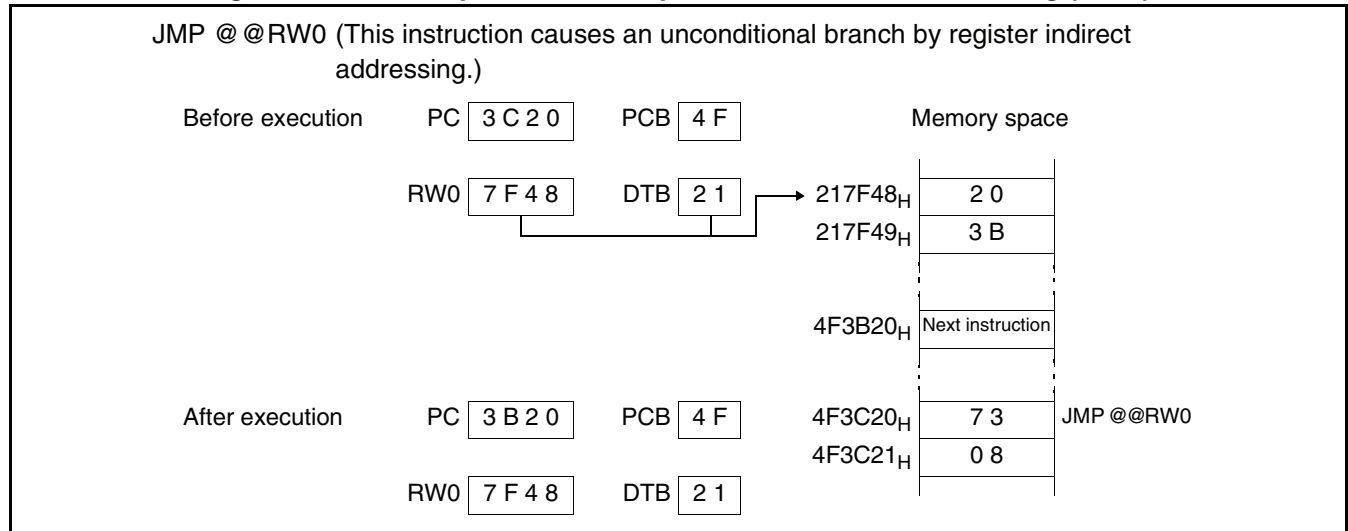
**Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)**



- Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

**Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)**

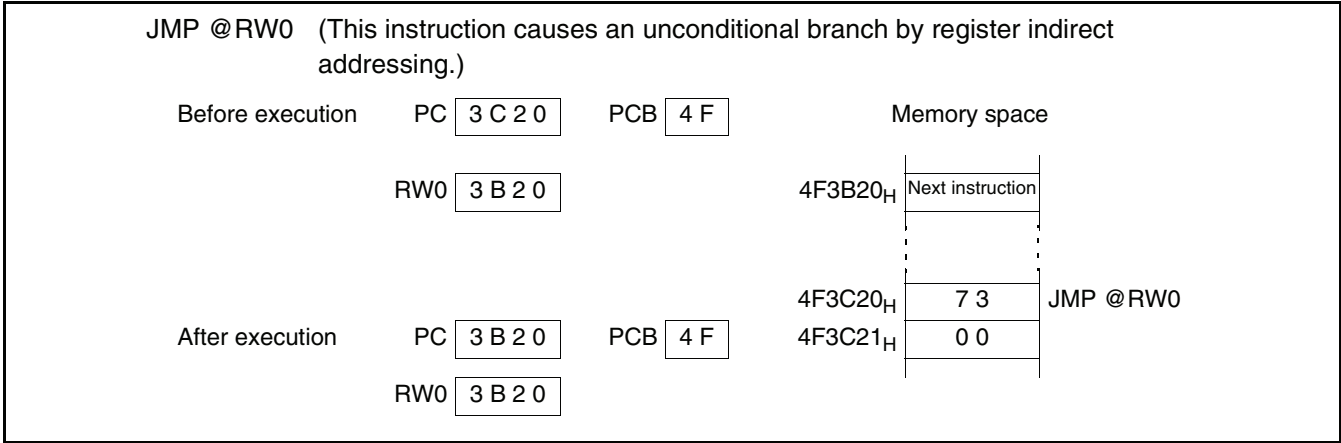


APPENDIX B Instructions

● Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



## B.5 Execution Cycle Count

---

**The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.**

---

### ■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.



## ■ Calculating the Execution Cycle Count

Table B.5-1 lists execution cycle counts and Table B.5-2 and Table B.5-3 summarize correction value data.

**Table B.5-1 Execution Cycle Counts in Each Addressing Mode**

Code	Operand	(a) *	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00   07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08   0B	@RWj	2	1
0C   0F	@RWj+	4	2
10   17	@RWi+disp8	2	1
18   1B	@RWi+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

\*: (a) is used for ~ (cycle count) and B (correction value) in "B.8 F<sup>2</sup>MC-16LX Instruction List".

**Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles**

Operand	(b) byte *		(c) word *		(d) long *	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus 16-bit even address	+1	1	+1	1	+2	2
External data bus 16-bit odd address	+1	1	+4	2	+8	4
External data bus 8-bits	+1	1	+4	2	+8	4

\*: (b), (c), and (d) are used for ~ (cycle count) and B (correction value) in "B.8 F<sup>2</sup>MC-16LX Instruction List".

---

**Note:**

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

---

**Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles**

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16-bits	-	+3
External data bus 8-bits	+3	-

---

**Notes:**

- When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.
  - Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.
-

## B.6 Effective address field

Table B.6-1 shows the effective address field.

### ■ Effective Address Field

**Table B.6-1 Effective Address Field**

Code	Representation			Address format	Byte count of extended address part *
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	0
09	@RW1				
0A	@RW2				
0B	@RW3				
0C	@RW0+			Register indirect with post increment	0
0D	@RW1+				
0E	@RW2+				
0F	@RW3+				
10	@RW0+disp8			Register indirect with 8-bit displacement	1
11	@RW1+disp8				
12	@RW2+disp8				
13	@RW3+disp8				
14	@RW4+disp8				
15	@RW5+disp8				
16	@RW6+disp8				
17	@RW7+disp8				
18	@RW0+disp16			Register indirect with 16-bit displacement	2
19	@RW1+disp16				
1A	@RW2+disp16				
1B	@RW3+disp16				
1C	@RW0+RW7			Register indirect with index	0
1D	@RW1+RW7			Register indirect with index	0
1E	@PC+disp16			PC indirect with 16-bit displacement	2
1F	addr16			Direct address	2

\*1: Each byte count of the extended address part applies to + in the # (byte count) column in "B.8 F<sup>2</sup>MC-16LX Instruction List".

## B.7 How to Read the Instruction List

Table B.7-1 describes the items used in "B.8 F<sup>2</sup>MC-16LX Instruction List", and Table B.7-2 describes the symbols used in the same list.

### ■ Description of Instruction Presentation Items and Symbols

**Table B.7-1 Description of Items in the Instruction List (1/2)**

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number of following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table B.2-1 for the alphabetical letters in items.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bit15 to bit08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00 <sub>H</sub> or FF <sub>H</sub> to AH after AL sign extension.
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change S: Set upon instruction execution. R: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	

**Table B.7-1 Description of Items in the Instruction List (1/2)**

Item	Description
RMW	<p>Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory).</p> <p>*: Read Modify Write instruction</p> <p> -: Not Read Modify Write instruction</p> <p><b>Note:</b></p> <p>Cannot be used for an address that has different meanings between read and write operations.</p>

**Table B.7-2 Explanation on Symbols in the Instruction List (1/2)**

Symbol	Explanation
A	<p>The bit length used varies depending on the 32-bit accumulator instruction.</p> <p>Byte: Low-order 8 bits of byte AL</p> <p>Word: 16 bits of word AL</p> <p>Long word: 32 bits of AL and AH</p>
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
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

**Table B.7-2 Explanation on Symbols in the Instruction List (1/2)**

Symbol	Explanation
ad24 16-23	Bit16 to bit23 of addr24
io	I/O area (000000 <sub>H</sub> to 0000FF <sub>H</sub> )
#imm4	4-bit immediate data
#imm8	8-bit immediate data
#imm16	16-bit immediate data
#imm32	32-bit immediate data
ext (imm8)	16-bit data obtained by sign extension of 8-bit immediate data
disp8	8-bit displacement
disp16	16-bit displacement
bp	Bit offset
vct4	Vector number (0 to 15)
vct8	Vector number (0 to 255)
( ) b	Bit address
rel	PC relative branch
ear	Effective addressing (code 00 to 07)
eam	Effective addressing (code 08 to 1F)
rlst	Register list

## B.8 F<sup>2</sup>MC-16LX Instruction List

Table B.8-1 to Table B.8-18 list the instructions used by the F<sup>2</sup>MC-16LX.

### ■ F<sup>2</sup>MC-16LX Instruction List

Table B.8-1 41 Transfer Instructions (Byte)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A,dir	2	3	0	(b)	byte (A) ← (dir)	Z	*	-	-	-	*	*	-	-	-
MOV A,addr16	3	4	0	(b)	byte (A) ← (addr16)	Z	*	-	-	-	*	*	-	-	-
MOV A,Ri	1	2	1	0	byte (A) ← (Ri)	Z	*	-	-	-	*	*	-	-	-
MOV A,ear	2	2	1	0	byte (A) ← (ear)	Z	*	-	-	-	*	*	-	-	-
MOV A,eam	2+	3 + (a)	0	(b)	byte (A) ← (eam)	Z	*	-	-	-	*	*	-	-	-
MOV A,io	2	3	0	(b)	byte (A) ← (io)	Z	*	-	-	-	*	*	-	-	-
MOV A,#imm8	2	2	0	0	byte (A) ← imm8	Z	*	-	-	-	*	*	-	-	-
MOV A,@A	2	3	0	(b)	byte (A) ← ((A))	Z	-	-	-	-	*	*	-	-	-
MOV A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	Z	*	-	-	-	*	*	-	-	-
MOVN A,#imm4	1	1	0	0	byte (A) ← imm4	Z	*	-	-	-	R	*	-	-	-
MOVX A,dir	2	3	0	(b)	byte (A) ← (dir)	X	*	-	-	-	*	*	-	-	-
MOVX A,addr16	3	4	0	(b)	byte (A) ← (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX A,Ri	2	2	1	0	byte (A) ← (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX A,ear	2	2	1	0	byte (A) ← (ear)	X	*	-	-	-	*	*	-	-	-
MOVX A,eam	2+	3 + (a)	0	(b)	byte (A) ← (eam)	X	*	-	-	-	*	*	-	-	-
MOVX A,io	2	3	0	(b)	byte (A) ← (io)	X	*	-	-	-	*	*	-	-	-
MOVX A,#imm8	2	2	0	0	byte (A) ← imm8	X	*	-	-	-	*	*	-	-	-
MOVX A,@A	2	3	0	(b)	byte (A) ← ((A))	X	-	-	-	-	*	*	-	-	-
MOVX A,@RWi+disp8	2	5	1	(b)	byte (A) ← ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOV dir,A	2	3	0	(b)	byte (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV addr16,A	3	4	0	(b)	byte (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,A	1	2	1	0	byte (Ri) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV ear,A	2	2	1	0	byte (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV eam,A	2+	3 + (a)	0	(b)	byte (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV io,A	2	3	0	(b)	byte (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV @RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,ear	2	3	2	0	byte (Ri) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOV Ri,eam	2+	4 + (a)	1	(b)	byte (Ri) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOV ear,Ri	2	4	2	0	byte (ear) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV eam,Ri	2+	5 + (a)	1	(b)	byte (eam) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV Ri,#imm8	2	2	1	0	byte (Ri) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV io,#imm8	3	5	0	(b)	byte (io) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV dir,#imm8	3	5	0	(b)	byte (dir) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV ear,#imm8	3	2	1	0	byte (ear) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV eam,#imm8	3+	4 + (a)	0	(b)	byte (eam) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV @AL,AH	2	3	0	(b)	byte ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCH A,ear	2	4	2	0	byte (A) ↔ (ear)	Z	-	-	-	-	-	-	-	-	-
XCH A,eam	2+	5 + (a)	0	2 × (b)	byte (A) ↔ (eam)	Z	-	-	-	-	-	-	-	-	-
XCH Ri,ear	2	7	4	0	byte (Ri) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCH Ri,eam	2+	9 + (a)	2	2 × (b)	byte (Ri) ↔ (eam)	-	-	-	-	-	-	-	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-2 38 Transfer Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,dir	2	3	0	(c)	word (A) ← (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	4	0	(c)	word (A) ← (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	1	1	0	0	word (A) ← (SP)	-	*	-	-	-	*	*	-	-	-
MOVW A,RWi	1	2	1	0	word (A) ← (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW A,ear	2	2	1	0	word (A) ← (ear)	-	*	-	-	-	*	*	-	-	-
MOVW A,eam	2+	3 + (a)	0	(c)	word (A) ← (eam)	-	*	-	-	-	*	*	-	-	-
MOVW A,io	2	3	0	(c)	word (A) ← (io)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	3	0	(c)	word (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVW A,#imm16	3	2	0	0	word (A) ← imm16	-	*	-	-	-	*	*	-	-	-
MOVW A,@RWi+disp8	2	5	1	(c)	word (A) ← ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@RLi+disp8	3	10	2	(c)	word (A) ← ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW dir,A	2	3	0	(c)	word (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW addr16,A	3	4	0	(c)	word (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW 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 B.5-1 and Table B.5-2 for information on (a), (c), and (d) in the table.



## APPENDIX B Instructions

**Table B.8-3 42 Addition/Subtraction Instructions (Byte, Word, Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD A,#imm8	2	2	0	0	byte (A) ← (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	5	0	(b)	byte (A) ← (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	3	1	0	byte (A) ← (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	3	2	0	byte (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD eam,A	2+	5 + (a)	0	2 × (b)	byte (eam) ← (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	0	byte (A) ← (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	3	1	0	byte (A) ← (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDDC A	1	3	0	0	byte (A) ← (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	0	byte (A) ← (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	5	0	(b)	byte (A) ← (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	3	1	0	byte (A) ← (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	3	2	0	byte (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB eam,A	2+	5 + (a)	0	2 × (b)	byte (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	0	byte (A) ← (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	3	1	0	byte (A) ← (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC A	1	3	0	0	byte (A) ← (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	0	word (A) ← (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	3	1	0	word (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	0	word (A) ← (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	3	2	0	word (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	3	1	0	word (A) ← (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	0	word (A) ← (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	3	1	0	word (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	0	word (A) ← (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	3	2	0	word (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	3	1	0	word (A) ← (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL A,ear	2	6	2	0	long (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	0	long (A) ← (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	6	2	0	long (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	0	long (A) ← (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

**Table B.8-4 12 Increment/decrement Instructions (Byte, Word, Long Word)**

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2 $\times$ (d)	long (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2 $\times$ (d)	long (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

**Table B.8-5 11 Compare Instructions (Byte, Word, Long Word)**

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

## APPENDIX B Instructions

**Table B.8-6 11 Unsigned Multiplication/Division Instructions (Word, Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU A	1	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW A	1	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

\*1: 3: Division by 0 7: Overflow 15: Normal  
 \*2: 4: Division by 0 8: Overflow 16: Normal  
 \*3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal  
 \*4: 4: Division by 0 7: Overflow 22: Normal  
 \*5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal  
 \*6: (b): Division by 0 or overflow 2 × (b): Normal  
 \*7: (c): Division by 0 or overflow 2 × (c): Normal  
 \*8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.  
 \*9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.  
 \*10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.  
 \*11: 3: Word (AH) is 0. 11: Word (AH) is not 0.  
 \*12: 4: Word (ear) is 0. 12: Word (ear) is not 0.  
 \*13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

### Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

**Table B.8-7 11 Signed Multiplication/Division Instructions (Word, Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV A	2	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL A	2	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW A	2	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

\*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal

\*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal

\*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal

\*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal  
When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal

\*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal  
When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal

\*6: (b): Division by 0 or overflow, 2 × (b): Normal

\*7: (c): Division by 0 or overflow, 2 × (c): Normal

\*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative

\*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative

\*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative

\*11: 3: Word (AH) is 0, 16: result is positive, 19: result is negative

\*12: 4: Word (ear) is 0, 17: result is positive, 20: result is negative

\*13: 5+(a): Word (eam) is 0, 18+(a): result is positive, 21+(a): result is negative

**Notes:**

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

## APPENDIX B Instructions

**Table B.8-8 39 Logic 1 Instructions (Byte, Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A,#imm8	2	2	0	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	3	1	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	2	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	3	1	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	2	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	3	1	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	2	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	3	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOT eam	2+	5+(a)	0	2 × (b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	3	1	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	2	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	3	1	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	2	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	3	1	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	2	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	3	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW eam	2+	5+(a)	0	2 × (c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

**Table B.8-9 6 Logic 2 Instructions (Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	6	2	0	long (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	long (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	long (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (d) in the table.

**Table B.8-10 6 Sign Inversion Instructions (Byte, Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	0	byte (A) ← 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	byte (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 × (b)	byte (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	word (A) ← 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 × (c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

**Table B.8-11 1 Normalization Instruction (Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A,R0	2	*1	1	0	long (A) ← Shift left to the position where '1' is set for the first time. byte (R0) ← Shift count at that time	-	-	-	-	-	-	*	-	-	-

\*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

## APPENDIX B Instructions

**Table B.8-12 18 Shift Instructions (Byte, Word, Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	3	2	0	byte (ear) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	*
ASR A,R0	2	*1	1	0	byte (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSR A,R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSL A,R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW A/SHRW A	1	2	0	0	word (A) ← Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW A/SHLW A	1	2	0	0	word (A) ← Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW A,R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW A,R0	2	*1	1	0	word (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL A,R0	2	*2	1	0	long (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL A,R0	2	*2	1	0	long (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL A,R0	2	*2	1	0	long (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

\*1: 6 when R0 is 0; otherwise, 5 + (R0)

\*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-13 31 Branch 1 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/ BNE	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/ BHS	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT rel	2	*1	0	0	Branch on (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE rel	2	*1	0	0	Branch on (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP @A	1	2	0	0	word (PC) ← (A)	-	-	-	-	-	-	-	-	-	-
JMP addr16	3	3	0	0	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
JMP @ear	2	3	1	0	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
JMP @eam	2+	4+(a)	0	(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
JMPP @ear *3	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP @eam *3	2+	6+(a)	0	(d)	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP addr24	4	4	0	0	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL @ear *4	2	6	1	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
CALL @eam *4	2+	7+(a)	0	2 × (c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
CALL addr16 *5	3	6	0	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
CALLV #vct4 *5	1	7	0	2 × (c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP @ear *6	2	10	2	2 × (c)	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
CALLP @eam *6	2+	11+(a)	0	*2	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
CALLP addr24 *7	4	10	0	2 × (c)	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-

\*1: 4 when a branch is made; otherwise, 3

\*2: 3 × (c) + (b)

\*3: Read (word) of branch destination address

\*4: W: Save to stack (word) R: Read (word) of branch destination address

\*5: Save to stack (word)

\*6: W: Save to stack (long word), R: Read (long word) of branch destination address

\*7: Save to stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.



## APPENDIX B Instructions

**Table B.8-14 19 Branch 2 Instructions**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE eam,#imm16,rel *9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ ear,rel	3	*5	2	0	byte (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ eam,rel	3+	*6	2	2 × (b)	byte (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*5	2	0	word (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ eam,rel	3+	*6	2	2 × (c)	word (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	16	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	17	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI	1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK #imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK	1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET *10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP *11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

\*1: 5 when a branch is made; otherwise, 4

\*2: 13 when a branch is made; otherwise, 12

\*3: 7+(a) when a branch is made; otherwise, 6+(a)

\*4: 8 when a branch is made; otherwise, 7

\*5: 7 when a branch is made; otherwise, 6

\*6: 8+(a) when a branch is made; otherwise, 7+(a)

\*7: 3 × (b) + 2 × (c) when jumping to the next interruption request; 6 × (c) when returning from the current interruption

\*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption

\*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.

\*10: Return from stack (word)

\*11: Return from stack (long word)

### Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

**Table B.8-15 28 Other Control Instructions (Byte, Word, Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	4	0	(c)	word (SP) $\leftarrow$ (SP) - 2, ((SP)) $\leftarrow$ (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	4	0	(c)	word (SP) $\leftarrow$ (SP) - 2, ((SP)) $\leftarrow$ (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	4	0	(c)	word (SP) $\leftarrow$ (SP) - 2, ((SP)) $\leftarrow$ (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*5	*4	(SP) $\leftarrow$ (SP) - 2n, ((SP)) $\leftarrow$ (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	0	(c)	word (A) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	0	(c)	word (AH) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	4	0	(c)	word (PS) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*5	*4	(rlst) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2n	-	-	-	-	-	-	-	-	-	-
JCTX @A	1	14	0	6 $\times$ (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	0	byte (CCR) $\leftarrow$ (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	0	byte (CCR) $\leftarrow$ (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	0	byte (RP) $\leftarrow$ imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	0	byte (ILM) $\leftarrow$ imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	1	0	word (RWi) $\leftarrow$ ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	1	0	word (RWi) $\leftarrow$ eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	1	0	0	word (A) $\leftarrow$ ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	0	word (A) $\leftarrow$ eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	0	word (SP) $\leftarrow$ (SP) + ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	0	word (SP) $\leftarrow$ (SP) + imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	0	byte (A) $\leftarrow$ (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	0	byte (brg2) $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	-
NOP	1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB	1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB	1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB	1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB	1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC	1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR	1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

\*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2

\*2:  $7 + 3 \times (\text{POP count}) + 2 \times (\text{POP last register number})$ , 7 when RLST = 0 (no transfer register)\*3:  $29 + 3 \times (\text{PUSH count}) - 3 \times (\text{PUSH last register number})$ , 8 when RLST = 0 (no transfer register)\*4:  $(\text{POP count}) \times (\text{c})$  or  $(\text{PUSH count}) \times (\text{c})$ \*5:  $(\text{POP count})$  or  $(\text{PUSH count})$ 

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (c) in the table.

## APPENDIX B Instructions

**Table B.8-16 21 Bit Operand Instructions**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A,dir:bp	3	5	0	(b)	byte (A) $\leftarrow$ (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,addr16:bp	4	5	0	(b)	byte (A) $\leftarrow$ (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,io:bp	3	4	0	(b)	byte (A) $\leftarrow$ (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB dir:bp,A	3	7	0	$2 \times (b)$	bit (dir:bp)b $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	7	0	$2 \times (b)$	bit (addr16:bp)b $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	6	0	$2 \times (b)$	bit (io:bp)b $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	7	0	$2 \times (b)$	bit (dir:bp)b $\leftarrow$ 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	7	0	$2 \times (b)$	bit (addr16:bp)b $\leftarrow$ 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	7	0	$2 \times (b)$	bit (io:bp)b $\leftarrow$ 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	7	0	$2 \times (b)$	bit (dir:bp)b $\leftarrow$ 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	7	0	$2 \times (b)$	bit (addr16:bp)b $\leftarrow$ 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	7	0	$2 \times (b)$	bit (io:bp)b $\leftarrow$ 0	-	-	-	-	-	-	-	-	-	*
BBC dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*3	0	$2 \times (b)$	Branch on (addr16:bp) b = 1, bit (addr16:bp) b $\leftarrow$ 1	-	-	-	-	-	-	*	-	-	*
WBTS io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

\*1: 8 when a branch is made; otherwise, 7

\*2: 7 when a branch is made; otherwise, 6

\*3: 10 when the condition is met; otherwise, 9

\*4: Undefined count

\*5: Until the condition is met

Note:

See Table B.5-1 and Table B.5-2 for information on (b) in the table.

**Table B.8-17 6 Accumulator Operation Instructions (Byte, Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 $\leftrightarrow$ (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW	1	2	0	0	word (AH) $\leftrightarrow$ (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extension	-	Z	-	-	-	R	*	-	-	-

**Table B.8-18 10 String Instructions**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*5	*3	byte transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*5	*3	byte transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*8	*4	byte search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*8	*4	byte search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*8	*3	byte fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*5	*6	word transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*5	*6	word transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*8	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*8	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*8	*6	word fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

\*1: 5 when RW0 is 0,  $4 + 7 \times (\text{RW0})$  when the counter expires, or  $7n + 5$  when a match occurs

\*2: 5 when RW0 is 0; otherwise,  $4 + 8 \times (\text{RW0})$

\*3:  $(b) \times (\text{RW0}) + (b) \times (\text{RW0})$  When the source and destination access different areas, calculate the (b) item individually.

\*4:  $(b) \times n$

\*5:  $2 \times (b) \times (\text{RW0})$

\*6:  $(c) \times (\text{RW0}) + (c) \times (\text{RW0})$  When the source and destination access different areas, calculate the (c) item individually.

\*7:  $(c) \times n$

\*8:  $(b) \times (\text{RW0})$

**Note:**

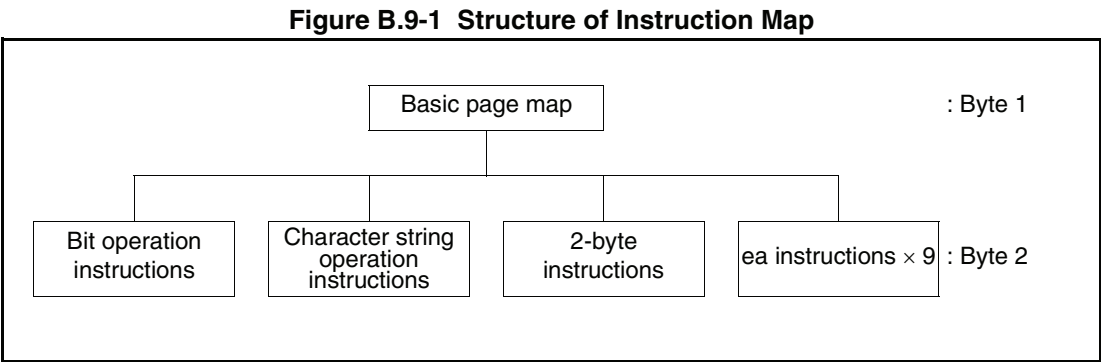
m: RW0 value (counter value), n: Loop count

See Table B.5-1 and Table B.5-2 for information on (b) and (c) in the table.

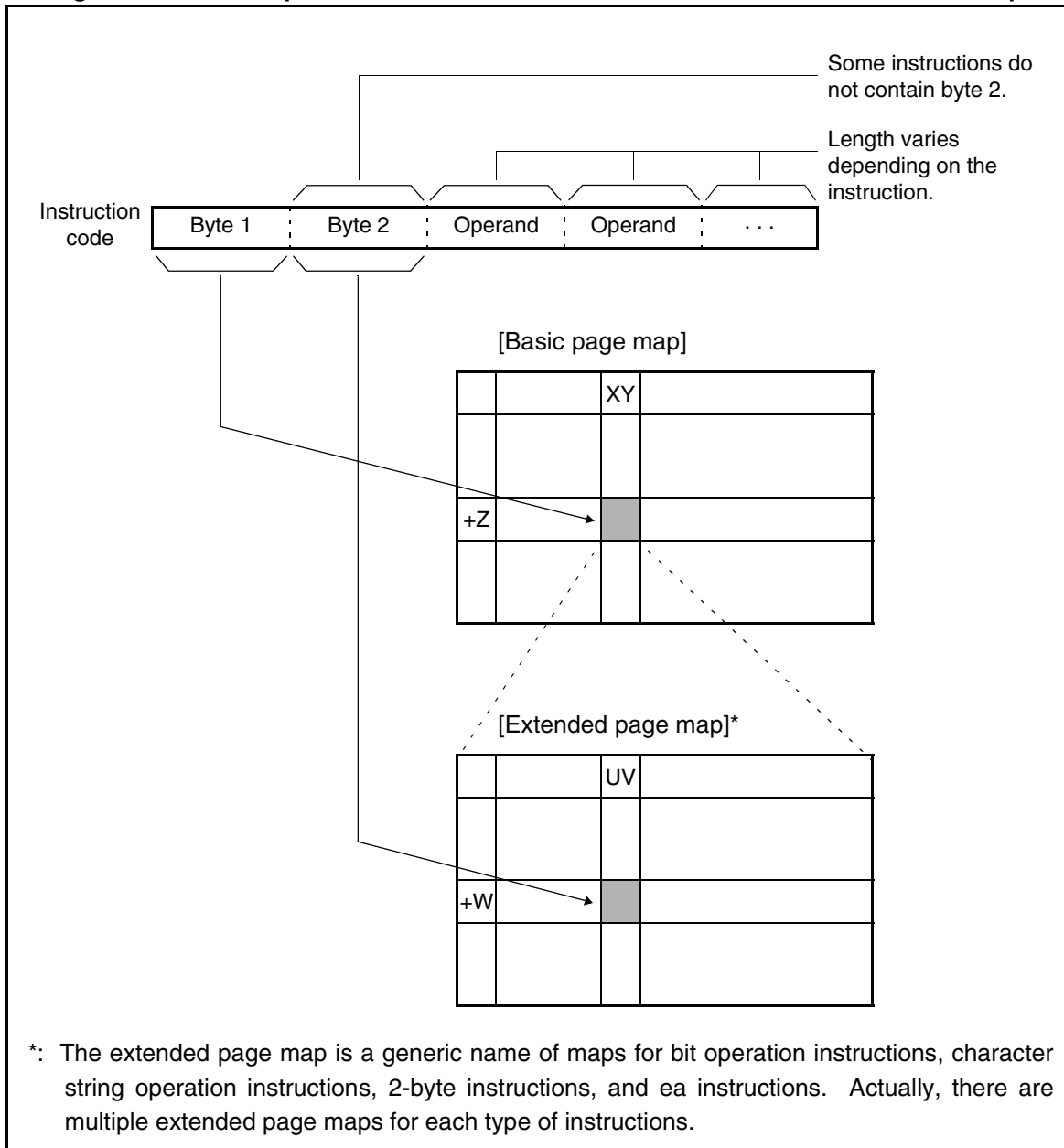
# B.9 Instruction Map

Each F<sup>2</sup>MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 to Table B.9-21 summarize the F<sup>2</sup>MC-16LX instruction map.

■ Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 shows the correspondence between an actual instruction code and instruction map.

**Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map**

An example of an instruction code is shown in Table B.9-1.

**Table B.9-1 Example of an Instruction Code**

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-
MOV A, ADB	60 +F=6F	00 +0=00
@RW2+d8, #8, rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	NOP	CMR	ADD A, dir	ADD A, #8	MOV A, dir	MOV A, io	BRA rel	ea instruction 1	MOV A, Ri	MOV Ri, A	MOV Ri, #8	MOV A, Ri	MOVX A, @RWI+d8	MOV A, #4	CALL #4	BZ/BEQ rel
+1	INT9	NCC	SUB A, dir	SUB A, #8	MOV dir, A	MOV io, A	JMP @A	ea instruction 2								BNZ/BNE rel
+2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A, #8	MOV A, addr16	JMP addr16	ea instruction 3								BC/BLO rel
+3	NEG A	JCTX @A	CMP A	CMP A, #8	MOVX A, #8	MOV addr16, A	JMPP addr24	ea instruction 4								BNC/BHS rel
+4	PCB	EXT	AND CCR, #8	AND A, #8	MOV dir, #8	MOV io, #8	CALL addr16	ea instruction 5								BN rel
+5	DTB	ZEXT	OR CCR, #8	OR A, #8	MOVX A, dir	MOVX A, io	CALLP addr24	ea instruction 6								BP rel
+6	ADB	SWAP	DIVU A	XOR A, #8	MOVW A, SP	MOVW io, #16	RETP	ea instruction 7								BV rel
+7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP, A	MOVX A, addr16	RET	ea instruction 8								BNV rel
+8	LINK #imm8	ADDL A, #32	ADDW A	ADDW A, #16	MOVW A, dir	MOVW A, io	INT #vct8	ea instruction 9	MOVW A, RWI	MOVW RWI, A	MOVW RWI, #16	MOVW 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 addr16	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 addr24	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 A	XORW A	XORW A, #16	PUSHW PS	POPW PS	Character string operation instruction	XCH Ri, ea								BLS rel
+F	LSRW A	ADDSP #16	MULW A	NOTW A	PUSHW rlist	POPW rlist	2-byte instruction	XCHW RWI, ea								BHI rel

Table B.9-3 Bit Operation Instruction Map (First Byte = 6C<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV B, A, io:bp		MOV B, io:bp, A		CLRB, io:bp		SETB, io:bp		BBC, io:bp, rel		BBS, io:bp, rel				WBTC, io:bp	
+1																
+2																
+3																
+4																
+5																
+6																
+7																
+8	MOV B, A, dir:bp	MOV B, A, addr16:bp	MOV B, dir:bp, A	MOV B, addr16:bp, A	CLRB, dir:bp	CLRB, addr16:bp	SETB, dir:bp	SETB, addr16:bp	BBC, dir:bp, rel	BBC, addr16:bp, rel	BBS, dir:bp, rel	BBS, addr16:bp, rel				SBBS, addr16:bp
+9																
+A																
+B																
+C																
+D																
+E																
+F																



Table B.9-4 Character String Operation Instruction Map (First Byte = 6E<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI PCB, PCB	MOVSD	MOVSWI	MOVSWD					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB			
+1	PCB, DTB								DTB	DTB	DTB	DTB	DTB		DTB	
+2	PCB, ADB								ADB	ADB	ADB	ADB	ADB		ADB	
+3	PCB, SPB								SPB	SPB	SPB	SPB	SPB		SPB	
+4	DTB, PCB															
+5	DTB, DTB															
+6	DTB, ADB															
+7	DTB, SPB															
+8	ADB, PCB															
+9	ADB, DTB															
+A	ADB, ADB															
+B	ADB, SPB															
+C	SPB, PCB															
+D	SPB, DTB															
+E	SPB, ADB															
+F	SPB, SPB															

Table B.9-5 2-byte Instruction Map (First Byte = 6F<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A, DTB	MOV DTB, A	MOVX A, @RL0+d8	MOV @RL0+d8, A	MOV A, @RL0+d8											
+1	MOV A, ADB	MOV ADB, A														
+2	MOV A, SSB	MOV SSB, A	MOVX A, @RL1+d8	MOV @RL1+d8, A	MOV A, @RL1+d8											
+3	MOV A, USB	MOV USB, A														
+4	MOV A, DPR	MOV DPR, A	MOVX A, @RL2+d8	MOV @RL2+d8, A	MOV A, @RL2+d8											
+5	MOV A, @A	MOV @AL, AH														
+6	MOV A, PCB	MOV A, @A	MOVX A, @RL3+d8	MOV @RL3+d8, A	MOV A, @RL3+d8											
+7	ROL A	ROL A														
+8				MOVW @RL0+d8, A	MOVW A, @RL0+d8		MUL A									
+9							MULW A									
+A				MOVW @RL1+d8, A	MOVW A, @RL1+d8		DIVU A									
+B																
+C	LSLW A, R0	LSLL A, R0	LSL A, R0	MOVW @RL2+d8, A	MOVW A, @RL2+d8											
+D	MOVW A, @A	MOVW @AL, AH	NRML A, R0													
+E	ASRW A, R0	ASRL A, R0	ASR A, R0	MOVW @RL3+d8, A	MOVW A, @RL3+d8											
+F	LSRW A, R0	LSRL A, R0	LSR A, R0													

Table B.9-6 ea Instruction 1 (First Byte = 70<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBN ↓, CWBNE ↓										CWBN ↓, CWBNE ↓	
+0	ADDL A, RLO', @RW0+d8	SUBL A, RLO', @RW0+d8	SUBL A, RLO', @RW0+d8	SUBL A, RLO', @RW0+d8	CMPL A, RLO', @RW0+d8	ANDL A, RLO', @RW0+d8	ANDL A, RLO', @RW0+d8	ANDL 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	R0', @RW0+d8, #8, rel
+1	ADDL A, RLO', @RW1+d8	SUBL A, RLO', @RW1+d8	SUBL A, RLO', @RW1+d8	SUBL A, RLO', @RW1+d8	CMPL A, RLO', @RW1+d8	ANDL A, RLO', @RW1+d8	ANDL A, RLO', @RW1+d8	ANDL 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	R1', @RW1+d8, #8, rel
+2	ADDL A, RL1', @RW2+d8	SUBL A, RL1', @RW2+d8	SUBL A, RL1', @RW2+d8	SUBL A, RL1', @RW2+d8	CMPL A, RL1', @RW2+d8	ANDL A, RL1', @RW2+d8	ANDL A, RL1', @RW2+d8	ANDL 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	R2', @RW2+d8, #8, rel
+3	ADDL A, RL1', @RW3+d8	SUBL A, RL1', @RW3+d8	SUBL A, RL1', @RW3+d8	SUBL A, RL1', @RW3+d8	CMPL A, RL1', @RW3+d8	ANDL A, RL1', @RW3+d8	ANDL A, RL1', @RW3+d8	ANDL 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	R3', @RW3+d8, #8, rel
+4	ADDL A, RL2', @RW4+d8	SUBL A, RL2', @RW4+d8	SUBL A, RL2', @RW4+d8	SUBL A, RL2', @RW4+d8	CMPL A, RL2', @RW4+d8	ANDL A, RL2', @RW4+d8	ANDL A, RL2', @RW4+d8	ANDL 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	R4', @RW4+d8, #8, rel
+5	ADDL A, RL2', @RW5+d8	SUBL A, RL2', @RW5+d8	SUBL A, RL2', @RW5+d8	SUBL A, RL2', @RW5+d8	CMPL A, RL2', @RW5+d8	ANDL A, RL2', @RW5+d8	ANDL A, RL2', @RW5+d8	ANDL 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	R5', @RW5+d8, #8, rel
+6	ADDL A, RL3', @RW6+d8	SUBL A, RL3', @RW6+d8	SUBL A, RL3', @RW6+d8	SUBL A, RL3', @RW6+d8	CMPL A, RL3', @RW6+d8	ANDL A, RL3', @RW6+d8	ANDL A, RL3', @RW6+d8	ANDL 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	R6', @RW6+d8, #8, rel
+7	ADDL A, RL3', @RW7+d8	SUBL A, RL3', @RW7+d8	SUBL A, RL3', @RW7+d8	SUBL A, RL3', @RW7+d8	CMPL A, RL3', @RW7+d8	ANDL A, RL3', @RW7+d8	ANDL A, RL3', @RW7+d8	ANDL 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	R7', @RW7+d8, #8, rel
+8	ADDL A, @RW0, @RW0+d16	SUBL A, @RW0, @RW0+d16	SUBL A, @RW0, @RW0+d16	SUBL A, @RW0, @RW0+d16	CMPL A, @RW0, @RW0+d16	ANDL A, @RW0, @RW0+d16	ANDL A, @RW0, @RW0+d16	ANDL A, @RW0, @RW0+d16	ANDL A, @RW0, @RW0+d16	ANDL A, @RW0, @RW0+d16	ORL A, @RW0, @RW0+d16	ORL A, @RW0, @RW0+d16	XORL A, @RW0, @RW0+d16	XORL A, @RW0, @RW0+d16	@RW0, @RW0+d16, #8, rel	@RW0, @RW0+d16, #8, rel
+9	ADDL A, @RW1, @RW1+d16	SUBL A, @RW1, @RW1+d16	SUBL A, @RW1, @RW1+d16	SUBL A, @RW1, @RW1+d16	CMPL A, @RW1, @RW1+d16	ANDL A, @RW1, @RW1+d16	ANDL A, @RW1, @RW1+d16	ANDL A, @RW1, @RW1+d16	ANDL A, @RW1, @RW1+d16	ANDL A, @RW1, @RW1+d16	ORL A, @RW1, @RW1+d16	ORL A, @RW1, @RW1+d16	XORL A, @RW1, @RW1+d16	XORL A, @RW1, @RW1+d16	@RW1, @RW1+d16, #8, rel	@RW1, @RW1+d16, #8, rel
+A	ADDL A, @RW2, @RW2+d16	SUBL A, @RW2, @RW2+d16	SUBL A, @RW2, @RW2+d16	SUBL A, @RW2, @RW2+d16	CMPL A, @RW2, @RW2+d16	ANDL A, @RW2, @RW2+d16	ANDL A, @RW2, @RW2+d16	ANDL A, @RW2, @RW2+d16	ANDL A, @RW2, @RW2+d16	ANDL A, @RW2, @RW2+d16	ORL A, @RW2, @RW2+d16	ORL A, @RW2, @RW2+d16	XORL A, @RW2, @RW2+d16	XORL A, @RW2, @RW2+d16	@RW2, @RW2+d16, #8, rel	@RW2, @RW2+d16, #8, rel
+B	ADDL A, @RW3, @RW3+d16	SUBL A, @RW3, @RW3+d16	SUBL A, @RW3, @RW3+d16	SUBL A, @RW3, @RW3+d16	CMPL A, @RW3, @RW3+d16	ANDL A, @RW3, @RW3+d16	ANDL A, @RW3, @RW3+d16	ANDL A, @RW3, @RW3+d16	ANDL A, @RW3, @RW3+d16	ANDL A, @RW3, @RW3+d16	ORL A, @RW3, @RW3+d16	ORL A, @RW3, @RW3+d16	XORL A, @RW3, @RW3+d16	XORL A, @RW3, @RW3+d16	@RW3, @RW3+d16, #8, rel	@RW3, @RW3+d16, #8, rel
+C	ADDL A, @RW0, @RW0+RW7	SUBL A, @RW0, @RW0+RW7	SUBL A, @RW0, @RW0+RW7	SUBL A, @RW0, @RW0+RW7	CMPL A, @RW0, @RW0+RW7	ANDL A, @RW0, @RW0+RW7	ANDL A, @RW0, @RW0+RW7	ANDL A, @RW0, @RW0+RW7	ANDL A, @RW0, @RW0+RW7	ANDL A, @RW0, @RW0+RW7	ORL A, @RW0, @RW0+RW7	ORL A, @RW0, @RW0+RW7	XORL A, @RW0, @RW0+RW7	XORL A, @RW0, @RW0+RW7	Use prohibited, @RW0+RW7, #8, rel	Use prohibited, @RW0+RW7, #8, rel
+D	ADDL A, @RW1, @RW1+RW7	SUBL A, @RW1, @RW1+RW7	SUBL A, @RW1, @RW1+RW7	SUBL A, @RW1, @RW1+RW7	CMPL A, @RW1, @RW1+RW7	ANDL A, @RW1, @RW1+RW7	ANDL A, @RW1, @RW1+RW7	ANDL A, @RW1, @RW1+RW7	ANDL A, @RW1, @RW1+RW7	ANDL A, @RW1, @RW1+RW7	ORL A, @RW1, @RW1+RW7	ORL A, @RW1, @RW1+RW7	XORL A, @RW1, @RW1+RW7	XORL A, @RW1, @RW1+RW7	Use prohibited, @RW1+RW7, #8, rel	Use prohibited, @RW1+RW7, #8, rel
+E	ADDL A, @RW2, @PC+d16	SUBL A, @RW2, @PC+d16	SUBL A, @RW2, @PC+d16	SUBL A, @RW2, @PC+d16	CMPL A, @RW2, @PC+d16	ANDL A, @RW2, @PC+d16	ANDL A, @RW2, @PC+d16	ANDL A, @RW2, @PC+d16	ANDL A, @RW2, @PC+d16	ANDL A, @RW2, @PC+d16	ORL A, @RW2, @PC+d16	ORL A, @RW2, @PC+d16	XORL A, @RW2, @PC+d16	XORL A, @RW2, @PC+d16	Use prohibited, @PC+d16, #8, rel	Use prohibited, @PC+d16, #8, rel
+F	ADDL A, @RW3, addr16	SUBL A, @RW3, addr16	SUBL A, @RW3, addr16	SUBL A, @RW3, addr16	CMPL A, @RW3, addr16	ANDL A, @RW3, addr16	ANDL A, @RW3, addr16	ANDL A, @RW3, addr16	ANDL A, @RW3, addr16	ANDL A, @RW3, addr16	ORL A, @RW3, addr16	ORL A, @RW3, addr16	XORL A, @RW3, addr16	XORL A, @RW3, addr16	Use prohibited, addr16, #8, rel	Use prohibited, addr16, #8, rel

Table B.9-7 ea Instruction 2 (First Byte = 71<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMPP @RL0, @RW0-d8	JMPP @RL0, @RW0-d8	CALLP @RL0, @RW0-d8	CALLP @RL0, @RW0-d8	INCL RL0, @RW0-d8	INCL RL0, @RW0-d8	DECL RL0, @RW0-d8	DECL RL0, @RW0-d8	MOVL A, RL0, @RW0-d8	MOVL A, RL0, @RW0-d8	MOVL RL0, A, @RW0-d8, A	MOVL RL0, A, @RW0-d8, A	MOV R0, #8, @RW0-d8, #8	MOV R0, #8, @RW0-d8, #8	MOVEA A, RW0, @RW0-d8	MOVEA A, RW0, @RW0-d8
+1	JMPP @RL0, @RW1-d8	JMPP @RL0, @RW1-d8	CALLP @RL0, @RW1-d8	CALLP @RL0, @RW1-d8	INCL RL0, @RW1-d8	INCL RL0, @RW1-d8	DECL RL0, @RW1-d8	DECL RL0, @RW1-d8	MOVL A, RL0, @RW1-d8	MOVL A, RL0, @RW1-d8	MOVL RL0, A, @RW1-d8, A	MOVL RL0, A, @RW1-d8, A	MOV R1, #8, @RW1-d8, #8	MOV R1, #8, @RW1-d8, #8	MOVEA A, RW1, @RW1-d8	MOVEA A, RW1, @RW1-d8
+2	JMPP @RL1, @RW2-d8	JMPP @RL1, @RW2-d8	CALLP @RL1, @RW2-d8	CALLP @RL1, @RW2-d8	INCL RL1, @RW2-d8	INCL RL1, @RW2-d8	DECL RL1, @RW2-d8	DECL RL1, @RW2-d8	MOVL A, RL1, @RW2-d8	MOVL A, RL1, @RW2-d8	MOVL RL1, A, @RW2-d8, A	MOVL RL1, A, @RW2-d8, A	MOV R2, #8, @RW2-d8, #8	MOV R2, #8, @RW2-d8, #8	MOVEA A, RW2, @RW2-d8	MOVEA A, RW2, @RW2-d8
+3	JMPP @RL1, @RW3-d8	JMPP @RL1, @RW3-d8	CALLP @RL1, @RW3-d8	CALLP @RL1, @RW3-d8	INCL RL1, @RW3-d8	INCL RL1, @RW3-d8	DECL RL1, @RW3-d8	DECL RL1, @RW3-d8	MOVL A, RL1, @RW3-d8	MOVL A, RL1, @RW3-d8	MOVL RL1, A, @RW3-d8, A	MOVL RL1, A, @RW3-d8, A	MOV R3, #8, @RW3-d8, #8	MOV R3, #8, @RW3-d8, #8	MOVEA A, RW3, @RW3-d8	MOVEA A, RW3, @RW3-d8
+4	JMPP @RL2, @RW4-d8	JMPP @RL2, @RW4-d8	CALLP @RL2, @RW4-d8	CALLP @RL2, @RW4-d8	INCL RL2, @RW4-d8	INCL RL2, @RW4-d8	DECL RL2, @RW4-d8	DECL RL2, @RW4-d8	MOVL A, RL2, @RW4-d8	MOVL A, RL2, @RW4-d8	MOVL RL2, A, @RW4-d8, A	MOVL RL2, A, @RW4-d8, A	MOV R4, #8, @RW4-d8, #8	MOV R4, #8, @RW4-d8, #8	MOVEA A, RW4, @RW4-d8	MOVEA A, RW4, @RW4-d8
+5	JMPP @RL2, @RW5-d8	JMPP @RL2, @RW5-d8	CALLP @RL2, @RW5-d8	CALLP @RL2, @RW5-d8	INCL RL2, @RW5-d8	INCL RL2, @RW5-d8	DECL RL2, @RW5-d8	DECL RL2, @RW5-d8	MOVL A, RL2, @RW5-d8	MOVL A, RL2, @RW5-d8	MOVL RL2, A, @RW5-d8, A	MOVL RL2, A, @RW5-d8, A	MOV R5, #8, @RW5-d8, #8	MOV R5, #8, @RW5-d8, #8	MOVEA A, RW5, @RW5-d8	MOVEA A, RW5, @RW5-d8
+6	JMPP @RL3, @RW6-d8	JMPP @RL3, @RW6-d8	CALLP @RL3, @RW6-d8	CALLP @RL3, @RW6-d8	INCL RL3, @RW6-d8	INCL RL3, @RW6-d8	DECL RL3, @RW6-d8	DECL RL3, @RW6-d8	MOVL A, RL3, @RW6-d8	MOVL A, RL3, @RW6-d8	MOVL RL3, A, @RW6-d8, A	MOVL RL3, A, @RW6-d8, A	MOV R6, #8, @RW6-d8, #8	MOV R6, #8, @RW6-d8, #8	MOVEA A, RW6, @RW6-d8	MOVEA A, RW6, @RW6-d8
+7	JMPP @RL3, @RW7-d8	JMPP @RL3, @RW7-d8	CALLP @RL3, @RW7-d8	CALLP @RL3, @RW7-d8	INCL RL3, @RW7-d8	INCL RL3, @RW7-d8	DECL RL3, @RW7-d8	DECL RL3, @RW7-d8	MOVL A, RL3, @RW7-d8	MOVL A, RL3, @RW7-d8	MOVL RL3, A, @RW7-d8, A	MOVL RL3, A, @RW7-d8, A	MOV R7, #8, @RW7-d8, #8	MOV R7, #8, @RW7-d8, #8	MOVEA A, RW7, @RW7-d8	MOVEA A, RW7, @RW7-d8
+8	JMPP @RW0, @RW0-d16	JMPP @RW0, @RW0-d16	CALLP @RW0, @RW0-d16	CALLP @RW0, @RW0-d16	INCL @RW0, @RW0-d16	INCL @RW0, @RW0-d16	DECL @RW0, @RW0-d16	DECL @RW0, @RW0-d16	MOVL A, @RW0, @RW0-d16	MOVL A, @RW0, @RW0-d16	MOVL @RW0, A, @RW0-d16, A	MOVL @RW0, A, @RW0-d16, A	MOV @RW0, #8, @RW0-d16, #8	MOV @RW0, #8, @RW0-d16, #8	MOVEA A, @RW0, @RW0-d16	MOVEA A, @RW0, @RW0-d16
+9	JMPP @RW1, @RW1-d16	JMPP @RW1, @RW1-d16	CALLP @RW1, @RW1-d16	CALLP @RW1, @RW1-d16	INCL @RW1, @RW1-d16	INCL @RW1, @RW1-d16	DECL @RW1, @RW1-d16	DECL @RW1, @RW1-d16	MOVL A, @RW1, @RW1-d16	MOVL A, @RW1, @RW1-d16	MOVL @RW1, A, @RW1-d16, A	MOVL @RW1, A, @RW1-d16, A	MOV @RW1, #8, @RW1-d16, #8	MOV @RW1, #8, @RW1-d16, #8	MOVEA A, @RW1, @RW1-d16	MOVEA A, @RW1, @RW1-d16
+A	JMPP @RW2, @RW2-d16	JMPP @RW2, @RW2-d16	CALLP @RW2, @RW2-d16	CALLP @RW2, @RW2-d16	INCL @RW2, @RW2-d16	INCL @RW2, @RW2-d16	DECL @RW2, @RW2-d16	DECL @RW2, @RW2-d16	MOVL A, @RW2, @RW2-d16	MOVL A, @RW2, @RW2-d16	MOVL @RW2, A, @RW2-d16, A	MOVL @RW2, A, @RW2-d16, A	MOV @RW2, #8, @RW2-d16, #8	MOV @RW2, #8, @RW2-d16, #8	MOVEA A, @RW2, @RW2-d16	MOVEA A, @RW2, @RW2-d16
+B	JMPP @RW3, @RW3-d16	JMPP @RW3, @RW3-d16	CALLP @RW3, @RW3-d16	CALLP @RW3, @RW3-d16	INCL @RW3, @RW3-d16	INCL @RW3, @RW3-d16	DECL @RW3, @RW3-d16	DECL @RW3, @RW3-d16	MOVL A, @RW3, @RW3-d16	MOVL A, @RW3, @RW3-d16	MOVL @RW3, A, @RW3-d16, A	MOVL @RW3, A, @RW3-d16, A	MOV @RW3, #8, @RW3-d16, #8	MOV @RW3, #8, @RW3-d16, #8	MOVEA A, @RW3, @RW3-d16	MOVEA A, @RW3, @RW3-d16
+C	JMPP @RW0+, @RW0-RW7	JMPP @RW0+, @RW0-RW7	CALLP @RW0+, @RW0-RW7	CALLP @RW0+, @RW0-RW7	INCL @RW0+, @RW0-RW7	INCL @RW0+, @RW0-RW7	DECL @RW0+, @RW0-RW7	DECL @RW0+, @RW0-RW7	MOVL A, @RW0+, @RW0-RW7	MOVL A, @RW0+, @RW0-RW7	MOVL @RW0+, A, @RW0-RW7, A	MOVL @RW0+, A, @RW0-RW7, A	MOV @RW0+, #8, @RW0-RW7, #8	MOV @RW0+, #8, @RW0-RW7, #8	MOVEA A, @RW0+, @RW0-RW7	MOVEA A, @RW0+, @RW0-RW7
+D	JMPP @RW1+, @RW1-RW7	JMPP @RW1+, @RW1-RW7	CALLP @RW1+, @RW1-RW7	CALLP @RW1+, @RW1-RW7	INCL @RW1+, @RW1-RW7	INCL @RW1+, @RW1-RW7	DECL @RW1+, @RW1-RW7	DECL @RW1+, @RW1-RW7	MOVL A, @RW1+, @RW1-RW7	MOVL A, @RW1+, @RW1-RW7	MOVL @RW1+, A, @RW1-RW7, A	MOVL @RW1+, A, @RW1-RW7, A	MOV @RW1+, #8, @RW1-RW7, #8	MOV @RW1+, #8, @RW1-RW7, #8	MOVEA A, @RW1+, @RW1-RW7	MOVEA A, @RW1+, @RW1-RW7
+E	JMPP @RW2+, @PC-d16	JMPP @RW2+, @PC-d16	CALLP @RW2+, @PC-d16	CALLP @RW2+, @PC-d16	INCL @RW2+, @PC-d16	INCL @RW2+, @PC-d16	DECL @RW2+, @PC-d16	DECL @RW2+, @PC-d16	MOVL A, @RW2+, @PC-d16	MOVL A, @RW2+, @PC-d16	MOVL @RW2+, A, @PC-d16, A	MOVL @RW2+, A, @PC-d16, A	MOV @RW2+, #8, @PC-d16, #8	MOV @RW2+, #8, @PC-d16, #8	MOVEA A, @RW2+, @PC-d16	MOVEA A, @RW2+, @PC-d16
+F	JMPP @RW3+, @addr16	JMPP @RW3+, @addr16	CALLP @RW3+, @addr16	CALLP @RW3+, @addr16	INCL @RW3+, @addr16	INCL @RW3+, @addr16	DECL @RW3+, @addr16	DECL @RW3+, @addr16	MOVL A, @RW3+, @addr16	MOVL A, @RW3+, @addr16	MOVL @RW3+, A, @addr16, A	MOVL @RW3+, A, @addr16, A	MOV @RW3+, #8, @addr16, #8	MOV @RW3+, #8, @addr16, #8	MOVEA A, @RW3+, @addr16	MOVEA A, @RW3+, @addr16

Table B.9-8 ea Instruction 3 (First Byte = 72<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R0' @RW0+d8	R0' @RW0+d8	R0' @RW0+d8	R0' @RW0+d8	R0' @RW0+d8	R0' @RW0+d8	R0' @RW0+d8	R0' @RW0+d8	A, R0' @RW0+d8	A, R0' @RW0+d8	R0, A' @RW0+d8	MOV	A, R0' @RW0+d8	MOVX	A, R0' @RW0+d8	A, R0' @RW0+d8
+1	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R1' @RW1+d8	R1' @RW1+d8	R1' @RW1+d8	R1' @RW1+d8	R1' @RW1+d8	R1' @RW1+d8	R1' @RW1+d8	R1' @RW1+d8	A, R1' @RW1+d8	A, R1' @RW1+d8	R1, A' @RW1+d8	MOV	A, R1' @RW1+d8	MOVX	A, R1' @RW1+d8	A, R1' @RW1+d8
+2	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R2' @RW2+d8	R2' @RW2+d8	R2' @RW2+d8	R2' @RW2+d8	R2' @RW2+d8	R2' @RW2+d8	R2' @RW2+d8	R2' @RW2+d8	A, R2' @RW2+d8	A, R2' @RW2+d8	R2, A' @RW2+d8	MOV	A, R2' @RW2+d8	MOVX	A, R2' @RW2+d8	A, R2' @RW2+d8
+3	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R3' @RW3+d8	R3' @RW3+d8	R3' @RW3+d8	R3' @RW3+d8	R3' @RW3+d8	R3' @RW3+d8	R3' @RW3+d8	R3' @RW3+d8	A, R3' @RW3+d8	A, R3' @RW3+d8	R3, A' @RW3+d8	MOV	A, R3' @RW3+d8	MOVX	A, R3' @RW3+d8	A, R3' @RW3+d8
+4	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R4' @RW4+d8	R4' @RW4+d8	R4' @RW4+d8	R4' @RW4+d8	R4' @RW4+d8	R4' @RW4+d8	R4' @RW4+d8	R4' @RW4+d8	A, R4' @RW4+d8	A, R4' @RW4+d8	R4, A' @RW4+d8	MOV	A, R4' @RW4+d8	MOVX	A, R4' @RW4+d8	A, R4' @RW4+d8
+5	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R5' @RW5+d8	R5' @RW5+d8	R5' @RW5+d8	R5' @RW5+d8	R5' @RW5+d8	R5' @RW5+d8	R5' @RW5+d8	R5' @RW5+d8	A, R5' @RW5+d8	A, R5' @RW5+d8	R5, A' @RW5+d8	MOV	A, R5' @RW5+d8	MOVX	A, R5' @RW5+d8	A, R5' @RW5+d8
+6	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R6' @RW6+d8	R6' @RW6+d8	R6' @RW6+d8	R6' @RW6+d8	R6' @RW6+d8	R6' @RW6+d8	R6' @RW6+d8	R6' @RW6+d8	A, R6' @RW6+d8	A, R6' @RW6+d8	R6, A' @RW6+d8	MOV	A, R6' @RW6+d8	MOVX	A, R6' @RW6+d8	A, R6' @RW6+d8
+7	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	R7' @RW7+d8	R7' @RW7+d8	R7' @RW7+d8	R7' @RW7+d8	R7' @RW7+d8	R7' @RW7+d8	R7' @RW7+d8	R7' @RW7+d8	A, R7' @RW7+d8	A, R7' @RW7+d8	R7, A' @RW7+d8	MOV	A, R7' @RW7+d8	MOVX	A, R7' @RW7+d8	A, R7' @RW7+d8
+8	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW0' @RW0+d16	@RW0' @RW0+d16	@RW0' @RW0+d16	@RW0' @RW0+d16	@RW0' @RW0+d16	@RW0' @RW0+d16	@RW0' @RW0+d16	@RW0' @RW0+d16	A, @RW0' @RW0+d16	A, @RW0' @RW0+d16	@RW0, A' @RW0+d16	MOV	A, @RW0' @RW0+d16	MOVX	A, @RW0' @RW0+d16	A, @RW0' @RW0+d16
+9	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW1' @RW1+d16	@RW1' @RW1+d16	@RW1' @RW1+d16	@RW1' @RW1+d16	@RW1' @RW1+d16	@RW1' @RW1+d16	@RW1' @RW1+d16	@RW1' @RW1+d16	A, @RW1' @RW1+d16	A, @RW1' @RW1+d16	@RW1, A' @RW1+d16	MOV	A, @RW1' @RW1+d16	MOVX	A, @RW1' @RW1+d16	A, @RW1' @RW1+d16
+A	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW2' @RW2+d16	@RW2' @RW2+d16	@RW2' @RW2+d16	@RW2' @RW2+d16	@RW2' @RW2+d16	@RW2' @RW2+d16	@RW2' @RW2+d16	@RW2' @RW2+d16	A, @RW2' @RW2+d16	A, @RW2' @RW2+d16	@RW2, A' @RW2+d16	MOV	A, @RW2' @RW2+d16	MOVX	A, @RW2' @RW2+d16	A, @RW2' @RW2+d16
+B	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW3' @RW3+d16	@RW3' @RW3+d16	@RW3' @RW3+d16	@RW3' @RW3+d16	@RW3' @RW3+d16	@RW3' @RW3+d16	@RW3' @RW3+d16	@RW3' @RW3+d16	A, @RW3' @RW3+d16	A, @RW3' @RW3+d16	@RW3, A' @RW3+d16	MOV	A, @RW3' @RW3+d16	MOVX	A, @RW3' @RW3+d16	A, @RW3' @RW3+d16
+C	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	@RW0+ @RW0+RW7	A, @RW0+ @RW0+RW7	A, @RW0+ @RW0+RW7	@RW0+, A' @RW0+RW7	MOV	A, @RW0+ @RW0+RW7	MOVX	A, @RW0+ @RW0+RW7	A, @RW0+ @RW0+RW7
+D	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	@RW1+ @RW1+RW7	A, @RW1+ @RW1+RW7	A, @RW1+ @RW1+RW7	@RW1+, A' @RW1+RW7	MOV	A, @RW1+ @RW1+RW7	MOVX	A, @RW1+ @RW1+RW7	A, @RW1+ @RW1+RW7
+E	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW2+ @PC+d16	@RW2+ @PC+d16	@RW2+ @PC+d16	@RW2+ @PC+d16	@RW2+ @PC+d16	@RW2+ @PC+d16	@RW2+ @PC+d16	@RW2+ @PC+d16	A, @RW2+ @PC+d16	A, @RW2+ @PC+d16	@RW2+, A' @PC+d16	MOV	A, @RW2+ @PC+d16	MOVX	A, @RW2+ @PC+d16	A, @RW2+ @PC+d16
+F	ROL	ROL	ROR	ROR	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
	@RW3+ addr16	@RW3+ addr16	@RW3+ addr16	@RW3+ addr16	@RW3+ addr16	@RW3+ addr16	@RW3+ addr16	@RW3+ addr16	A, @RW3+ addr16	A, @RW3+ addr16	@RW3+, A' addr16	MOV	A, @RW3+ addr16	MOVX	A, @RW3+ addr16	A, @RW3+ addr16

Table B.9-9 ea Instruction 4 (First Byte = 73<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMP @RW0, @RW0-d8	CALL RW0, @RW0-d8	CALL RW0, @RW0-d8	CALL RW0, @RW0-d8	INCW RW0, @RW0-d8	INCW RW0, @RW0-d8	DECW RW0, @RW0-d8	DECW RW0, @RW0-d8	MOVW A, RW0, @RW0-d8	MOVW A, RW0, @RW0-d8	MOVW RW0, A, @RW0-d8	MOVW A, @RW0-d8	MOVW RW0, #16, @RW0-d8, #16	MOVW A, @RW0-d8, #16	XCHW A, RW0, @RW0-d8	XCHW A, RW0, @RW0-d8
+1	JMP @RW1, @RW1-d8	CALL RW1, @RW1-d8	CALL RW1, @RW1-d8	CALL RW1, @RW1-d8	INCW RW1, @RW1-d8	INCW RW1, @RW1-d8	DECW RW1, @RW1-d8	DECW RW1, @RW1-d8	MOVW A, RW1, @RW1-d8	MOVW A, RW1, @RW1-d8	MOVW RW1, A, @RW1-d8	MOVW A, @RW1-d8	MOVW RW1, #16, @RW1-d8, #16	MOVW A, @RW1-d8, #16	XCHW A, RW1, @RW1-d8	XCHW A, RW1, @RW1-d8
+2	JMP @RW2, @RW2-d8	CALL RW2, @RW2-d8	CALL RW2, @RW2-d8	CALL RW2, @RW2-d8	INCW RW2, @RW2-d8	INCW RW2, @RW2-d8	DECW RW2, @RW2-d8	DECW RW2, @RW2-d8	MOVW A, RW2, @RW2-d8	MOVW A, RW2, @RW2-d8	MOVW RW2, A, @RW2-d8	MOVW A, @RW2-d8	MOVW RW2, #16, @RW2-d8, #16	MOVW A, @RW2-d8, #16	XCHW A, RW2, @RW2-d8	XCHW A, RW2, @RW2-d8
+3	JMP @RW3, @RW3-d8	CALL RW3, @RW3-d8	CALL RW3, @RW3-d8	CALL RW3, @RW3-d8	INCW RW3, @RW3-d8	INCW RW3, @RW3-d8	DECW RW3, @RW3-d8	DECW RW3, @RW3-d8	MOVW A, RW3, @RW3-d8	MOVW A, RW3, @RW3-d8	MOVW RW3, A, @RW3-d8	MOVW A, @RW3-d8	MOVW RW3, #16, @RW3-d8, #16	MOVW A, @RW3-d8, #16	XCHW A, RW3, @RW3-d8	XCHW A, RW3, @RW3-d8
+4	JMP @RW4, @RW4-d8	CALL RW4, @RW4-d8	CALL RW4, @RW4-d8	CALL RW4, @RW4-d8	INCW RW4, @RW4-d8	INCW RW4, @RW4-d8	DECW RW4, @RW4-d8	DECW RW4, @RW4-d8	MOVW A, RW4, @RW4-d8	MOVW A, RW4, @RW4-d8	MOVW RW4, A, @RW4-d8	MOVW A, @RW4-d8	MOVW RW4, #16, @RW4-d8, #16	MOVW A, @RW4-d8, #16	XCHW A, RW4, @RW4-d8	XCHW A, RW4, @RW4-d8
+5	JMP @RW5, @RW5-d8	CALL RW5, @RW5-d8	CALL RW5, @RW5-d8	CALL RW5, @RW5-d8	INCW RW5, @RW5-d8	INCW RW5, @RW5-d8	DECW RW5, @RW5-d8	DECW RW5, @RW5-d8	MOVW A, RW5, @RW5-d8	MOVW A, RW5, @RW5-d8	MOVW RW5, A, @RW5-d8	MOVW A, @RW5-d8	MOVW RW5, #16, @RW5-d8, #16	MOVW A, @RW5-d8, #16	XCHW A, RW5, @RW5-d8	XCHW A, RW5, @RW5-d8
+6	JMP @RW6, @RW6-d8	CALL RW6, @RW6-d8	CALL RW6, @RW6-d8	CALL RW6, @RW6-d8	INCW RW6, @RW6-d8	INCW RW6, @RW6-d8	DECW RW6, @RW6-d8	DECW RW6, @RW6-d8	MOVW A, RW6, @RW6-d8	MOVW A, RW6, @RW6-d8	MOVW RW6, A, @RW6-d8	MOVW A, @RW6-d8	MOVW RW6, #16, @RW6-d8, #16	MOVW A, @RW6-d8, #16	XCHW A, RW6, @RW6-d8	XCHW A, RW6, @RW6-d8
+7	JMP @RW7, @RW7-d8	CALL RW7, @RW7-d8	CALL RW7, @RW7-d8	CALL RW7, @RW7-d8	INCW RW7, @RW7-d8	INCW RW7, @RW7-d8	DECW RW7, @RW7-d8	DECW RW7, @RW7-d8	MOVW A, RW7, @RW7-d8	MOVW A, RW7, @RW7-d8	MOVW RW7, A, @RW7-d8	MOVW A, @RW7-d8	MOVW RW7, #16, @RW7-d8, #16	MOVW A, @RW7-d8, #16	XCHW A, RW7, @RW7-d8	XCHW A, RW7, @RW7-d8
+8	JMP @RW0, @RW0-d16	CALL @RW0, @RW0-d16	CALL @RW0, @RW0-d16	CALL @RW0, @RW0-d16	INCW @RW0, @RW0-d16	INCW @RW0, @RW0-d16	DECW @RW0, @RW0-d16	DECW @RW0, @RW0-d16	MOVW A, @RW0, @RW0-d16	MOVW A, @RW0, @RW0-d16	MOVW @RW0, A, @RW0-d16	MOVW A, @RW0-d16	MOVW @RW0, #16, @RW0-d16, #16	MOVW A, @RW0-d16, #16	XCHW A, @RW0, @RW0-d16	XCHW A, @RW0, @RW0-d16
+9	JMP @RW1, @RW1-d16	CALL @RW1, @RW1-d16	CALL @RW1, @RW1-d16	CALL @RW1, @RW1-d16	INCW @RW1, @RW1-d16	INCW @RW1, @RW1-d16	DECW @RW1, @RW1-d16	DECW @RW1, @RW1-d16	MOVW A, @RW1, @RW1-d16	MOVW A, @RW1, @RW1-d16	MOVW @RW1, A, @RW1-d16	MOVW A, @RW1-d16	MOVW @RW1, #16, @RW1-d16, #16	MOVW A, @RW1-d16, #16	XCHW A, @RW1, @RW1-d16	XCHW A, @RW1, @RW1-d16
+A	JMP @RW2, @RW2-d16	CALL @RW2, @RW2-d16	CALL @RW2, @RW2-d16	CALL @RW2, @RW2-d16	INCW @RW2, @RW2-d16	INCW @RW2, @RW2-d16	DECW @RW2, @RW2-d16	DECW @RW2, @RW2-d16	MOVW A, @RW2, @RW2-d16	MOVW A, @RW2, @RW2-d16	MOVW @RW2, A, @RW2-d16	MOVW A, @RW2-d16	MOVW @RW2, #16, @RW2-d16, #16	MOVW A, @RW2-d16, #16	XCHW A, @RW2, @RW2-d16	XCHW A, @RW2, @RW2-d16
+B	JMP @RW3, @RW3-d16	CALL @RW3, @RW3-d16	CALL @RW3, @RW3-d16	CALL @RW3, @RW3-d16	INCW @RW3, @RW3-d16	INCW @RW3, @RW3-d16	DECW @RW3, @RW3-d16	DECW @RW3, @RW3-d16	MOVW A, @RW3, @RW3-d16	MOVW A, @RW3, @RW3-d16	MOVW @RW3, A, @RW3-d16	MOVW A, @RW3-d16	MOVW @RW3, #16, @RW3-d16, #16	MOVW A, @RW3-d16, #16	XCHW A, @RW3, @RW3-d16	XCHW A, @RW3, @RW3-d16
+C	JMP @RW0+, @RW0-RW7	CALL @RW0+, @RW0-RW7	CALL @RW0+, @RW0-RW7	CALL @RW0+, @RW0-RW7	INCW @RW0+, @RW0-RW7	INCW @RW0+, @RW0-RW7	DECW @RW0+, @RW0-RW7	DECW @RW0+, @RW0-RW7	MOVW A, @RW0+, @RW0-RW7	MOVW A, @RW0+, @RW0-RW7	MOVW @RW0+, A, @RW0-RW7	MOVW A, @RW0-RW7	MOVW @RW0+, #16, @RW0-RW7, #16	MOVW A, @RW0-RW7, #16	XCHW A, @RW0+, @RW0-RW7	XCHW A, @RW0+, @RW0-RW7
+D	JMP @RW1+, @RW1-RW7	CALL @RW1+, @RW1-RW7	CALL @RW1+, @RW1-RW7	CALL @RW1+, @RW1-RW7	INCW @RW1+, @RW1-RW7	INCW @RW1+, @RW1-RW7	DECW @RW1+, @RW1-RW7	DECW @RW1+, @RW1-RW7	MOVW A, @RW1+, @RW1-RW7	MOVW A, @RW1+, @RW1-RW7	MOVW @RW1+, A, @RW1-RW7	MOVW A, @RW1-RW7	MOVW @RW1+, #16, @RW1-RW7, #16	MOVW A, @RW1-RW7, #16	XCHW A, @RW1+, @RW1-RW7	XCHW A, @RW1+, @RW1-RW7
+E	JMP @RW2+, @RW2-PC+d16	CALL @RW2+, @RW2-PC+d16	CALL @RW2+, @RW2-PC+d16	CALL @RW2+, @RW2-PC+d16	INCW @RW2+, @RW2-PC+d16	INCW @RW2+, @RW2-PC+d16	DECW @RW2+, @RW2-PC+d16	DECW @RW2+, @RW2-PC+d16	MOVW A, @RW2+, @RW2-PC+d16	MOVW A, @RW2+, @RW2-PC+d16	MOVW @RW2+, A, @RW2-PC+d16	MOVW A, @RW2-PC+d16	MOVW @RW2+, #16, @RW2-PC+d16, #16	MOVW A, @RW2-PC+d16, #16	XCHW A, @RW2+, @RW2-PC+d16	XCHW A, @RW2+, @RW2-PC+d16
+F	JMP @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	INCW @RW3+, @RW3+addr16	INCW @RW3+, @RW3+addr16	DECW @RW3+, @RW3+addr16	DECW @RW3+, @RW3+addr16	MOVW A, @RW3+, @RW3+addr16	MOVW A, @RW3+, @RW3+addr16	MOVW @RW3+, A, @RW3+addr16	MOVW A, @RW3+addr16	MOVW @RW3+, #16, @RW3+addr16, #16	MOVW A, @RW3+addr16, #16	XCHW A, @RW3+, @RW3+addr16	XCHW A, @RW3+, @RW3+addr16

Table B.9-10 ea Instruction 5 (First Byte = 74<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0', @RW0+d8	ADD A, R0', @RW0+d8	SUB A, R0', @RW0+d8	SUB A, R0', @RW0+d8	ADDC A, R0', @RW0+d8	ADDC A, R0', @RW0+d8	CMP A, R0', @RW0+d8	CMP A, R0', @RW0+d8	AND A, R0', @RW0+d8	AND A, R0', @RW0+d8	OR A, R0', @RW0+d8	OR A, R0', @RW0+d8	XOR A, R0', @RW0+d8	XOR A, R0', @RW0+d8	DBNZ R0, r', RW0+d8, r	DBNZ R0, r', RW0+d8, r
+1	ADD A, R1', @RW1+d8	ADD A, R1', @RW1+d8	SUB A, R1', @RW1+d8	SUB A, R1', @RW1+d8	ADDC A, R1', @RW1+d8	ADDC A, R1', @RW1+d8	CMP A, R1', @RW1+d8	CMP A, R1', @RW1+d8	AND A, R1', @RW1+d8	AND A, R1', @RW1+d8	OR A, R1', @RW1+d8	OR A, R1', @RW1+d8	XOR A, R1', @RW1+d8	XOR A, R1', @RW1+d8	DBNZ R1, r', RW1+d8, r	DBNZ R1, r', RW1+d8, r
+2	ADD A, R2', @RW2+d8	ADD A, R2', @RW2+d8	SUB A, R2', @RW2+d8	SUB A, R2', @RW2+d8	ADDC A, R2', @RW2+d8	ADDC A, R2', @RW2+d8	CMP A, R2', @RW2+d8	CMP A, R2', @RW2+d8	AND A, R2', @RW2+d8	AND A, R2', @RW2+d8	OR A, R2', @RW2+d8	OR A, R2', @RW2+d8	XOR A, R2', @RW2+d8	XOR A, R2', @RW2+d8	DBNZ R2, r', RW2+d8, r	DBNZ R2, r', RW2+d8, r
+3	ADD A, R3', @RW3+d8	ADD A, R3', @RW3+d8	SUB A, R3', @RW3+d8	SUB A, R3', @RW3+d8	ADDC A, R3', @RW3+d8	ADDC A, R3', @RW3+d8	CMP A, R3', @RW3+d8	CMP A, R3', @RW3+d8	AND A, R3', @RW3+d8	AND A, R3', @RW3+d8	OR A, R3', @RW3+d8	OR A, R3', @RW3+d8	XOR A, R3', @RW3+d8	XOR A, R3', @RW3+d8	DBNZ R3, r', RW3+d8, r	DBNZ R3, r', RW3+d8, r
+4	ADD A, R4', @RW4+d8	ADD A, R4', @RW4+d8	SUB A, R4', @RW4+d8	SUB A, R4', @RW4+d8	ADDC A, R4', @RW4+d8	ADDC A, R4', @RW4+d8	CMP A, R4', @RW4+d8	CMP A, R4', @RW4+d8	AND A, R4', @RW4+d8	AND A, R4', @RW4+d8	OR A, R4', @RW4+d8	OR A, R4', @RW4+d8	XOR A, R4', @RW4+d8	XOR A, R4', @RW4+d8	DBNZ R4, r', RW4+d8, r	DBNZ R4, r', RW4+d8, r
+5	ADD A, R5', @RW5+d8	ADD A, R5', @RW5+d8	SUB A, R5', @RW5+d8	SUB A, R5', @RW5+d8	ADDC A, R5', @RW5+d8	ADDC A, R5', @RW5+d8	CMP A, R5', @RW5+d8	CMP A, R5', @RW5+d8	AND A, R5', @RW5+d8	AND A, R5', @RW5+d8	OR A, R5', @RW5+d8	OR A, R5', @RW5+d8	XOR A, R5', @RW5+d8	XOR A, R5', @RW5+d8	DBNZ R5, r', RW5+d8, r	DBNZ R5, r', RW5+d8, r
+6	ADD A, R6', @RW6+d8	ADD A, R6', @RW6+d8	SUB A, R6', @RW6+d8	SUB A, R6', @RW6+d8	ADDC A, R6', @RW6+d8	ADDC A, R6', @RW6+d8	CMP A, R6', @RW6+d8	CMP A, R6', @RW6+d8	AND A, R6', @RW6+d8	AND A, R6', @RW6+d8	OR A, R6', @RW6+d8	OR A, R6', @RW6+d8	XOR A, R6', @RW6+d8	XOR A, R6', @RW6+d8	DBNZ R6, r', RW6+d8, r	DBNZ R6, r', RW6+d8, r
+7	ADD A, R7', @RW7+d8	ADD A, R7', @RW7+d8	SUB A, R7', @RW7+d8	SUB A, R7', @RW7+d8	ADDC A, R7', @RW7+d8	ADDC A, R7', @RW7+d8	CMP A, R7', @RW7+d8	CMP A, R7', @RW7+d8	AND A, R7', @RW7+d8	AND A, R7', @RW7+d8	OR A, R7', @RW7+d8	OR A, R7', @RW7+d8	XOR A, R7', @RW7+d8	XOR A, R7', @RW7+d8	DBNZ R7, r', RW7+d8, r	DBNZ R7, r', RW7+d8, r
+8	ADD A, @RW0', @RW0+d16	ADD A, @RW0', @RW0+d16	SUB A, @RW0', @RW0+d16	SUB A, @RW0', @RW0+d16	ADDC A, @RW0', @RW0+d16	ADDC A, @RW0', @RW0+d16	CMP A, @RW0', @RW0+d16	CMP A, @RW0', @RW0+d16	AND A, @RW0', @RW0+d16	AND A, @RW0', @RW0+d16	OR A, @RW0', @RW0+d16	OR A, @RW0', @RW0+d16	XOR A, @RW0', @RW0+d16	XOR A, @RW0', @RW0+d16	DBNZ @RW0, r', W0+d16, r	DBNZ @RW0, r', W0+d16, r
+9	ADD A, @RW1', @RW1+d16	ADD A, @RW1', @RW1+d16	SUB A, @RW1', @RW1+d16	SUB A, @RW1', @RW1+d16	ADDC A, @RW1', @RW1+d16	ADDC A, @RW1', @RW1+d16	CMP A, @RW1', @RW1+d16	CMP A, @RW1', @RW1+d16	AND A, @RW1', @RW1+d16	AND A, @RW1', @RW1+d16	OR A, @RW1', @RW1+d16	OR A, @RW1', @RW1+d16	XOR A, @RW1', @RW1+d16	XOR A, @RW1', @RW1+d16	DBNZ @RW1, r', W1+d16, r	DBNZ @RW1, r', W1+d16, r
+A	ADD A, @RW2', @RW2+d16	ADD A, @RW2', @RW2+d16	SUB A, @RW2', @RW2+d16	SUB A, @RW2', @RW2+d16	ADDC A, @RW2', @RW2+d16	ADDC A, @RW2', @RW2+d16	CMP A, @RW2', @RW2+d16	CMP A, @RW2', @RW2+d16	AND A, @RW2', @RW2+d16	AND A, @RW2', @RW2+d16	OR A, @RW2', @RW2+d16	OR A, @RW2', @RW2+d16	XOR A, @RW2', @RW2+d16	XOR A, @RW2', @RW2+d16	DBNZ @RW2, r', W2+d16, r	DBNZ @RW2, r', W2+d16, r
+B	ADD A, @RW3', @RW3+d16	ADD A, @RW3', @RW3+d16	SUB A, @RW3', @RW3+d16	SUB A, @RW3', @RW3+d16	ADDC A, @RW3', @RW3+d16	ADDC A, @RW3', @RW3+d16	CMP A, @RW3', @RW3+d16	CMP A, @RW3', @RW3+d16	AND A, @RW3', @RW3+d16	AND A, @RW3', @RW3+d16	OR A, @RW3', @RW3+d16	OR A, @RW3', @RW3+d16	XOR A, @RW3', @RW3+d16	XOR A, @RW3', @RW3+d16	DBNZ @RW3, r', W3+d16, r	DBNZ @RW3, r', W3+d16, r
+C	ADD A, @RW0+', @RW0+RW7	ADD A, @RW0+', @RW0+RW7	SUB A, @RW0+', @RW0+RW7	SUB A, @RW0+', @RW0+RW7	ADDC A, @RW0+', @RW0+RW7	ADDC A, @RW0+', @RW0+RW7	CMP A, @RW0+', @RW0+RW7	CMP A, @RW0+', @RW0+RW7	AND A, @RW0+', @RW0+RW7	AND A, @RW0+', @RW0+RW7	OR A, @RW0+', @RW0+RW7	OR A, @RW0+', @RW0+RW7	XOR A, @RW0+', @RW0+RW7	XOR A, @RW0+', @RW0+RW7	DBNZ @RW0+, r', W0+RW7, r	DBNZ @RW0+, r', W0+RW7, r
+D	ADD A, @RW1+', @RW1+RW7	ADD A, @RW1+', @RW1+RW7	SUB A, @RW1+', @RW1+RW7	SUB A, @RW1+', @RW1+RW7	ADDC A, @RW1+', @RW1+RW7	ADDC A, @RW1+', @RW1+RW7	CMP A, @RW1+', @RW1+RW7	CMP A, @RW1+', @RW1+RW7	AND A, @RW1+', @RW1+RW7	AND A, @RW1+', @RW1+RW7	OR A, @RW1+', @RW1+RW7	OR A, @RW1+', @RW1+RW7	XOR A, @RW1+', @RW1+RW7	XOR A, @RW1+', @RW1+RW7	DBNZ @RW1+, r', W1+RW7, r	DBNZ @RW1+, r', W1+RW7, r
+E	ADD A, @RW2+', @PC+d16	ADD A, @RW2+', @PC+d16	SUB A, @RW2+', @PC+d16	SUB A, @RW2+', @PC+d16	ADDC A, @RW2+', @PC+d16	ADDC A, @RW2+', @PC+d16	CMP A, @RW2+', @PC+d16	CMP A, @RW2+', @PC+d16	AND A, @RW2+', @PC+d16	AND A, @RW2+', @PC+d16	OR A, @RW2+', @PC+d16	OR A, @RW2+', @PC+d16	XOR A, @RW2+', @PC+d16	XOR A, @RW2+', @PC+d16	DBNZ @RW2+, r', PC+d16, r	DBNZ @RW2+, r', PC+d16, r
+F	ADD A, @RW3+', A, addr16	ADD A, @RW3+', A, addr16	SUB A, @RW3+', A, addr16	SUB A, @RW3+', A, addr16	ADDC A, @RW3+', A, addr16	ADDC A, @RW3+', A, addr16	CMP A, @RW3+', A, addr16	CMP A, @RW3+', A, addr16	AND A, @RW3+', A, addr16	AND A, @RW3+', A, addr16	OR A, @RW3+', A, addr16	OR A, @RW3+', A, addr16	XOR A, @RW3+', A, addr16	XOR A, @RW3+', A, addr16	DBNZ @RW3+, r', addr16, r	DBNZ @RW3+, r', addr16, r

Table B.9-11 ea Instruction 6 (First Byte = 75<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUBC A, R0, @RW0+d8, A	SUBC A, R0, @RW0+d8, A	NEG R0, @RW0+d8, A	NEG A, R0, @RW0+d8, A	AND R0, A, @RW0+d8, A	AND A, R0, @RW0+d8, A	OR R0, A, @RW0+d8, A	OR A, R0, @RW0+d8, A	XOR R0, A, @RW0+d8, A	XOR A, R0, @RW0+d8, A	NOT R0, @RW0+d8, A	NOT A, R0, @RW0+d8, A
+1	ADD R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUBC A, R1, @RW1+d8, A	SUBC A, R1, @RW1+d8, A	NEG R1, @RW1+d8, A	NEG A, R1, @RW1+d8, A	AND R1, A, @RW1+d8, A	AND A, R1, @RW1+d8, A	OR R1, A, @RW1+d8, A	OR A, R1, @RW1+d8, A	XOR R1, A, @RW1+d8, A	XOR A, R1, @RW1+d8, A	NOT R1, @RW1+d8, A	NOT A, R1, @RW1+d8, A
+2	ADD R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUBC A, R2, @RW2+d8, A	SUBC A, R2, @RW2+d8, A	NEG R2, @RW2+d8, A	NEG A, R2, @RW2+d8, A	AND R2, A, @RW2+d8, A	AND A, R2, @RW2+d8, A	OR R2, A, @RW2+d8, A	OR A, R2, @RW2+d8, A	XOR R2, A, @RW2+d8, A	XOR A, R2, @RW2+d8, A	NOT R2, @RW2+d8, A	NOT A, R2, @RW2+d8, A
+3	ADD R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUBC A, R3, @RW3+d8, A	SUBC A, R3, @RW3+d8, A	NEG R3, @RW3+d8, A	NEG A, R3, @RW3+d8, A	AND R3, A, @RW3+d8, A	AND A, R3, @RW3+d8, A	OR R3, A, @RW3+d8, A	OR A, R3, @RW3+d8, A	XOR R3, A, @RW3+d8, A	XOR A, R3, @RW3+d8, A	NOT R3, @RW3+d8, A	NOT A, R3, @RW3+d8, A
+4	ADD R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUBC A, R4, @RW4+d8, A	SUBC A, R4, @RW4+d8, A	NEG R4, @RW4+d8, A	NEG A, R4, @RW4+d8, A	AND R4, A, @RW4+d8, A	AND A, R4, @RW4+d8, A	OR R4, A, @RW4+d8, A	OR A, R4, @RW4+d8, A	XOR R4, A, @RW4+d8, A	XOR A, R4, @RW4+d8, A	NOT R4, @RW4+d8, A	NOT A, R4, @RW4+d8, A
+5	ADD R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUBC A, R5, @RW5+d8, A	SUBC A, R5, @RW5+d8, A	NEG R5, @RW5+d8, A	NEG A, R5, @RW5+d8, A	AND R5, A, @RW5+d8, A	AND A, R5, @RW5+d8, A	OR R5, A, @RW5+d8, A	OR A, R5, @RW5+d8, A	XOR R5, A, @RW5+d8, A	XOR A, R5, @RW5+d8, A	NOT R5, @RW5+d8, A	NOT A, R5, @RW5+d8, A
+6	ADD R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUBC A, R6, @RW6+d8, A	SUBC A, R6, @RW6+d8, A	NEG R6, @RW6+d8, A	NEG A, R6, @RW6+d8, A	AND R6, A, @RW6+d8, A	AND A, R6, @RW6+d8, A	OR R6, A, @RW6+d8, A	OR A, R6, @RW6+d8, A	XOR R6, A, @RW6+d8, A	XOR A, R6, @RW6+d8, A	NOT R6, @RW6+d8, A	NOT A, R6, @RW6+d8, A
+7	ADD R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUBC A, R7, @RW7+d8, A	SUBC A, R7, @RW7+d8, A	NEG R7, @RW7+d8, A	NEG A, R7, @RW7+d8, A	AND R7, A, @RW7+d8, A	AND A, R7, @RW7+d8, A	OR R7, A, @RW7+d8, A	OR A, R7, @RW7+d8, A	XOR R7, A, @RW7+d8, A	XOR A, R7, @RW7+d8, A	NOT R7, @RW7+d8, A	NOT A, R7, @RW7+d8, A
+8	ADD @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUBC A, @RW0, @RW0+d16, A	SUBC A, @RW0, @RW0+d16, A	NEG @RW0, @RW0+d16, A	NEG A, @RW0, @RW0+d16, A	AND @RW0, A, @RW0+d16, A	AND A, @RW0, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	OR A, @RW0, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	XOR A, @RW0, @RW0+d16, A	NOT @RW0, @RW0+d16, A	NOT A, @RW0, @RW0+d16, A
+9	ADD @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUBC A, @RW1, @RW1+d16, A	SUBC A, @RW1, @RW1+d16, A	NEG @RW1, @RW1+d16, A	NEG A, @RW1, @RW1+d16, A	AND @RW1, A, @RW1+d16, A	AND A, @RW1, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	OR A, @RW1, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	XOR A, @RW1, @RW1+d16, A	NOT @RW1, @RW1+d16, A	NOT A, @RW1, @RW1+d16, A
+A	ADD @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUBC A, @RW2, @RW2+d16, A	SUBC A, @RW2, @RW2+d16, A	NEG @RW2, @RW2+d16, A	NEG A, @RW2, @RW2+d16, A	AND @RW2, A, @RW2+d16, A	AND A, @RW2, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	OR A, @RW2, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	XOR A, @RW2, @RW2+d16, A	NOT @RW2, @RW2+d16, A	NOT A, @RW2, @RW2+d16, A
+B	ADD @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUBC A, @RW3, @RW3+d16, A	SUBC A, @RW3, @RW3+d16, A	NEG @RW3, @RW3+d16, A	NEG A, @RW3, @RW3+d16, A	AND @RW3, A, @RW3+d16, A	AND A, @RW3, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	OR A, @RW3, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	XOR A, @RW3, @RW3+d16, A	NOT @RW3, @RW3+d16, A	NOT A, @RW3, @RW3+d16, A
+C	ADD @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7, A	NEG @RW0+, @RW0+RW7, A	NEG A, @RW0+, @RW0+RW7, A	AND @RW0+, A, @RW0+RW7, A	AND A, @RW0+, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	OR A, @RW0+, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	XOR A, @RW0+, @RW0+RW7, A	NOT @RW0+, @RW0+RW7, A	NOT A, @RW0+, @RW0+RW7, A
+D	ADD @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7, A	NEG @RW1+, @RW1+RW7, A	NEG A, @RW1+, @RW1+RW7, A	AND @RW1+, A, @RW1+RW7, A	AND A, @RW1+, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	OR A, @RW1+, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	XOR A, @RW1+, @RW1+RW7, A	NOT @RW1+, @RW1+RW7, A	NOT A, @RW1+, @RW1+RW7, A
+E	ADD @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUBC A, @RW2+, @PC+d16, A	SUBC A, @RW2+, @PC+d16, A	NEG @RW2+, @PC+d16, A	NEG A, @RW2+, @PC+d16, A	AND @RW2+, A, @PC+d16, A	AND A, @RW2+, @PC+d16, A	OR @RW2+, A, @PC+d16, A	OR A, @RW2+, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	XOR A, @RW2+, @PC+d16, A	NOT @RW2+, @PC+d16, A	NOT A, @RW2+, @PC+d16, A
+F	ADD @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUBC A, @RW3+, addr16, A	SUBC A, @RW3+, addr16, A	NEG @RW3+, addr16, A	NEG A, @RW3+, addr16, A	AND @RW3+, A, addr16, A	AND A, @RW3+, addr16, A	OR @RW3+, A, addr16, A	OR A, @RW3+, addr16, A	XOR @RW3+, A, addr16, A	XOR A, @RW3+, addr16, A	NOT @RW3+, addr16, A	NOT A, @RW3+, addr16, A



Table B.9-12 ea Instruction 7 (First Byte = 76<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW A, RW0, @RW0+d8	ADDW A, RW0, @RW0+d8	SUBW A, RW0, @RW0+d8	SUBW A, RW0, @RW0+d8	ADDCW A, RW0, @RW0+d8	ADDCW A, RW0, @RW0+d8	CMPW A, RW0, @RW0+d8	CMPW A, RW0, @RW0+d8	ANDW A, RW0, @RW0+d8	ANDW A, RW0, @RW0+d8	ORW A, RW0, @RW0+d8	ORW A, RW0, @RW0+d8	XORW A, RW0, @RW0+d8	XORW A, RW0, @RW0+d8	DWBZ RW0, r, @RW0+d8,r	DWBZ RW0, r, @RW0+d8,r
+1	ADDW A, RW1, @RW1+d8	ADDW A, RW1, @RW1+d8	SUBW A, RW1, @RW1+d8	SUBW A, RW1, @RW1+d8	ADDCW A, RW1, @RW1+d8	ADDCW A, RW1, @RW1+d8	CMPW A, RW1, @RW1+d8	CMPW A, RW1, @RW1+d8	ANDW A, RW1, @RW1+d8	ANDW A, RW1, @RW1+d8	ORW A, RW1, @RW1+d8	ORW A, RW1, @RW1+d8	XORW A, RW1, @RW1+d8	XORW A, RW1, @RW1+d8	DWBZ RW1, r, @RW1+d8,r	DWBZ RW1, r, @RW1+d8,r
+2	ADDW A, RW2, @RW2+d8	ADDW A, RW2, @RW2+d8	SUBW A, RW2, @RW2+d8	SUBW A, RW2, @RW2+d8	ADDCW A, RW2, @RW2+d8	ADDCW A, RW2, @RW2+d8	CMPW A, RW2, @RW2+d8	CMPW A, RW2, @RW2+d8	ANDW A, RW2, @RW2+d8	ANDW A, RW2, @RW2+d8	ORW A, RW2, @RW2+d8	ORW A, RW2, @RW2+d8	XORW A, RW2, @RW2+d8	XORW A, RW2, @RW2+d8	DWBZ RW2, r, @RW2+d8,r	DWBZ RW2, r, @RW2+d8,r
+3	ADDW A, RW3, @RW3+d8	ADDW A, RW3, @RW3+d8	SUBW A, RW3, @RW3+d8	SUBW A, RW3, @RW3+d8	ADDCW A, RW3, @RW3+d8	ADDCW A, RW3, @RW3+d8	CMPW A, RW3, @RW3+d8	CMPW A, RW3, @RW3+d8	ANDW A, RW3, @RW3+d8	ANDW A, RW3, @RW3+d8	ORW A, RW3, @RW3+d8	ORW A, RW3, @RW3+d8	XORW A, RW3, @RW3+d8	XORW A, RW3, @RW3+d8	DWBZ RW3, r, @RW3+d8,r	DWBZ RW3, r, @RW3+d8,r
+4	ADDW A, RW4, @RW4+d8	ADDW A, RW4, @RW4+d8	SUBW A, RW4, @RW4+d8	SUBW A, RW4, @RW4+d8	ADDCW A, RW4, @RW4+d8	ADDCW A, RW4, @RW4+d8	CMPW A, RW4, @RW4+d8	CMPW A, RW4, @RW4+d8	ANDW A, RW4, @RW4+d8	ANDW A, RW4, @RW4+d8	ORW A, RW4, @RW4+d8	ORW A, RW4, @RW4+d8	XORW A, RW4, @RW4+d8	XORW A, RW4, @RW4+d8	DWBZ RW4, r, @RW4+d8,r	DWBZ RW4, r, @RW4+d8,r
+5	ADDW A, RW5, @RW5+d8	ADDW A, RW5, @RW5+d8	SUBW A, RW5, @RW5+d8	SUBW A, RW5, @RW5+d8	ADDCW A, RW5, @RW5+d8	ADDCW A, RW5, @RW5+d8	CMPW A, RW5, @RW5+d8	CMPW A, RW5, @RW5+d8	ANDW A, RW5, @RW5+d8	ANDW A, RW5, @RW5+d8	ORW A, RW5, @RW5+d8	ORW A, RW5, @RW5+d8	XORW A, RW5, @RW5+d8	XORW A, RW5, @RW5+d8	DWBZ RW5, r, @RW5+d8,r	DWBZ RW5, r, @RW5+d8,r
+6	ADDW A, RW6, @RW6+d8	ADDW A, RW6, @RW6+d8	SUBW A, RW6, @RW6+d8	SUBW A, RW6, @RW6+d8	ADDCW A, RW6, @RW6+d8	ADDCW A, RW6, @RW6+d8	CMPW A, RW6, @RW6+d8	CMPW A, RW6, @RW6+d8	ANDW A, RW6, @RW6+d8	ANDW A, RW6, @RW6+d8	ORW A, RW6, @RW6+d8	ORW A, RW6, @RW6+d8	XORW A, RW6, @RW6+d8	XORW A, RW6, @RW6+d8	DWBZ RW6, r, @RW6+d8,r	DWBZ RW6, r, @RW6+d8,r
+7	ADDW A, RW7, @RW7+d8	ADDW A, RW7, @RW7+d8	SUBW A, RW7, @RW7+d8	SUBW A, RW7, @RW7+d8	ADDCW A, RW7, @RW7+d8	ADDCW A, RW7, @RW7+d8	CMPW A, RW7, @RW7+d8	CMPW A, RW7, @RW7+d8	ANDW A, RW7, @RW7+d8	ANDW A, RW7, @RW7+d8	ORW A, RW7, @RW7+d8	ORW A, RW7, @RW7+d8	XORW A, RW7, @RW7+d8	XORW A, RW7, @RW7+d8	DWBZ RW7, r, @RW7+d8,r	DWBZ RW7, r, @RW7+d8,r
+8	ADDW A, RW0, @RW0+d16	ADDW A, RW0, @RW0+d16	SUBW A, RW0, @RW0+d16	SUBW A, RW0, @RW0+d16	ADDCW A, RW0, @RW0+d16	ADDCW A, RW0, @RW0+d16	CMPW A, RW0, @RW0+d16	CMPW A, RW0, @RW0+d16	ANDW A, RW0, @RW0+d16	ANDW A, RW0, @RW0+d16	ORW A, RW0, @RW0+d16	ORW A, RW0, @RW0+d16	XORW A, RW0, @RW0+d16	XORW A, RW0, @RW0+d16	DWBZ @RW0, r, @RW0+d16,r	DWBZ @RW0, r, @RW0+d16,r
+9	ADDW A, RW1, @RW1+d16	ADDW A, RW1, @RW1+d16	SUBW A, RW1, @RW1+d16	SUBW A, RW1, @RW1+d16	ADDCW A, RW1, @RW1+d16	ADDCW A, RW1, @RW1+d16	CMPW A, RW1, @RW1+d16	CMPW A, RW1, @RW1+d16	ANDW A, RW1, @RW1+d16	ANDW A, RW1, @RW1+d16	ORW A, RW1, @RW1+d16	ORW A, RW1, @RW1+d16	XORW A, RW1, @RW1+d16	XORW A, RW1, @RW1+d16	DWBZ @RW1, r, @RW1+d16,r	DWBZ @RW1, r, @RW1+d16,r
+A	ADDW A, RW2, @RW2+d16	ADDW A, RW2, @RW2+d16	SUBW A, RW2, @RW2+d16	SUBW A, RW2, @RW2+d16	ADDCW A, RW2, @RW2+d16	ADDCW A, RW2, @RW2+d16	CMPW A, RW2, @RW2+d16	CMPW A, RW2, @RW2+d16	ANDW A, RW2, @RW2+d16	ANDW A, RW2, @RW2+d16	ORW A, RW2, @RW2+d16	ORW A, RW2, @RW2+d16	XORW A, RW2, @RW2+d16	XORW A, RW2, @RW2+d16	DWBZ @RW2, r, @RW2+d16,r	DWBZ @RW2, r, @RW2+d16,r
+B	ADDW A, RW3, @RW3+d16	ADDW A, RW3, @RW3+d16	SUBW A, RW3, @RW3+d16	SUBW A, RW3, @RW3+d16	ADDCW A, RW3, @RW3+d16	ADDCW A, RW3, @RW3+d16	CMPW A, RW3, @RW3+d16	CMPW A, RW3, @RW3+d16	ANDW A, RW3, @RW3+d16	ANDW A, RW3, @RW3+d16	ORW A, RW3, @RW3+d16	ORW A, RW3, @RW3+d16	XORW A, RW3, @RW3+d16	XORW A, RW3, @RW3+d16	DWBZ @RW3, r, @RW3+d16,r	DWBZ @RW3, r, @RW3+d16,r
+C	ADDW A, RW0+, @RW0-RW7	ADDW A, RW0+, @RW0-RW7	SUBW A, RW0+, @RW0-RW7	SUBW A, RW0+, @RW0-RW7	ADDCW A, RW0+, @RW0-RW7	ADDCW A, RW0+, @RW0-RW7	CMPW A, RW0+, @RW0-RW7	CMPW A, RW0+, @RW0-RW7	ANDW A, RW0+, @RW0-RW7	ANDW A, RW0+, @RW0-RW7	ORW A, RW0+, @RW0-RW7	ORW A, RW0+, @RW0-RW7	XORW A, RW0+, @RW0-RW7	XORW A, RW0+, @RW0-RW7	DWBZ @RW0+, r, @RW0-RW7,r	DWBZ @RW0+, r, @RW0-RW7,r
+D	ADDW A, RW1+, @RW1-RW7	ADDW A, RW1+, @RW1-RW7	SUBW A, RW1+, @RW1-RW7	SUBW A, RW1+, @RW1-RW7	ADDCW A, RW1+, @RW1-RW7	ADDCW A, RW1+, @RW1-RW7	CMPW A, RW1+, @RW1-RW7	CMPW A, RW1+, @RW1-RW7	ANDW A, RW1+, @RW1-RW7	ANDW A, RW1+, @RW1-RW7	ORW A, RW1+, @RW1-RW7	ORW A, RW1+, @RW1-RW7	XORW A, RW1+, @RW1-RW7	XORW A, RW1+, @RW1-RW7	DWBZ @RW1+, r, @RW1-RW7,r	DWBZ @RW1+, r, @RW1-RW7,r
+E	ADDW A, RW2+, @PC+d16	ADDW A, RW2+, @PC+d16	SUBW A, RW2+, @PC+d16	SUBW A, RW2+, @PC+d16	ADDCW A, RW2+, @PC+d16	ADDCW A, RW2+, @PC+d16	CMPW A, RW2+, @PC+d16	CMPW A, RW2+, @PC+d16	ANDW A, RW2+, @PC+d16	ANDW A, RW2+, @PC+d16	ORW A, RW2+, @PC+d16	ORW A, RW2+, @PC+d16	XORW A, RW2+, @PC+d16	XORW A, RW2+, @PC+d16	DWBZ @RW2+, r, @PC+d16,r	DWBZ @RW2+, r, @PC+d16,r
+F	ADDW A, RW3+, addr 16	ADDW A, RW3+, addr 16	SUBW A, RW3+, addr 16	SUBW A, RW3+, addr 16	ADDCW A, RW3+, addr 16	ADDCW A, RW3+, addr 16	CMPW A, RW3+, addr 16	CMPW A, RW3+, addr 16	ANDW A, RW3+, addr 16	ANDW A, RW3+, addr 16	ORW A, RW3+, addr 16	ORW A, RW3+, addr 16	XORW A, RW3+, addr 16	XORW A, RW3+, addr 16	DWBZ @RW3+, r, addr 16, r	DWBZ @RW3+, r, addr 16, r

Table B.9-13 ea Instruction 8 (First Byte = 77<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A, @RW0+d8, A	SUBW RW0, A, @RW0+d8, A	SUBW RW0, A, @RW0+d8, A	SUBW RW0, A, @RW0+d8, A	SUBCW A, RW0, @RW0+d8	SUBCW A, RW0, @RW0+d8	NEGW RW0, @RW0+d8	NEGW RW0, @RW0+d8	ANDW RW0, A, @RW0+d8, A	ANDW RW0, A, @RW0+d8, A	ORW RW0, A, @RW0+d8, A	ORW RW0, A, @RW0+d8, A	XORW RW0, A, @RW0+d8, A	XORW RW0, A, @RW0+d8, A	NOTW RW0, @RW0+d8	NOTW RW0, @RW0+d8
+1	ADDW RW1, A, @RW1+d8, A	SUBW RW1, A, @RW1+d8, A	SUBW RW1, A, @RW1+d8, A	SUBW RW1, A, @RW1+d8, A	SUBCW A, RW1, @RW1+d8	SUBCW A, RW1, @RW1+d8	NEGW RW1, @RW1+d8	NEGW RW1, @RW1+d8	ANDW RW1, A, @RW1+d8, A	ANDW RW1, A, @RW1+d8, A	ORW RW1, A, @RW1+d8, A	ORW RW1, A, @RW1+d8, A	XORW RW1, A, @RW1+d8, A	XORW RW1, A, @RW1+d8, A	NOTW RW1, @RW1+d8	NOTW RW1, @RW1+d8
+2	ADDW RW2, A, @RW2+d8, A	SUBW RW2, A, @RW2+d8, A	SUBW RW2, A, @RW2+d8, A	SUBW RW2, A, @RW2+d8, A	SUBCW A, RW2, @RW2+d8	SUBCW A, RW2, @RW2+d8	NEGW RW2, @RW2+d8	NEGW RW2, @RW2+d8	ANDW RW2, A, @RW2+d8, A	ANDW RW2, A, @RW2+d8, A	ORW RW2, A, @RW2+d8, A	ORW RW2, A, @RW2+d8, A	XORW RW2, A, @RW2+d8, A	XORW RW2, A, @RW2+d8, A	NOTW RW2, @RW2+d8	NOTW RW2, @RW2+d8
+3	ADDW RW3, A, @RW3+d8, A	SUBW RW3, A, @RW3+d8, A	SUBW RW3, A, @RW3+d8, A	SUBW RW3, A, @RW3+d8, A	SUBCW A, RW3, @RW3+d8	SUBCW A, RW3, @RW3+d8	NEGW RW3, @RW3+d8	NEGW RW3, @RW3+d8	ANDW RW3, A, @RW3+d8, A	ANDW RW3, A, @RW3+d8, A	ORW RW3, A, @RW3+d8, A	ORW RW3, A, @RW3+d8, A	XORW RW3, A, @RW3+d8, A	XORW RW3, A, @RW3+d8, A	NOTW RW3, @RW3+d8	NOTW RW3, @RW3+d8
+4	ADDW RW4, A, @RW4+d8, A	SUBW RW4, A, @RW4+d8, A	SUBW RW4, A, @RW4+d8, A	SUBW RW4, A, @RW4+d8, A	SUBCW A, RW4, @RW4+d8	SUBCW A, RW4, @RW4+d8	NEGW RW4, @RW4+d8	NEGW RW4, @RW4+d8	ANDW RW4, A, @RW4+d8, A	ANDW RW4, A, @RW4+d8, A	ORW RW4, A, @RW4+d8, A	ORW RW4, A, @RW4+d8, A	XORW RW4, A, @RW4+d8, A	XORW RW4, A, @RW4+d8, A	NOTW RW4, @RW4+d8	NOTW RW4, @RW4+d8
+5	ADDW RW5, A, @RW5+d8, A	SUBW RW5, A, @RW5+d8, A	SUBW RW5, A, @RW5+d8, A	SUBW RW5, A, @RW5+d8, A	SUBCW A, RW5, @RW5+d8	SUBCW A, RW5, @RW5+d8	NEGW RW5, @RW5+d8	NEGW RW5, @RW5+d8	ANDW RW5, A, @RW5+d8, A	ANDW RW5, A, @RW5+d8, A	ORW RW5, A, @RW5+d8, A	ORW RW5, A, @RW5+d8, A	XORW RW5, A, @RW5+d8, A	XORW RW5, A, @RW5+d8, A	NOTW RW5, @RW5+d8	NOTW RW5, @RW5+d8
+6	ADDW RW6, A, @RW6+d8, A	SUBW RW6, A, @RW6+d8, A	SUBW RW6, A, @RW6+d8, A	SUBW RW6, A, @RW6+d8, A	SUBCW A, RW6, @RW6+d8	SUBCW A, RW6, @RW6+d8	NEGW RW6, @RW6+d8	NEGW RW6, @RW6+d8	ANDW RW6, A, @RW6+d8, A	ANDW RW6, A, @RW6+d8, A	ORW RW6, A, @RW6+d8, A	ORW RW6, A, @RW6+d8, A	XORW RW6, A, @RW6+d8, A	XORW RW6, A, @RW6+d8, A	NOTW RW6, @RW6+d8	NOTW RW6, @RW6+d8
+7	ADDW RW7, A, @RW7+d8, A	SUBW RW7, A, @RW7+d8, A	SUBW RW7, A, @RW7+d8, A	SUBW RW7, A, @RW7+d8, A	SUBCW A, RW7, @RW7+d8	SUBCW A, RW7, @RW7+d8	NEGW RW7, @RW7+d8	NEGW RW7, @RW7+d8	ANDW RW7, A, @RW7+d8, A	ANDW RW7, A, @RW7+d8, A	ORW RW7, A, @RW7+d8, A	ORW RW7, A, @RW7+d8, A	XORW RW7, A, @RW7+d8, A	XORW RW7, A, @RW7+d8, A	NOTW RW7, @RW7+d8	NOTW RW7, @RW7+d8
+8	ADDW RW0, A, @RW0+d16, A	SUBW RW0, A, @RW0+d16, A	SUBW RW0, A, @RW0+d16, A	SUBW RW0, A, @RW0+d16, A	SUBCW A, @RW0, @RW0+d16	SUBCW A, @RW0, @RW0+d16	NEGW @RW0, @RW0+d16	NEGW @RW0, @RW0+d16	ANDW @RW0, A, @RW0+d16, A	ANDW @RW0, A, @RW0+d16, A	ORW @RW0, A, @RW0+d16, A	ORW @RW0, A, @RW0+d16, A	XORW @RW0, A, @RW0+d16, A	XORW @RW0, A, @RW0+d16, A	NOTW @RW0, @RW0+d16	NOTW @RW0, @RW0+d16
+9	ADDW @RW1, A, @RW1+d16, A	SUBW @RW1, A, @RW1+d16, A	SUBW @RW1, A, @RW1+d16, A	SUBW @RW1, A, @RW1+d16, A	SUBCW A, @RW1, @RW1+d16	SUBCW A, @RW1, @RW1+d16	NEGW @RW1, @RW1+d16	NEGW @RW1, @RW1+d16	ANDW @RW1, A, @RW1+d16, A	ANDW @RW1, A, @RW1+d16, A	ORW @RW1, A, @RW1+d16, A	ORW @RW1, A, @RW1+d16, A	XORW @RW1, A, @RW1+d16, A	XORW @RW1, A, @RW1+d16, A	NOTW @RW1, @RW1+d16	NOTW @RW1, @RW1+d16
+A	ADDW @RW2, A, @RW2+d16, A	SUBW @RW2, A, @RW2+d16, A	SUBW @RW2, A, @RW2+d16, A	SUBW @RW2, A, @RW2+d16, A	SUBCW A, @RW2, @RW2+d16	SUBCW A, @RW2, @RW2+d16	NEGW @RW2, @RW2+d16	NEGW @RW2, @RW2+d16	ANDW @RW2, A, @RW2+d16, A	ANDW @RW2, A, @RW2+d16, A	ORW @RW2, A, @RW2+d16, A	ORW @RW2, A, @RW2+d16, A	XORW @RW2, A, @RW2+d16, A	XORW @RW2, A, @RW2+d16, A	NOTW @RW2, @RW2+d16	NOTW @RW2, @RW2+d16
+B	ADDW @RW3, A, @RW3+d16, A	SUBW @RW3, A, @RW3+d16, A	SUBW @RW3, A, @RW3+d16, A	SUBW @RW3, A, @RW3+d16, A	SUBCW A, @RW3, @RW3+d16	SUBCW A, @RW3, @RW3+d16	NEGW @RW3, @RW3+d16	NEGW @RW3, @RW3+d16	ANDW @RW3, A, @RW3+d16, A	ANDW @RW3, A, @RW3+d16, A	ORW @RW3, A, @RW3+d16, A	ORW @RW3, A, @RW3+d16, A	XORW @RW3, A, @RW3+d16, A	XORW @RW3, A, @RW3+d16, A	NOTW @RW3, @RW3+d16	NOTW @RW3, @RW3+d16
+C	ADDW @RW0+, A, @RW0+RW7, A	SUBW @RW0+, A, @RW0+RW7, A	SUBW @RW0+, A, @RW0+RW7, A	SUBW @RW0+, A, @RW0+RW7, A	SUBCW A, @RW0+, @RW0+RW7	SUBCW A, @RW0+, @RW0+RW7	NEGW @RW0+, @RW0+RW7	NEGW @RW0+, @RW0+RW7	ANDW @RW0+, A, @RW0+RW7, A	ANDW @RW0+, A, @RW0+RW7, A	ORW @RW0+, A, @RW0+RW7, A	ORW @RW0+, A, @RW0+RW7, A	XORW @RW0+, A, @RW0+RW7, A	XORW @RW0+, A, @RW0+RW7, A	NOTW @RW0+, @RW0+RW7	NOTW @RW0+, @RW0+RW7
+D	ADDW @RW1+, A, @RW1+RW7, A	SUBW @RW1+, A, @RW1+RW7, A	SUBW @RW1+, A, @RW1+RW7, A	SUBW @RW1+, A, @RW1+RW7, A	SUBCW A, @RW1+, @RW1+RW7	SUBCW A, @RW1+, @RW1+RW7	NEGW @RW1+, @RW1+RW7	NEGW @RW1+, @RW1+RW7	ANDW @RW1+, A, @RW1+RW7, A	ANDW @RW1+, A, @RW1+RW7, A	ORW @RW1+, A, @RW1+RW7, A	ORW @RW1+, A, @RW1+RW7, A	XORW @RW1+, A, @RW1+RW7, A	XORW @RW1+, A, @RW1+RW7, A	NOTW @RW1+, @RW1+RW7	NOTW @RW1+, @RW1+RW7
+E	ADDW @RW2+, A, @PC+d16, A	SUBW @RW2+, A, @PC+d16, A	SUBW @RW2+, A, @PC+d16, A	SUBW @RW2+, A, @PC+d16, A	SUBCW A, @RW2+, @PC+d16	SUBCW A, @RW2+, @PC+d16	NEGW @RW2+, @PC+d16	NEGW @RW2+, @PC+d16	ANDW @RW2+, A, @PC+d16, A	ANDW @RW2+, A, @PC+d16, A	ORW @RW2+, A, @PC+d16, A	ORW @RW2+, A, @PC+d16, A	XORW @RW2+, A, @PC+d16, A	XORW @RW2+, A, @PC+d16, A	NOTW @RW2+, @PC+d16	NOTW @RW2+, @PC+d16
+F	ADDW @RW3+, A, addr16, A	SUBW @RW3+, A, addr16, A	SUBW @RW3+, A, addr16, A	SUBW @RW3+, A, addr16, A	SUBCW A, @RW3+, addr16	SUBCW A, @RW3+, addr16	NEGW @RW3+, addr16	NEGW @RW3+, addr16	ANDW @RW3+, A, addr16, A	ANDW @RW3+, A, addr16, A	ORW @RW3+, A, addr16, A	ORW @RW3+, A, addr16, A	XORW @RW3+, A, addr16, A	XORW @RW3+, A, addr16, A	NOTW @RW3+, addr16	NOTW @RW3+, addr16

## 496

**Table B.9-14 ea Instruction 9 (First Byte = 78<sub>H</sub>)**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MULU A, R0' @RW0+d8	MULU A, R0' @RW0+d8	MULUW A, RW0' @RW0+d8	MULUW A, RW0' @RW0+d8	MUL A, R0' @RW0+d8	MUL A, R0' @RW0+d8	MULW A, RW0' @RW0+d8	MULW A, RW0' @RW0+d8	DIVU A, R0' @RW0+d8	DIVU A, R0' @RW0+d8	DIVUW A, RW0' @RW0+d8	DIVUW A, RW0' @RW0+d8	DIV A, R0' @RW0+d8	DIV A, R0' @RW0+d8	DIVW A, RW0' @RW0+d8	DIVW A, RW0' @RW0+d8
+1	MULU A, R1' @RW1+d8	MULU A, R1' @RW1+d8	MULUW A, RW1' @RW1+d8	MULUW A, RW1' @RW1+d8	MUL A, R1' @RW1+d8	MUL A, R1' @RW1+d8	MULW A, RW1' @RW1+d8	MULW A, RW1' @RW1+d8	DIVU A, R1' @RW1+d8	DIVU A, R1' @RW1+d8	DIVUW A, RW1' @RW1+d8	DIVUW A, RW1' @RW1+d8	DIV A, R1' @RW1+d8	DIV A, R1' @RW1+d8	DIVW A, RW1' @RW1+d8	DIVW A, RW1' @RW1+d8
+2	MULU A, R2' @RW2+d8	MULU A, R2' @RW2+d8	MULUW A, RW2' @RW2+d8	MULUW A, RW2' @RW2+d8	MUL A, R2' @RW2+d8	MUL A, R2' @RW2+d8	MULW A, RW2' @RW2+d8	MULW A, RW2' @RW2+d8	DIVU A, R2' @RW2+d8	DIVU A, R2' @RW2+d8	DIVUW A, RW2' @RW2+d8	DIVUW A, RW2' @RW2+d8	DIV A, R2' @RW2+d8	DIV A, R2' @RW2+d8	DIVW A, RW2' @RW2+d8	DIVW A, RW2' @RW2+d8
+3	MULU A, R3' @RW3+d8	MULU A, R3' @RW3+d8	MULUW A, RW3' @RW3+d8	MULUW A, RW3' @RW3+d8	MUL A, R3' @RW3+d8	MUL A, R3' @RW3+d8	MULW A, RW3' @RW3+d8	MULW A, RW3' @RW3+d8	DIVU A, R3' @RW3+d8	DIVU A, R3' @RW3+d8	DIVUW A, RW3' @RW3+d8	DIVUW A, RW3' @RW3+d8	DIV A, R3' @RW3+d8	DIV A, R3' @RW3+d8	DIVW A, RW3' @RW3+d8	DIVW A, RW3' @RW3+d8
+4	MULU A, R4' @RW4+d8	MULU A, R4' @RW4+d8	MULUW A, RW4' @RW4+d8	MULUW A, RW4' @RW4+d8	MUL A, R4' @RW4+d8	MUL A, R4' @RW4+d8	MULW A, RW4' @RW4+d8	MULW A, RW4' @RW4+d8	DIVU A, R4' @RW4+d8	DIVU A, R4' @RW4+d8	DIVUW A, RW4' @RW4+d8	DIVUW A, RW4' @RW4+d8	DIV A, R4' @RW4+d8	DIV A, R4' @RW4+d8	DIVW A, RW4' @RW4+d8	DIVW A, RW4' @RW4+d8
+5	MULU A, R5' @RW5+d8	MULU A, R5' @RW5+d8	MULUW A, RW5' @RW5+d8	MULUW A, RW5' @RW5+d8	MUL A, R5' @RW5+d8	MUL A, R5' @RW5+d8	MULW A, RW5' @RW5+d8	MULW A, RW5' @RW5+d8	DIVU A, R5' @RW5+d8	DIVU A, R5' @RW5+d8	DIVUW A, RW5' @RW5+d8	DIVUW A, RW5' @RW5+d8	DIV A, R5' @RW5+d8	DIV A, R5' @RW5+d8	DIVW A, RW5' @RW5+d8	DIVW A, RW5' @RW5+d8
+6	MULU A, R6' @RW6+d8	MULU A, R6' @RW6+d8	MULUW A, RW6' @RW6+d8	MULUW A, RW6' @RW6+d8	MUL A, R6' @RW6+d8	MUL A, R6' @RW6+d8	MULW A, RW6' @RW6+d8	MULW A, RW6' @RW6+d8	DIVU A, R6' @RW6+d8	DIVU A, R6' @RW6+d8	DIVUW A, RW6' @RW6+d8	DIVUW A, RW6' @RW6+d8	DIV A, R6' @RW6+d8	DIV A, R6' @RW6+d8	DIVW A, RW6' @RW6+d8	DIVW A, RW6' @RW6+d8
+7	MULU A, R7' @RW7+d8	MULU A, R7' @RW7+d8	MULUW A, RW7' @RW7+d8	MULUW A, RW7' @RW7+d8	MUL A, R7' @RW7+d8	MUL A, R7' @RW7+d8	MULW A, RW7' @RW7+d8	MULW A, RW7' @RW7+d8	DIVU A, R7' @RW7+d8	DIVU A, R7' @RW7+d8	DIVUW A, RW7' @RW7+d8	DIVUW A, RW7' @RW7+d8	DIV A, R7' @RW7+d8	DIV A, R7' @RW7+d8	DIVW A, RW7' @RW7+d8	DIVW A, RW7' @RW7+d8
+8	MULU A, @RW0, @RW0+d16	MULU A, @RW0, @RW0+d16	MULUW A, RW0' @RW0+d16	MULUW A, RW0' @RW0+d16	MUL A, @RW0' @RW0+d16	MUL A, @RW0' @RW0+d16	MULW A, RW0' @RW0+d16	MULW A, RW0' @RW0+d16	DIVU A, @RW0' @RW0+d16	DIVU A, @RW0' @RW0+d16	DIVUW A, RW0' @RW0+d16	DIVUW A, RW0' @RW0+d16	DIV A, @RW0' @RW0+d16	DIV A, @RW0' @RW0+d16	DIVW A, @RW0' @RW0+d16	DIVW A, @RW0' @RW0+d16
+9	MULU A, @RW1, @RW1+d16	MULU A, @RW1, @RW1+d16	MULUW A, RW1' @RW1+d16	MULUW A, RW1' @RW1+d16	MUL A, @RW1' @RW1+d16	MUL A, @RW1' @RW1+d16	MULW A, RW1' @RW1+d16	MULW A, RW1' @RW1+d16	DIVU A, @RW1' @RW1+d16	DIVU A, @RW1' @RW1+d16	DIVUW A, RW1' @RW1+d16	DIVUW A, RW1' @RW1+d16	DIV A, @RW1' @RW1+d16	DIV A, @RW1' @RW1+d16	DIVW A, @RW1' @RW1+d16	DIVW A, @RW1' @RW1+d16
+A	MULU A, @RW2, @RW2+d16	MULU A, @RW2, @RW2+d16	MULUW A, RW2' @RW2+d16	MULUW A, RW2' @RW2+d16	MUL A, @RW2' @RW2+d16	MUL A, @RW2' @RW2+d16	MULW A, RW2' @RW2+d16	MULW A, RW2' @RW2+d16	DIVU A, @RW2' @RW2+d16	DIVU A, @RW2' @RW2+d16	DIVUW A, RW2' @RW2+d16	DIVUW A, RW2' @RW2+d16	DIV A, @RW2' @RW2+d16	DIV A, @RW2' @RW2+d16	DIVW A, @RW2' @RW2+d16	DIVW A, @RW2' @RW2+d16
+B	MULU A, @RW3, @RW3+d16	MULU A, @RW3, @RW3+d16	MULUW A, RW3' @RW3+d16	MULUW A, RW3' @RW3+d16	MUL A, @RW3' @RW3+d16	MUL A, @RW3' @RW3+d16	MULW A, RW3' @RW3+d16	MULW A, RW3' @RW3+d16	DIVU A, @RW3' @RW3+d16	DIVU A, @RW3' @RW3+d16	DIVUW A, RW3' @RW3+d16	DIVUW A, RW3' @RW3+d16	DIV A, @RW3' @RW3+d16	DIV A, @RW3' @RW3+d16	DIVW A, @RW3' @RW3+d16	DIVW A, @RW3' @RW3+d16
+C	MULU A, @RW0+, @RW0+RW7	MULU A, @RW0+, @RW0+RW7	MULUW A, RW0' @RW0+RW7	MULUW A, RW0' @RW0+RW7	MUL A, @RW0' @RW0+RW7	MUL A, @RW0' @RW0+RW7	MULW A, RW0' @RW0+RW7	MULW A, RW0' @RW0+RW7	DIVU A, @RW0' @RW0+RW7	DIVU A, @RW0' @RW0+RW7	DIVUW A, RW0' @RW0+RW7	DIVUW A, RW0' @RW0+RW7	DIV A, @RW0' @RW0+RW7	DIV A, @RW0' @RW0+RW7	DIVW A, @RW0' @RW0+RW7	DIVW A, @RW0' @RW0+RW7
+D	MULU A, @RW1+, @RW1+RW7	MULU A, @RW1+, @RW1+RW7	MULUW A, RW1' @RW1+RW7	MULUW A, RW1' @RW1+RW7	MUL A, @RW1' @RW1+RW7	MUL A, @RW1' @RW1+RW7	MULW A, RW1' @RW1+RW7	MULW A, RW1' @RW1+RW7	DIVU A, @RW1' @RW1+RW7	DIVU A, @RW1' @RW1+RW7	DIVUW A, RW1' @RW1+RW7	DIVUW A, RW1' @RW1+RW7	DIV A, @RW1' @RW1+RW7	DIV A, @RW1' @RW1+RW7	DIVW A, @RW1' @RW1+RW7	DIVW A, @RW1' @RW1+RW7
+E	MULU A, @RW2+, @PC+d16	MULU A, @RW2+, @PC+d16	MULUW A, RW2' @RW2+ A, @PC+d16	MULUW A, RW2' @RW2+ A, @PC+d16	MUL A, @RW2' @RW2+ A, @PC+d16	MUL A, @RW2' @RW2+ A, @PC+d16	MULW A, RW2' @RW2+ A, @PC+d16	MULW A, RW2' @RW2+ A, @PC+d16	DIVU A, @RW2' @RW2+ A, @PC+d16	DIVU A, @RW2' @RW2+ A, @PC+d16	DIVUW A, RW2' @RW2+ A, @PC+d16	DIVUW A, RW2' @RW2+ A, @PC+d16	DIV A, @RW2' @RW2+ A, @PC+d16	DIV A, @RW2' @RW2+ A, @PC+d16	DIVW A, @RW2' @RW2+ A, @PC+d16	DIVW A, @RW2' @RW2+ A, @PC+d16
+F	MULU A, @RW3+, addr16	MULU A, @RW3+, addr16	MULUW A, RW3' @RW3+ addr16	MULUW A, RW3' @RW3+ addr16	MUL A, @RW3' @RW3+ addr16	MUL A, @RW3' @RW3+ addr16	MULW A, RW3' @RW3+ addr16	MULW A, RW3' @RW3+ addr16	DIVU A, @RW3' @RW3+ addr16	DIVU A, @RW3' @RW3+ addr16	DIVUW A, RW3' @RW3+ addr16	DIVUW A, RW3' @RW3+ addr16	DIV A, @RW3' @RW3+ addr16	DIV A, @RW3' @RW3+ addr16	DIVW A, @RW3' @RW3+ addr16	DIVW A, @RW3' @RW3+ addr16

**Table B.9-15 MOVEA RWi, ea Instruction (First Byte = 79<sub>H</sub>)**

[illegible]

**Table B.9-16 MOV Ri, ea Instruction (First Byte = 7A<sub>H</sub>)**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0 @RW0+d8	MOV R0, R0 @RW0+d8	MOV R1, R1 @RW0+d8	MOV R1, R0 @RW0+d8	MOV R2, R0 @RW0+d8	MOV R2, R0 @RW0+d8	MOV R3, R0 @RW0+d8	MOV R3, R0 @RW0+d8	MOV R4, R0 @RW0+d8	MOV R4, R0 @RW0+d8	MOV R5, R0 @RW0+d8	MOV R5, R0 @RW0+d8	MOV R6, R0 @RW0+d8	MOV R6, R0 @RW0+d8	MOV R7, R0 @RW0+d8	MOV R7, R0 @RW0+d8
+1	MOV R0, R1 @RW1+d8	MOV R0, R0 @RW1+d8	MOV R1, R1 @RW1+d8	MOV R1, R1 @RW1+d8	MOV R2, R1 @RW1+d8	MOV R2, R1 @RW1+d8	MOV R3, R1 @RW1+d8	MOV R3, R1 @RW1+d8	MOV R4, R1 @RW1+d8	MOV R4, R1 @RW1+d8	MOV R5, R1 @RW1+d8	MOV R5, R1 @RW1+d8	MOV R6, R1 @RW1+d8	MOV R6, R1 @RW1+d8	MOV R7, R1 @RW1+d8	MOV R7, R1 @RW1+d8
+2	MOV R0, R2 @RW2+d8	MOV R0, R0 @RW2+d8	MOV R1, R2 @RW2+d8	MOV R1, R2 @RW2+d8	MOV R2, R2 @RW2+d8	MOV R2, R2 @RW2+d8	MOV R3, R2 @RW2+d8	MOV R3, R2 @RW2+d8	MOV R4, R2 @RW2+d8	MOV R4, R2 @RW2+d8	MOV R5, R2 @RW2+d8	MOV R5, R2 @RW2+d8	MOV R6, R2 @RW2+d8	MOV R6, R2 @RW2+d8	MOV R7, R2 @RW2+d8	MOV R7, R2 @RW2+d8
+3	MOV R0, R3 @RW3+d8	MOV R0, R0 @RW3+d8	MOV R1, R3 @RW3+d8	MOV R1, R3 @RW3+d8	MOV R2, R3 @RW3+d8	MOV R2, R3 @RW3+d8	MOV R3, R3 @RW3+d8	MOV R3, R3 @RW3+d8	MOV R4, R3 @RW3+d8	MOV R4, R3 @RW3+d8	MOV R5, R3 @RW3+d8	MOV R5, R3 @RW3+d8	MOV R6, R3 @RW3+d8	MOV R6, R3 @RW3+d8	MOV R7, R3 @RW3+d8	MOV R7, R3 @RW3+d8
+4	MOV R0, R4 @RW4+d8	MOV R0, R0 @RW4+d8	MOV R1, R4 @RW4+d8	MOV R1, R4 @RW4+d8	MOV R2, R4 @RW4+d8	MOV R2, R4 @RW4+d8	MOV R3, R4 @RW4+d8	MOV R3, R4 @RW4+d8	MOV R4, R4 @RW4+d8	MOV R4, R4 @RW4+d8	MOV R5, R4 @RW4+d8	MOV R5, R4 @RW4+d8	MOV R6, R4 @RW4+d8	MOV R6, R4 @RW4+d8	MOV R7, R4 @RW4+d8	MOV R7, R4 @RW4+d8
+5	MOV R0, R5 @RW5+d8	MOV R0, R0 @RW5+d8	MOV R1, R5 @RW5+d8	MOV R1, R5 @RW5+d8	MOV R2, R5 @RW5+d8	MOV R2, R5 @RW5+d8	MOV R3, R5 @RW5+d8	MOV R3, R5 @RW5+d8	MOV R4, R5 @RW5+d8	MOV R4, R5 @RW5+d8	MOV R5, R5 @RW5+d8	MOV R5, R5 @RW5+d8	MOV R6, R5 @RW5+d8	MOV R6, R5 @RW5+d8	MOV R7, R5 @RW5+d8	MOV R7, R5 @RW5+d8
+6	MOV R0, R6 @RW6+d8	MOV R0, R0 @RW6+d8	MOV R1, R6 @RW6+d8	MOV R1, R6 @RW6+d8	MOV R2, R6 @RW6+d8	MOV R2, R6 @RW6+d8	MOV R3, R6 @RW6+d8	MOV R3, R6 @RW6+d8	MOV R4, R6 @RW6+d8	MOV R4, R6 @RW6+d8	MOV R5, R6 @RW6+d8	MOV R5, R6 @RW6+d8	MOV R6, R6 @RW6+d8	MOV R6, R6 @RW6+d8	MOV R7, R6 @RW6+d8	MOV R7, R6 @RW6+d8
+7	MOV R0, R7 @RW7+d8	MOV R0, R0 @RW7+d8	MOV R1, R7 @RW7+d8	MOV R1, R7 @RW7+d8	MOV R2, R7 @RW7+d8	MOV R2, R7 @RW7+d8	MOV R3, R7 @RW7+d8	MOV R3, R7 @RW7+d8	MOV R4, R7 @RW7+d8	MOV R4, R7 @RW7+d8	MOV R5, R7 @RW7+d8	MOV R5, R7 @RW7+d8	MOV R6, R7 @RW7+d8	MOV R6, R7 @RW7+d8	MOV R7, R7 @RW7+d8	MOV R7, R7 @RW7+d8
+8	MOV R0, R8 @RW8+d8	MOV R0, R0 @RW8+d8	MOV R1, R8 @RW8+d8	MOV R1, R8 @RW8+d8	MOV R2, R8 @RW8+d8	MOV R2, R8 @RW8+d8	MOV R3, R8 @RW8+d8	MOV R3, R8 @RW8+d8	MOV R4, R8 @RW8+d8	MOV R4, R8 @RW8+d8	MOV R5, R8 @RW8+d8	MOV R5, R8 @RW8+d8	MOV R6, R8 @RW8+d8	MOV R6, R8 @RW8+d8	MOV R7, R8 @RW8+d8	MOV R7, R8 @RW8+d8
+9	MOV R0, R9 @RW9+d8	MOV R0, R0 @RW9+d8	MOV R1, R9 @RW9+d8	MOV R1, R9 @RW9+d8	MOV R2, R9 @RW9+d8	MOV R2, R9 @RW9+d8	MOV R3, R9 @RW9+d8	MOV R3, R9 @RW9+d8	MOV R4, R9 @RW9+d8	MOV R4, R9 @RW9+d8	MOV R5, R9 @RW9+d8	MOV R5, R9 @RW9+d8	MOV R6, R9 @RW9+d8	MOV R6, R9 @RW9+d8	MOV R7, R9 @RW9+d8	MOV R7, R9 @RW9+d8
+A	MOV R0, R10 @RW10+d8	MOV R0, R0 @RW10+d8	MOV R1, R10 @RW10+d8	MOV R1, R10 @RW10+d8	MOV R2, R10 @RW10+d8	MOV R2, R10 @RW10+d8	MOV R3, R10 @RW10+d8	MOV R3, R10 @RW10+d8	MOV R4, R10 @RW10+d8	MOV R4, R10 @RW10+d8	MOV R5, R10 @RW10+d8	MOV R5, R10 @RW10+d8	MOV R6, R10 @RW10+d8	MOV R6, R10 @RW10+d8	MOV R7, R10 @RW10+d8	MOV R7, R10 @RW10+d8
+B	MOV R0, R11 @RW11+d8	MOV R0, R0 @RW11+d8	MOV R1, R11 @RW11+d8	MOV R1, R11 @RW11+d8	MOV R2, R11 @RW11+d8	MOV R2, R11 @RW11+d8	MOV R3, R11 @RW11+d8	MOV R3, R11 @RW11+d8	MOV R4, R11 @RW11+d8	MOV R4, R11 @RW11+d8	MOV R5, R11 @RW11+d8	MOV R5, R11 @RW11+d8	MOV R6, R11 @RW11+d8	MOV R6, R11 @RW11+d8	MOV R7, R11 @RW11+d8	MOV R7, R11 @RW11+d8
+C	MOV R0, R12 @RW12+d8	MOV R0, R0 @RW12+d8	MOV R1, R12 @RW12+d8	MOV R1, R12 @RW12+d8	MOV R2, R12 @RW12+d8	MOV R2, R12 @RW12+d8	MOV R3, R12 @RW12+d8	MOV R3, R12 @RW12+d8	MOV R4, R12 @RW12+d8	MOV R4, R12 @RW12+d8	MOV R5, R12 @RW12+d8	MOV R5, R12 @RW12+d8	MOV R6, R12 @RW12+d8	MOV R6, R12 @RW12+d8	MOV R7, R12 @RW12+d8	MOV R7, R12 @RW12+d8
+D	MOV R0, R13 @RW13+d8	MOV R0, R0 @RW13+d8	MOV R1, R13 @RW13+d8	MOV R1, R13 @RW13+d8	MOV R2, R13 @RW13+d8	MOV R2, R13 @RW13+d8	MOV R3, R13 @RW13+d8	MOV R3, R13 @RW13+d8	MOV R4, R13 @RW13+d8	MOV R4, R13 @RW13+d8	MOV R5, R13 @RW13+d8	MOV R5, R13 @RW13+d8	MOV R6, R13 @RW13+d8	MOV R6, R13 @RW13+d8	MOV R7, R13 @RW13+d8	MOV R7, R13 @RW13+d8
+E	MOV R0, R14 @RW14+d8	MOV R0, R0 @RW14+d8	MOV R1, R14 @RW14+d8	MOV R1, R14 @RW14+d8	MOV R2, R14 @RW14+d8	MOV R2, R14 @RW14+d8	MOV R3, R14 @RW14+d8	MOV R3, R14 @RW14+d8	MOV R4, R14 @RW14+d8	MOV R4, R14 @RW14+d8	MOV R5, R14 @RW14+d8	MOV R5, R14 @RW14+d8	MOV R6, R14 @RW14+d8	MOV R6, R14 @RW14+d8	MOV R7, R14 @RW14+d8	MOV R7, R14 @RW14+d8
+F	MOV R0, R15 @RW15+d8	MOV R0, R0 @RW15+d8	MOV R1, R15 @RW15+d8	MOV R1, R15 @RW15+d8	MOV R2, R15 @RW15+d8	MOV R2, R15 @RW15+d8	MOV R3, R15 @RW15+d8	MOV R3, R15 @RW15+d8	MOV R4, R15 @RW15+d8	MOV R4, R15 @RW15+d8	MOV R5, R15 @RW15+d8	MOV R5, R15 @RW15+d8	MOV R6, R15 @RW15+d8	MOV R6, R15 @RW15+d8	MOV R7, R15 @RW15+d8	MOV R7, R15 @RW15+d8

**Table B.9-17 MOVW RWi, ea Instruction (First Byte = 7B<sub>H</sub>)**

[illegible]

Table B.9-18 MOV ea, Ri Instruction (First Byte = 7C<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0, @RW0+d8, R0	MOV R0, R1, @RW0+d8, R1	MOV R0, R2, @RW0+d8, R2	MOV R0, R3, @RW0+d8, R3	MOV R0, R4, @RW0+d8, R4	MOV R0, R5, @RW0+d8, R5	MOV R0, R6, @RW0+d8, R6	MOV R0, R7, @RW0+d8, R7	MOV R0, R8, @RW0+d8, R8	MOV R0, R9, @RW0+d8, R9	MOV R0, R10, @RW0+d8, R10	MOV R0, R11, @RW0+d8, R11	MOV R0, R12, @RW0+d8, R12	MOV R0, R13, @RW0+d8, R13	MOV R0, R14, @RW0+d8, R14	MOV R0, R15, @RW0+d8, R15
+1	MOV R1, R0, @RW1+d8, R0	MOV R1, R1, @RW1+d8, R1	MOV R1, R2, @RW1+d8, R2	MOV R1, R3, @RW1+d8, R3	MOV R1, R4, @RW1+d8, R4	MOV R1, R5, @RW1+d8, R5	MOV R1, R6, @RW1+d8, R6	MOV R1, R7, @RW1+d8, R7	MOV R1, R8, @RW1+d8, R8	MOV R1, R9, @RW1+d8, R9	MOV R1, R10, @RW1+d8, R10	MOV R1, R11, @RW1+d8, R11	MOV R1, R12, @RW1+d8, R12	MOV R1, R13, @RW1+d8, R13	MOV R1, R14, @RW1+d8, R14	MOV R1, R15, @RW1+d8, R15
+2	MOV R2, R0, @RW2+d8, R0	MOV R2, R1, @RW2+d8, R1	MOV R2, R2, @RW2+d8, R2	MOV R2, R3, @RW2+d8, R3	MOV R2, R4, @RW2+d8, R4	MOV R2, R5, @RW2+d8, R5	MOV R2, R6, @RW2+d8, R6	MOV R2, R7, @RW2+d8, R7	MOV R2, R8, @RW2+d8, R8	MOV R2, R9, @RW2+d8, R9	MOV R2, R10, @RW2+d8, R10	MOV R2, R11, @RW2+d8, R11	MOV R2, R12, @RW2+d8, R12	MOV R2, R13, @RW2+d8, R13	MOV R2, R14, @RW2+d8, R14	MOV R2, R15, @RW2+d8, R15
+3	MOV R3, R0, @RW3+d8, R0	MOV R3, R1, @RW3+d8, R1	MOV R3, R2, @RW3+d8, R2	MOV R3, R3, @RW3+d8, R3	MOV R3, R4, @RW3+d8, R4	MOV R3, R5, @RW3+d8, R5	MOV R3, R6, @RW3+d8, R6	MOV R3, R7, @RW3+d8, R7	MOV R3, R8, @RW3+d8, R8	MOV R3, R9, @RW3+d8, R9	MOV R3, R10, @RW3+d8, R10	MOV R3, R11, @RW3+d8, R11	MOV R3, R12, @RW3+d8, R12	MOV R3, R13, @RW3+d8, R13	MOV R3, R14, @RW3+d8, R14	MOV R3, R15, @RW3+d8, R15
+4	MOV R4, R0, @RW4+d8, R0	MOV R4, R1, @RW4+d8, R1	MOV R4, R2, @RW4+d8, R2	MOV R4, R3, @RW4+d8, R3	MOV R4, R4, @RW4+d8, R4	MOV R4, R5, @RW4+d8, R5	MOV R4, R6, @RW4+d8, R6	MOV R4, R7, @RW4+d8, R7	MOV R4, R8, @RW4+d8, R8	MOV R4, R9, @RW4+d8, R9	MOV R4, R10, @RW4+d8, R10	MOV R4, R11, @RW4+d8, R11	MOV R4, R12, @RW4+d8, R12	MOV R4, R13, @RW4+d8, R13	MOV R4, R14, @RW4+d8, R14	MOV R4, R15, @RW4+d8, R15
+5	MOV R5, R0, @RW5+d8, R0	MOV R5, R1, @RW5+d8, R1	MOV R5, R2, @RW5+d8, R2	MOV R5, R3, @RW5+d8, R3	MOV R5, R4, @RW5+d8, R4	MOV R5, R5, @RW5+d8, R5	MOV R5, R6, @RW5+d8, R6	MOV R5, R7, @RW5+d8, R7	MOV R5, R8, @RW5+d8, R8	MOV R5, R9, @RW5+d8, R9	MOV R5, R10, @RW5+d8, R10	MOV R5, R11, @RW5+d8, R11	MOV R5, R12, @RW5+d8, R12	MOV R5, R13, @RW5+d8, R13	MOV R5, R14, @RW5+d8, R14	MOV R5, R15, @RW5+d8, R15
+6	MOV R6, R0, @RW6+d8, R0	MOV R6, R1, @RW6+d8, R1	MOV R6, R2, @RW6+d8, R2	MOV R6, R3, @RW6+d8, R3	MOV R6, R4, @RW6+d8, R4	MOV R6, R5, @RW6+d8, R5	MOV R6, R6, @RW6+d8, R6	MOV R6, R7, @RW6+d8, R7	MOV R6, R8, @RW6+d8, R8	MOV R6, R9, @RW6+d8, R9	MOV R6, R10, @RW6+d8, R10	MOV R6, R11, @RW6+d8, R11	MOV R6, R12, @RW6+d8, R12	MOV R6, R13, @RW6+d8, R13	MOV R6, R14, @RW6+d8, R14	MOV R6, R15, @RW6+d8, R15
+7	MOV R7, R0, @RW7+d8, R0	MOV R7, R1, @RW7+d8, R1	MOV R7, R2, @RW7+d8, R2	MOV R7, R3, @RW7+d8, R3	MOV R7, R4, @RW7+d8, R4	MOV R7, R5, @RW7+d8, R5	MOV R7, R6, @RW7+d8, R6	MOV R7, R7, @RW7+d8, R7	MOV R7, R8, @RW7+d8, R8	MOV R7, R9, @RW7+d8, R9	MOV R7, R10, @RW7+d8, R10	MOV R7, R11, @RW7+d8, R11	MOV R7, R12, @RW7+d8, R12	MOV R7, R13, @RW7+d8, R13	MOV R7, R14, @RW7+d8, R14	MOV R7, R15, @RW7+d8, R15
+8	MOV @RW0, R0, @RW0+d16, R0	MOV @RW0, R1, @RW0+d16, R1	MOV @RW0, R2, @RW0+d16, R2	MOV @RW0, R3, @RW0+d16, R3	MOV @RW0, R4, @RW0+d16, R4	MOV @RW0, R5, @RW0+d16, R5	MOV @RW0, R6, @RW0+d16, R6	MOV @RW0, R7, @RW0+d16, R7	MOV @RW0, R8, @RW0+d16, R8	MOV @RW0, R9, @RW0+d16, R9	MOV @RW0, R10, @RW0+d16, R10	MOV @RW0, R11, @RW0+d16, R11	MOV @RW0, R12, @RW0+d16, R12	MOV @RW0, R13, @RW0+d16, R13	MOV @RW0, R14, @RW0+d16, R14	MOV @RW0, R15, @RW0+d16, R15
+9	MOV @RW1, R0, @RW1+d16, R0	MOV @RW1, R1, @RW1+d16, R1	MOV @RW1, R2, @RW1+d16, R2	MOV @RW1, R3, @RW1+d16, R3	MOV @RW1, R4, @RW1+d16, R4	MOV @RW1, R5, @RW1+d16, R5	MOV @RW1, R6, @RW1+d16, R6	MOV @RW1, R7, @RW1+d16, R7	MOV @RW1, R8, @RW1+d16, R8	MOV @RW1, R9, @RW1+d16, R9	MOV @RW1, R10, @RW1+d16, R10	MOV @RW1, R11, @RW1+d16, R11	MOV @RW1, R12, @RW1+d16, R12	MOV @RW1, R13, @RW1+d16, R13	MOV @RW1, R14, @RW1+d16, R14	MOV @RW1, R15, @RW1+d16, R15
+A	MOV @RW2, R0, @RW2+d16, R0	MOV @RW2, R1, @RW2+d16, R1	MOV @RW2, R2, @RW2+d16, R2	MOV @RW2, R3, @RW2+d16, R3	MOV @RW2, R4, @RW2+d16, R4	MOV @RW2, R5, @RW2+d16, R5	MOV @RW2, R6, @RW2+d16, R6	MOV @RW2, R7, @RW2+d16, R7	MOV @RW2, R8, @RW2+d16, R8	MOV @RW2, R9, @RW2+d16, R9	MOV @RW2, R10, @RW2+d16, R10	MOV @RW2, R11, @RW2+d16, R11	MOV @RW2, R12, @RW2+d16, R12	MOV @RW2, R13, @RW2+d16, R13	MOV @RW2, R14, @RW2+d16, R14	MOV @RW2, R15, @RW2+d16, R15
+B	MOV @RW3, R0, @RW3+d16, R0	MOV @RW3, R1, @RW3+d16, R1	MOV @RW3, R2, @RW3+d16, R2	MOV @RW3, R3, @RW3+d16, R3	MOV @RW3, R4, @RW3+d16, R4	MOV @RW3, R5, @RW3+d16, R5	MOV @RW3, R6, @RW3+d16, R6	MOV @RW3, R7, @RW3+d16, R7	MOV @RW3, R8, @RW3+d16, R8	MOV @RW3, R9, @RW3+d16, R9	MOV @RW3, R10, @RW3+d16, R10	MOV @RW3, R11, @RW3+d16, R11	MOV @RW3, R12, @RW3+d16, R12	MOV @RW3, R13, @RW3+d16, R13	MOV @RW3, R14, @RW3+d16, R14	MOV @RW3, R15, @RW3+d16, R15
+C	MOV @RW0+, R0, @RW0+RW7, R0	MOV @RW0+, R1, @RW0+RW7, R1	MOV @RW0+, R2, @RW0+RW7, R2	MOV @RW0+, R3, @RW0+RW7, R3	MOV @RW0+, R4, @RW0+RW7, R4	MOV @RW0+, R5, @RW0+RW7, R5	MOV @RW0+, R6, @RW0+RW7, R6	MOV @RW0+, R7, @RW0+RW7, R7	MOV @RW0+, R8, @RW0+RW7, R8	MOV @RW0+, R9, @RW0+RW7, R9	MOV @RW0+, R10, @RW0+RW7, R10	MOV @RW0+, R11, @RW0+RW7, R11	MOV @RW0+, R12, @RW0+RW7, R12	MOV @RW0+, R13, @RW0+RW7, R13	MOV @RW0+, R14, @RW0+RW7, R14	MOV @RW0+, R15, @RW0+RW7, R15
+D	MOV @RW1+, R0, @RW1+RW7, R0	MOV @RW1+, R1, @RW1+RW7, R1	MOV @RW1+, R2, @RW1+RW7, R2	MOV @RW1+, R3, @RW1+RW7, R3	MOV @RW1+, R4, @RW1+RW7, R4	MOV @RW1+, R5, @RW1+RW7, R5	MOV @RW1+, R6, @RW1+RW7, R6	MOV @RW1+, R7, @RW1+RW7, R7	MOV @RW1+, R8, @RW1+RW7, R8	MOV @RW1+, R9, @RW1+RW7, R9	MOV @RW1+, R10, @RW1+RW7, R10	MOV @RW1+, R11, @RW1+RW7, R11	MOV @RW1+, R12, @RW1+RW7, R12	MOV @RW1+, R13, @RW1+RW7, R13	MOV @RW1+, R14, @RW1+RW7, R14	MOV @RW1+, R15, @RW1+RW7, R15
+E	MOV @RW2+, R0, @PC+df6, R0	MOV @RW2+, R1, @PC+df6, R1	MOV @RW2+, R2, @PC+df6, R2	MOV @RW2+, R3, @PC+df6, R3	MOV @RW2+, R4, @PC+df6, R4	MOV @RW2+, R5, @PC+df6, R5	MOV @RW2+, R6, @PC+df6, R6	MOV @RW2+, R7, @PC+df6, R7	MOV @RW2+, R8, @PC+df6, R8	MOV @RW2+, R9, @PC+df6, R9	MOV @RW2+, R10, @PC+df6, R10	MOV @RW2+, R11, @PC+df6, R11	MOV @RW2+, R12, @PC+df6, R12	MOV @RW2+, R13, @PC+df6, R13	MOV @RW2+, R14, @PC+df6, R14	MOV @RW2+, R15, @PC+df6, R15
+F	MOV @RW3+, R0, addr16, R0	MOV @RW3+, R1, addr16, R1	MOV @RW3+, R2, addr16, R2	MOV @RW3+, R3, addr16, R3	MOV @RW3+, R4, addr16, R4	MOV @RW3+, R5, addr16, R5	MOV @RW3+, R6, addr16, R6	MOV @RW3+, R7, addr16, R7	MOV @RW3+, R8, addr16, R8	MOV @RW3+, R9, addr16, R9	MOV @RW3+, R10, addr16, R10	MOV @RW3+, R11, addr16, R11	MOV @RW3+, R12, addr16, R12	MOV @RW3+, R13, addr16, R13	MOV @RW3+, R14, addr16, R14	MOV @RW3+, R15, addr16, R15



**Table B.9-19 MOVW ea, Rwi Instruction (First Byte = 7D<sub>H</sub>)**

[illegible]



## 502

[illegible]

**Table B.9-21 XCHW RWi, ea Instruction (First Byte = 7F<sub>H</sub>)**

[illegible]



# INDEX

---

**The index follows on the next page.  
This is listed in alphabetic order.**

---

# Index

## Numerics

16-bit free run timer .....	185
16-bit free run timer (x 1) .....	180
16-bit free run timer register .....	185
16-bit free run timer, block diagram of .....	185
16-bit free run timer, operation of .....	201
16-bit free-running timer, timing for .....	202
16-bit I/O timer selection, register of entire .....	181
16-bit input capture, operation of .....	206
16-bit input/output timer register .....	183
16-bit output compare, operation of .....	203
16-bit output compare, timing for .....	204
2M-bit flash memory, feature of .....	396
2M-bit flash memory, sector configuration of .....	397
8 bits x 2 ch, 16 bits x 1 ch operation .....	256
8/16-bit PPG interrupt .....	229
8/16-bit PPG operating mode .....	223
8/16-bit PPG output operation .....	224
8/16-bit PPG, block diagram of .....	211
8/16-bit PPG, controlling pulse pin output of .....	227
8/16-bit PPG, initial value of each hardware component in .....	230
8/16-bit PPG, operation of .....	222
8/16-bit PPG, overview of .....	210
8/16-bit PPG, register of .....	213
8/16-bit PPG, selecting count clock for .....	226
8/16-bit PPG, timing of writing reload register in .....	228
8/16-bit up/down counter/timer, block diagram of .....	234
8/16-bit up/down counter/timer, count mode selection for .....	247
8/16-bit up/down counter/timer, function of .....	232
8/16-bit up/down counter/timer, register of .....	236
8/16-bit up/down counter/timer, reload function and compare function .....	250
8/16-bit up/down counter/timer, simultaneous activation of reload/compare function .....	252

## A

A/D converter, block diagram of .....	275
A/D converter, note on using .....	293
A/D converter, overview of .....	274
A/D converter, register of .....	276
access mode .....	126

access to many-byte length data .....	28
accessing low-power consumption mode control register (LPMCR) .....	96
accumulator (A) .....	32
acknowledgement .....	364
activating EI <sup>2</sup> OS in continuous mode, example of .....	289
activating EI <sup>2</sup> OS in single mode, example of .....	287
activating EI <sup>2</sup> OS in stop mode, example of .....	291
ADCR2 and ADCR1 .....	282
ADCS1 and ADCS2 .....	277
address match detection function, block diagram of .....	382
address match detection function, note on .....	386
address match detection function, operation of ...	386
address match detection function, register of .....	383
address match detection function, system configuration of .....	387
address register (IADR) .....	362
addressing .....	364
Addressing .....	445
addressing in write data cycle .....	411
addressing using bank method .....	26
addressing using linear method .....	25
allocating many-byte length data in memory space .....	28
analog input enable register (ADER) .....	160
application of UART (example of system configuration in mode 1) .....	329
arbitration .....	364
ARSR .....	134
asynchronous mode, receiving operation of .....	322
asynchronous mode, sending operation of .....	322
asynchronous mode, transfer data format of .....	322
automatic ready function selection register (ARSR) .....	134

## B

bank method, addressing using .....	26
bank register .....	40
bank select prefix .....	43
basic configuration of MB90F574/A serial programming connection .....	424
baud rate generator, dedicated .....	318
block diagram of MB90570 series .....	6

buffer address pointer (BAP) .....	76
bus control register (IBCR) .....	356
bus control signal selection register (ECSR) .....	137
bus error.....	365
bus mode .....	126
bus status register (IBSR) .....	353

## C

Calculating the Execution Cycle Count.....	462
CCRH0.....	241
CCRH1.....	243
CCRL0/1 .....	245
chip erase or sector erase operation, at .....	404
chip select control register (CSCR0 to CSCR6) ..	371
chip select function, block diagram of .....	370
chip select function, decode address space of ....	373
chip select function, operation of .....	372
chip select function, overview of .....	370
clearing watchdog counter .....	172
CLK synchronous mode being used, setting for control register when .....	324
CLK synchronous mode, starting communication in.....	324
CLK synchronous mode, terminating communication in.....	324
CLK synchronous mode, transfer data format of .	323
clock control register (ICCR).....	359
clock generator .....	84
clock generator, note for .....	84
clock mode, external shift .....	341
clock mode, internal shift .....	341
clock monitor, block diagram of .....	378
clock output enable register (CLKR) .....	379
clock selection register (CKSCR).....	97
clock selection, status transition for .....	100
clock supply map .....	85
command sequence table .....	401
common register bank prefix (CMR) .....	44
communication flow chart of UART.....	329
communication prescaler control register (CDCR).....	316
compare detection flag.....	255
competition among SCC, MSS, and INT bits.....	358
condition code register (CCR).....	36
condition of peripheral circuits connected externally when DTP being used .....	267
continuous mode.....	284
control register, timebase timer (TBTC).....	164
control register, watch timer (WTC) .....	176
control register, watchdog timer (WDTC) .....	170
control signal, external memory access .....	140
control status register .....	277
control status register, output compare .....	193
controlling pulse pin output of 8/16-bit PPG .....	227
conversion data protection function.....	294
conversion using EI <sup>2</sup> OS .....	286
count clear/gate function .....	254
count direction flag, count direction change flag .....	255
count mode selection for 8/16-bit up/down counter/timer .....	247
counter control register high ch.0 (CCRH0) .....	241
counter control register high ch.1 (CCRH1) .....	243
counter control register low ch.0/1 (CCRL0/1) .....	245
counter status register 0/1 (CSR0/1) .....	239
CSR0/1 .....	239

## D

D/A converter register.....	298
D/A converter register (DACR0 and DACR1) .....	300
D/A converter, block diagram of .....	299
D/A converter, operation of.....	302
D/A data register (DADR0 and DADR1).....	300
data counter (DCT) .....	74
data polling flag state, transition of .....	404
data register.....	282
data register (IDAR).....	363
data write, note on .....	411
decode address space of chip select function .....	373
dedicated baud rate generator .....	318
dedicated register .....	30
delayed interrupt requesting module, block diagram of .....	270
delayed interrupt requesting module, operation of .....	271
delayed interrupt requesting module, precaution for using .....	271
delayed interrupt requesting module, register of .....	270
Description of Instruction Presentation Items and Symbols.....	465
direct page register (DPR).....	39
Direct Addressing .....	447
DIV A, Ri and DIVW A, RWi instruction, note on using .....	48, 49
DTP and external interrupt .....	258
DTP operation .....	264
DTP request and external interrupt request, switching between .....	265

## INDEX

DTP/external interrupt, block diagram of .....	258
DTP/external interrupt, operation of .....	264
DTP/external interrupt, operation procedure of .....	267
DTP/external interrupt, register of .....	259
DTP/interrupt enable register (ENIR) .....	260
DTP/interrupt source register (EIRR) .....	261

## E

E2PROM memory map .....	387
ECSR .....	137
Effective Address Field .....	446, 464
EI <sup>2</sup> OS , conversion using .....	286
EI <sup>2</sup> OS in continuous mode, example of activating .....	289
EI <sup>2</sup> OS in single mode, example of activating .....	287
EI <sup>2</sup> OS in stop mode, example of activating .....	291
EI <sup>2</sup> OS interrupt processing, overview of .....	67
EI <sup>2</sup> OS status register (ISCS) .....	75
EI <sup>2</sup> OS, execution time of .....	79
EI <sup>2</sup> OS, operation of .....	68
EI <sup>2</sup> OS, procedure for .....	77
EI <sup>2</sup> OS, procedure for using .....	78
EI <sup>2</sup> OS, structure of .....	69
entire 16-bit I/O timer, block diagram of .....	182
erasing all chips .....	413
erasing data (erasing all chips) .....	413
erasing data (erasing sector) .....	414
erasing sector .....	414
exception occurrence due to execution of undefined instruction .....	82
Execution Cycle Count .....	461
explanation of operation .....	284
extended intelligent I/O service (EI <sup>2</sup> OS) .....	67, 330
extended intelligent I/O service descriptor (ISD) .....	73
extended intelligent I/O service, execution time of .....	79
extended intelligent I/O service, operation of .....	68
extended intelligent I/O service, procedure for .....	77
extended intelligent I/O service, procedure for using .....	78
extended intelligent I/O service, structure of .....	69
extended serial I/O interface, block diagram of .....	333
extended serial I/O interface, interrupt function of .....	347
extended serial I/O interface, operation of .....	340
extended serial I/O interface, overview of .....	332
extended serial I/O interface, register of .....	334

external address output control register (HACR) .....	136
external bus pin control circuit .....	132
external clock .....	320
external memory access .....	132
external memory access control signal .....	140
external memory access register, configuration of .....	133
external memory access, block diagram of .....	133
external shift clock mode .....	341

## F

F <sup>2</sup> MC-16LX Instruction List .....	468
flag change suppression prefix (NCC) .....	44
flag of UART .....	325
flag, compare detection .....	255
flag, count direction/count direction change .....	255
flash memory control status register (FMCS) .....	398
flash memory program, example of .....	418
flash memory register .....	396
flash memory write procedure .....	411
flash memory write/erase method .....	396
flash memory write/erase, detailed explanation of .....	409
flash memory, writing data to .....	411
FMCS .....	398
FPT-120P-M05, package dimension of .....	8
FPT-120P-M13, package dimension of .....	9
FPT-120P-M21, package dimension of .....	10
function of 8/16-bit up/down counter/timer .....	232

## G

general-purpose register .....	41
generator, clock .....	84

## H

HACR .....	136
handling device, note on .....	20
hardware interrupt mechanism .....	57
hardware interrupt operation .....	60
hardware interrupt operation flowchart .....	63
hardware interrupt, note on .....	59
hardware interrupt, overview of .....	57
hardware interrupt, processing time for .....	61
hardware sequence flag .....	402
hardware standby mode, releasing .....	114
hardware standby mode, transition to .....	114
hold function .....	145

**I**

I/O circuit type .....	17
I/O map .....	436
I/O port register, configuration of .....	150
I/O port, overview of .....	148
I/O register address pointer (IOA) .....	74
I/O service descriptor, extended intelligent (ISD) ...	73
I <sup>2</sup> C Interface, block diagram of .....	351
I <sup>2</sup> C Interface, feature of .....	350
I <sup>2</sup> C interface, operation flow of .....	366
I <sup>2</sup> C Interface, register of .....	352
IADR .....	362
IBCR .....	356
IBSR .....	353
ICCR .....	359
ICR .....	70
IDAR .....	363
Indirect Addressing .....	453
initial status .....	387
initial value of each hardware component in 8/16-bit PPG .....	230
input capture .....	196
input capture (x 2) .....	180
input capture control status register (ICS01) .....	199
input capture input timing .....	207
input capture register (IPCP0, IPCP1) .....	198
input capture register, entire .....	196
input capture, block diagram of entire .....	197
input resistance register (PDR), note on .....	158
input resistor register (RDR) .....	158
input resistor register, block diagram of .....	158
Instruction Types .....	444
INT9 interrupt .....	388
intelligent I/O service (EI <sup>2</sup> OS), extended .....	330
intermittent CPU operation function .....	91
internal shift clock mode .....	341
internal timer .....	319
interrupt control register (ICR) .....	70
interrupt function of extended serial I/O interface .....	347
interrupt level mask register (ILM) .....	37
interrupt request during writing in internal resource area .....	58
interrupt source .....	53
interrupt source of UART .....	325
interrupt suppress instruction .....	58
interrupt vector .....	55
interrupt, overview of .....	52
interrupt, software .....	65

interrupt/hold suppression instruction and prefix code .....	46
interval interrupt function of timebase timer .....	166
interval interrupt function of watch timer .....	178
ISCS .....	75

**L**

linear method, addressing using .....	25
low-power consumption control circuit, block diagram of .....	93
low-power consumption control circuit, operating mode of .....	90
low-power consumption control circuit, register in .....	94
low-power consumption mode .....	104
low-power consumption mode control register (LPMCR) .....	95
low-power consumption mode control register (LPMCR), accessing .....	96
low-power consumption mode, operating state in .....	105
low-power consumption mode, status transition diagram of .....	119
low-power consumption mode, status transition in .....	115

**M**

machine clock, switching of .....	91
main clock, setting of oscillation stabilization time for .....	91
many-byte length data in memory space, allocating .....	28
many-byte length data, access to .....	28
MB90570 series, block diagram .....	6
MB90570 series, feature of .....	2
MB90570 series, pin assignment .....	7
MB90F574/A serial write connection, basic configuration of .....	424
memory access mode .....	126
memory space .....	24
memory space for mode .....	129
minimum connection to flash microcomputer programmer (power supplied from the programmer), example of .....	433
minimum connection to flash microcomputer programmer (User Power Supply Used), example of .....	431
mode data .....	128
mode pin .....	127
multiple interrupts .....	58



## INDEX

<b>O</b>	
operating mode .....	126
operating mode of low-power consumption control circuit .....	90
operating state in low-power consumption mode .....	105
operation after reset is released .....	88
operation flow of I <sup>2</sup> C interface .....	366
operation of chip select function .....	372
operation of delayed interrupting requesting module .....	271
operation procedure of DTP/external interrupt .....	267
operation, explanation of .....	284
operation, hardware interrupt .....	60
oscillation clock frequency and serial clock input frequency .....	426
oscillation stabilization time for main clock, setting of .....	91
output compare .....	190
output compare (x 4) .....	180
output compare control status register (OCS0 to OCS3) .....	193
output compare register .....	190
output compare register (OCCP0 to OCCP3) .....	192
output compare, block diagram of .....	191
output control register, external address .....	136
output pin register (ODR) .....	157
output pin register (ODR), block diagram of .....	157
output pin register (ODR), note on .....	157
overview of I/O port .....	148
overview of the interrupt .....	52
<b>P</b>	
package dimension of FPT-120P-M05 .....	8
package dimension of FPT-120P-M13 .....	9
package dimension of FPT-120P-M21 .....	10
peripheral circuits connected externally when DTP being used, condition of .....	267
pin assignment of MB90570 series .....	7
pin description .....	11
PLL clock multiplication function .....	92
port data register (PDR) .....	152
port direction register (DDR) .....	155
port output being set by PDR register, operation of port used as resource when ...	149
port used as resource when port output being set by PDR register, operation of .....	149
PPG0 and PPG1 output pin control register (PPGOE) .....	219
PPG0 operating mode control register (PPGC0) .....	214
PPG1 operating mode control register (PPGC1) .....	216
precaution for using delayed interrupting requesting module .....	271
Precautions .....	330
prefix code and interrupt/hold suppression instruction .....	46
prefix code being consecutive .....	47
preventing watchdog timer reset .....	172
processing time for hardware interrupt .....	61
processor status (PS) .....	36
product lineup .....	5
program address detection control status register (PACSR) .....	385
program address detection register (PADR0 and PADR1) .....	384
program counter (PC) .....	38
program error occurs .....	388
program patch processing, example and flow of .....	389
pseudo watch mode, releasing .....	109
pseudo watch mode, transition to .....	108
putting flash memory into read/reset state .....	410
<b>R</b>	
R/W wait state, serial data register .....	342
RCR0/1 .....	238
read/reset state, putting flash memory into .....	410
ready function .....	143
receiving operation of asynchronous (start-stop synchronous) mode .....	322
recommended setting, example of .....	130
register .....	276
register bank .....	42
register bank pointer (RP) .....	37
register in low-power consumption control circuit .....	94
register of 8/16-bit up/down counter/timer .....	236
register of address match detection function .....	383
register of delayed interrupt requesting module ...	270
register of entire 16-bit I/O timer section .....	181
register of I <sup>2</sup> C interface .....	352
register of ROM mirror function selection module (ROMM) .....	393
register saved on stack when interrupt occurs .....	59
register, dedicated .....	30
relationship between reload value and pulse width .....	224

releasing hardware standby mode .....	114
releasing pseudo watch mode .....	109
releasing sleep mode .....	107
releasing stop mode .....	112
releasing watch mode .....	111
reload function and compare function of 8/16-bit up/down counter/timer .....	250
reload register (PRL and PRLH) .....	221
reload value and pulse width, relationship between .....	224
reload/compare register 0/1 (RCR0/1) .....	238
request level setting register (ELVR) .....	262
reset being released .....	88
reset factor .....	86
reset released, operation after .....	88
reset sequence .....	388
reset source .....	86
restarting sector erase operation .....	417
return from standby state .....	267
ROM mirror function selection module (ROMM), register of .....	393
ROM mirror function selection module, block diagram of .....	392
<b>S</b>	
sample use procedure, flow chart of .....	64
SCC, MSS, and INT bit, competition among .....	358
sector configuration of 2M-bit flash memory .....	397
sector erase operation, at .....	408
sector erase procedure .....	414
sector erase temporarily, stopping .....	416
sector erase temporary stop, at .....	404, 406
sector erase timer flag state, transition of .....	408
sector specification method .....	414
selecting count clock for 8/16-bit PPG .....	226
selection register, automatic ready function (ARSR) .....	134
selection register, bus control signal (ECSR) .....	137
sending operation of asynchronous (start-stop synchronous) mode .....	322
serial control register (SCR) .....	309
serial data I/O shift timing .....	346
serial data register R/W wait state .....	342
serial input data register (SIDR)/serial output data register (SODR), configuration of .....	312
serial mode control status register (SMCS) .....	335
serial mode register (SMR) .....	307
serial programming connection (power supplied from the programmer), example of .....	429
serial programming connection (user power supply used), example of .....	427
serial shift data register (SDR) .....	339
serial status register (SSR) .....	313
setting of oscillation stabilization time for main clock .....	91
settings for control register when CLK synchronous mode being used .....	324
shift operation start/stop timing .....	344
SIDR/SODR .....	312
simultaneous activation of reload/compare function of 8/16-bit up/down counter/timer .....	252
single mode .....	284
sleep mode, releasing .....	107
sleep mode, transition to .....	107
software interrupt .....	65
software interrupt mechanism .....	65
software interrupt operation .....	65
software interrupt, note on .....	66
software interrupt, overview of .....	65
specification of several sectors, note on .....	414
stack pointer, user (USP) and system (SSP) .....	34
standby state, return from .....	267
start condition .....	364
start/stop timing, shift operation .....	344
starting communication in CLK synchronous mode .....	324
starting watchdog timer .....	172
status transition diagram of low-power consumption mode .....	119
status transition for clock selection .....	100
status transition in low-power consumption mode .....	115
stop condition .....	364
stop mode .....	285
stop mode, releasing .....	112
stop mode, transition to .....	112
STOP state .....	342
stopped state .....	342
stopping sector erase temporarily .....	416
stopping watchdog .....	172
Structure of Instruction Map .....	482
switching between DTP request and external interrupt request .....	265
switching of machine clock .....	91
system configuration in mode 1, example of .....	329
system configuration of address match detection function .....	387

## INDEX

<b>T</b>	
TCCS .....	187
TCDT .....	186
terminating communication in CLK synchronous mode .....	324
Timebase .....	166
timebase counter .....	166
timebase timer control register (TBTC) .....	164
timebase timer register, configuration of .....	162
timebase timer, block diagram of .....	163
timebase timer, interval interrupt function of .....	166
timer counter control status register (TCCS) .....	187
timer counter data register (TCDT) .....	186
timer register, 16-bit input/output .....	183
timing for 16-bit free-running timer .....	202
timing for 16-bit output compare .....	204
timing limit excess flag state, transition of .....	407
timing of writing reload register in 8/16-bit PPG .....	228
timing to set interrupt and flag of UART .....	326
toggle bit flag state, transition of .....	406
transfer data format of asynchronous (start-stop synchronous) mode .....	322
transfer data format of CLK synchronous mode .....	323
transfer state .....	342
transfer time one operation .....	79
transition of data polling flag state .....	404
transition of sector erase timer flag state .....	408
transition of timing limit excess flag state .....	407
transition of toggle bit flag state .....	406
transition to hardware standby mode .....	114
transition to pseudo watch mode .....	108
transition to sleep mode .....	107
transition to stop mode .....	112
transition to watch mode .....	110
<b>U</b>	
UART operation mode .....	321
UART, application of .....	329
UART, block diagram of .....	305
UART, communication flow chart of .....	329
UART, feature of .....	304
UART, flag of .....	325
UART, interrupt source of .....	325
UART, precaution for using .....	330
UART, register of .....	306
UART, timing to set interrupt and flag of .....	326
UDCR, writing data to .....	254
UDCR0/1 .....	237
undefined instruction, exception occurrence due to execution of .....	82
up/down count register ch.0/1 (UDCR0/1) .....	237
user stack pointer (USP) and system stack pointer (SSP) .....	34
<b>W</b>	
watch counter .....	178
watch mode, releasing .....	111
watch mode, transition to .....	110
watch timer block diagram, block diagram of .....	175
watch timer control register (WTC) .....	176
watch timer control register, configuration of .....	174
watch timer, interval interrupt function of .....	178
watchdog timer control register (WDTC) .....	170
watchdog timer control register, configuration of .....	168
watchdog timer counter, clearing .....	172
watchdog timer reset, preventing .....	172
watchdog timer, block diagram of .....	169
watchdog timer, starting .....	172
watchdog timer, stopping .....	172
write operation, at .....	404
write/chip or sector erase operation, at .....	406
write/chip or sector erase, at .....	407
writing data to flash memory .....	411
writing data to UDCR .....	254

CM44-10102-8E

---

**FUJITSU MICROELECTRONICS • CONTROLLER MANUAL**

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

16-BIT MICROCONTROLLER

MB90570 series

HARDWARE MANUAL

---

July1 2008 the eighth edition

Published **FUJITSU MICROELECTRONICS LIMITED**

Edited Business & Media Promotion Dept.

---

