



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.

F²MCTM-16LX
16-BIT MICROCONTROLLER
MB90580C Series
HARDWARE MANUAL

F²MCTM-16LX

16-BIT MICROCONTROLLER

MB90580C Series

HARDWARE MANUAL

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

"Check Sheet" is seen at the following support page

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

PREFACE

■ Objective and Intended Reader

Thank you for your continued preference for Fujitsu semiconductor products.

The MB90580C Series was developed as general-purpose version of the F²MC-16LX Series, which is a proprietary 16-bit single-chip microcontroller that supports application-specific ICs (ASICs).

This manual is intended for engineers who design products using this semiconductor. Consult this manual for information on the functions and operations of the MB90580C Series.

■ Trademark

F²MC 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.

■ Structure of This Manual

This manual consists of the following 26 chapters:

Chapter 1 "OVERVIEW"

This chapter describes the configuration of the MB90580C series models and gives an outline of each model.

Chapter 2 "CPU"

This chapter describes the functions and operation of the CPU.

Chapter 3 "INTERRUPTS"

This chapter describes the features and operation of interrupts.

Chapter 4 "GENERATING AND RESETTING CLOCKS"

This chapter describes clock and reset functions and operations.

Chapter 5 "LOW-POWER CONSUMPTION CONTROL CIRCUIT"

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

Chapter 6 "MEMORY ACCESS MODES"

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

Chapter 7 "I/O PORTS"

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

Chapter 8 "TIME-BASED TIMER"

This chapter describes the functions and operations of the time-based timer.

Chapter 9 "WATCHDOG TIMER"

This chapter describes the functions and operations of the watchdog timer.

Chapter 10 "WATCH TIMER"

This chapter describes the functions and operations of the watch timer.

Chapter 11 "PWC TIMER"

This chapter describes the functions and operations of the PWC timer.

Chapter 12 "16-BIT I/O TIMER"

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

Chapter 13 "16-BIT RELOAD TIMER (WITH THE EVENT COUNT FUNCTION)"

This chapter gives an overview of the 16-bit reload timer (with the event count function) and explains its functions.

Chapter 14 "8/16-BIT PPG"

This chapter describes the function and operation of the 8/16-bit PPG.

Chapter 15 "DTP/EXTERNAL INTERRUPT CIRCUIT"

This chapter describes the function and operation of the DTP/external interrupt circuit.

Chapter 16 "DELAYED INTERRUPT GENERATING MODULE"

This chapter describes the function and operation of the delayed interrupt generating module.

Chapter 17 "A/D CONVERTER"

This chapter describes the functions and provides an overview of the A/D converter.

Chapter 18 "D/A CONVERTER"

This chapter explains the functions and operation of the D/A converter.

Chapter 19 "COMMUNICATION PRESCALER REGISTER"

This chapter describes the functions and overview of the communication prescaler register.

Chapter 20 "UART"

This chapter describes the UART functions and operations.

Chapter 21 "IEBusTM CONTROLLER"

This chapter describes the functions and operation of the IEBusTM controller.

Chapter 22 "CLOCK MONITOR FUNCTION"

This chapter describes the functions and operation of the clock monitor.

Chapter 23 "ADDRESS MATCH DETECTION FUNCTION"

This chapter describes the address match detection function and operation.

Chapter 24 "ROM MIRROR FUNCTION SELECTION MODULE"

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

Chapter 25 "1M-BIT FLASH MEMORY"

This chapter describes the functions and operations of the 1M-bit flash memory.

Chapter 26 "EXAMPLE OF MB90F583C/CA SERIAL PROGRAMMING CONNECTION"

This chapter provides examples of serial programming connection using the flash microcomputer programmer manufactured by YDC corporation.

Appendix

The appendix describes the I/O map and instructions.

- 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

CHAPTER 1 OVERVIEW	1
1.1 Features	2
1.2 Models Available	5
1.3 Block Diagram for MB90580C Series	6
1.4 Package Dimensions	7
1.5 Pin Layout	9
1.6 Pin Functions	11
1.7 I/O Circuit Formats	19
1.8 Precautions on Handling of Device	22
 CHAPTER 2 CPU	 27
2.1 Memory Space	28
2.2 Addressing	29
2.2.1 Allocating Multiple-byte Data in a Memory Space	32
2.3 Dedicated Registers	33
2.3.1 Accumulator (A)	35
2.3.2 User Stack Pointer (USP) and System Stack Pointer (SSP)	37
2.3.3 Processor Status (PS)	38
2.3.4 Program Counter (PC)	41
2.3.5 Direct Page Register (DPR)	42
2.3.6 Bank Registers	43
2.4 General-purpose Registers	44
2.5 Prefix Codes	46
2.6 Interrupt Suppression Instructions and Prefix Codes	49
2.7 Notes on Use of the DIV A, Ri and DIVW A, RWi Instructions	51
 CHAPTER 3 INTERRUPTS	 53
3.1 Overview of Interrupts	54
3.2 Interrupt Causes	55
3.3 Interrupt Vectors	57
3.4 Hardware Interrupts	59
3.4.1 Operation of Hardware Interrupts	62
3.4.2 Operating Flow for Hardware Interrupts	64
3.4.3 Example of Procedure for Using Hardware Interrupts	65
3.5 Software Interrupts	66
3.6 Expanded Intelligent I/O Service (EI ² OS)	68
3.6.1 Interrupt Control Register (ICR)	70
3.6.2 Expanded Intelligent I/O Service Descriptor (ISD)	73
3.6.3 Operation of the Expanded Intelligent I/O Service (EI ² OS)	77
3.6.4 Execution Time of the Expanded Intelligent I/O Service (EI ² OS)	79
3.7 Exceptions because of Executing Undefined Instructions	81
 CHAPTER 4 GENERATING AND RESETTING CLOCKS	 83
4.1 Clock Generator	84

4.2	Reset Causes	85
4.3	Operation after a Reset is Released	87
CHAPTER 5 LOW-POWER CONSUMPTION CONTROL CIRCUIT		91
5.1	Overview of the Low-power Consumption Control Circuit	92
5.2	Low-power Consumption Mode Control Register (LPMCR)	95
5.3	Clock Selection Register (CKSCR)	97
5.4	Operation of the Low-power Consumption Control Circuit	100
5.4.1	Sleep Mode	102
5.4.2	Pseudo Watch Mode	103
5.4.3	Watch Mode	104
5.4.4	Stop Mode	105
5.4.5	Hardware Standby Mode	106
5.5	Intermittent CPU Operation Function	107
5.6	Setting the Oscillation Stabilization Time for the Main Clock	108
5.7	Switching the Machine Clock	109
5.8	Status Transition	113
5.9	Status Transition Diagrams for Low Power-Consumption Modes	119
CHAPTER 6 MEMORY ACCESS MODES		127
6.1	Memory Access Mode Overview	128
6.1.1	Mode Pins	129
6.1.2	Mode Data	130
6.1.3	Memory Space for Each Bus Mode	131
6.2	External Memory Access (External Bus Pin Control Circuit)	134
6.2.1	Registers for External Memory Access (External Bus Pin Control Circuit)	135
6.2.2	Automatic Ready Function Selection Register (ARSR)	136
6.2.3	External Address Output Control Register (HACR)	138
6.2.4	Bus Control Signal Selection Register (ECSR)	139
6.3	Operation of the External Memory Access Control Signals	142
6.3.1	Ready Function	144
6.3.2	Hold Function	146
CHAPTER 7 I/O PORTS		147
7.1	I/O Port Overview	148
7.2	I/O Port Block Diagram	149
7.3	I/O Port Registers	151
7.3.1	Port Data Registers (PDRx)	153
7.3.2	Port Data Direction Registers (DDRx)	154
7.3.3	Port 4 Output Pin Register (ODR4)	156
7.3.4	Input Pull-up Resistor Setting Registers (RDR0, RDR1, and RDR6)	157
7.3.5	Port 5 Analog Input Enable Register (ADER)	158
CHAPTER 8 TIME-BASED TIMER		159
8.1	Overview of the Time-Based Timer	160
8.2	Time-Based Timer Control Register (TBTC)	162
8.3	Time-Based Timer Operations	164

CHAPTER 9 WATCHDOG TIMER	165
9.1 Overview of the Watchdog Timer	166
9.2 Watchdog Timer Control Register (WDTC)	168
9.3 Watchdog Timer Operations	171
CHAPTER 10 WATCH TIMER	173
10.1 Overview of the Watch Timer	174
10.2 Watch Timer Control Register (WTC)	176
10.3 Watch Timer Operations	178
CHAPTER 11 PWC TIMER	179
11.1 Overview of the PWC Timer	180
11.2 PWC Timer Block Diagram	181
11.3 PWC Timer Registers	182
11.3.1 PWC control status register (PWCSR)	184
11.3.2 PWC data buffer register (PWCR)	190
11.3.3 Division rate control register (DIVR)	191
11.3.4 PWC noise filter register (RNCR)	192
11.4 PWC Timer Operations	194
11.4.1 Count clock selection	197
11.4.2 Operation mode selection	198
11.4.3 Starting and stopping the timer and pulse-width measurement and clearing the timer	200
11.5 Details of Timer Mode Operation	202
11.6 Flowchart of Timer Mode Operation	204
11.7 Details of Pulse Width Measurement Mode Operation	205
11.7.1 Measurement mode and measurement operation	208
11.7.2 Flowchart of pulse-width measurement operation	211
11.8 Notes on Handling the PWC Timer	212
CHAPTER 12 16-BIT I/O TIMER	215
12.1 Overview of the 16-Bit I/O Timer	216
12.2 16-Bit I/O Timer Block Diagram	218
12.3 16-Bit I/O Timer Registers	219
12.3.1 16-bit Free-run Timer	221
12.3.2 Output Compare	225
12.3.3 Input Capture	229
12.4 16-Bit Free-Run Timer Operations	231
12.5 16-Bit Output Compare Operations	233
12.6 16-Bit Input Capture Operations	236
CHAPTER 13 16-BIT RELOAD TIMER (WITH THE EVENT COUNT FUNCTION)	239
13.1 Overview of the 16-Bit Reload Timer (with the Event Count Function)	240
13.2 Registers of the 16-Bit Reload Timer (with the Event Count Function)	241
13.2.1 Timer Control Status Register (TMCSR)	242
13.2.2 16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)	245
13.3 Clock Operations	246
13.4 Underflow Operation	247
13.5 I/O Pin Functions (for the Internal Clock Mode)	248

13.6 Counter Operation Statuses	250
CHAPTER 14 8/16-BIT PPG	251
14.1 Overview of the 8/16-Bit PPG	252
14.2 Block Diagrams of the 8/16-Bit PPG	253
14.3 Registers in the 8/16-Bit PPG	255
14.3.1 PPG0 Operation Mode Control Register (PPGC0)	256
14.3.2 PPG1 Operation Mode Control Register (PPGC1)	258
14.3.3 PPG0/1 Output Pin Control Register (PPGOE)	261
14.3.4 Reload Registers (PRLH/PRLH)	263
14.4 8/16-Bit PPG Operation	264
14.4.1 8/16-bit PPG Operation Modes	266
14.4.2 PPG Output Operation	267
14.4.3 Selecting a Count Clock	269
14.4.4 Controlling Pulse Output on Pins	270
14.4.5 Write Timing for the Reload Registers	271
CHAPTER 15 DTP/EXTERNAL INTERRUPT CIRCUIT	273
15.1 Overview of the DTP/External Interrupt Circuit	274
15.2 Registers in the DTP/External Interrupt Circuit	276
15.3 Operation of DTP/External Interrupt Circuit	278
15.4 Notes on Using the DTP/External Interrupt Circuit	281
CHAPTER 16 DELAYED INTERRUPT GENERATING MODULE	283
16.1 Overview of the Delayed Interrupt Generating Module	284
16.2 Operation of the Delayed Interrupt Generating Module	285
CHAPTER 17 A/D CONVERTER	287
17.1 Overview of the A/D Converter	288
17.2 A/D Converter Block Diagram	290
17.3 Registers of the A/D Converter	291
17.3.1 Control Status Registers (ADCS1 and ADCS2)	292
17.3.2 Data Register (ADCR1 and ADCR2)	297
17.4 Operation of A/D Converter	299
17.4.1 Example of EI ² OS Activation in Single Mode	301
17.4.2 Example of EI ² OS Activation in Successive Mode	303
17.4.3 Example of EI ² OS Activation in Pause Mode	305
17.5 Conversion Data Protection Function	307
CHAPTER 18 D/A CONVERTER	309
18.1 Overview of D/A Converter	310
18.2 D/A Converter Registers	312
18.3 Operation of D/A Converter	314
CHAPTER 19 COMMUNICATION PRESCALER REGISTER	315
19.1 Overview of Communication Prescaler Register	316
19.2 Operation of Communication Prescaler Register	318

CHAPTER 20	UART	319
20.1	Overview of UART	320
20.2	UART Block Diagram	321
20.3	UART Registers	322
20.3.1	Serial Mode Register (SMR0 to 4)	323
20.3.2	Serial Control Register (SCR0 to 4)	326
20.3.3	Serial Input Data Register (SIDR0 to 4) and Serial Output Data Register (SODR0 to 4)	329
20.3.4	Serial Status Register (SSR0 to 4)	330
20.4	UART Operations	333
20.4.1	UART Clock Selection	334
20.4.2	Asynchronous (Start-stop Synchronous) Mode	336
20.4.3	CLK-synchronous Mode	338
20.4.4	Occurrence of Interrupt and Flag Setting Timing	340
20.5	Application of UART (During Operation in Mode 1)	343
CHAPTER 21	IEBusTM CONTROLLER	345
21.1	Overview of IEBus TM Controller	346
21.2	Block Diagram for IEBus TM Controller	347
21.3	Registers of IEBus TM Controller	348
21.3.1	Local Address Set Registers (MAWH and HAWL)	351
21.3.2	Slave Address Set Registers (SAWH and SAWL)	352
21.3.3	Broadcast Control Bit Set Register (DCWR)	353
21.3.4	Text Length Bit Set Register (DEWR)	355
21.3.5	Command Register (Higher 8 Bits) (CMRH)	356
21.3.6	Command Register (Lower 8 Bits) (CMRL)	358
21.3.7	Status Register (Higher 8 Bits) (STRH)	361
21.3.8	Status Register (Lower 8 Bits) (STRL)	363
21.3.9	Lock Read Registers (LRRH and LRRL)	365
21.3.10	Master Address Read Registers (MARH and MARL)	366
21.3.11	Broadcast Control Bit Read Register (DCRR)	367
21.3.12	Text Length Bit Read Register (Lower 8 Bits) (DERR)	368
21.3.13	Read Data Buffer (RDB)	369
21.3.14	Write Data Buffer (WDB)	370
21.4	IEBus TM Transmission Control	371
21.5	IEBus TM Reception Control	374
21.6	Communication Control Status	376
21.7	Examples of the Flows of Main and Interrupt Processing Routines for IEBus TM Controller	379
21.7.1	Initialization Routine	380
21.7.2	Master Transmission Routine	381
21.7.3	Slave Data Transmission Routine	382
21.7.4	Master Reception Routine	383
21.8	IEBus TM Controller Operation at Transmission	386
21.9	IEBus TM Protocol Operation	390
21.10	Transmission Protocol	393
21.10.1	Header in Transmission Protocol Signal Format	394
21.10.2	Master Address Field in Transmission Protocol Signal Format	395
21.10.3	Slave Address Field in Transmission Protocol Signal Format	396
21.10.4	Control Field in Transmission Protocol Signal Format	397

21.10.5 Text Length Field	398
21.10.6 Data Field	399
21.10.7 Parity Bit	400
21.10.8 Acknowledgment Bit	401
21.11 Transmission Data	403
21.12 Bit Format	407
CHAPTER 22 CLOCK MONITOR FUNCTION	409
22.1 Overview of the Clock Monitor Functions	410
22.2 Clock Output Permission Register (CLKR)	411
CHAPTER 23 ADDRESS MATCH DETECTION FUNCTION	413
23.1 Overview of the Address Match Detection Function	414
23.2 Registers of the Address Match Detection Function	415
23.3 Operation of the Address Match Detection Function	417
23.4 Example of the Address Match Detection Function	418
CHAPTER 24 ROM MIRROR FUNCTION SELECTION MODULE	421
24.1 Overview of the ROM Mirror Function Selection Module	422
24.2 ROM Mirror Function Selection Register (ROMM)	423
CHAPTER 25 1M-BIT FLASH MEMORY	425
25.1 Overview of the 1M-Bit Flash Memory	426
25.2 Sector Configuration of the Flash Memory	427
25.3 Flash Memory Control Status Register (FMCS)	428
25.4 Activating the Automatic Algorithm of the Flash Memory	430
25.5 Confirming the Automatic Algorithm Execution Status	432
25.5.1 Data Polling Flag (DQ7)	434
25.5.2 Toggle Bit Flag (DQ6)	436
25.5.3 Timing Limit Excess Flag (DQ5)	438
25.5.4 Sector Deletion Timer Flag (DQ3)	439
25.5.5 Toggle Bit 2 Flag (DQ2)	440
25.6 Detailed Explanations of Flash Memory Writing and Deletion	442
25.6.1 Setting the Flash Memory in the Read or Reset Status	443
25.6.2 Writing Data in the Flash Memory	444
25.6.3 Deleting all Data Items from the Flash Memory (Chip Deletion)	446
25.6.4 Deleting any Data Item from the Flash Memory (Sector Deletion)	447
25.6.5 Temporarily Stopping the Sector Deletion from the Flash Memory	449
25.6.6 Restarting the Flash Memory Sector Deletion	450
25.7 Example of the 1M-Bit Flash Memory Program	451
CHAPTER 26 EXAMPLE OF MB90F583C/CA SERIAL PROGRAMMING CONNECTION	457
26.1 Basic Configuration of MB90F583C/CA Serial Programming Connection	458
26.2 Example of Serial Programming Connection (When User Power Supply Is Used)	462
26.3 Example of Serial Programming Connection (When Power is Supplied from a Programmer)	464
26.4 Example of Minimum Connection with the Flash Microcomputer Programmer (When User Power Supply is Used)	466

26.5	Example of Minimum Connection with the Flash Microcomputer Programmer (When Power is Supplied from a Programmer)	468
APPENDIX		471
APPENDIX A	I/O Map	472
APPENDIX B	Instructions	480
B.1	Instruction Types	481
B.2	Addressing	482
B.3	Direct Addressing	484
B.4	Indirect Addressing	490
B.5	Execution Cycle Count	498
B.6	Effective Address Field	501
B.7	How to Read the Instruction List	502
B.8	F ² MC-16LX Instruction List	505
B.9	Instruction Map	519
INDEX		541

Main changes in this edition

Page	Changes (For details, refer to main body.)
480 to 540	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 the configuration of the MB90580C series models and gives an outline of each model.

- 1.1 "Features"
- 1.2 "Models Available"
- 1.3 "Block Diagram for MB90580C Series"
- 1.4 "Package Dimensions"
- 1.5 "Pin Layout"
- 1.6 "Pin Functions"
- 1.7 "I/O Circuit Formats"
- 1.8 "Precautions on Handling of Device"

1.1 Features

The MB90580C series is a Fujitsu's general-purpose 16-bit microcontroller designed for the process control on products for private use and other equipment that require high-speed realtime processing. The MB90580C series inherits the AT architecture of the F²MC series as the instruction set and features additional instructions conforming to high-level language, extended addressing modes, reinforced multiplication and division instructions, and enhanced bit processing. Mounting of a 32-bit accumulator enables the MB90580C series to process long-word data. A built-in IEBusTM Controller simplifies communication with other devices. With these features, the MB90580C series matches component-type audio equipment and VTR systems.

■ Features of MB90580C Series

- **Minimum execution time**
62.5 ns at 4 MHz, 4 times multiplied (PLL clock multiplication system)
- **Maximum memory space**
 - 16 Mbyte
 - Linear bank access
- **Instruction set optimized for use of controllers**
 - Data types that can be handled: Bit, byte, word, and long word
 - Standard addressing modes: 23 modes
 - Reinforced high-accuracy arithmetic operations due to the use of 32-bit accumulator
 - Multiplication and division operations of signed numerics and extended RETI instructions
- **Instruction set conforming to high-level language (C language) and multitask**
 - Use of system stack pointer
 - Symmetry of instruction set and barrel shift instructions
- **Program patch function (2-address pointer)**
- **Increase in execution speed with 4-byte queue**
- **Reinforced interrupt functions**
 - Programmable setting of eight priority levels
 - Eight external interrupt inputs

○ **Data transfer functions independent of the CPU**

- Extended intelligent I/O service with up to 16 channels
- Eight DTP request inputs

○ **Internal ROM**

- Flash ROM: 128 Kbytes
- Mask ROM: 128 Kbytes (MB90583C/CA) or 64 Kbytes (MB90587C/CA)

○ **Internal RAM**

- Flash RAM: 6 Kbytes
- Mask ROM: 6 Kbytes (MB90583C/CA) or 4 Kbytes (MB90587C/CA)

○ **General-purpose ports**

Up to 77 ports (including 22 ports allowing input pull-up resistance setting and 8 ports allowing output open-drain setting)

○ **IEBus™ Controller***

Three data transfer rates available for selection:

- Mode 0: 3.9 kbps (16 bytes per frame)
- Mode 1: 17.0 kbps (32 bytes per frame)
- Mode 2: 26.0 kbps (128 bytes per frame)

*: IEBus™ Controller is a trademark of NEC Corporation.

○ **A/D converter (RC successive approximation type): 8 channels**

- Resolution: 8 or 10 bits
- Conversion time: 34.7 μs(minimum) at 12 MHz

○ **D/A converter: 2 channels**

- Resolution: 8 bits
- Setting time: 12.5 μs

○ **UART: 5 channels**

○ **8/16-bit PPG: 1 channel**

PPG with mode switching function to switch between 8 bits by 2 channels and 16 bits by 1 channel

○ **16-bit reload timer: 3 channels**

○ **16-bit PWC timer: 1 channel**

PWC timer with noise filter mounted and usable for pulse width counter

○ **16-bit I/O timers**

- Input capture timer: 4 channels

CHAPTER 1 OVERVIEW

- Output compare timer: 2 channels
- Free-run timer: 1 channel
- **Built-in clock generator**
- **Time base counter and watchdog timer: 18 bits**
- **Built-in clock monitor function**
- **Low power consumption modes**
 - Sleep
 - Stop
 - Hardware standby mode
 - Intermittent CPU operation mode
- **Package**
 - LQFP-100
 - QFP-100

1.2 Models Available

Table 1.2-1 "MB90580C Series Models" lists the available models of the MB90580C series. Functions other than ROM and RAM capacity and clocks are common to all models. The MB90587C/CA does not have the IEBus™ Controller.

■ Models Available

Table 1.2-1 MB90580C Series Models

Item	MB90583C	MB90583CA	MB90587C	MB90587CA	MB90F583C	MB90F583CA	MB90V580B
ROM capacity	Mask ROM 128 Kbytes	Mask ROM 128 Kbytes	Mask ROM 64 Kbytes	Mask ROM 64 Kbytes	FLASH ROM 128 Kbytes	FLASH ROM 128 Kbytes	-
RAM capacity	6 Kbytes	6 Kbytes	4 Kbytes	4 Kbytes	6 Kbytes	6 Kbytes	6 Kbytes
Clock	Two clocks system	One clock system	Two clocks system	One clock system	Two clocks system	One clock system	Two clocks system
IEBus™ controller	Available	Available	None	None	Available	Available	Available
Dedicated power supply for emulator*	-	-	-	-	-	-	None

*: Setting of DIP switch S2 for using the emulation pod MB2145-507. For details, see Section 2.7 "Dedicated Power Pin for Emulator" in the Hardware Manual for MB2145-507.

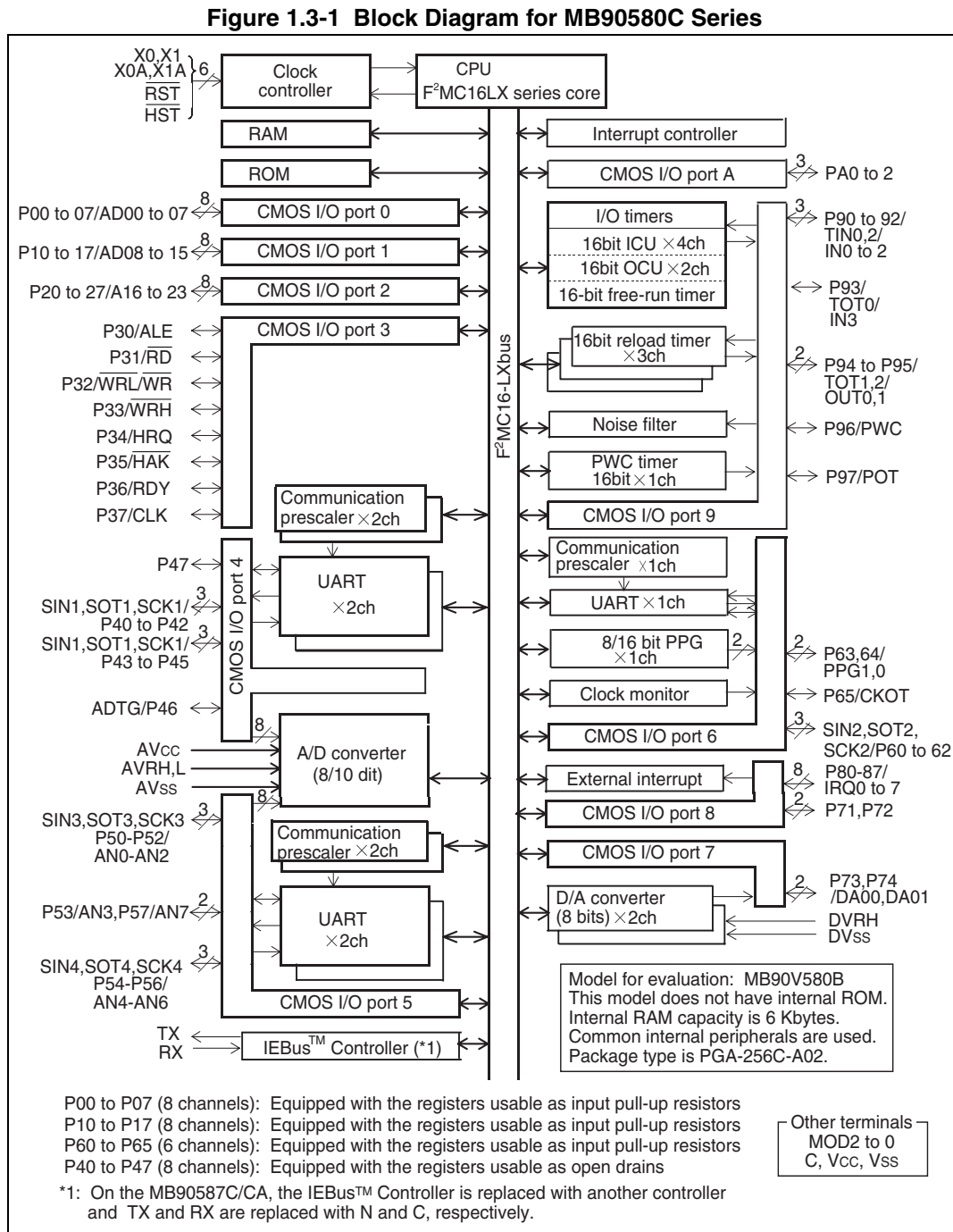
Note:

For the evaluation device, use the MB90V580B. Also, if the one clock system is used, equip X0A and X1A with clocks from the tool side.

1.3 Block Diagram for MB90580C Series

Figure 1.3-1 "Block Diagram for MB90580C Series" is a block diagram for the MB90580C series.

■ Block Diagram for MB90580C Series



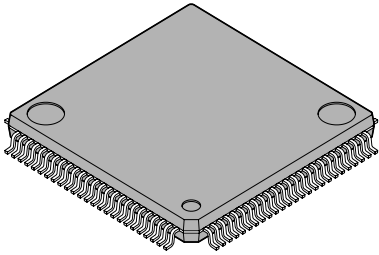
1.4 Package Dimensions

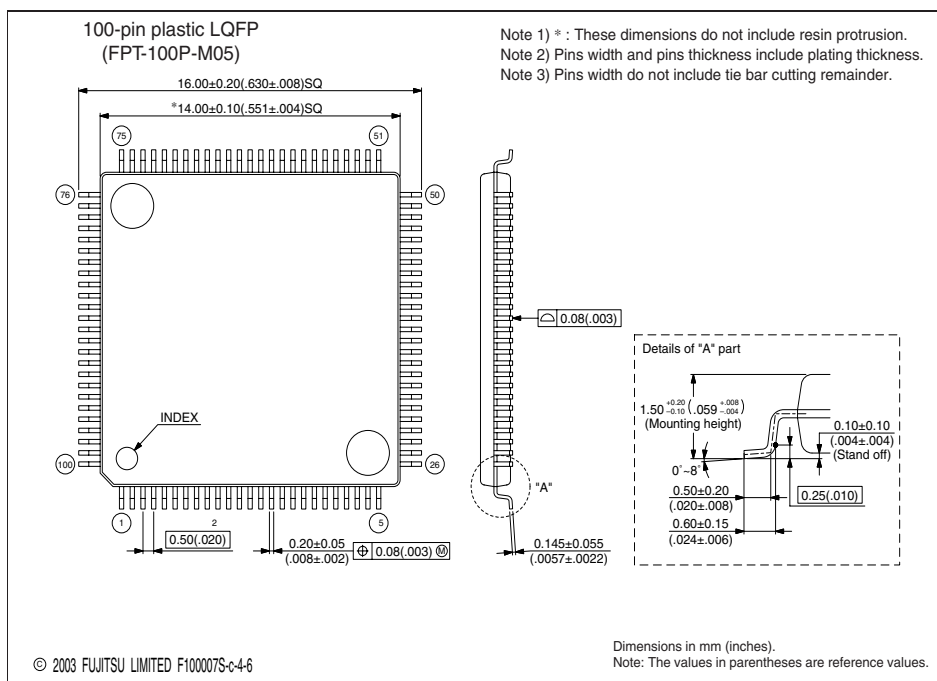
Figure 1.4-1 "Outside Dimensions of FPT-100P-M05 (LQFP-100)" shows the outside dimensions of the FPT-100P-M05 (LQFP-100) package. Figure 1.4-2 "Outside Dimensions of FPT-100P-M06 (QFP-100)" shows the outside dimensions of the FPT-100P-M06 (QFP-100) package.

Note that the dimensions shown below are reference dimensions. For formal dimensions of each package, contact us.

■ Outside Dimensions of FPT-100P-M05 Package

Figure 1.4-1 Outside Dimensions of FPT-100P-M05 (LQFP-100)

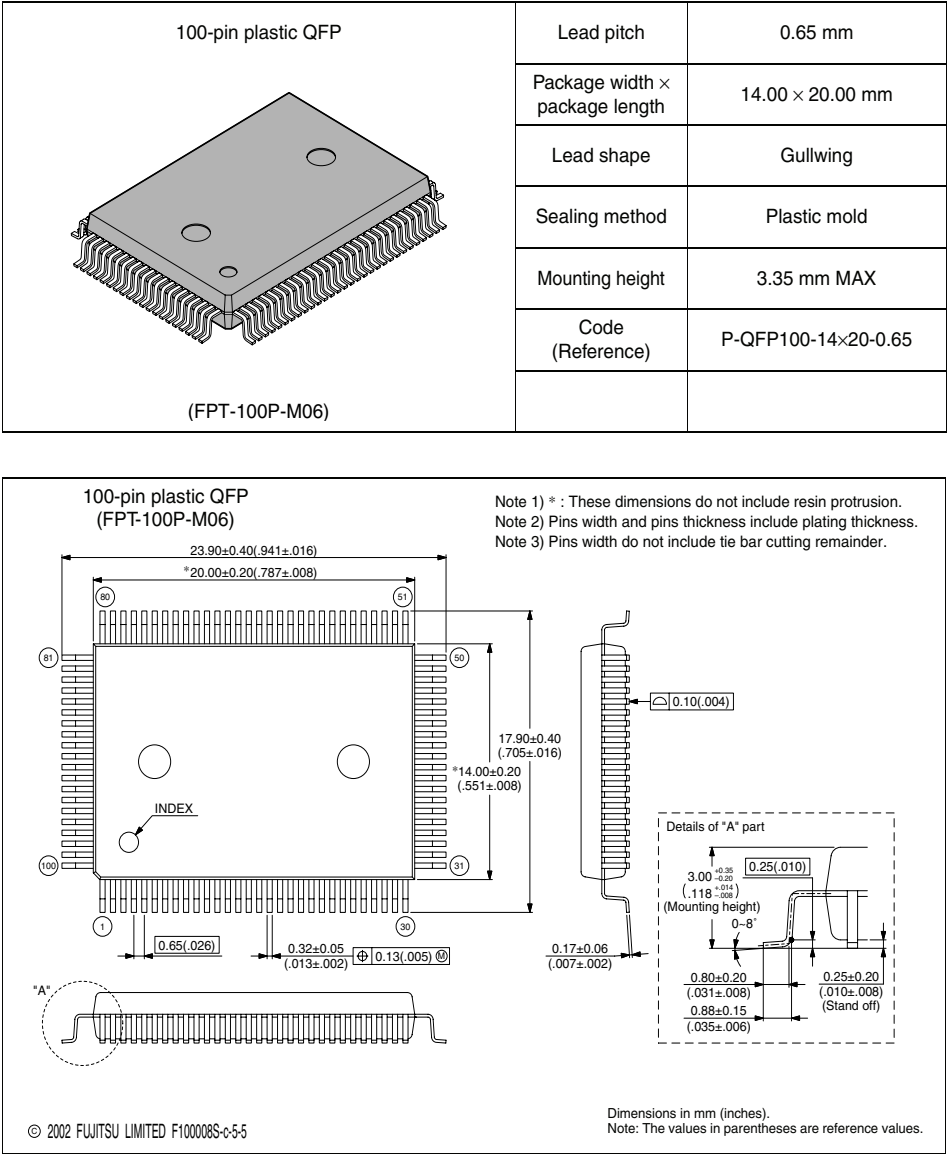
 <p>100-pin plastic LQFP</p> <p>(FPT-100P-M05)</p>	Lead pitch	0.50 mm
	Package width × package length	14.0 × 14.0 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	1.70 mm MAX
	Weight	0.65g
	Code (Reference)	P-LFQFP100-14×14-0.50



CHAPTER 1 OVERVIEW

■ Outside Dimensions of FPT-100P-M06 Package

Figure 1.4-2 Outside Dimensions of FPT-100P-M06 (QFP-100)

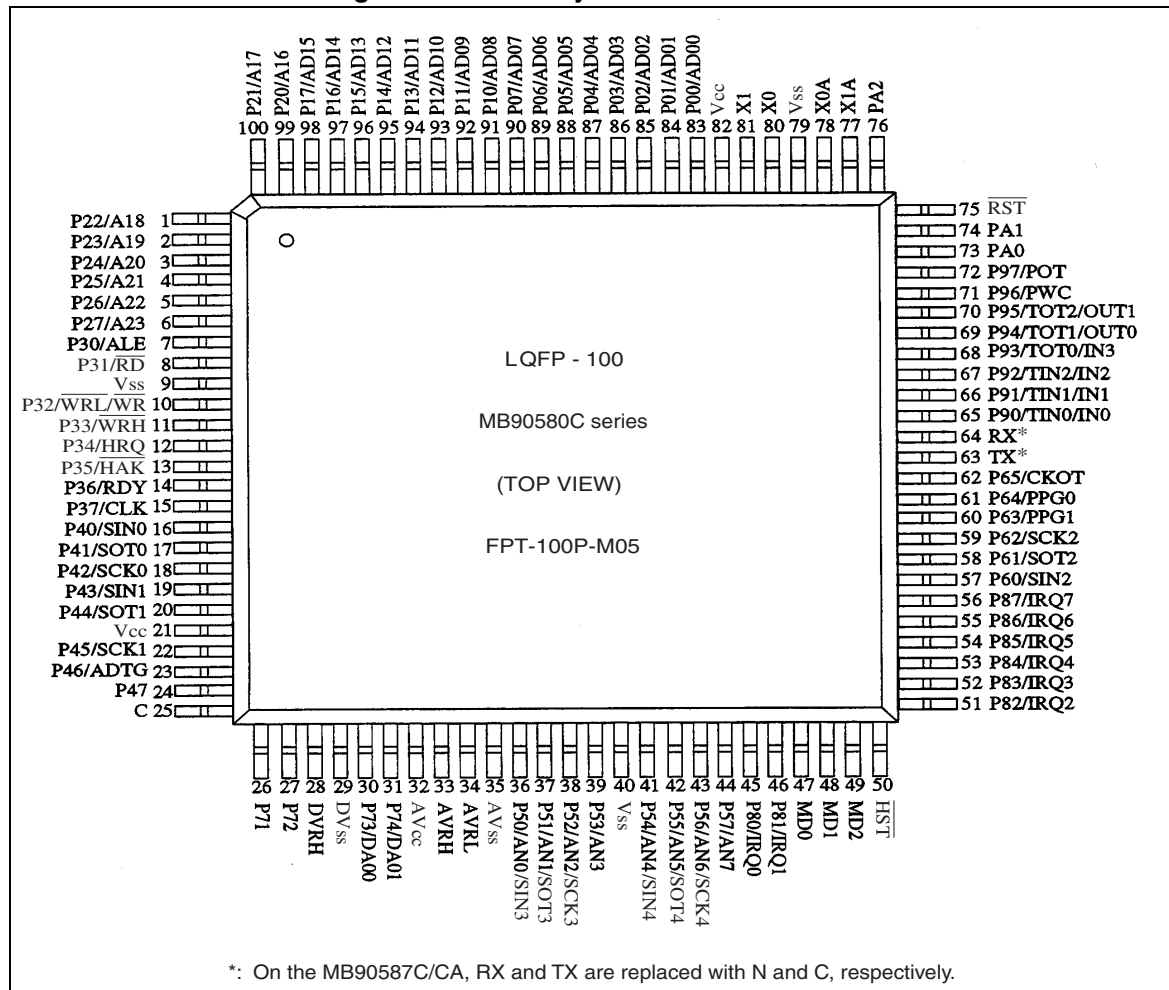


1.5 Pin Layout

Figure 1.5-1 "Pin Layout of FTP-100P-M05" shows the pin layout of the FTP-100P-M05. Figure 1.5-2 "Pin Layout of FPT-100P-M06" shows the pin layout of the FPT-100P-M06.

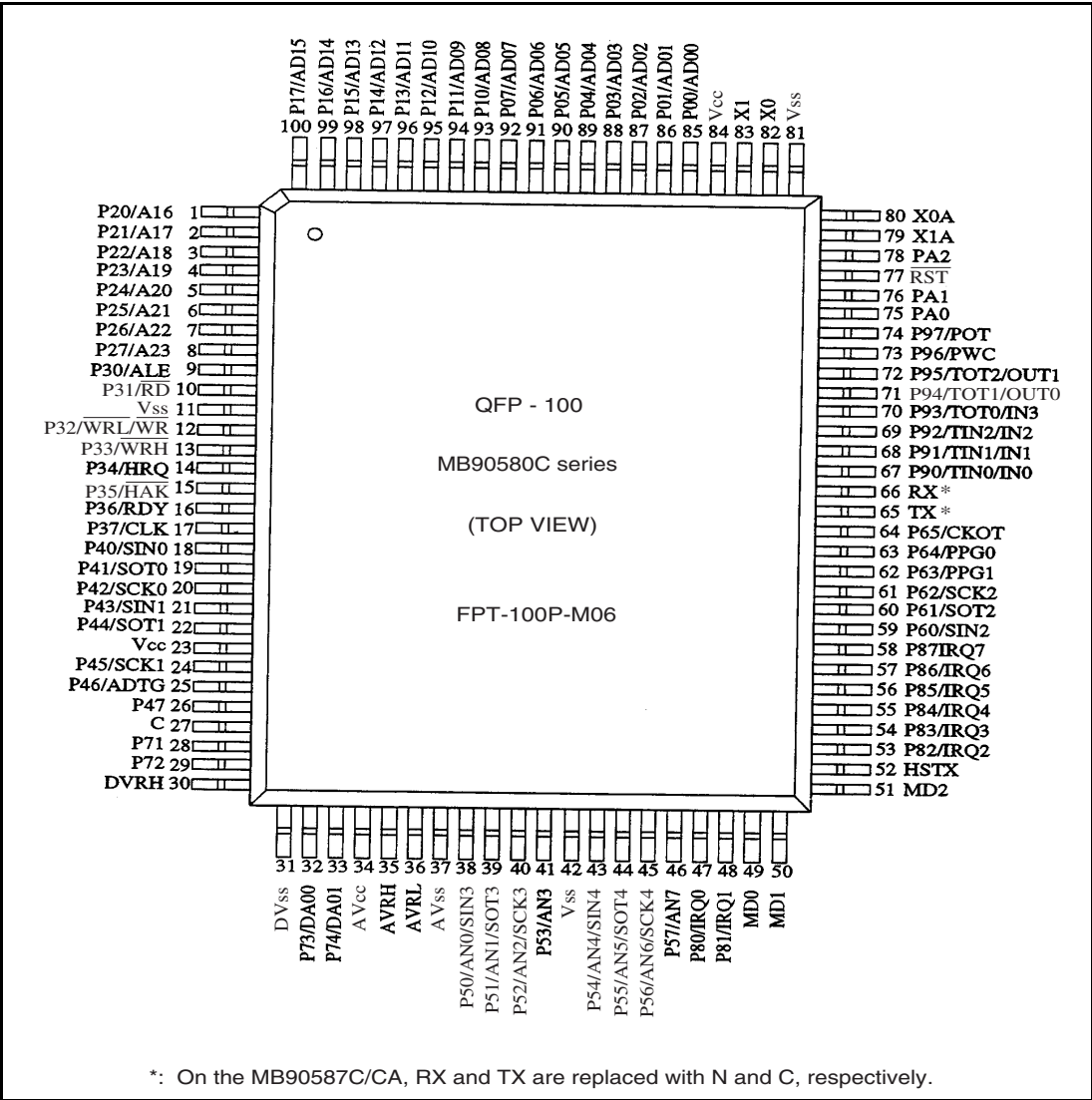
■ Pin Layout of FTP-100P-M05

Figure 1.5-1 Pin Layout of FTP-100P-M05



■ Pin Layout of FPT-100P-M06

Figure 1.5-2 Pin Layout of FPT-100P-M06



*: On the MB90587C/CA, RX and TX are replaced with N and C, respectively.

1.6 Pin Functions

Table 1.6-1 "Pin Functions" lists the pin functions of the MB90580C series. The alphabetic letters shown in the "Circuit format" column of Table 1.6-1 "Pin Functions" correspond to those shown in the "Classification" column of Table 1.7-1 "I/O Circuit Types".

■ Pin Functions

Table 1.6-1 Pin Functions

QFP	LQFP	Pin name	Circuit format	Function
82	80	X0	A	Oscillation output pin
83	81	X1	A	Oscillation output pin
52	50	$\overline{\text{HST}}$	C	Hardware standby input pin
77	75	$\overline{\text{RST}}$	B	Reset input pin
85 to 92	83 to 90	P00 to P07	D (CMOS/H)	General-purpose I/O ports Pull-up resistors can be assigned to these pins by the setting (RD07 to RD00 = 1) of pull-up resistor setting register (RDR0). (When these pins are set for output, the setting [D07 to D00 = 1] of register DDR0 is invalid.)
		AD00 to AD07		In external bus mode, these pins operate (as AD00 to AD07) to input or output the lower byte of data or output the lower byte of address.
93 to 100	91 to 98	P10 to P17	D (CMOS/H)	General-purpose I/O ports Pull-up resistors can be assigned to these pins by the setting (RD17 to RD10 = 1) of pull-up resistor setting register (RDR1). (When these pins are set for output, the setting [D17 to D10 = 1] of register DDR1 is invalid.)
		AD08 to AD15		In external bus mode with bus width of 16 bits, these pins operate (as AD08 to AD15) to input or output the higher byte of data or output the middle byte of address.
1 to 8	99 to 6	P20 to P27	F (CMOS/H)	General-purpose I/O ports In external bus mode, these pins operate as pins A16 to A23 when the corresponding bits of register HACR are 0.
		A16 to A23		In external bus mode, these pins operate (as A16 to A23) to output the higher byte of address when the corresponding bits of register HACR are 0.

CHAPTER 1 OVERVIEW

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
9	7	P30	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates as the ALE pin.
		ALE		In external bus mode, this pin operates to output the address input enable signal (ALE).
10	8	P31	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates as the \overline{RD} pin.
		\overline{RD}		In external bus mode, this pin operates to output the read strobe signal (\overline{RD}).
12	10	P32	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates as the $\overline{WRL}/\overline{WR}$ pin when the WRE bit is 1.
		\overline{WRL}		In external bus mode, this pin operates to output the lower bit of data strobe signal ($\overline{WRL}/\overline{WR}$). \overline{WRL} is used for strobe write for the lower 8 bits of the data bus in 16-bit access mode. \overline{WR} is used for strobe write for 8 bits of the data bus in 8-bit access mode.
		\overline{WR}		
13	11	P33	F (CMOS/H)	General-purpose I/O port. In external bus mode with bus width of 16 bits, this pin operates as the \overline{WRH} pin when the WRE bit of register EPCR is 1.
		\overline{WRH}		In external bus mode, this pin operates to output the higher bit of data strobe signal (\overline{WRH}).
14	12	P34	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates as the HRQ pin when the HDE bit of register EPCR is 1.
		HRQ		In external bus mode, this pin operates to input the hold request signal (HRQ).
15	13	P35	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates as the \overline{HAK} pin when the HDE bit of register EPCR is 1.
		\overline{HAK}		In external bus mode, this pin operates to output the hold acknowledge signal (\overline{HAK}).
16	14	P36	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates to input the external ready signal (RDY) when the RYE bit of register EPCR is 1.
		RDY		In external bus mode, this pin operates to input the external ready signal (RDY).

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
17	15	P37	F (CMOS/H)	General-purpose I/O port. In external bus mode, this pin operates as the CLK pin when the CKE bit of register EPCR is 1.
		CLK		In external bus mode, this pin operates to output the machine cycle clock signal (CLK).
18	16	P40	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD40 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D40 bit = 1] of register ODR4 is invalid.)
		SIN0		Serial data input (SIN0) pin for UART0 This pin is used for input occasionally during the input operation by UART0. Output from this pin by another function must be disabled except for intentional use.
19	17	P41	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD41 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D41 bit = 1] of register DDR4 is invalid.)
		SOT0		Serial data output (SOT0) pin for UART0 Function of this pin is valid when the serial data output from UART0 is enabled.
20	18	P42	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD42 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D42 bit = 1] of register DDR4 is invalid.)
		SCK0		Serial data I/O (SCK0) pin for UART0 Function of this pin is valid when the clock output from UART0 is enabled.
21	19	P43	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD43 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D43 bit = 1] of register DDR4 is invalid.)
		SIN1		Serial data input (SIN1) pin for UART1 This pin is used for input occasionally during the input operation by UART1. Output from this pin by another function must be disabled except for intentional use.

CHAPTER 1 OVERVIEW

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
22	20	P44	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD44 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D44 bit = 1] of register DDR4 is invalid.)
		SOT1		Serial data output (SOT1) pin for UART1 Function of this pin is valid when the serial data output from UART1 is enabled.
24	22	P45	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD45 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D45 bit = 0] of register DDR4 is invalid.)
		SCK1		Serial clock I/O (SCK1) pin for UART1 Function of this pin is valid when the clock output from UART1 is enabled.
25	23	P46	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD46 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D46 bit = 0] of register DDR4 is invalid.)
		ADTG		External trigger input (ADTG) pin for A/D converter
26	24	P47	E (CMOS/H)	General-purpose I/O port. This pin operates as an open-drain output port when the OD47 bit of open drain control setting register (ODR4) is 1. (When these pins are set for input, the setting [D47 bit = 0] of register DDR4 is invalid.)
38	36	P50	G (CMOS/H)	General-purpose I/O port.
		AN0		Analog input (AN0) pin for A/D converter
		SIN3		Serial data input (SIN3) pin for UART3 This pin is used for input occasionally during the input operation by UART3. Output from this pin by another function must be disabled except for intentional use.
39	37	P51	G (CMOS/H)	General-purpose I/O port.
		AN1		Analog input (AN1) pin for A/D converter
		SOT3		Serial data output (SOT3) pin for UART3 Function of this pin is valid when the serial data output from UART3 is enabled.

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
40	38	P52	G (CMOS/H)	General-purpose I/O port.
		AN2		Analog input (AN2) pin for A/D converter
		SCK3		Serial clock I/O (SCK3) pin for UART3 Function of this pin is valid when the clock output from UART3 is enabled.
41	39	P53	G (CMOS/H)	General-purpose I/O port.
		AN3		Analog input (AN3) pin for A/D converter
43	41	P54	G (CMOS/H)	General-purpose I/O port.
		AN4		Analog input (AN4) pin for A/D converter
		SIN4		Serial data input (SIN4) pin for UART4 This pin is used for input occasionally during the input operation by UART4. Output from this pin by another function must be disabled except for intentional use.
44	42	P55	G (CMOS/H)	General-purpose I/O port.
		AN5		Analog input (AN5) pin for A/D converter
		SOT4		Serial data output (SOT4) pin for UART4 Function of this pin is valid when the serial data output from UART4 is enabled.
45	43	P56	G (CMOS/H)	General-purpose I/O port.
		AN6		Analog input (AN6) pin for A/D converter
		SCK4		Serial clock output (SCK4) pin for UART4 Function of this pin is valid when the clock output from UART4 is enabled.
46	44	P57	G (CMOS/H)	General-purpose I/O port.
		AN7		Analog input (AN7) pin for A/D converter
27	25	C	-	Pin to connect 0.1 μ F capacitor to regulate supplied power voltage
28	26	P71	F (CMOS/H)	General-purpose I/O port.
29	27	P72	F (CMOS/H)	General-purpose I/O port
32	30	P73	H (CMOS/H)	General-purpose I/O port This pin operates as a D/A converter output pin (DA00) when the DAE0 bit of D/A control register (DACR) is 1.
		DAO0		D/A converter output 0 (DAO0) pin

CHAPTER 1 OVERVIEW

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
33	31	P74	H (CMOS/H)	General-purpose I/O port This pin operates as a D/A converter output pin (DAO1) when the DAE1 bit of D/A control register (DACR) is 1.
		DAO1		D/A converter output 1 (DAO1) pin
47	45	P80	F (CMOS/H)	General-purpose I/O port
		IRQ0		External interrupt request 0 (IRQ0) pin
48	46	P81	F (CMOS/H)	General-purpose I/O port
		IRQ1		External interrupt request 1 (IRQ1) pin
53	51	P82	F (CMOS/H)	General-purpose I/O port
		IRQ2		External interrupt request 2 (IRQ2) pin
54	52	P83	F (CMOS/H)	General-purpose I/O port
		IRQ3		External interrupt request 3 (IRQ3) pin
55	53	P84	F (CMOS/H)	General-purpose I/O port
		IRQ4		External interrupt request 4 (IRQ4) pin
56	54	P85	F (CMOS/H)	General-purpose I/O port
		IRQ5		External interrupt request 5 (IRQ5) pin
57	55	P86	F (CMOS/H)	General-purpose I/O port
		IRQ6		External interrupt request 6 (IRQ6) pin
58	56	P87	F (CMOS/H)	General-purpose I/O port
		IRQ7		External interrupt request 7 (IRQ7) pin
59	57	P60	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned to this pin by the setting (RD60 = 1) of pull-up resistor setting register (RDR6). (When this pin set for output, the setting [D60 = 1] of register DDR6 is invalid.)
		SIN2		Serial data input (SIN2) pin for UART2 This pin is used for input occasionally during the input operation by UART2. Output from this pin by another function must be disabled except for intentional use.
60	58	P61	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned to this pin by the setting (RD61 = 1) of pull-up resistor setting register (RDR6). (When this pin set for output, the setting [D61 = 1] of register DDR6 is invalid.)
		SOT2		Serial data output (SOT2) pin for UART2 Function of this pin is valid when the serial data output from UART2 is enabled.

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
61	59	P62	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned to this pin by the setting (RD62 = 1) of pull-up resistor setting register (RDR6). (When this pin set for output, the setting [D62 = 1] of register DDR6 is invalid.)
		SCK2		Serial clock output (SCK2) pin for UART2 Function of this pin is valid when the clock output from UART2 is enabled.
62	60	P63	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned to this pin by the setting (RD63 = 1) of pull-up resistor setting register (RDR6). (When this pin set for output, the setting [D63 = 1] of register DDR6 is invalid.)
		PPG1		This pin operates to output PPG1 signal when PPG is valid.
63	61	P64	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned to this pin by the setting (RD64 = 1) of pull-up resistor setting register (RDR6). (When this pin set for output, the setting [D64 = 1] of register DDR6 is invalid.)
		PPG0		This pin operates to output PPG0 signal when PPG is valid.
64	62	P65	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned to this pin by the setting (RD65 = 1) of pull-up resistor setting register (RDR6). (When this pin set for output, the setting [D65 = 1] of register RDR6 is invalid.)
		CKOT		This pin operates to output CKOT signal when CKOT is valid.
65	63	TX*	I	This pin operates to output IEBus TM signal.
66	64	RX*	J (CMOS)	This pin operates to input IEBus TM signal.
67 to 69	65 to 67	P90 to P92	F (CMOS/H)	General-purpose I/O port
		TIN0 to TIN2		Event input pins for reload timers 0, 1, and 2 These pins are used for input continuously during the input from reload timers. Output to this pin from other functions must be disabled except for intentional use.
		IN0 to IN2		Trigger input pins for input capture channels 0 to 2

CHAPTER 1 OVERVIEW

Table 1.6-1 Pin Functions (Continued)

QFP	LQFP	Pin name	Circuit format	Function
70	68	P93	F (CMOS/H)	General-purpose I/O port
		TOTO		Reload timer output pin This pin function is applied for the output from reload timer 0.
		IN3		Trigger input pin for input capture channel 3
71,72	69,70	P94,P95	F (CMOS/H)	General-purpose I/O port.
		TOT1, TOT2		Reload timer output pins This pin function is applied for the output from reload timers 1 and 2.
		OUT0, OUT1		Event output pins for output compare channels 0 and 1
73	71	P96	F (CMOS/H)	General-purpose I/O port.
		PWC		This pin operates to input PWC signal when the PWC timer is valid.
74	72	P97	F (CMOS/H)	General-purpose I/O port.
		POT		This pin operates to output PWC signal when the PWC timer is valid.
75,76	73,74	PA0,PA1	F (CMOS/H)	General-purpose I/O port.
78	76	PA2	F (CMOS/H)	General-purpose I/O port.
79	77	X1A	A	Oscillation input. For the one clock system, leave it open.
80	78	X0A	A	Oscillation input. For the one clock system, perform external pull-down processing.
34	32	AV _{CC}	-	A/D converter power supply pin
37	35	AV _{SS}	-	A/D converter power supply pin
35	33	AVRH	-	External reference power supply pin for A/D converter
36	34	AVRL	-	External reference power supply pin for A/D converter
30	28	DVRH	-	External reference power supply pin for D/A converter
31	29	DV _{SS}	-	External reference power supply pin for D/A converter
49 to 51	47 to 49	MD0 to MD2	C	Input pins for operation mode specification These pins are connected directly to V _{CC} or V _{SS} .
23,84	21,82	V _{CC}	-	Power supply (5 V) input pins
11,42,81	9,40,79	V _{SS}	-	Power supply (0 V) input pins

*1: On the MB90587C/CA, RX and TX are replaced with N and C, respectively.

1.7 I/O Circuit Formats

Table 1.7-1 "I/O Circuit Types" shows the formats of I/O circuits. The alphabetic letters shown in the "Classification" column of Table 1.7-1 "I/O Circuit Types" correspond to those shown in the "Circuit format" column of Table 1.6-1 "Pin Functions".

■ I/O Circuit Types

Table 1.7-1 I/O Circuit Types

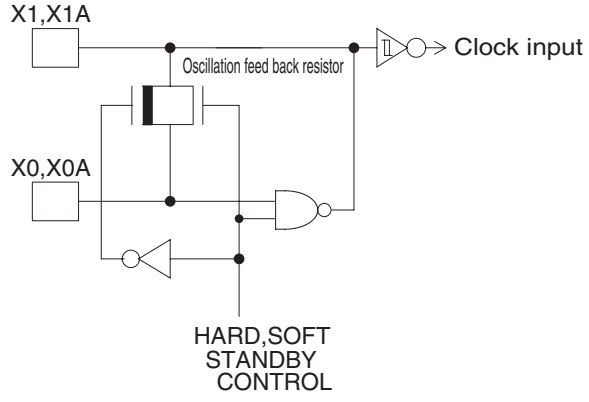
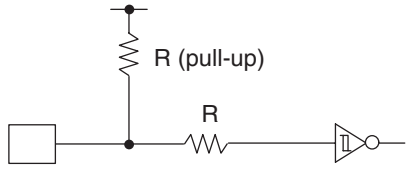
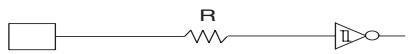
Classification	Circuit	Remark
A	 <p>The diagram shows an oscillator circuit. It includes two square-wave components labeled X1, X1A and X0, X0A. X1, X1A is connected to an oscillation feedback resistor, which is then connected to a clock input. X0, X0A is connected to an AND gate. The output of the AND gate is connected to a clock input. There is also a HARD, SOFT STANDBY CONTROL input connected to the AND gate.</p>	<ul style="list-style-type: none">Oscillation feedback resistor: About 1 MΩ (High speed oscillator) About 10 MΩ (Low speed oscillator)
B	 <p>The diagram shows a hysteresis input circuit. It consists of a square-wave component connected to a pull-up resistor R, which is then connected to a hysteresis input.</p>	<ul style="list-style-type: none">Hysteresis input with pull-up resistor Pull-up resistor: About 50 kΩ
C	 <p>The diagram shows a hysteresis input circuit. It consists of a square-wave component connected to a resistor R, which is then connected to a hysteresis input.</p>	<ul style="list-style-type: none">Hysteresis input

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

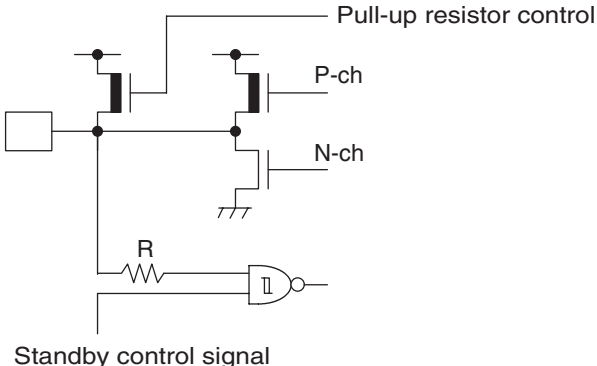
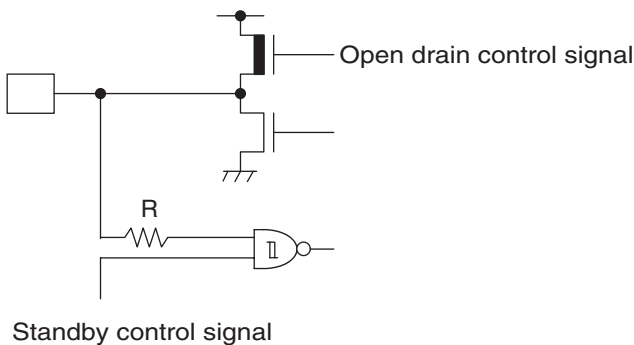
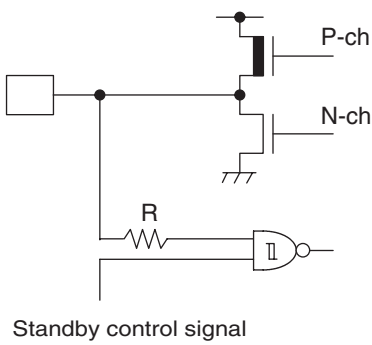
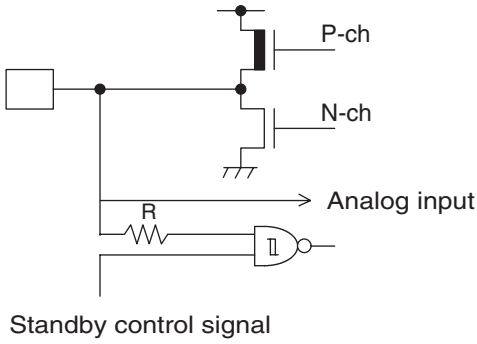
Classification	Circuit	Remark
D	 <p>Pull-up resistor control</p> <p>P-ch</p> <p>N-ch</p> <p>R</p> <p>Standby control signal</p>	<ul style="list-style-type: none">• Input with pull-up resistor control• CMOS level output• Hysteresis input with standby control <p>Pull-up resistor: About 50 kΩ</p>
E	 <p>Open drain control signal</p> <p>R</p> <p>Standby control signal</p>	<ul style="list-style-type: none">• CMOS level output• Hysteresis input with standby control• Open drain control signal
F	 <p>P-ch</p> <p>N-ch</p> <p>R</p> <p>Standby control signal</p>	<ul style="list-style-type: none">• CMOS level output• Hysteresis input with standby control
G	 <p>P-ch</p> <p>N-ch</p> <p>R</p> <p>Analog input</p> <p>Standby control signal</p>	<ul style="list-style-type: none">• CMOS level output• Hysteresis input with standby control• Analog input

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

Classification	Circuit	Remark
H	<p>Standby control signal</p>	<ul style="list-style-type: none">• CMOS level output• Hysteresis input with standby control• DA output
I	<p>Standby control signal</p>	<ul style="list-style-type: none">• CMOS level output
J	<p>Standby control signal</p>	<ul style="list-style-type: none">• CMOS input with standby control

1.8 Precautions on Handling of Device

Take special care for the following points when handling and operating the device:

- Prevention of latchup
 - Treatment of unused input pins
 - Treatment of A/D converter power supply pins
 - Treatment of D/A converter power supply pins
 - Treatment of TX and RX pins when IEBusTM is not used
 - Power supply pins
 - Using REALOS
 - Startup of power
 - Use of subclock mode and external clock
 - Indeterminate outputs from ports 0 and 1
 - Notes on the use of "DIV A, Ri" and "DIVW A Rwi" instructions
 - Stabilization of supply voltage
-

■ Precautions on the Handling of the Device

○ Prevention of latchup

Latchup may occur on a CMOS IC chip if:

- a voltage higher than V_{CC} or lower than V_{SS} is applied to an input or output pin,
- a voltage over the rated voltage is applied between V_{CC} and V_{SS} .
- voltage AV_{CC} is supplied before voltage V_{CC} .

An latchup increases remarkably the supply current that may cause an device overheat destruction.

○ Treatment of unused pins

Unused input pins left open may cause abnormal operation, or latch up leading to permanent damage. Unused input pins should be pulled up or pulled down through at least 2 k Ω resistance. Unused input/output pins may be left open in output state, but if such pins are in input state they should be handled in the same way as input pins.

○ Treatment of A/D converter power supply pins

If the A/D converter is not used, make connections of its power supply pins so that AV_{CC} is equal to V_{CC} and AV_{SS} is equal to $AVRH$, $AVRL$, and V_{SS} .

○ Treatment of D/A converter power supply pins

If the D/A converter is not used, make connections of its power supply pins so that $DVRH$ is equal to V_{SS} and DV_{SS} is equal to V_{SS} .

○ Treatment of TX and RX pins when IEBus™ is not used

If IEBus is not used, connect a pull-down resistor to the TX pin and a pull-down or pull-up resistor to the RX pin.

○ Power supply pins

The device is designed to connect those V_{CC} and V_{SS} pins that must be at the same potential in the device to prevent malfunctions, including latchup. When using the device, connect all the V_{CC} and V_{SS} pins externally to the power supply and ground to reduce unnecessary radiation, prevent the incorrect operation of strobe signals due to a rise of ground level, and keep to the standard for total output power.

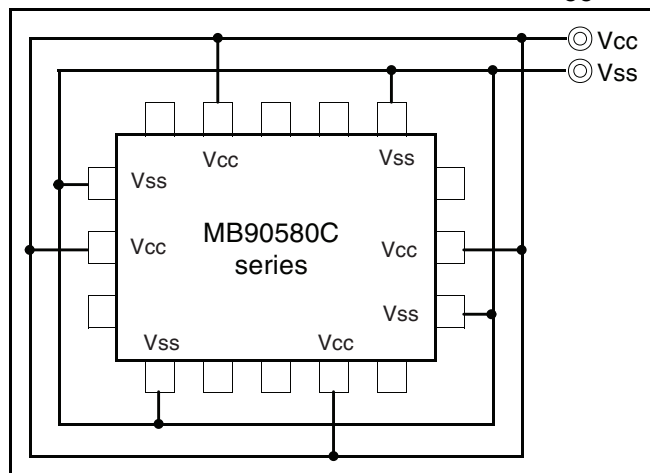
○ Using REALOS

The use of EI²OS is not passible with the REALOS real time operating system.

○ Startup of power

To prevent a malfunction in the internal step-down circuit, be sure that the voltage start-up time after the power is turned on is at least 50 μ s (between 0.2 and 2.7 V).

Figure 1.8-1 Treatment of Power Supply Pins (V_{CC} and V_{SS})

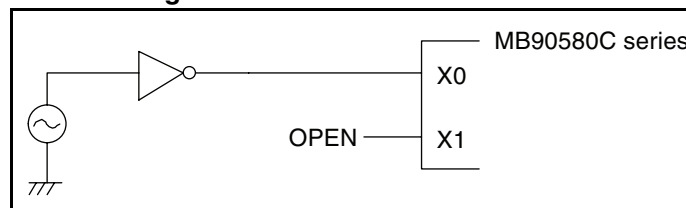


○ Use of subclock mode and external clock

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

When using external clock, drive only pin X0 and leave pin X1 open. (See Figure 1.8-2 "Use of External Clock".)

Figure 1.8-2 Use of external clock



○ Indeterminate outputs from ports 0 and 1

During oscillation setting time of step-down circuit (during a power-on reset) after the power is turned on, the outputs from ports 0 and 1 become following state.

- If $\overline{\text{RST}}$ pin is "H", the outputs become indeterminate.
- If $\overline{\text{RST}}$ pin is "L", the outputs become high-impedance.

Pay attention to the port output timing shown as follow

Figure 1.8-3 Indeterminate output from ports 0 and 1 ($\overline{\text{RST}}$ pin is "H")

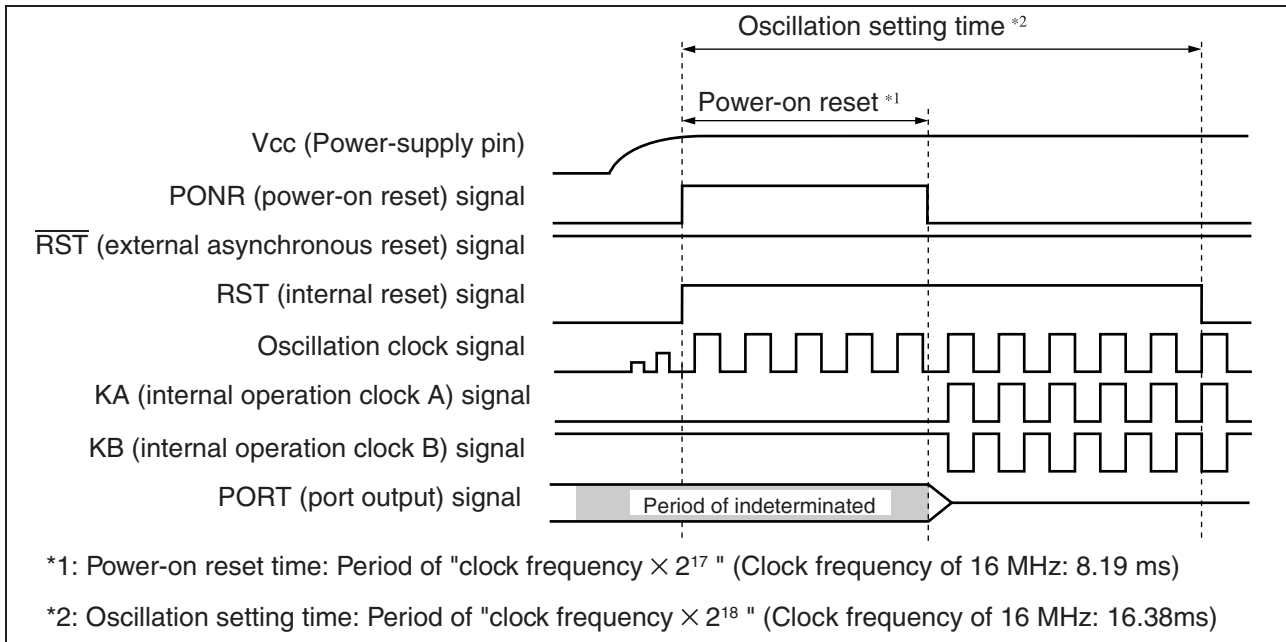
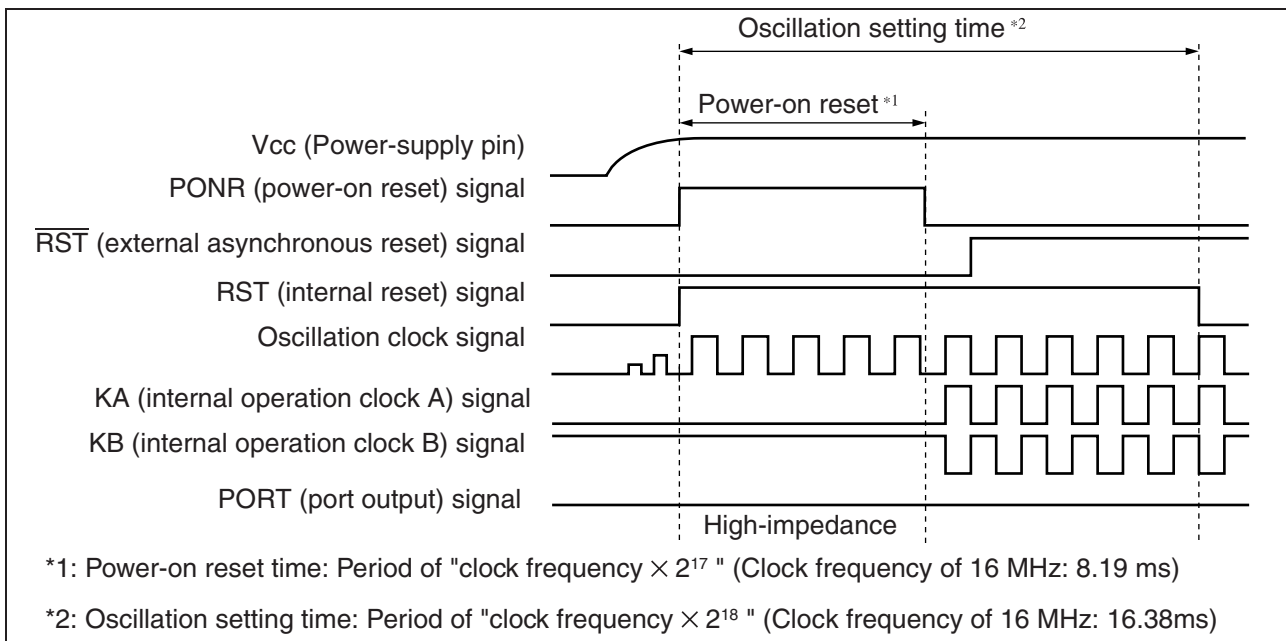


Figure 1.8-4 High-impedance output from ports 0 and 1 ($\overline{\text{RST}}$ pin is "L")



○ Notes on the use of "DIV A, Ri" and "DIVW A Rwi" instructions

The remainder obtained as the result of the execution of the signed division instructions "DIV A, Ri" and "DIVW A, Rwi" is determined by the bank register and stored in the address containing the memory bank set by the bank register.

For details, see Section 2.7 "Notes on Use of the DIV A, Ri and DIVW A, Rwi Instructions". Also, for information on bank registers, see Section 2.3.6 "Bank Registers".

○ Stabilization of supply voltage

A sudden change in the supply voltage may cause the device to malfunction even within the specified Vcc supply voltage operation range. Therefore, the Vcc supply voltage should be stabilized.

For reference, supply voltage should be controlled so that Vcc ripple variations (peak-to-peak values) at commercial frequencies (50 to 60Hz) fall below 10% of the standard Vcc supply voltage and the coefficient of fluctuation does not exceed 0.1 V/ms at instantaneous power switching.

○ Notes on the during operation of PLL clock mode

If the PLL clock mode is selected, the microcontroller attempt to be working with the self-oscillating circuit even when there is no external oscillator or external colck input is stopped. Performance of this operation, however, cannot be guaranteed.

CHAPTER 2 CPU

This chapter describes the functions and operation of the CPU.

- 2.1 "Memory Space"
- 2.2 "Addressing"
- 2.3 "Dedicated Registers"
- 2.4 "General-purpose Registers"
- 2.5 "Prefix Codes"
- 2.6 "Interrupt Suppression Instructions and Prefix Codes"
- 2.7 "Notes on Use of the DIV A, Ri and DIVW A, RWi Instructions"

2.1 Memory Space

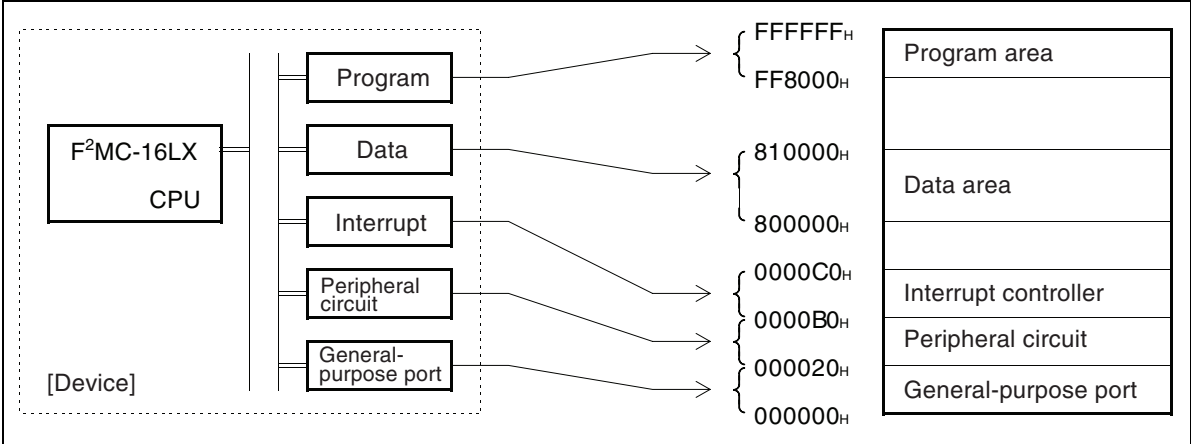
The F²MC-16LX CPU core is a 16-bit CPU that was designed for applications requiring high-speed real-time processing in the areas welfare and car-mounted products. The F²MC-16LX instruction set is designed for controller applications and enables high-speed and high-efficiency processing of various types of control.

In addition to 16-bit data processing, the F²MC-16LX can perform 32-bit data processing because it contains an internal 32-bit accumulator. The maximum size of the memory space is 16 MB (expandable) and can be accessed by the linear or the bank method. The instruction system was enhanced based on the F²MC-8 A-T architecture by adding high-level language support instructions, expanding the addressing modes, and enhancing the multiplication and division instructions and provides substantial bit processing capabilities.

■ Memory Space

All data/program I/O channels managed by the F²MC-16LX CPU are allocated in the 16 MB memory space of the F²MC-16LX CPU. Specifying these addresses via the 24-bit address bus enables the CPU to access each of the resources.

Figure 2.1-1 Example of the Relationship between the F²MC-16LX System and the Memory Map



2.2 Addressing

The following two methods can be used to specify addresses of the F²MC-16LX

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

■ Linear Addressing Methods

The linear addressing methods can be classified into the following two types:

- **24-bit operand specification:** A 24-bit address is directly specified 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 the 24-bit Operand Specification in the Linear Addressing Method

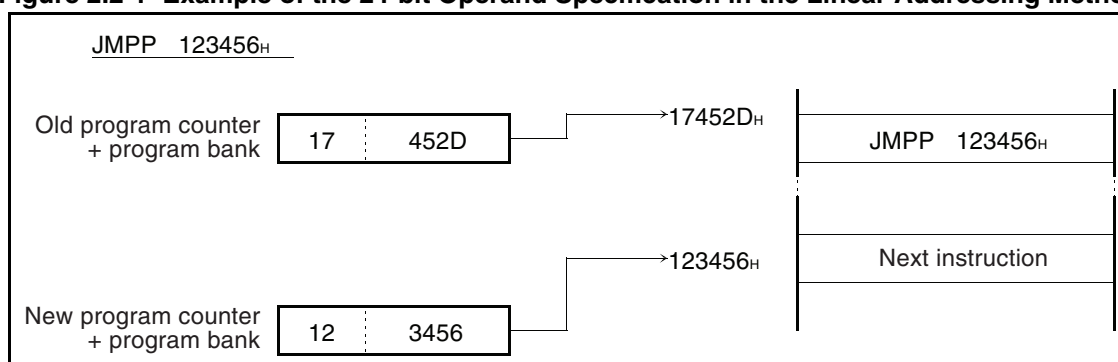
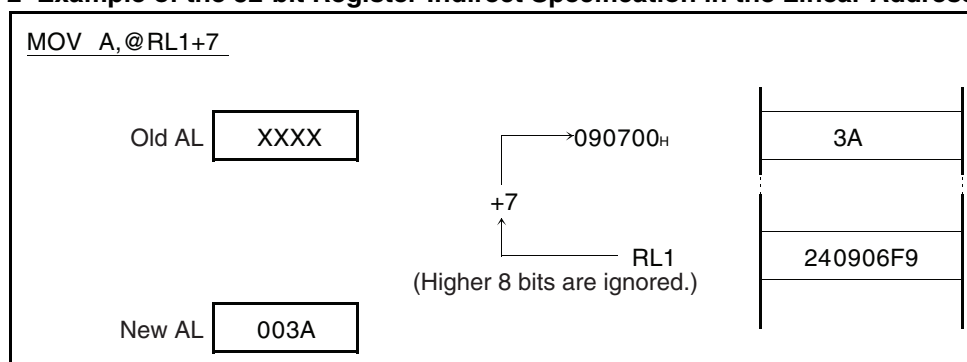


Figure 2.2-2 Example of the 32-bit Register-Indirect Specification in the Linear Addressing Method



■ Addressing by the Bank Method

The bank method divides the 16 MB memory space into 256 banks of each 64 KB and specifies the bank associated with each space using the following five bank registers:

○ **Program bank register (PCB): Initial reset value FF_H**

The 64 KB bank specified by PCB is called program (PC) space. The program space mainly contains instruction codes, the vector table, and immediate data.

○ **Data bank register (DTB): Initial reset value 00_H**

The 64 KB bank specified by DTB is called data (DT) space. The data space mainly contains readable and writable data as well as the control and data registers for external and internal resources.

○ **Initial reset value 00_H of the user stack bank register (USB) and initial reset value 00_H of the system stack bank register (SSB)**

The 64 KB bank specified by USB or SSB is called stack (SP) space. The stack space is accessed when a stack is used to save the contents of a register during the execution of a push or pop instruction or an interrupt. The space to be used depends on the S flag in the condition code register.

○ **Additional bank register (ADB): Initial reset value 00_H**

The 64 KB bank specified by ADB is called additional (AD) space. The additional space mainly contains data that overflowed from the DT space.

To increase instruction code efficiency, a default space is determined for instructions of each addressing mode, as listed in Table 2.2-1 "Default Spaces and Addressing Mode". To use a non-default space when using an addressing mode, specify the prefix code associated with the bank before the instruction. This makes it possible to access any bank space that is associated with the prefix code.

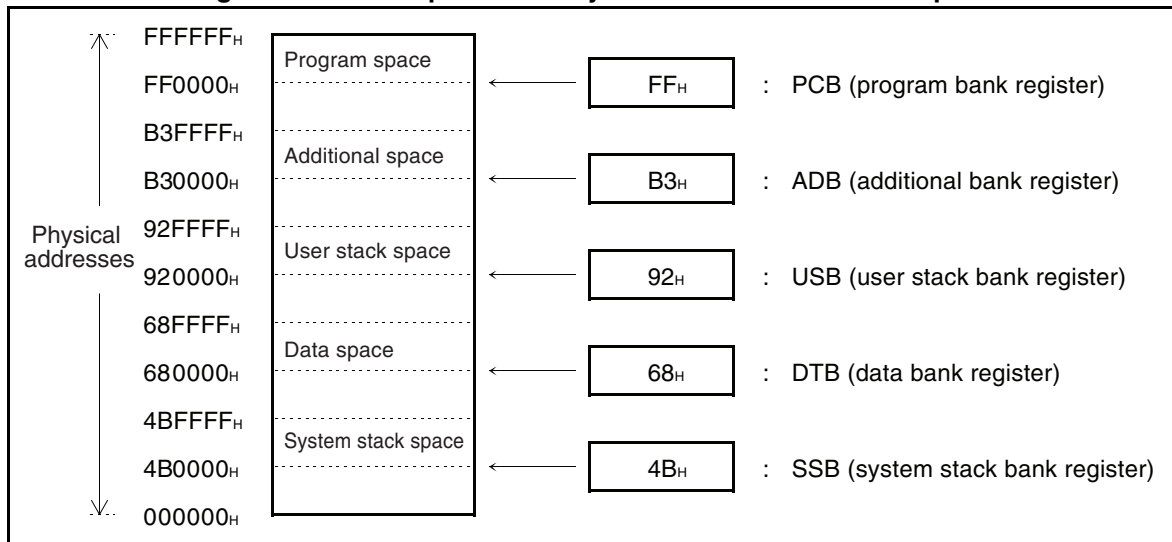
DTB, USB, SSB, and ADB are initialized to 00_H by a reset. PCB is initialized to a value specified by the reset vector. After the reset, the DT, SP, or AD space is allocated in bank 00_H (000000_H-00FFFF_H). The PC space is allocated in the bank specified by the reset vector.

Table 2.2-1 Default Spaces and Addressing Mode

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

Figure 2.2-3 "Example of the Physical Addresses of Each Space" is an example of memory space divided into register banks.

Figure 2.2-3 Example of the Physical Addresses of Each Space



2.2.1 Allocating Multiple-byte Data in a Memory Space

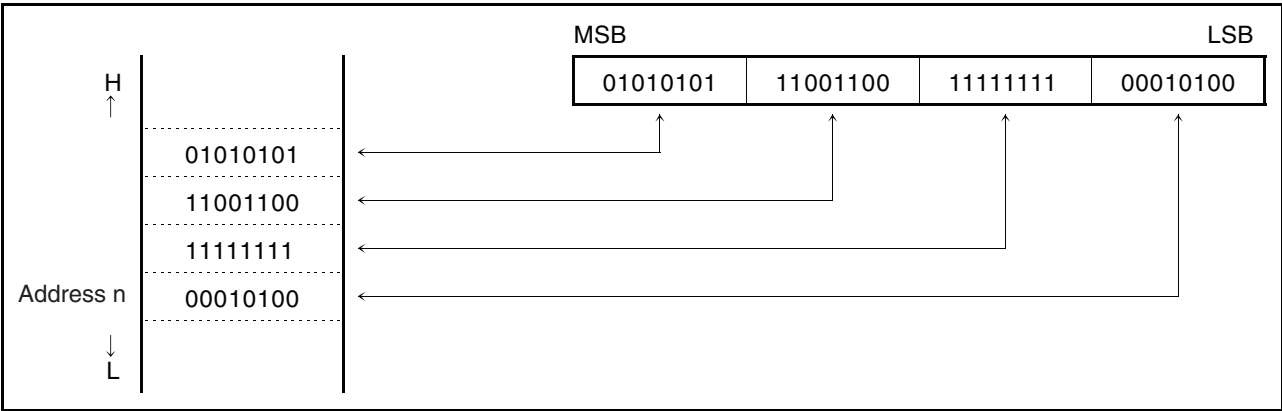
In a memory space, the lower eight bits of multiple-byte data are stored at address n . The remaining bits are stored at the addresses $n + 1$, $n + 2$, $n + 3$, ... in that order.

■ Allocating Multiple-byte Data in a Memory Space

As shown in Figure 2.2-4 "Example of Allocating Multiple-byte Data in a Memory Space" data is written to memory in ascending order of addresses. Therefore, if the data is 32 bit long, the lower 16 bits are first transferred and the higher 16 bits are transferred subsequently.

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

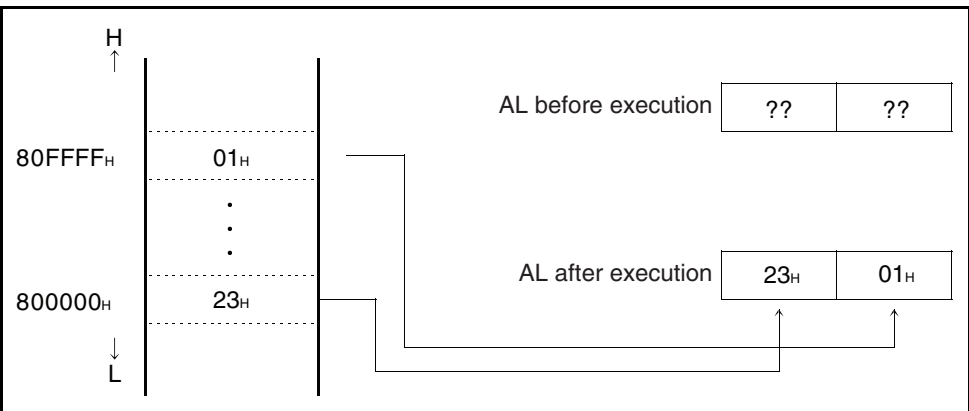
Figure 2.2-4 Example of Allocating Multiple-byte Data in a Memory Space



■ Access of Multiple-byte Data

As shown in Figure 2.2-5 "Example of Accessing Multiple-byte Data (Execution of MOVWA, 080FFFF_H)" all accesses are basically made within a bank. For an instruction that accesses multiple-byte data, the next address after address FFFF_H is 0000_H in the same bank.

Figure 2.2-5 Example of Accessing Multiple-byte Data (Execution of MOVWA, 080FFFF_H)



2.3 Dedicated Registers

Dedicated registers are implemented in the CPU by dedicated hardware. The application of these registers is restricted by the CPU architecture.

■ Dedicated Registers

Dedicated registers are implemented in the CPU by dedicated hardware. The application of these registers is restricted by the CPU architecture.

The F²MC-16LX supports the following 13 dedicated registers:

- **Accumulator (A=AH:AL)**

Two 16-bit accumulators (can be used as a single accumulator containing a total of 32 bits).

- **User stack pointer (USP)**

A 16-bit pointer to the user stack area.

- **System stack pointer (SSP)**

A 16-bit pointer to the system stack area.

- **Processor status (PS)**

A 16-bit register indicating the system status.

- **Program counter (PC)**

16-bit register for the address at which the program is stored.

- **Program bank register (PCB)**

An 8-bit register indicating the PC space.

- **Data bank register (DTB)**

An 8-bit register indicating the DT space.

- **User stack bank register (USB)**

An 8-bit register indicating the user stack space.

- **System stack bank register (SSB)**

An 8-bit register indicating the system stack space.

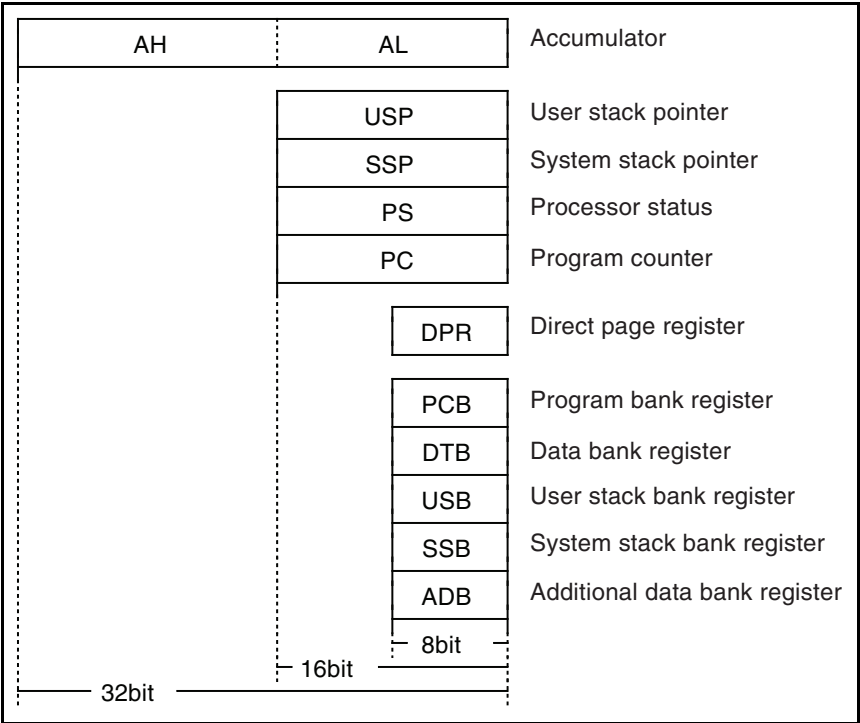
- **Additional bank register (ADB)**

An 8-bit register indicating the AD space.

○ Direct page register (DPR)

An 8-bit register indicating the direct page.

Figure 2.3-1 Dedicated Registers



2.3.1 Accumulator (A)

The accumulator (A) consists of the two 16-bit arithmetic operation registers AH and AL. It is used as temporary storage for arithmetic operation results or for data transfer. When AH and AL are used for 32-bit data processing, they are connected. Only AL is used for word processing of 16-bit data or byte processing of 8-bit data.

■ Accumulator (A)

Data in the accumulator (A) can be used for arithmetic operations with data in the memory and registers (Ri, RWi, and RLi). As with the F²MC-8L, when the F²MC-16LX transfers data to the AL that is not longer than a single word, the data in AL before the transfer is automatically transferred to the AH (data retention function). In other words, processing efficiency can be increased by using the data retention function and AL-AH arithmetic operations.

Figure 2.3-2 Example of 32-bit Data Transfer

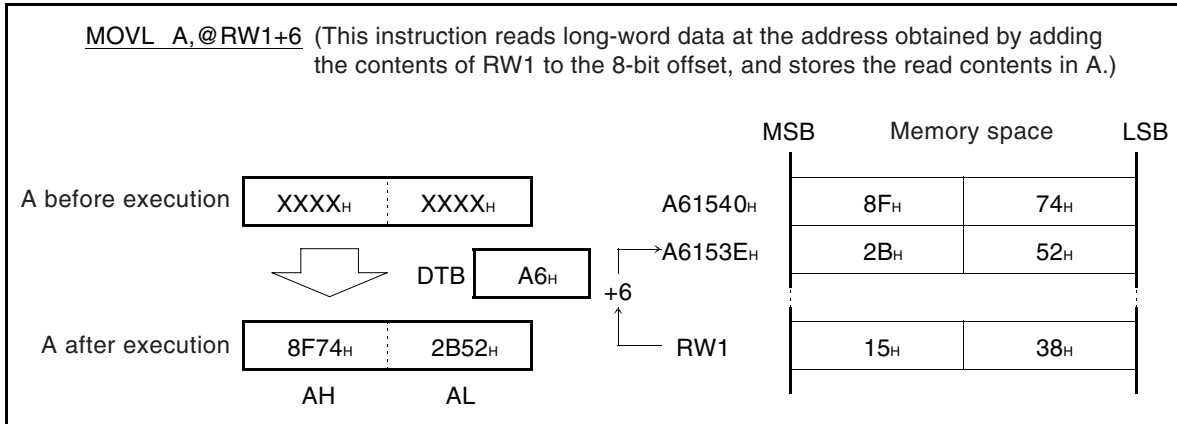
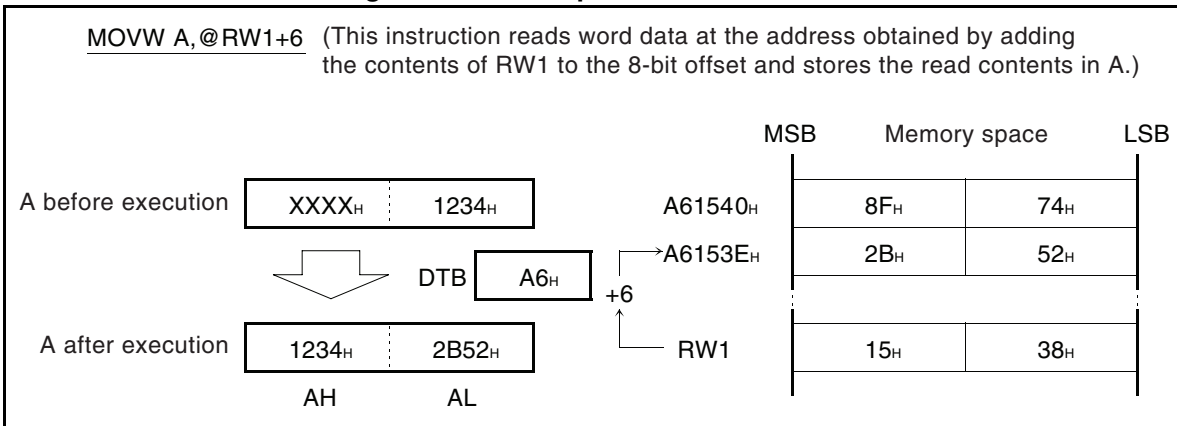


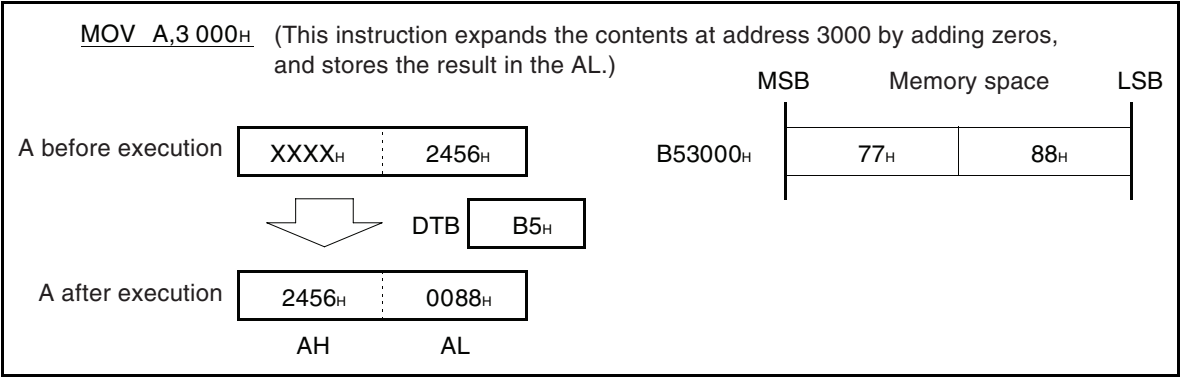
Figure 2.3-3 Example of AL-AH Transfer



As shown in Figure 2.3-4 "Example of Zero Extension" when data is transferred to the AL that is not longer than a byte, it is expanded to 16 bits by signs or zeros and stored in AL. The data in the AL can also be handled both as word or byte data. If an arithmetic operation instruction for byte processing is executed, the higher eight bits of AL before the arithmetic operations are ignored. All higher eight bits of the arithmetic operation result are set to 0.

The accumulator (A) is not initialized by a reset. Immediately after a reset, its value becomes undefined .

Figure 2.3-4 Example of Zero Extension



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

The user stack pointer (USP) and system stack pointer (SSP) are 16-bit registers, and indicate an address for data saving and return during execution of a push or pop instruction or a subroutine.

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

The user stack pointer (USP) and system stack pointer (SSP) are used by stack instructions. However, if the S flag for the processor status is 0, the USP register becomes valid. If the S flag is 1, the SSP register becomes valid (see Figure 2.3-6 "Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 0)" and Figure 2.3-7 "Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 1)"). If an interrupt is accepted, the S flag is set. In other words, register saving at an interrupt always occurs in the memory area indicated by the SSP. The SSP is used for stack processing by an interrupt routine. The USP is used for stack processing by a routine other than an interrupt routine. If it is unnecessary to split the stack space, use only the SSP. SSP --> SSB, USP --> USB indicate the higher eight bits of an address for stack processing. USP and SSP are not initialized by a reset and their values become undefined in that case.

Figure 2.3-6 Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 0)

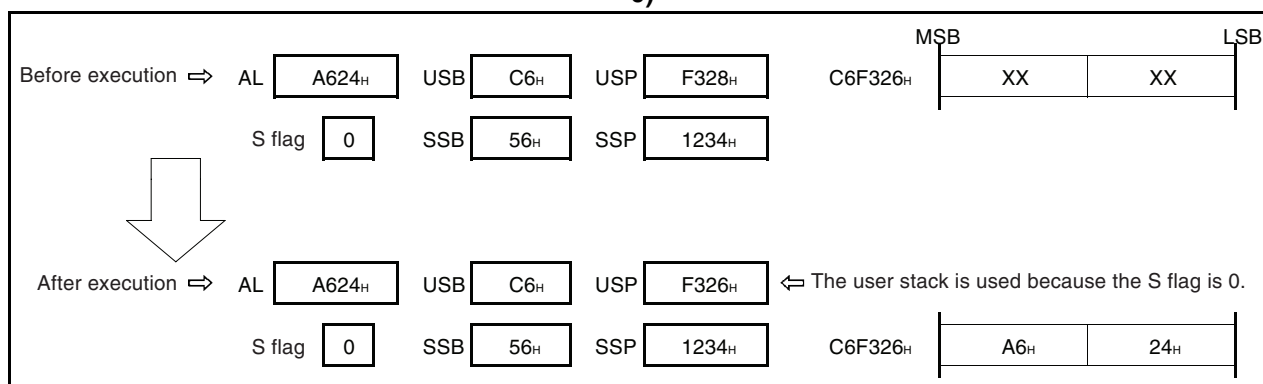
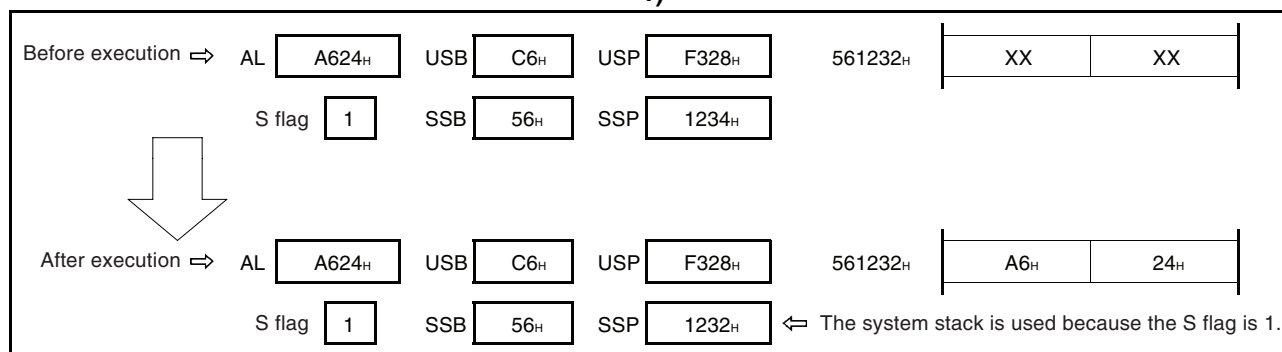


Figure 2.3-7 Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is 1)



Note:

As a general rule, use an even-numbered address as the value for the stack pointer.

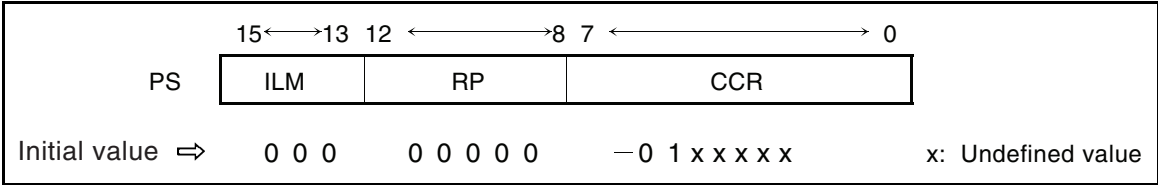
2.3.3 Processor Status (PS)

The processor status (PS) consists of bits for controlling CPU operations and bits indicating the CPU status.

■ Processor Status (PS)

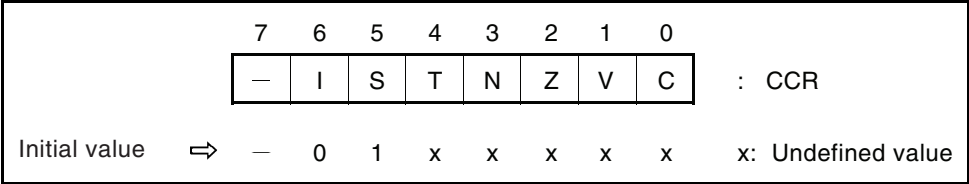
The higher bytes of the processor status (PS) consist of the register bank pointer (RP), which indicates the starting address of the register bank, and the interrupt level mask register (ILM). The lower bytes of the PS consist of the condition code register (CCR) formed by flags to be set or reset by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

Figure 2.3-8 Structure of the Processor Status (PS)



■ Condition Code Register (CCR)

Figure 2.3-9 Configuration of the Condition Code Register (CCR)



○ Interrupt enable flag (I)

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

○ Stack flag (S)

When S is 0, the USP becomes effective as the stack operation pointer. If S is 1, SSP becomes effective. S is set when an interrupt is accepted or a reset is made.

○ Sticky bit flag (T)

This flag is 1 if the data shifted out from the carry contains at least one 1 after a 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.

○ Negative flag (N)

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

○ **Zero flag (Z)**

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

○ **Overflow flag (V)**

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

○ **Carry flag (C)**

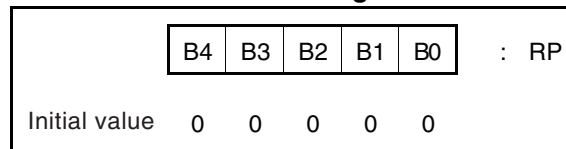
This flag is set when a carry-up or carry-down is generated from the MSB as a result of an arithmetic operation. It is cleared if no carry-up or carry-down occurs.

■ **Register Bank Pointer (RP)**

The register bank pointer (RP) indicates the relationship between the general-purpose register of the F²MC-16LX and its internal RAM address. The RP indicates the first memory address of the register bank which is currently used by the conversion expression $[000180_H + (RP) \cdot 10_H]$. The RP consists of five bits and can have a value from 00_H to $1F_H$. It can also assign a register bank in the memory area 000180_H to $00037F_H$.

However, when the memory in this range is not internal RAM, the register cannot be used as a general-purpose register. The contents of RP are all initialized to 0 by a reset. From the viewpoint of an instruction, it is possible to transfer an 8-bit immediate value to the RP, but only the lower five bits of the data are actually used.

Figure 2.3-10 Structure of the Register Bank Pointer (RP)



■ Interrupt Level Mask Register (ILM)

The interrupt level mask register (ILM) consists of three bits that 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. As shown in Figure 2.3-1 "Dedicated Registers" level 0 is defined as the highest and level 7 is defined as the lowest. In order that an interrupt is accepted, a value which is smaller than the value retained by the current ILM must be requested. When the interrupt is accepted, its level value is stored in the ILM, and any subsequent interrupt with equal or lower priority is not accepted. The contents of ILM are all initialized to 0 by a reset. From the viewpoint of the instruction, it is possible to transfer an 8-bit immediate value to the ILM, but only the lower three bits of the data are actually used.

Figure 2.3-11 Structure of the Interrupt Level Register (ILM)

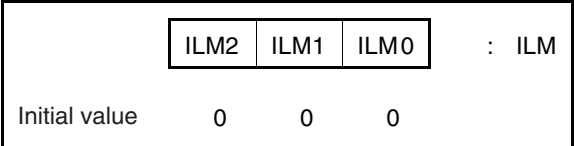


Table 2.3-1 Level Hierarchy of the Levels Indicated by the Interrupt Level Mask Register (ILM)

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

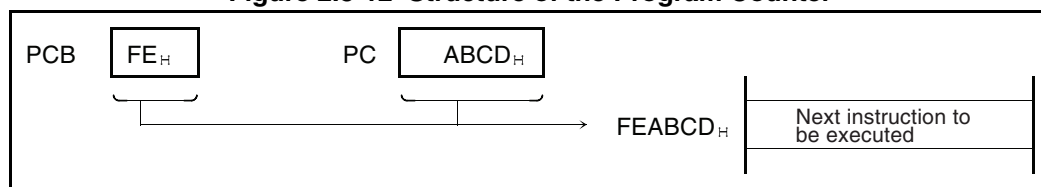
2.3.4 Program Counter (PC)

The program counter (PC) is a 16-bit counter that indicates the lower 16 bits of the memory address for the instruction code to be executed by the CPU. The higher 8-bit of the address are indicated by the PCB.

■ Program Counter (PC)

The program counter (PC) is updated by conditional branch instructions, subroutine call instructions, interrupts, or resets. It can also be used as the base pointer to an operand access.

Figure 2.3-12 Structure of the Program Counter



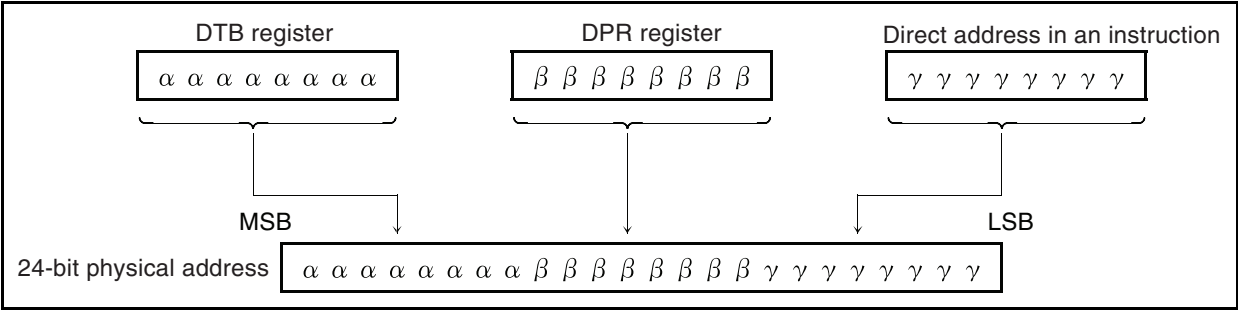
2.3.5 Direct Page Register (DPR)

The direct page register (DPR) specifies an operand between addr8 and addr15 when a direct addressing instruction is executed. DPR is eight bits long and is initialized to 01_H by a reset. DPR can be read or written by an instruction.

■ Direct Page Register (DPR)

Figure 2.3-13 "Generating a Physical Address by Direct Addressing" shows physical address generation by direct addressing.

Figure 2.3-13 Generating a Physical Address by Direct Addressing



2.3.6 Bank Registers

Bank registers can be classified into the following five types:

- Program counter bank register (PCB) <initial value: value of the reset vector>
 - Data bank register (DTB) <initial value: 00_H>
 - User stack bank register (USB) <initial value: 00_H>
 - System stack bank register (SSB) <initial value: 00_H>
 - Additional data bank register (ADB) <initial value: 00_H>
-

■ Bank Registers

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 the bank registers are of byte length, and the PCB is initialized to "00_H" by a reset vector. Non-PCB bank registers are readable and writable. The PCB is readable but not writable.

The PCB is rewritten at an interrupt or when a JMPP, CALLP, RETP, RETI, or RETF instruction which branches to all of the 16 MB space is executed. For register operations, see Section 2.1 "Memory Space".

2.4 General-purpose Registers

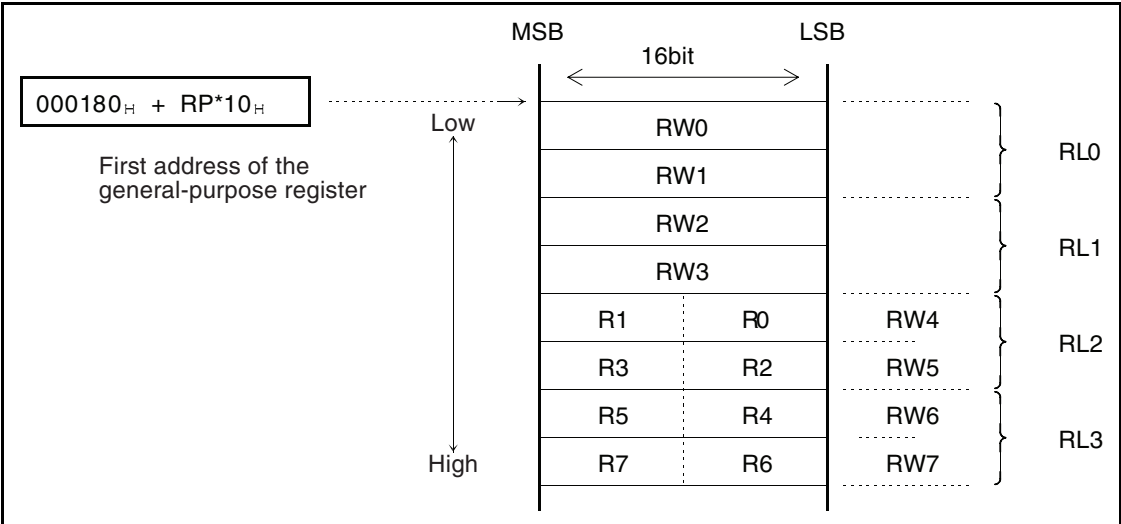
General-purpose registers are the same as dedicated registers in the sense that they coexist with the RAM in the address space of the CPU and that they can be accessed without specifying an address. Similar to ordinary memory, the user can specify the use of general-purpose registers.

■ General-purpose Registers

The general-purpose registers of the F²MC-16LX are located at 000180_H to 00037F_H (maximum) of the RAM. The register bank pointer (RP) specifies the portion of the previously mentioned register bank currently used. Each bank contains the three types of registers listed below. These registers are not independent but have the following relationship as shown in Figure 2.4-1 "General-purpose Registers".

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

Figure 2.4-1 General-purpose Registers



The relationship between higher and lower bytes of the byte and word registers can be expressed by the following expression:

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

The relationship between the higher and lower bytes of the RL_i and RW can be expressed by the following expression:

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

■ Register Banks

A register bank consists of eight words. As with ordinary RAM, the contents of the register bank are not initialized by a reset and the status before the reset is retained. However, the values contained in the register bank are undefined at power-on.

As shown in Table 2.4-1 "Functions of Register Banks" register banks can be used as general-purpose registers in the type of byte registers R0 to R7, word registers RW0 to RW7, and long word registers RL0 to RL3. They can also be used for arithmetic operations to store an instruction or a pointer.

Table 2.4-2 "Relationship between Register in Register Bank" shows the relationship of registers in a register bank.

Table 2.4-1 Functions of Register Banks

Register	Function
R0 to R7	Used as operands of instructions. Note R0 is also used as a barrel shift counter and normalize instruction counter.
RW0 to RW7	Used as pointers and operands of instructions. Note RW0 is also used as a string instruction counter.
RL0 to RL3	Used as long pointers and operands of instructions.

Table 2.4-2 Relationship between Register in Register Bank

	RW0	RL0
	RW1	
	RW2	RL1
	RW3	
R0	RW4	RL2
R1		
R2	RW5	
R3		
R4	RW6	RL3
R5		
R6	RW7	
R7		

2.5 Prefix Codes

Prefix codes can be classified into three types: bank selection prefixes, common register bank prefixes, and flag change suppression prefixes. Adding these prefix codes at the front of instructions can change a part of the operation.

■ Bank Selection Prefixes

The memory space to be used at data access is determined for each addressing mode. By adding bank selection prefixes at the front of an instruction, the memory space for data access by an instruction can be freely selected regardless of the addressing mode.

Table 2.5-1 "Bank Selection Prefixes" lists the bank selection prefixes and the memory spaces selected by them.

Table 2.5-1 Bank Selection Prefixes

Bank selection prefix	Selected space
PCB	Program counter space
DTB	Data space
ADB	Additional space
SPB	The system stack space or user stack space is used depending on the contents of the stack flag at that time.

However, note the following in connection with using bank selection prefixes for the following instructions:

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

Use the bank register specified by the operand regardless of whether there is a prefix.

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

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

○ **I/O access instructions [MOV A, io/MOV io, A/MOVX A, io/MOVW A, io/MOVW io, A/MOV io, #imm8 MOVW io, #imm16 / MOVB A, io:bp / MOVB io:bp, A / SETB io:bp / CLRB io:bp BBC io:bp, rel / BBS io:bp, rel / WBTC, WBTS]**

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

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

The operation of the instruction itself is as normal, however, the prefix has an effect on the next instruction.

○ **POPW ps**

The SSB or the USB is used according to the S flag regardless of whether there is a prefix. The prefix has an effect on the next instruction.

- **MOV ILM, #imm8**

The operation of the instruction itself is normal, however, the prefix has an effect on the next instruction.

- **RETI**

SSB is used regardless of whether there is a prefix.

■ Common Register Bank Prefix (CMR)

To simplify the data exchange between multiple tasks, a relatively easy means of accessing the same register bank regardless of the RP value at that time is required. Adding the common register bank prefix (CMR) in front of instructions that access this register bank simplifies all register access of the instruction to common banks between 000180_H and 00018F_H (register bank selected when RP = 0) regardless of the current RP value.

However, note the following remark about instructions when using the common register bank prefix (CMR):

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

If an interrupt is requested during the execution of a string instruction to which a prefix code has been added, a malfunction occurs after the return from the interrupt because the prefix has become invalid. Therefore, do not add the CMR prefix to the above string instructions.

- **Flag change instructions [AND CCR, #imm8/OR CCR, #imm8/POPW PS]**

The operation of these instructions is normal, but the prefix has an effect on the next instruction.

- **MOV ILM, #imm8**

The operation of this instruction is normal, but the prefix has an effect on the next instruction.

■ Flag Change Suppression Prefix (NCC)

To suppress a flag change, use the flag change suppression prefix code (NCC). By placing the prefix code in front of the instruction to suppress an unwanted flag change, a flag change during the execution of the instruction can be suppressed.

However, note the following remarks about instructions when using the flag change suppression prefix (NCC):

○ 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 after the return from the interrupt because the prefix has become invalid. Therefore, do not add the NCC prefix to the above string instructions.

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

The operation of these instructions is normal, but the prefix has an effect on the next instruction.

○ Interrupt instructions [INT #vct8/INT9/INT addr16/INTP addr24/RETI]

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

○ JCTX @A

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

○ MOV ILM, #imm8

The operation of this instructions is normal, but the prefix has an effect on the next instruction.

2.6 Interrupt Suppression Instructions and Prefix Codes

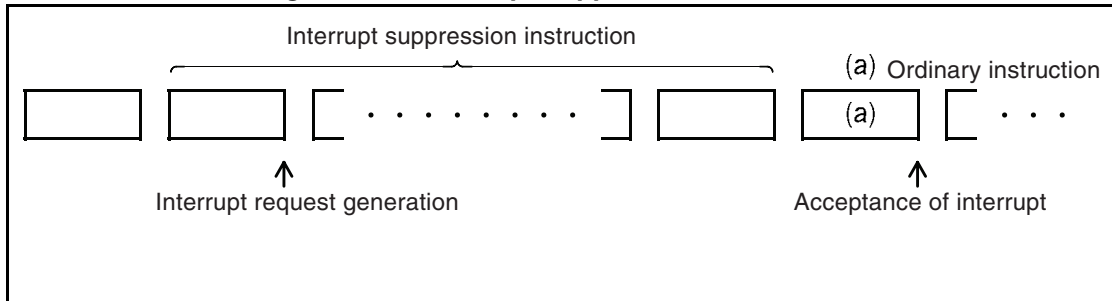
The following 10 types of interrupt suppression instructions do not detect whether there is a hardware interrupt request and ignore any such interrupt request.

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

■ Interrupt Suppression Instructions

As shown in Figure 2.6-1 "Interrupt Suppression Instructions" assume that a valid hardware interrupt request is issued during the execution of an interrupt suppression instruction. This interrupt will only be processed in an instruction other than an interrupt suppression instruction and after the present interrupt suppression instruction.

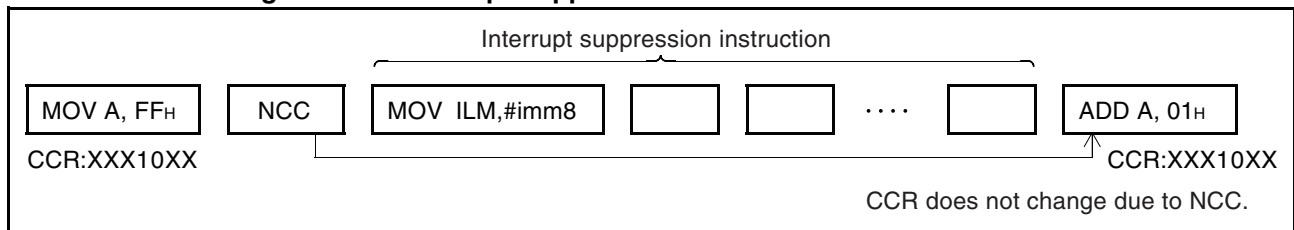
Figure 2.6-1 Interrupt Suppression Instructions



■ Restrictions on Interrupt Suppression and Prefix Instructions

As shown in Figure 2.6-2 "Interrupt Suppression Instructions and Prefix Codes" if a prefix code is added in front of the interrupt suppression instruction, the effect of the prefix code expands to the first instruction other than the interrupt suppression instruction itself after the prefix code.

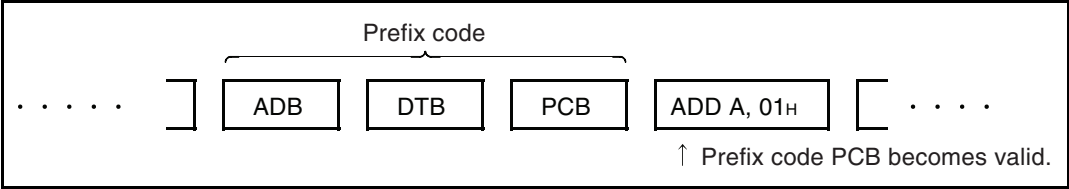
Figure 2.6-2 Interrupt Suppression Instructions and Prefix Codes



■ In the Case of Consecutive Prefix Codes

As shown in Figure 2.6-3 "Consecutive Prefix Codes" when conflicting prefix codes are specified consecutively, only the last prefix code is valid. In the figure below, PCB, ADB, DTB, and SPB are conflicting prefix codes.

Figure 2.6-3 Consecutive Prefix Codes



2.7 Notes on Use of the DIV A, Ri and DIVW A, RWi Instructions

If the DIV A, Ri and DIVW A, RWi instructions are used, set the bank registers to "00_H".

■ Notes on use of the DIV A, Ri and DIVW A, RWi instructions

Table 2.7-1 Notes on use of the DIV A, Ri and DIVW A, RWi instructions (i=0 to 7)

Instruction	Name of the bank register that is affected when the instruction at left is executed	Remainder storage address
DIV A, R0	DTB	(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 8 _H : Lower 16 bits)
DIV A, R1		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 9 _H : Lower 16 bits)
DIV A, R4		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + C _H : Lower 16 bits)
DIV A, R5		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + D _H : Lower 16 bits)
DIVW A, RW0		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 0 _H : Lower 16 bits)
DIVW A, RW1		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 2 _H : Lower 16 bits)
DIVW A, RW4		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 8 _H : Lower 16 bits)
DIVW A, RW5		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + A _H : Lower 16 bits)
DIV A, R2	ADB	(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + A _H : Lower 16 bits)
DIV A, R6		(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + E _H : Lower 16 bits)
DIVW A, RW2		(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + 4 _H : Lower 16 bits)
DIVW A, RW6		(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + E _H : Lower 16 bits)
DIV A, R3	USB SSB*1	(USB*2: Upper 8 bits) + (0180 _H + RP x 10 _H + B _H : Lower 16 bits)
DIV A, R7		(USB*2: Upper 8 bits) + (0180 _H + RP x 10 _H + F _H : Lower 16 bits)
DIVW A, RW3		(USB*2: Upper 8 bits) + (0180 _H + RP x 10 _H + 6 _H : Lower 16 bits)
DIVW A, RW7		(USB*2: Upper 8 bits) + (0180 _H + RP x 10 _H + E _H : Lower 16 bits)

*1: Depending on the S bit in the CCR register

*2: If the S bit in the CCR register is zero

The values of the bank registers (DTB, ADB, USB, SSB) are "00_H", the remainder of the division result is stored in the register of the instruction operand. If the value of the bank register is other than "00_H", the upper 8-bit address is specified by the bank register corresponding to the register of the instruction operand, while the lower 16-bit address becomes the same address of that of the instruction operand. The remainder is stored in the register of the bank specified by the upper eight bits.

Example

If DIV A, R0 is executed where DTB="053_H" and RP="03_H", the address of R0 is "0001B8_H" from "0180_H" + RP ("03_H") x "10_H" + "08_H" (address equivalent to R0).

Here, since the bank register specified in DIV A, R0 is the data bank register (DTB), the remainder is stored at "05301B8_H", which is obtained by prefixing the above address with bank address "053_H". (For information on the Ri and RWi registers, Section 2.4 "General-purpose Registers").

■ Avoiding being subject to the notes

To enable the user to develop programs that are not covered by the notes on the use of DIV A, Ri and DIVW A, RWi instructions, Fujitsu provides modified compilers that do not generate the instructions in Table 2.7-1 "Notes on use of the DIV A, Ri and DIVW A, RWi instructions (i=0 to 7)" and assemblers that add the function of replacing the instructions in Table 2.7-1 with an equivalent instruction string. Use the following compiler and assembler.

- Compiler
 - cc907 V02L06 and later versions and fcc907s V30L02 and later versions
- Assembler
 - asm907a V03L04 and later versions and fasm907s V30L04 (Rev. 300004) and later versions

CHAPTER 3 INTERRUPTS

This chapter describes the features and operation of interrupts.

3.1 "Overview of Interrupts"

3.2 "Interrupt Causes"

3.3 "Interrupt Vectors"

3.4 "Hardware Interrupts"

3.5 "Software Interrupts"

3.6 "Expanded Intelligent I/O Service (EI²OS)"

3.7 "Exceptions because of Executing Undefined Instructions"

3.1 Overview of Interrupts

F²MC-16LX provides interrupt features for suspending the currently executed processing when a certain event occurs and transferring the control to another pre-defined program.

■ Overview of Interrupts

The provided interrupt features can be classified into four types:

- Hardware interrupts: Interrupt due to an event of an internal resource
- Software interrupts: Interrupt due to an event because of a software instruction
- Extended intelligent I/O service (EI²OS): Transfer processing due to an event of an internal resource
- Exception: Suspension due to an exceptional operation

3.2 Interrupt Causes

Table 3.2-1 "Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers" lists interrupt causes, interrupt vectors, and interrupt control registers.

■ Interrupt Causes

Table 3.2-1 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Interrupt cause	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Reset	N	#08	FFFFDC _H	-	-
INT9 instruction	N	#09	FFFFD8 _H	-	-
Exception	N	#10	FFFFD4 _H	-	-
A/D converter	Y	#11	FFFFD0 _H	ICR00	0000B0 _H
Time base timer	N	#12	FFFFCC _H		
DTP0 (external interrupt #0)/ end of UART3 reception	Y	#13	FFFFC8 _H	ICR01	0000B1 _H
DTP1 (external interrupt #1)/ end of UART4 reception	Y	#14	FFFFC4 _H		
DTP2 (external interrupt #2)/ end of UART3 transmission	Y	#15	FFFFC0 _H	ICR02	0000B2 _H
DTP3 (external interrupt #3)/ end of UART4 transmission	Y	#16	FFFFBC _H		
DTP4 to DTP7 (external interrupts #4 to #7)	Y	#17	FFFFB8 _H	ICR03	0000B3 _H
Output compare (channel 1) matching (I/O timer)	Y	#18	FFFFB4 _H		
End of UART2 reception	Y	#19	FFFFB0 _H	ICR04	0000B4 _H
End of UART1 reception	Y	#20	FFFFAC _H		
Input capture (channel 3) fetching (I/O timer)	Y	#21	FFFFA8 _H	ICR05	0000B5 _H
Input capture (channel 2) fetching (I/O timer)	Y	#22	FFFFA4 _H		
Input capture (channel 1) fetching (I/O timer)	Y	#23	FFFFA0 _H	ICR06	0000B6 _H
Input capture (channel 0) fetching (I/O timer)	Y	#24	FFFF9C _H		

CHAPTER 3 INTERRUPTS

Table 3.2-1 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers (Continued)

Interrupt cause	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
8/16-bit PPG0 counter borrow	N	#25	FFFF98 _H	ICR07	0000B7 _H
16-bit reload timer 2 to 0	Y	#26	FFFF94 _H		
Watch prescaler	N	#27	FFFF90 _H	ICR08	0000B8 _H
Output compare (channel 0) matching (I/O timer)	Y	#28	FFFF8C _H		
End of UART2 transmission	Y	#29	FFFF88 _H	ICR09	0000B9 _H
End of measurement by PWC timer/PWC timer overflow	Y	#30	FFFF84 _H		
End of UART1 transmission	Y	#31	FFFF80 _H	ICR10	0000BA _H
16-bit free-run timer (I/O timer) overflow	Y	#32	FFFF7C _H		
End of UART0 transmission	Y	#33	FFFF78 _H	ICR11	0000BB _H
8/16-bit PPG1 counter borrow	N	#34	FFFF74 _H		
End of IEBus reception	y	#35	FFFF70 _H	ICR12	0000BC _H
Start of IEBus transmission	y	#37	FFFF68 _H	ICR13	0000BD _H
End of UART0 reception	y	#39	FFFF60 _H	ICR14	0000BE _H
Flash memory status	N	#41	FFFF58 _H	ICR15	0000BF _H
Delay interrupt	N	#42	FFFF54 _H		

y: An interrupt request flag is cleared by the EI²OS interrupt clear signal. There is a stop request.

Y: An interrupt request flag is cleared by the EI²OS interrupt clear signal.

N: An interrupt request flag is not cleared by the EI²OS interrupt clear signal.

Note:

If a resource has two possible interrupt causes with the same interrupt number, both interrupt request flags for the two interrupt causes are cleared by an EI²OS interrupt clear signal. Therefore, if the EI²OS feature is used for one interrupt cause, the other interrupt feature cannot be used. To resolve this, set the interrupt request permission bit to 0 and use software polling.

3.3 Interrupt Vectors

Table 3.3-1 "List of Interrupt Vectors" shows 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 _H	FFFFFD _H	FFFFFE _H	Not used	#0	None
:	:	:	:	:	:	:
INT 7	FFFFE0 _H	FFFFE1 _H	FFFFE2 _H	Not used	#7	None
INT 8	FFFFDC _H	FFFFDD _H	FFFFDE _H	FFFFDF	#8	(RESET vector)
INT 9	FFFFD8 _H	FFFFD9 _H	FFFFDA _H	Not used	#9	None
INT 10	FFFFD4 _H	FFFFD5 _H	FFFFD6 _H	Not used	#10	<Exception>
INT11	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Not used	#11	A/D converter
INT 12	FFFFCC _H	FFFFCD _H	FFFFCE _H	Not used	#12	Time base timer
INT 13	FFFFC8 _H	FFFFC9 _H	FFFFCA _H	Not used	#13	DTP0 (external interrupt #0)/end of UART3 reception
INT 14	FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Not used	#14	DTP1 (external interrupt #1)/end of UART4 reception
INT 15	FFFFC0 _H	FFFFC1 _H	FFFFC2 _H	Not used	#15	DTP2 (external interrupt #2)/end of UART3 transmission
INT 16	FFFFBC _H	FFFFBD _H	FFFFBE _H	Not used	#16	DTP3 (external interrupt #3)/end of UART4 transmission
INT 17	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Not used	#17	DTP4 to DTP7 (external interrupt #4 to 7)
INT 18	FFFFB4 _H	FFFFB5 _H	FFFFB6 _H	Not used	#18	Output compare (channel 1) matching (I/O timer)
INT 19	FFFFB0 _H	FFFFB1 _H	FFFFB2 _H	Not used	#19	End of UART2 reception
INT 20	FFFFAC _H	FFFFAD _H	FFFFAE _H	Not used	#20	End of UART1 reception
INT 21	FFFFA8 _H	FFFFA9 _H	FFFFAA _H	Not used	#21	Input capture (channel 3) fetching (I/O timer)

CHAPTER 3 INTERRUPTS

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 22	FFFFA4 _H	FFFFA5 _H	FFFFA6 _H	Not used	#22	Input capture (channel 2) fetching (I/O timer)
INT 23	FFFFA0 _H	FFFFA1 _H	FFFFA2 _H	Not used	#23	Input capture (channel 1) fetching (I/O timer)
INT 24	FFFF9C _H	FFFF9D _H	FFFF9E _H	Not used	#24	Input capture (channel 0) fetching (I/O timer)
INT 25	FFFF98 _H	FFFF99 _H	FFFF9A _H	Not used	#25	8/16-bit PPG0 counter borrow
INT 26	FFFF94 _H	FFFF95 _H	FFFF96 _H	Not used	#26	16-bit reload timer 2 to 0
INT 27	FFFF90 _H	FFFF91 _H	FFFF92 _H	Not used	#27	Watch prescaler
INT 28	FFFF8C _H	FFFF8D _H	FFFF8E _H	Not used	#28	Output compare (channel 0) matching (I/O timer)
INT 29	FFFF88 _H	FFFF89 _H	FFFF8A _H	Not used	#29	End of UART2 transmission
INT 30	FFFF84 _H	FFFF85 _H	FFFF86 _H	Not used	#30	End of measurement by PWC timer/PWC timer overflow
INT 31	FFFF80 _H	FFFF81 _H	FFFF82 _H	Not used	#31	End of UART1 transmission
INT 32	FFFF7C _H	FFFF7D _H	FFFF7E _H	Not used	#32	16-bit free-run timer (I/O timer) overflow
INT 33	FFFF78 _H	FFFF79 _H	FFFF7A _H	Not used	#33	End of UART0 transmission
INT 34	FFFF74 _H	FFFF75 _H	FFFF76 _H	Not used	#34	8/16-bit PPG1 counter borrow
INT 35	FFFF70 _H	FFFF71 _H	FFFF72 _H	Not used	#35	End of IEBus reception
INT 37	FFFF68 _H	FFFF69 _H	FFFF6A _H	Not used	#37	Start of IEBus transmission
INT 38	FFFF64 _H	FFFF65 _H	FFFF66 _H	Not used	#38	None
INT 39	FFFF60 _H	FFFF61 _H	FFFF62 _H	Not used	#39	End of UART0 reception
INT 41	FFFF58 _H	FFFF59 _H	FFFF5A _H	Not used	#41	Flash memory status
INT 42	FFFF54 _H	FFFF55 _H	FFFF56 _H	Not used	#42	Delay interrupt
:	:	:	:	:	:	:
INT 254	FFFC04 _H	FFFC05 _H	FFFC06 _H	Not used	#254	None
INT 255	FFFC00 _H	FFFC01 _H	FFFC02 _H	Not used	#255	None

3.4 Hardware Interrupts

A hardware interrupt suspends the program the CPU is executing in response to an interrupt request signal from an internal resource and transfers the control to a program that the user has defined for interrupt processing.

■ Overview of Hardware Interrupts

A hardware interrupt occurs after comparing the interrupt level for an interrupt request with the interrupt level mask register (ILM) in the PS of the CPU and after referencing the contents of the I flag in the PS by hardware if the interrupt condition is satisfied.

The CPU performs one of the following operations when a hardware interrupt occurs:

- Saving data to the system stack of the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers in the CPU.
- Setting the ILM in the PS register. The ILM is automatically set to the same level as the currently requesting interrupt level.
- Incorporating the contents of the corresponding interrupt vector and branching to the interrupt vector.

■ Structure of Hardware Interrupts

The processing related to a hardware interrupt can be classified into the following three structure elements:

○ Internal Resources

Interrupt permission bit, interrupt request bit: Control interrupt requests from a resource.

○ Interrupt Controller

ICR: Assigns an interrupt level, and evaluates the priority among simultaneous interrupt requests.

○ CPU

I and ILM: Compare the request interrupt level with the current level and distinguish between interrupt permission statuses.

Microcode: Contains the steps for interrupt processing.

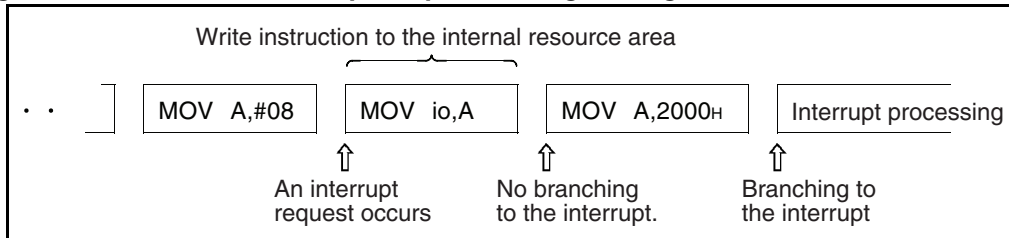
The status of an internal resource is defined by the relevant resource control register, the status of the interrupt controller is defined by ICR, and the status of the CPU is defined by the value of CCR. For using a hardware interrupt, it is necessary to define these three structure parts in advance on the software level. For information on ICR, see the section entitled, "Expanded Intelligent I/O Service Interrupt Control Register (ICR)".

The table of interrupt vectors referenced during interrupt processing is allocated to FFFC00_H to FFFFFF_H , and is shared by hardware and software interrupts.

■ Hardware Interrupt Request during Writing to the Internal Resource Area

No hardware interrupt requests are accepted during writing to the internal resource area. This was implemented to prevent CPU malfunctions due to interrupt conflicts in connection with overwriting the interrupt control registers for each resource. The internal resource area represents the area allocated to the control register or data register of the internal resource rather than the I/O addressing area of 000000_H to 0000FF_H.

Figure 3.4-1 Hardware Interrupt Request During Writing to the Internal Resource Area



■ Interrupt Stop Instruction

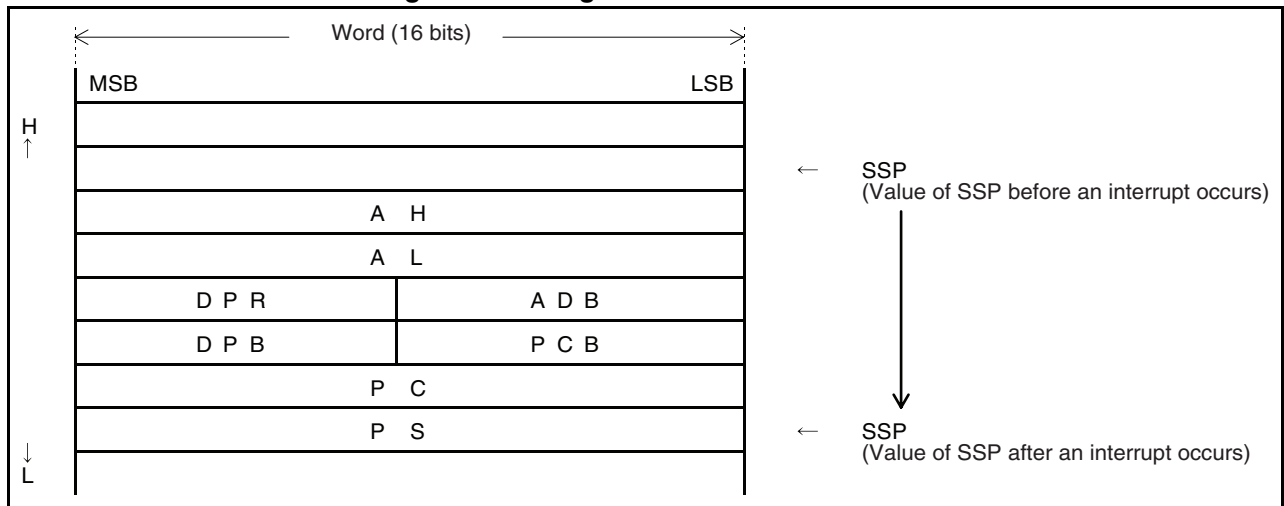
The F²MC-16LX is provided with an interrupt stop instruction that disables the detection of hardware interrupt requests. (See Section 2.6 "Interrupt Suppression Instructions and Prefix Codes".)

■ Multiple Interrupts

The F²MC-16LX CPU supports multiple interrupts. If an interrupt with a higher level than a currently processed interrupt occurs, control is transferred to the interrupt with the higher level after finishing the currently executed instruction. After completing the execution of this interrupt, the control returns to the execution of the previous interrupt. If an interrupt with the same or a lower level occurs during interrupt processing, the new interrupt is put on hold until the currently processed interrupt is completed, unless the contents of the ILM are changed or the respective interrupt levels change by an instruction to change the I flag. The extended intelligent I/O service cannot be started multiple times simultaneously. While one instance of the extended intelligent I/O service is being processed, all other interrupt requests and extended intelligent I/O service requests are put on hold.

■ Saving a Register to the Stack

Figure 3.4-2 Registers Saved to the Stack



■ Notes on the Use of Hardware Interrupts

To avoid a malfunction during a hardware interrupt, it is necessary to clear the interrupt request flag before exiting the corresponding interrupt routine.

When a specific register is read, interrupt request flags that refer to certain resources are cleared automatically. In this case, these registers are read and the interrupt request flag is cleared before exiting the interrupt routine.

3.4.1 Operation of Hardware Interrupts

The internal resources for providing the hardware interrupt request feature include the interrupt request flag and interrupt permission flag. The interrupt request flag indicates whether an interrupt request is present. The interrupt permission flag indicates whether an interrupt request to the CPU by the corresponding internal resource is present. The interrupt request flag is set when a specific event occurs in an internal resource.

Depending on permission by the interrupt permission flag, the resource generates an interrupt request to the interrupt controller.

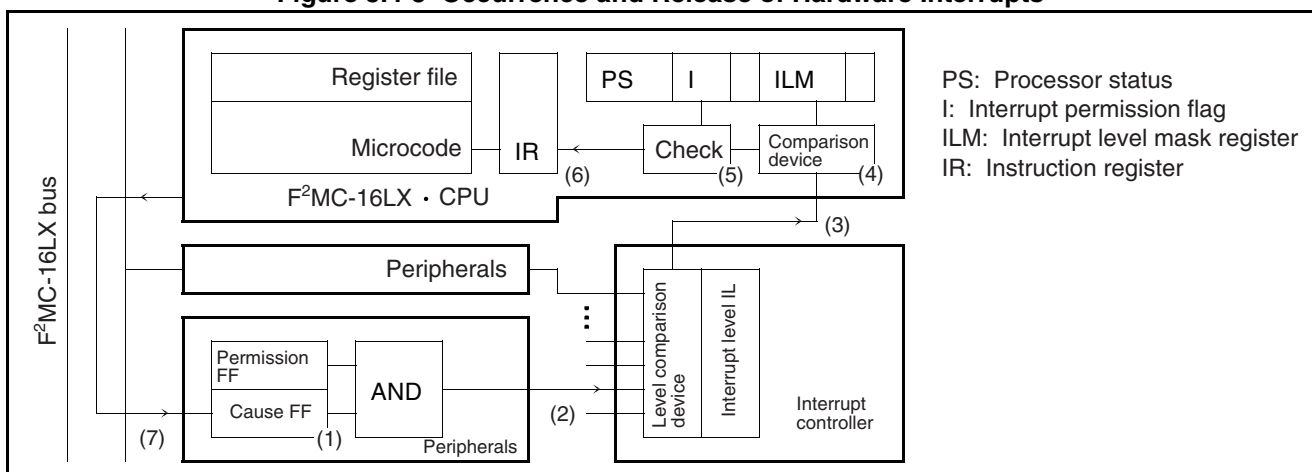
■ Operations of Hardware Interrupts

The interrupt controller compares the interrupt levels (ILs) in the ICR for all interrupt requests received at the same time, selects the request with the highest level (in other words, the value of the respective IL is lowest), and reports it to the CPU. If multiple requests have the same level, the request with the lowest interrupt number is prioritized. The relationship between interrupt requests and the ICR depends on the hardware.

The CPU compares the received interrupt level (IL) with the ILM in the PS register. When the interrupt level (IL) is lower than the ILM and the I bit in the PS register is set to 1, the microcode for interrupt processing is processed after finishing the current instruction. At the beginning of processing the microcode for interrupt processing, the ISE bit in the ICR of the interrupt controller is referenced. After confirming that the ISE bit is set to 0 (this means an interrupt), the main part of interrupt processing is started.

During the main part of interrupt processing, the 12 bytes of PS, PC, PCB, DTB, ADB, DPR, and A are saved to the memory area indicated by SSB and SSP. Three bytes from the interrupt vector are then loaded to PC and PCB. Branch processing is performed by updating the ILM in the PS to the level of the received interrupt request and setting the S flag to 1. Consequently, the instruction to be executed next is the interrupt processing program defined by the user.

Figure 3.4-3 Occurrence and Release of Hardware Interrupts



The meanings of the items 1 to 7 in Figure 3.4-3 "Occurrence and Release of Hardware Interrupts" are described below:

1. An interrupt cause occurs in one of the peripherals.
2. The interrupt permission bit in the peripheral device is referenced. If interrupt permission is set, an interrupt request from the peripheral to the interrupt controller is generated.
3. The interrupt controller that receives an interrupt request, evaluates the priority of simultaneous interrupt requests, and transfers the interrupt level corresponding to the relevant interrupt request 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. When the comparison shows a higher priority level than for the currently processed interrupt, the content of the I flag in the same processor status register is checked.
6. When the check in 5 shows that the I flag is in interrupt permission status, the ILM bit is set to the requested level so that interrupt processing is performed immediately after the currently executed instruction is completed. Thereafter, control is transferred to the interrupt processing routine.
7. The interrupt request ends when the interrupt cause of item 1 is cleared by the interrupt processing routine defined by the user.

The execution time for the interrupt processing steps performed by the CPU in item 6 and 7 is listed below. The time it takes to transfer to the interrupt sequence differs depending on the address to which the stack pointer points.

○ Time required for the interrupt processing by CPU

Delay before the CPU starts an interrupt sequence (The CPU does not start the interrupt sequence in the middle of the execution of an instruction.)

○ Time required to execute an interrupt sequence

Interrupt start : $24 + 6$ times machine cycles according to Table 3.4-1 "Corrective Value of the Number of Cycles for Interrupt Processing"

Interrupt recover: $15 + 6$ times machine cycles according to Table 3.4-1 "Corrective Value of the Number of Cycles for Interrupt Processing" (RETI instruction)

Table 3.4-1 Corrective Value of the Number of Cycles for Interrupt Processing

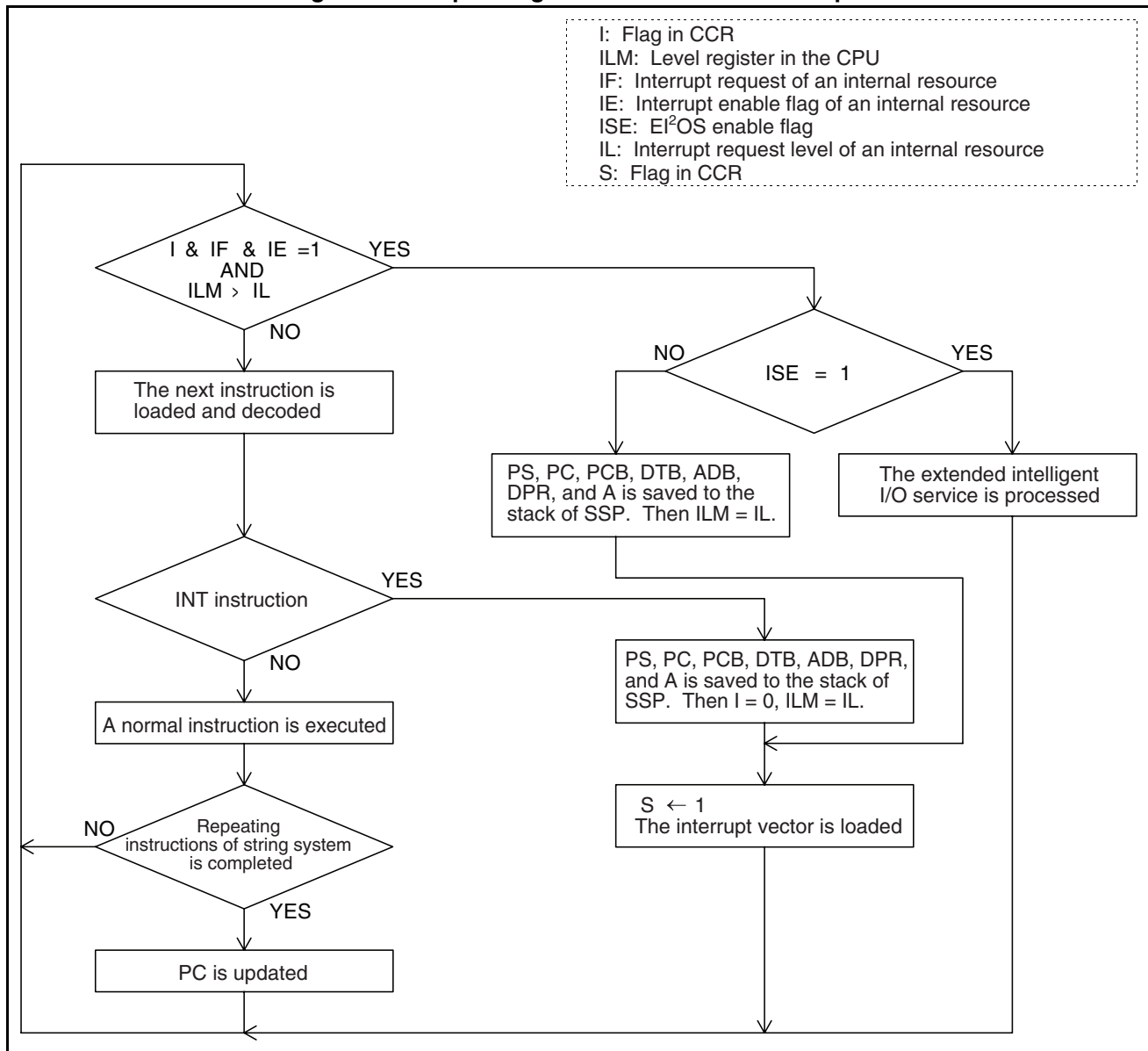
Address to which the stack pointer points	Corrective value of the number of cycles
External area 8-bit data bus	+4
External area even-numbered address	+1
External area odd-numbered address	+4
Internal area even-numbered address	0
Internal area odd-numbered address	+2

3.4.2 Operating Flow for Hardware Interrupts

Figure 3.4-4 "Operating Flow for Hardware Interrupts" shows the flow of operation for hardware interrupts.

■ Operating Flow for Hardware Interrupts

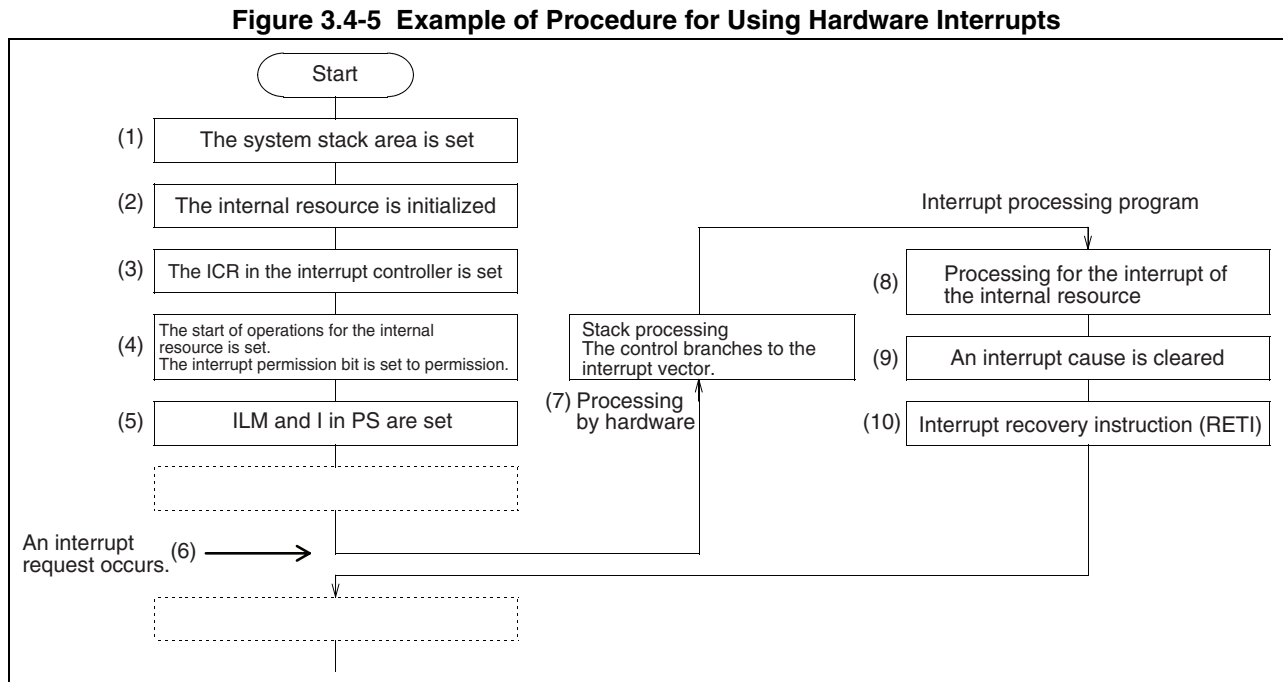
Figure 3.4-4 Operating Flow for Hardware Interrupts



3.4.3 Example of Procedure for Using Hardware Interrupts

Figure 3.4-5 "Example of Procedure for Using Hardware Interrupts" shows an example of a procedure for using hardware interrupts.

■ Example of Procedure for Using Hardware Interrupts



The usage of the items 1 to 10 in Figure 3.4-5 "Example of Procedure for Using Hardware Interrupts" is as follows:

1. The system stack area is set.
2. The internal resource for generating an interrupt request is initialized.
3. The ICR in the interrupt controller is set.
4. The internal resource is set to start operation status. The interrupt permission bit is set to permission.
5. The ILM and I flags in the CPU are set in such a way that an interrupt is accepted.
6. A hardware interrupt request occurs due to an internal resource interrupt.
7. The respective register is saved by the interrupt processing hardware and the control branches to the interrupt processing program.
8. The processing for preventing interrupts in the internal resource is performed by the interrupt processing program.
9. The interrupt request of the internal resource circuit is released.
10. The interrupt recovery instruction is executed and the control is returned to the program that was executed before the branch.

3.5 Software Interrupts

Software interrupts transfer the control from the execution of the program that is currently executed by the CPU to a program for interrupt processing that was defined by the user for this specific instruction.

■ Overview of Software Interrupts

A software interrupt occurs when a software interrupt instruction is executed. The CPU performs one of the following types of processing when a software interrupt occurs:

- Saving data of the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers in the CPU to the system stack.
- Setting the I flag of the PS register. This automatically prohibits further interrupts.
- Determining the value of the corresponding interrupt vector and branching the control to a processing according to the value.

A software interrupt request issued by an INT instruction does not include an interrupt request flag or permission flag. Software interrupt requests are always issued due to the execution of an INT instruction.

The INT instruction does not include an interrupt level. Therefore, the INT instruction does not update the ILM. The INT instruction clears the I flag and puts subsequent interrupt requests on hold.

■ Structure of Software Interrupts

All software interrupts are processed by the CPU.

○ CPU

- Microcode: Interrupt processing step

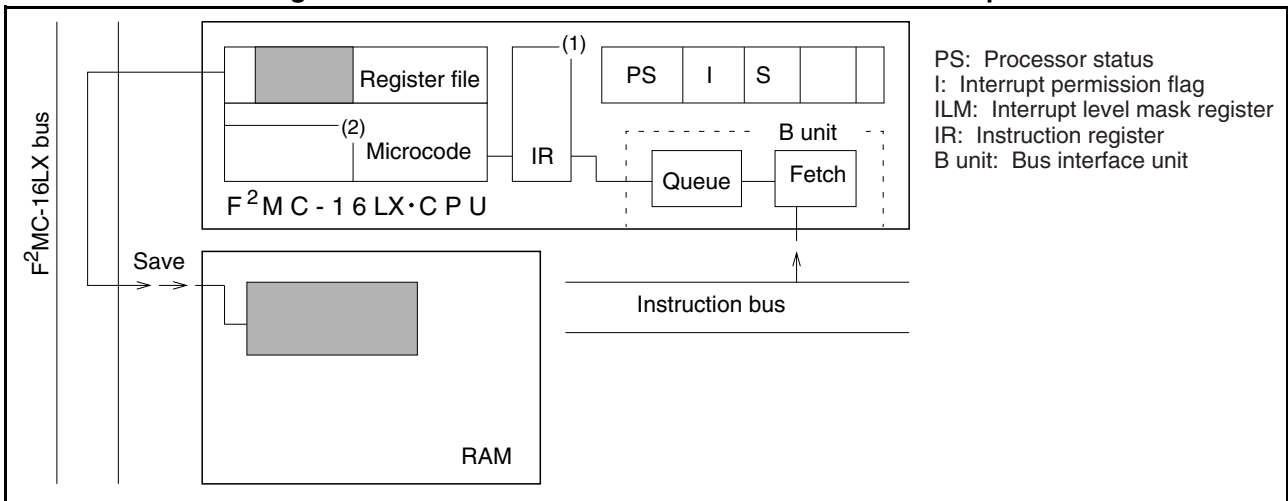
If a software interrupt is issued, it is necessary to execute the corresponding instruction.

As shown in Table 3.3-1 "List of Interrupt Vectors" software interrupts share the interrupt vector area with hardware interrupts. For example, interrupt request number INT 13 is used to indicate not only external interrupt #0/end of UART3 reception as a hardware interrupt cause but also INT #13 as a software interrupt cause. Therefore, external interrupt #0/end of UART3 reception and INT #13 call the same interrupt processing routine.

■ Operation of Software Interrupts

Execution of the microcode for software interrupt processing is started when the CPU loads and executes a software interrupt instruction. The microcode for software interrupt processing saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP. Three bytes from the interrupt vector are then stored to PC and PCB according to the microcode. The I flag is reset (set to 0) and the S flag is set to 1. Consequently, the interrupt processing program defined by the user application program is executed next.

Figure 3.5-1 Occurrence and Release of Software Interrupts



The meanings of the items 1 to 3 in Figure 3.5-1 "Occurrence and Release of Software Interrupts" are as follows:

1. The software interrupt instruction is executed.
2. The internal special CPU register defined by the register file is saved according to the microcode for the software interrupt instruction.
3. The interrupt processing ends by the RETI instruction in the user's interrupt processing routine.

■ Notes on Software Interrupts

If the program bank register (PCB) is FF_H, the vector area of the CALLV instruction overlaps to the table of the INT #vct8 instruction. When designing software, be sure that the CALLV instruction never uses the same address as the INT #vct8 instruction.

3.6 Expanded Intelligent I/O Service (EI²OS)

The expanded intelligent I/O service (EI²OS), which automatically transfers data between an I/O and memory, is a hardware interrupt handling program. Interrupt handling programs ordinarily transfer data between I/O and memory, but the EI²OS can also transfer such data as DMA mode.

However, the use of EI²OS is not possible with the REALOS real time operating system.

■ Overview of the Expanded Intelligent I/O Service (EI²OS)

The expanded intelligent I/O service (EI²OS) provides the following advantages over conventional interrupt handling programs:

- Reduction of the total program size because a transfer program does not have to be generated.
- Improving the transfer rate, because as internal registers are not used for transfer, they do not have to be saved.
- Avoiding unnecessary data transfers, because the I/O system can stop the transfer.
- Capability of selecting increment, decrement, or non-updating of a buffer address
- Capability of selecting increment, decrement, or non-updating of an I/O register address (at updating of a buffer address)

Upon completion of the EI²OS, the CPU automatically branches to the interrupt handling routine after setting a termination condition. Therefore, the user can determine the type of the termination condition.

The hardware for implementing the EI²OS is structured in two separate blocks, each of which contains the following register and descriptor:

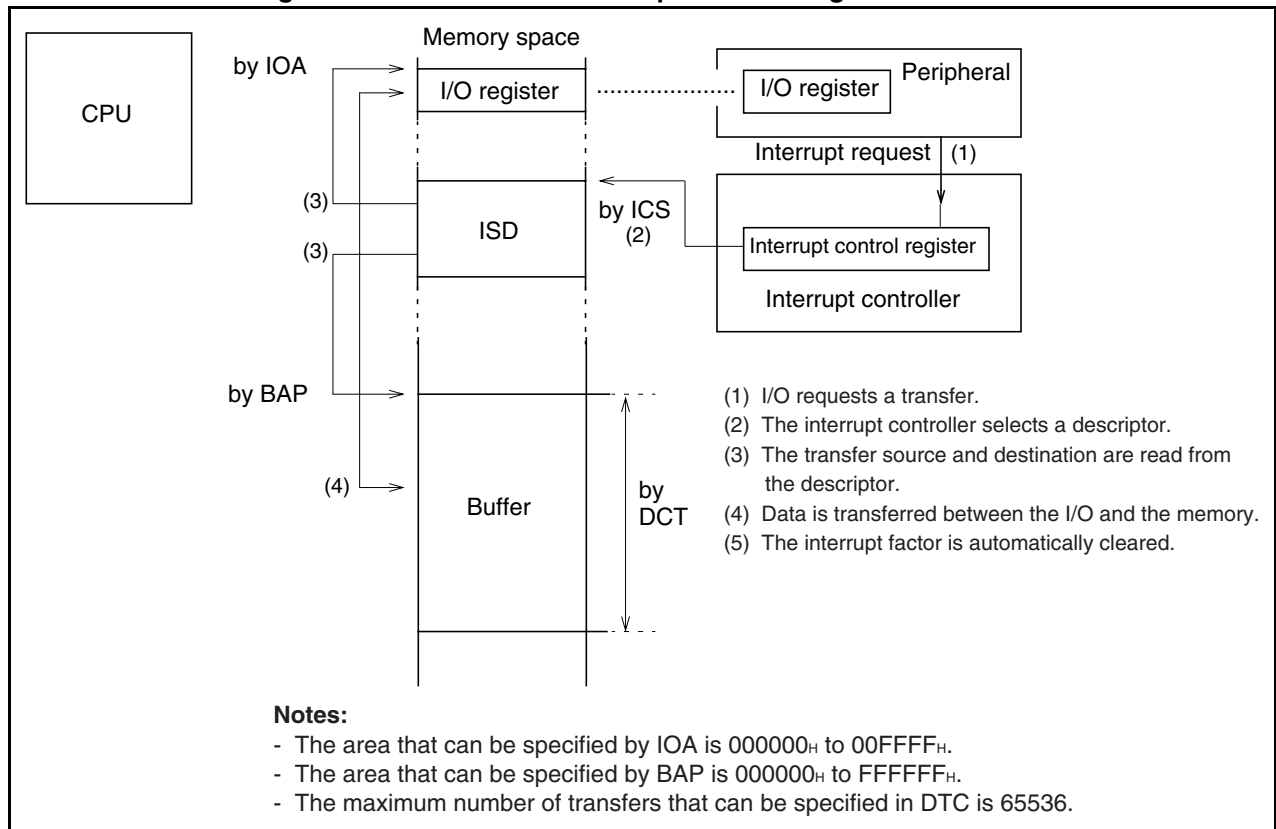
○ Interrupt control register

This register is located in the interrupt controller and indicates an ISD address.

○ Expanded intelligent I/O service descriptor

This descriptor is located in RAM and contains information on the transfer mode, the I/O address, the number of transfers, and the buffer address.

Figure 3.6-1 Overview of the Expanded Intelligent I/O Service



■ Configuration of the Expanded Intelligent I/O Service (EI²OS)

The mechanism of the EI²OS consists of the following four parts:

○ Built-in resources

- Interrupt enable bit and interrupt request bit: Control interrupt requests from resources.

○ Interrupt controller

- ICR: Assigns levels to interrupts, determines a priority of simultaneously requested interrupts, and selects EI²OS operation.

○ CPU

- I and ILM: Compare the requested interrupt level against the current level and determine the status of interrupt capability.
- Micro-code: for EI²OS processing step

○ RAM

- Descriptor: Describes the transfer information for the EI²OS.

3.6.1 Interrupt Control Register (ICR)

The interrupt control register is located in the interrupt controller, which corresponds to all I/Os that support interrupt functions. The interrupt control register has the following three functions:

- Specifying an interrupt level for the corresponding peripheral resource.
- Selecting whether the interrupts of the corresponding peripheral should be handled as normal interrupts or by the expanded intelligent I/O service.
- Selecting a channel for the expanded intelligent I/O service.

Do not access this register with read-modify-write instructions, as this may cause 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 No.
Address: B0 _H to BF _H		ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	When writing
Read/write ⇐		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇐		(0)	(0)	(0)	(0)	(0)	(1)	(1)	(1)	
Address: B0 _H to BF _H		—	—	S1	S0	ISE	IL2	IL1	IL0	⇐ Bit No.
Read/write ⇐		(-)	(-)	(R)	(R)	(R)	(R)	(R)	(R)	When reading
Initial value ⇐		(X)	(X)	(0)	(0)	(0)	(1)	(1)	(1)	

Note:

ICS3 to ICS0 are effective only when the EI²OS is executed. Set the ISE to 1 when executing the EI²OS, and to 0 otherwise. When the EI²OS is not executed, ICS3 to ICS0 may have any value.

ICS1 and ICS0 are effective only during writing, while S1 and S0 are effective only during reading.

[Bits 15 to 12 and 7 to 4] ICS3 to ICS0 (EI²OS channel selection bits)

The bits ICS3 to ICS0 are the channel selection bits for EI²OS.

These bits are write only and specify a channel for the EI²OS.

The value in these bits determines the address of an expanded intelligent I/O service descriptor in the memory. The ICS is initialized to 0000 by a reset.

Table 3.6-1 ICS3 to ICS0 (EI²OS Channel Selection Bits)

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

[Bits 13, 12, 5, and 4] S0 and S1 (EI²OS status bits)

S0 and S1 are the EI²OS termination status bits.

S0 and S1 are read only and allow to determine the termination condition for the termination of the EI²OS. They are initialized to 00 by a reset.

Table 3.6-2 Termination Conditions within the Status of the Expanded Intelligent I/O Service

S1	S0	Termination condition
0	0	During operation of the EI ² OS or when not executing it
0	1	Stopped by count-out
1	0	Reserved
1	1	Stopped by request from a built-in resource

[Bits 11 and 3] ISE (EI²OS enable bits)

The ISE bit specifies whether the EI²OS can be used.

If this bit is 1 when an interrupt request occurs, the EI²OS is executed; if it is 0, the interrupt sequence is executed instead. Also, when the EI²OS is terminated by a count-out or request from the built-in resource, the ISE bit becomes 0. When a corresponding built-in resource does not support the EI²OS function, set the ISE by software to 0. This bit is readable and writable and is initialized to 0 by a reset.

[Bits 10 to 8 and 2 to 0] IL0, IL1, and IL2 (Interrupt level setting bits)

The IL0, IL1, and IL2 bits specify an interrupt level.

These bits specify the interrupt level of the corresponding built-in resource. They are readable and writable. They are initialized to level 7 (no interrupt) by a reset.

Table 3.6-3 Level Settings of Interrupt Level Set Bits

IL2	IL1	IL0	Interrupt level
0	0	0	0 (Highest interrupt level)
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 interrupt level)
1	1	1	7 (No interrupt)

3.6.2 Expanded Intelligent I/O Service Descriptor (ISD)

The expanded intelligent I/O service descriptor is located in internal RAM between 000100_H and 00017F_H. The descriptor contains:

- Control data for data transfer
- Status data
- A buffer address pointer

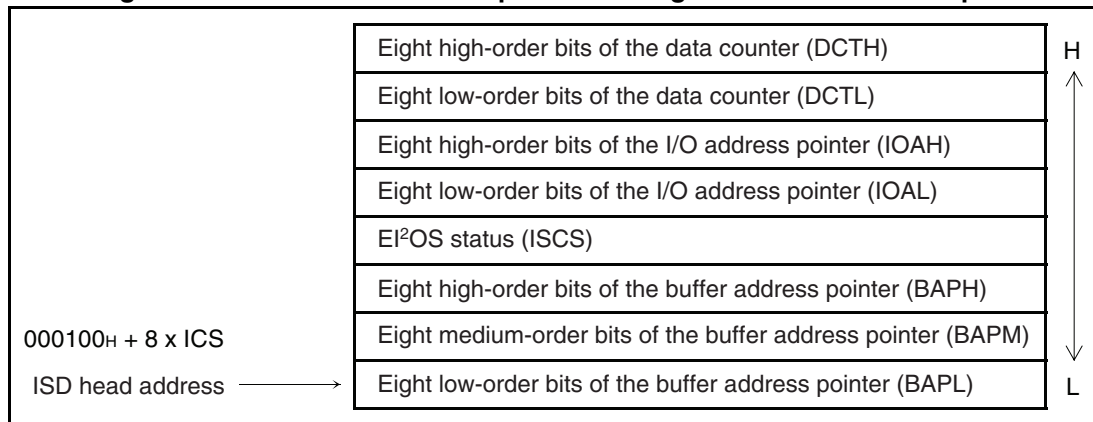
■ Expanded Intelligent I/O Service Descriptor (ISD)

The expanded intelligent I/O service descriptor (ISD) is located in internal RAM between 000100_H and 00017F_H. The descriptor contains:

- Control data for data transfer
- Status data
- A buffer address pointer

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

Figure 3.6-3 Structure of the Expanded Intelligent I/O Service Descriptor



■ Data Counter (DCT)

The DCT is a 16-bit register that functions as a counter of the number of transferred data elements. This counter is decremented by one before the transfer of a data element. When this counter becomes 0, the EI²OS terminates.

Figure 3.6-4 Structure of the Data Counter (DCT)

High-order byte of the data counter	15	14	13	12	11	10	9	8	⇐ Bit No.
	B15	B14	B13	B12	B11	B10	B09	B08	DCTH
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
Low-order byte of the data counter	7	6	5	4	3	2	1	0	⇐ Bit No.
	B07	B06	B05	B04	B03	B02	B01	B00	DCTL
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

■ I/O Register Address Pointer (IOA)

The I/O register address pointer (IOA) is a 16-bit register that indicates the lower address (A15 to A0) of the I/O register that transfers data to or from the buffer. Because the upper part of the register address (A23 to A16) is all 0s, the pointer can specify any I/O register between 000000_H and 00FFFF_H.

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

High-order byte of the I/O address pointer	15	14	13	12	11	10	9	8	⇐ Bit No.
	A15	A14	A13	A12	A11	A10	A09	A08	IOAH
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
Low-order byte of the I/O address pointer	7	6	5	4	3	2	1	0	⇐ Bit No.
	A07	A06	A05	A04	A03	A02	A01	A00	IOAL
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

■ EI²OS Status Register (ISCS)

The EI²OS status register (ISCS) is an eight-bit register that indicates an updating mode (increment or decrement), the format of transferred data (byte or word), and the data transfer direction between the buffer address pointer and the I/O register address pointer. Also, this register indicates whether the buffer address pointer and I/O register address pointer has been updated or left unchanged.

Figure 3.6-6 Configuration of EI²OS Status Register (ISCS)

EI ² OS status register (ISCS)	7	6	5	4	3	2	1	0	⇐ Bit No.
	Reserved	Reserved	Reserved	IF	BW	BF	DIR	SE	
Read/write ⇒	(-)	(-)	(-)	(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. Always set these bits to 0 when setting the ISCS.

[Bit 4] IF

The IF bit indicates whether the I/O register address pointer has been updated or fixed.

Table 3.6-4 Updated/unchanged Specification Bit for the I/O Register Address Pointer (IF)

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

The BW bit indicates the transfer data length.

Table 3.6-5 Bit Specifying the Transfer Data Length (BW)

BW	Function
0	Byte
1	Word

[Bit 2] BF

The BF bit specifies whether the buffer address pointer has been updated or fixed.

Table 3.6-6 Updated/unchanged Specification Bit for Buffer Address Pointer (BF)

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

Note:

When the buffer address pointer is updated, only the lower 16 bits change.

[Bit 1] DIR

The DIR bit indicates the direction of the data transfer.

Table 3.6-7 Setting of the Data Transfer Direction Bit (DIR)

DIR	Setting
0	I/O address pointer ---> Buffer address pointer
1	Buffer address pointer ---> B/O address pointer

[Bit 0] SE

The SE bit controls the termination of the expanded intelligent I/O service by requests from the built-in resource.

Table 3.6-8 EI²OS Termination Control Bit

SE	Setting
0	Service is terminated by a request from the built-in resource.
1	Service is not terminated by a request from the built-in resource.

■ Buffer Address Pointer (BAP)

The buffer address pointer is a 24-bit register for storing the address to be used next by the EI²OS. A separate BAP exists for each channel of the EI²OS, so each channel of EI²OS can transfer data to any address within the 16 MB space.

Note:

When the BF bit of ISCS is set to "0" (update), only the lower 16 bits of BAP change while BAPH does not change.

3.6.3 Operation of the Expanded Intelligent I/O Service (EI²OS)

Figure 3.6-7 "Operational Flow of the Extended Intelligent I/O Service (EI²OS)" shows the operational flow of the expanded intelligent I/O service (EI²OS), while Figure 3.6-8 "Procedural Flow of the Expanded Intelligent I/O Service (EI²OS)" shows the procedural flow of the expanded intelligent I/O service (EI²OS).

■ Operational Flow of the Expanded Intelligent I/O Service (EI²OS)

Figure 3.6-7 Operational Flow of the Extended Intelligent I/O Service (EI²OS)

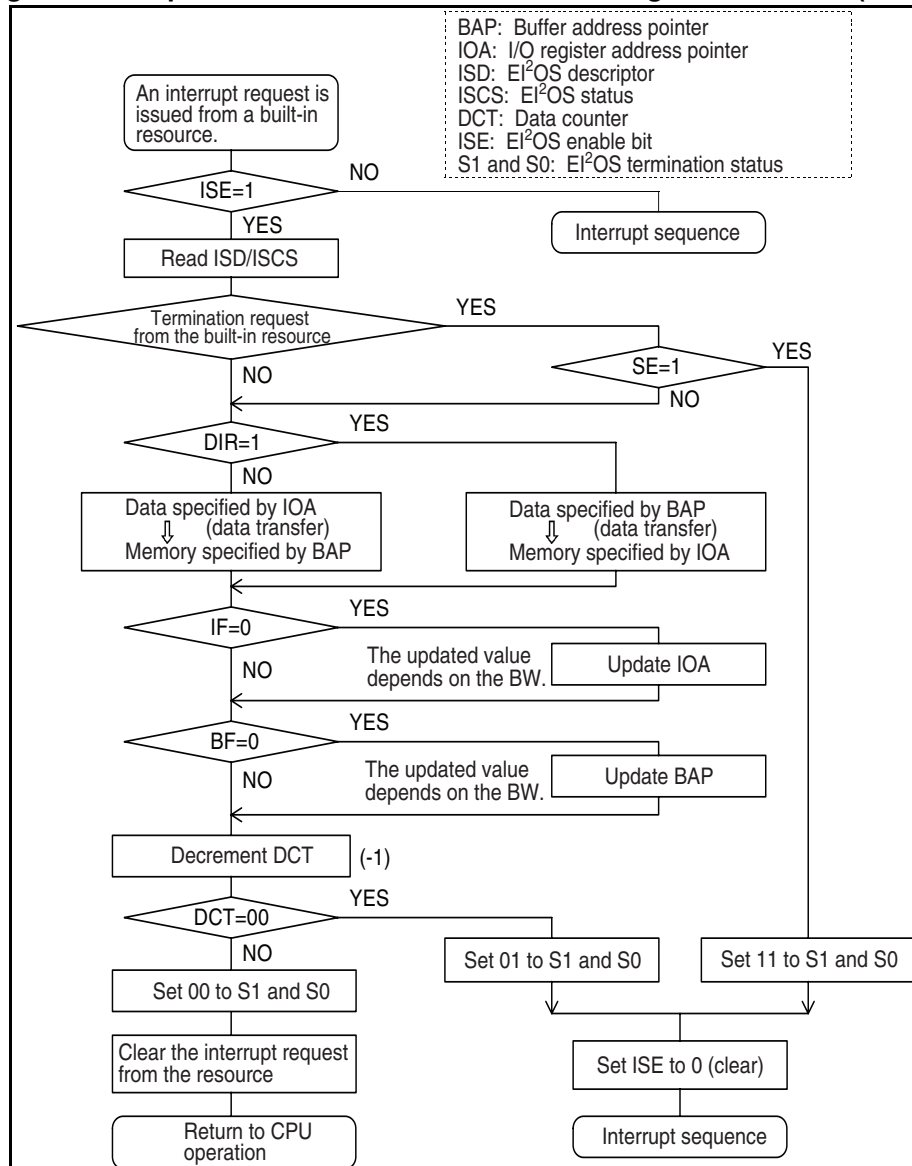
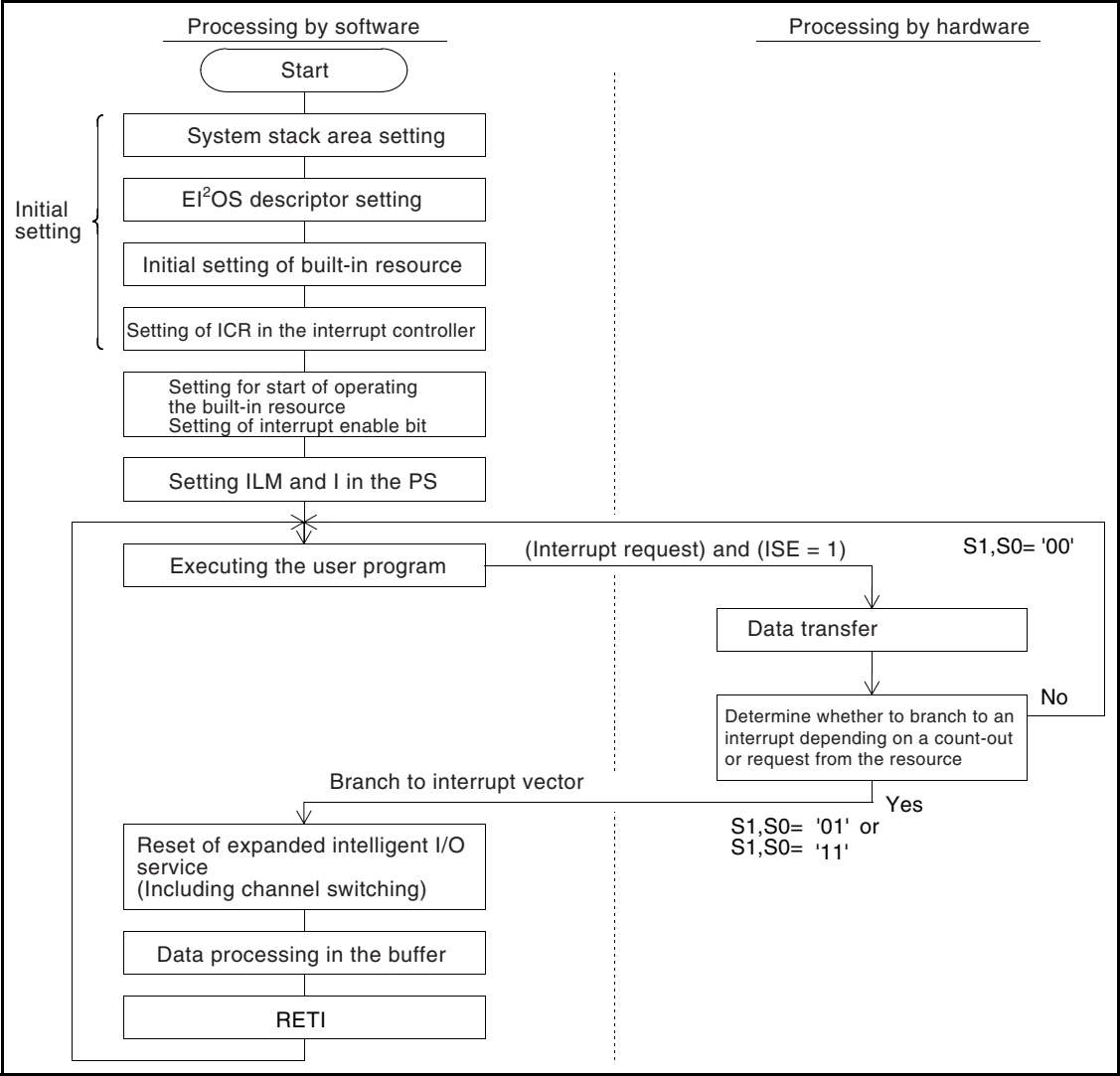


Figure 3.6-8 Procedural Flow of the Expanded Intelligent I/O Service (EI²OS)



3.6.4 Execution Time of the Expanded Intelligent I/O Service (EI²OS)

The execution time of the expanded intelligent I/O service (EI²OS) can be structured into three types:

- During data transfer (while no termination condition is satisfied)
- At a termination request from the resource
- At a count-out

■ Execution Time of the Expanded Intelligent I/O Service (EI²OS)

○ During data transfer (while no termination condition is satisfied)

(Table 3.6-9 "Execution Time of EI²OS During Data Transfer" + Table 3.6-10 "Compensation Value for data Transfer in the Execution Time of EI²OS") machine cycles

Table 3.6-9 Execution Time of EI²OS During Data Transfer

Bit SE of ISCS		Set "0"		Set "1"	
I/O address pointer		Fixed	Updated	Fixed	Updated
Buffer address pointer	Fixed	32	34	33	35
	Updated	34	36	35	37

○ In case of a termination request from the resource

(36 + 6 x Table 3.4-1 "Corrective Value of the Number of Cycles for Interrupt Processing") machine cycles

CHAPTER 3 INTERRUPTS

○ In case of a count-out

(Table 3.6-9 "Execution Time of EI²OS During Data Transfer" + Table 3.6-10 "Compensation Value for data Transfer in the Execution Time of EI²OS" + (21 + 6 x Table 3.4-1 "Corrective Value of the Number of Cycles for Interrupt Processing")) machine cycles

Table 3.6-10 Compensation Value for data Transfer in the Execution Time of EI²OS

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

B: Byte data transfer

8: Word transfer for 8-bit external bus

Even: Word transfer for even address

Odd: Word transfer for odd address

3.7 Exceptions because of Executing Undefined Instructions

When an undefined instruction is executed in the F²MC-16LX, an exception occurs and exception processing is initiated.

The exception processing is basically the same as the processing for an interrupt.

When an exception within the instructions is detected, the control is transferred from normal processing to exception processing. Generally, exception processing is a result of an unpredicted operation. Therefore, use it only for debugging, for software recovery in an emergency, and similar cases.

■ Occurrence of Exceptions because of Executing Undefined Instructions

The F²MC-16LX handles all codes not defined in the instruction map as undefined instructions. When an undefined instruction is executed, the same type of processing as for the software interrupt instruction, "INT 10", is performed. In other words, after saving the contents of AL, AH, DPR, DTB, ADB, PCB, PC, and PS to the system stack, the control sets the I flag to 0, sets the S flag to 1, and then branches to the routine specified by the vector of interrupt number 10. The PC contents saved to the stack consist of the address where the undefined instruction is stored. If an undefined code was detected for an instruction code of two or more bytes, the PC value indicates the address where the undefined code is stored. Therefore, it is possible but ineffectual to recover the system with an RETI instruction because the same exception will occur again.

CHAPTER 4 GENERATING AND RESETTING CLOCKS

This chapter describes clock and reset functions and operations.

4.1 "Clock Generator"

4.2 "Reset Causes"

4.3 "Operation after a Reset Is Released"

4.1 Clock Generator

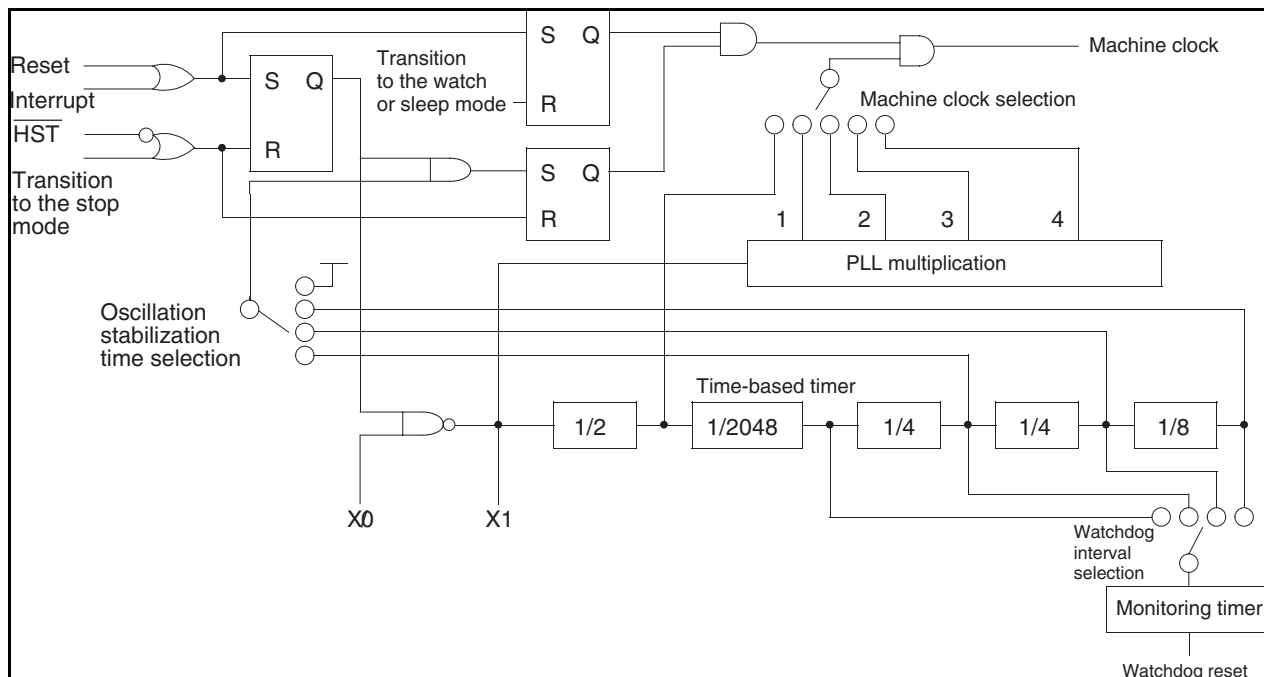
The clock generator controls internal clock operations such as the sleep, watch, and stop modes and the PLL clock multiplication function. This internal clock is called the machine clock. One cycle of the machine clock is used as a machine cycle. The clock generated by OSC oscillation is called the main clock. The clock generated by internal VCO oscillation is called the PLL clock.

■ Notes as to 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 resource circuits do not operate normally. For example, if the OSC oscillation frequency 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 this frequency cannot be specified.

Figure 4.1-1 Clock Generator Block Diagram



4.2 Reset Causes

The five types of reset causes 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 $\overline{\text{RST}}$ pin
- Occurrence of a reset request by software

■ Reset Causes

When the stop mode is released or a power-on reset occurs, operation starts after the oscillation stabilization time has elapsed. When a reset cause occurs, the F²MC-16LX immediately stops executing the current processing and enters the reset release wait state. The machine clock and watchdog function initial states differ depending on the reset cause.

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

Note:

Because the external reset input is sampled in synchronization with the internal clock in 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 cause occurs, the address generated by each device during reset is undefined. External bus access signals such as $\overline{\text{RD}}$ and $\overline{\text{WR}}$ become inactive.

Table 4.2-1 Reset Causes

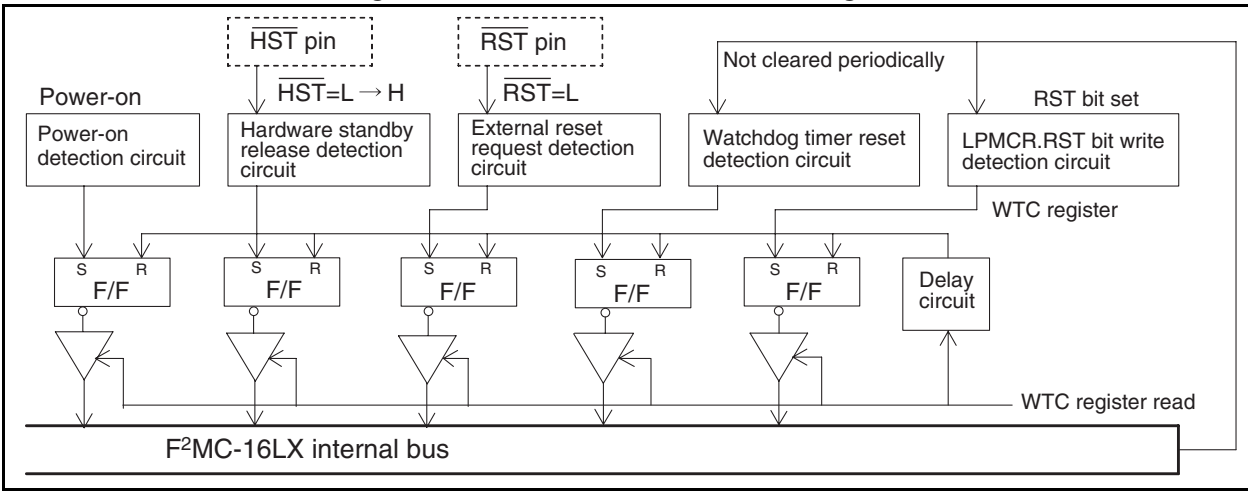
Reset	Cause	Machine clock		Watch-dog timer	Oscillation stabilization wait
		at sub clock	at PLL clock		
Power-on	When the power is turned on	Main clock *	Main clock *	Stop	Yes
Hardware standby	"L" level input to $\overline{\text{HST}}$ pin	Main clock *	Main clock *	Stop	Yes
Watch-dog timer	Watch-dog timer overflow	Main clock *	Main clock *	Stop	Yes
External pin	"L" level input to $\overline{\text{RST}}$ pin	Main clock * or PLL clock	PLL clock	Previous status maintained	No
Software	0 written to the RST bit in the LPMCR register	Main clock * or PLL clock	PLL clock	Previous status maintained	No

*: $f_{\text{OSC}}/2$ (f_{OSC} : the source oscillation)

- When the reset input is received in the stop or hardware standby mode, the oscillation stabilization time is required for any reset cause.
- The oscillation stabilization time required for a power-on reset is fixed to 2^{18} cycles of OSC oscillation. The oscillation stabilization time required for another reset is determined by WS1 and WS0 in the clock selection register.

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

Figure 4.2-1 Reset Cause Bit Block Diagram



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

This rule does not apply to a power-on reset. Because when the PONR bit is 1, the values of other bits do not indicate normal reset causes, a software program must be created that ignores the values of other bits when the PONR bit is 1.

Table 4.2-2 Correspondence between the Reset Causes and the Values of the Reset Cause Bits

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

*: The value before reset is retained.

Note:

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

For the configuration of the watchdog control register and the reset cause bits, see Chapter 9 "WATCHDOG TIMER".

4.3 Operation after a Reset is Released

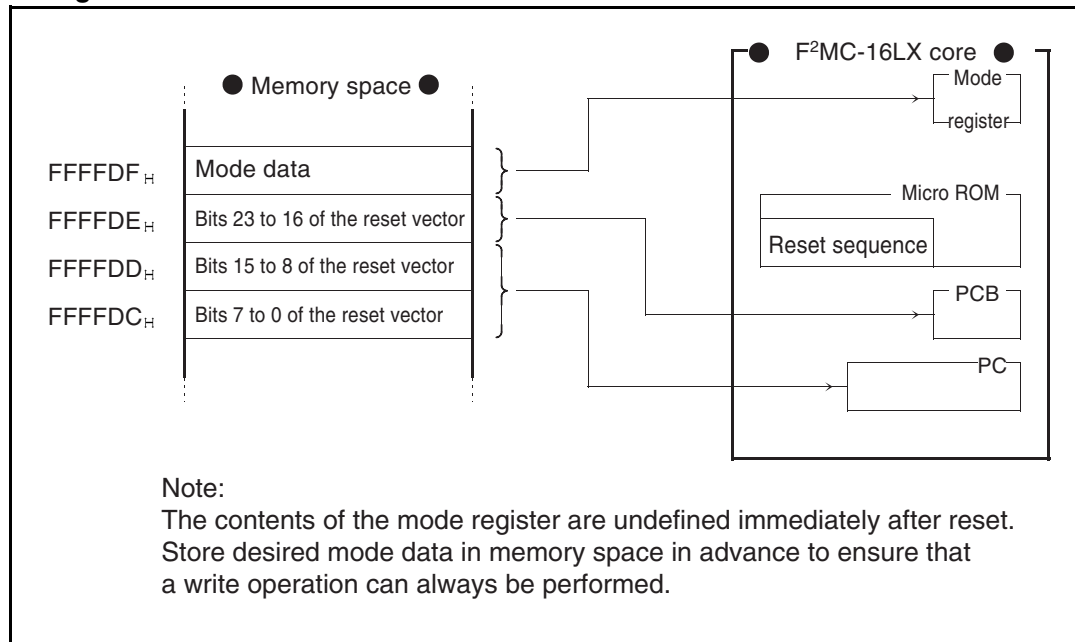
When a reset cause is removed, the F²MC-16LX immediately outputs the address at which the reset vector is stored and fetches the reset vector and mode data. A 4-byte area at FFFFDC_H to FFFFDF_H is allocated for the reset vector and mode data. The reset vector and mode data are transferred to corresponding registers by hardware after the reset is released.

■ Operation after a 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. Because 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, when the microcontroller is to be used in the single chip mode or internal ROM and external bus mode, it is recommended to specify the internal vector mode using the mode pins.

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

Figure 4.3-1 Locations and Destinations of the Reset Vector and of Mode Data



CHAPTER 4 GENERATING AND RESETTING CLOCKS

■ Registers not Initialized by Reset Input

This microcontroller contains registers initialized only by a power-on reset.

Table 4.3-1 "Registers not Initialized by Reset Input" lists registers not initialized by each reset cause.

Table 4.3-1 Registers not Initialized by Reset Input

Type of reset		CKSCR					WTC	LPMCR	
		WS1	WS0	MCS	CS1	CS0	WDCS	CG1	CG0
Software reset (Only \overline{RST} is used.)		N	N	N	N	N	N	N	N
Watchdog reset		N	N	Y	N	N	N	Y	Y
Power-on reset		Y	Y	Y	Y	Y	Y	Y	Y
Hardware standby	Main mode	N	N	Y	N	N	N	Y	Y
	Sub mode	Y	Y	Y	Y	Y	Y	Y	Y

WS1 and WS0: Set the oscillation stabilization time for the main clock.

MCS: Specifies the machine clock (0 = PLL clock or 1 = main clock).

CS1 and CS0: Set the multiplication factor for the PLL clock.

WDCS: Specifies the input clock source for the watchdog timer (0 = watch timer or 1 = time base timer).

Y: Initialized

N: Not initialized

In particular, handle the MCS bit carefully because it sets the machine clock. For example, if power-on does not satisfy the power-on reset specification, no power-on reset occurs. For this reason, the internal operating frequency may become outside the valid operation range, because MCS is not initialized, and the microcontroller may not operate normally.

If the CPU crashes for some reason and MCS, CS1, or CS0 is rewritten, the internal operating frequency may also become outside the valid operation range. The microcontroller may not be able to recover normally from this status by \overline{RST} input only (however, if the internal watchdog state occurs, MCS is initialized and the microcontroller operates normally).

When either of the above cases occurs, use of \overline{HST} plus \overline{RST} (connecting \overline{HST} and \overline{RST} with a jumper) is recommended.

Table 4.3-2 "Registers not Initialized by Reset Input" lists registers that are not initialized by reset input using \overline{HST} plus \overline{RST} . Note that the operation status after the reset is released differs depending on the reset input type, \overline{HST} plus \overline{RST} reset input, or only \overline{RST} input, as listed in Table 4.3-2 "Registers not Initialized by Reset Input".

Table 4.3-2 Registers not Initialized by Reset Input

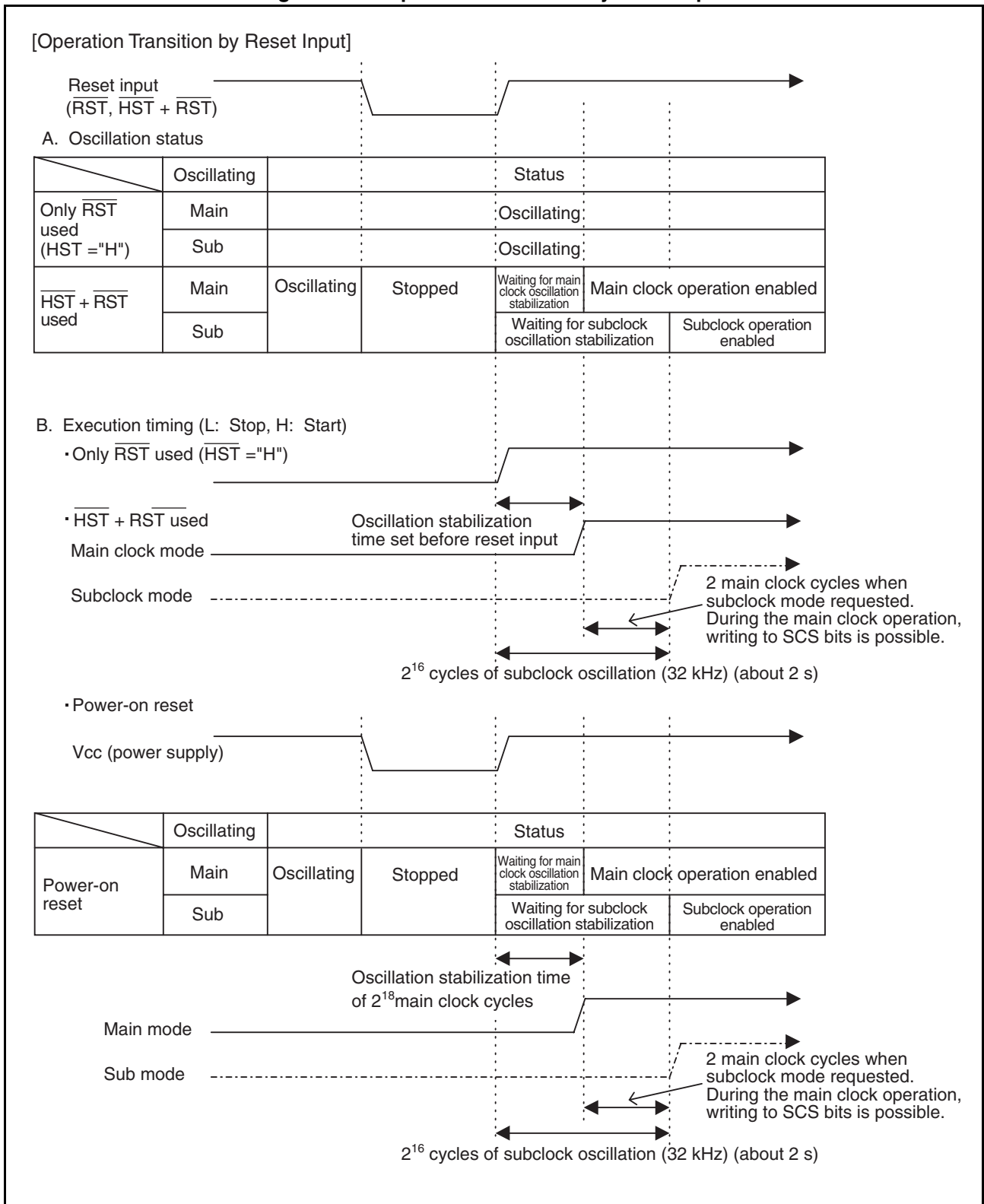
Type of reset		CKSCR					WTC	LPMCR	
		WS1	WS0	MCS	CS1	CS0	WDCS	CG1	CG0
$\overline{HST} + \overline{RST}$	Main mode	N	N	Y	N	N	N	Y	Y
	Sub mode *	Y	Y	Y	Y	Y	Y	Y	Y

Y: Initialized

N: Not initialized

*: Including the sub mode transition period.

Figure 4.3-2 Operation Transition by Reset Input



CHAPTER 5 LOW-POWER CONSUMPTION CONTROL CIRCUIT

This chapter describes the functions and operation of the low-power consumption control circuit (intermittent CPU operation function, oscillation stabilization time, and clock multiplication function).

- 5.1 "Overview of the Low-power Consumption Control Circuit"
- 5.2 "Low-power Consumption Mode Control Register (LPMCR)"
- 5.3 "Clock Selection Register (CKSCR)"
- 5.4 "Operation of the Low-power Consumption Control Circuit"
- 5.5 "Intermittent CPU Operation Function"
- 5.6 "Setting the Oscillation Stabilization Time for the Main Clock"
- 5.7 "Switching the Machine Clock"
- 5.8 "Status Transition"
- 5.9 "Status Transition Diagrams for Low Power-Consumption Modes"

5.1 Overview 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 sleep mode
- Stop mode
- Sub clock mode
- Sub sleep mode
- Watch mode
- Hardware standby mode

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

■ Overview of the Low-power Consumption Control Circuit

○ Main clock mode and main sleep mode

The microcontroller only operates using the oscillation clock (OSC oscillation). The main clock is used as the operating clock and the PLL clock (VCO oscillation) is stopped.

○ PLL sleep mode and main sleep mode

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

○ Watch mode

Only the time-based timer is operating.

○ Stop mode and hardware standby mode

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

The intermittent CPU operation function intermittently operates the clock supplied 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 lowered by the above intermittent operation while supplying the internal peripherals with a high-speed clock.

A PLL clock multiplication factor can be selected among 1, 2, 3, and 4 using the CS1 and CS0 bits in the clock selection register.

The WS1 and WS0 bits can be used to set the oscillation stabilization time for the main clock required when the stop or hardware standby mode is released.

5.1 Overview of the Low-power Consumption Control Circuit

Note:

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

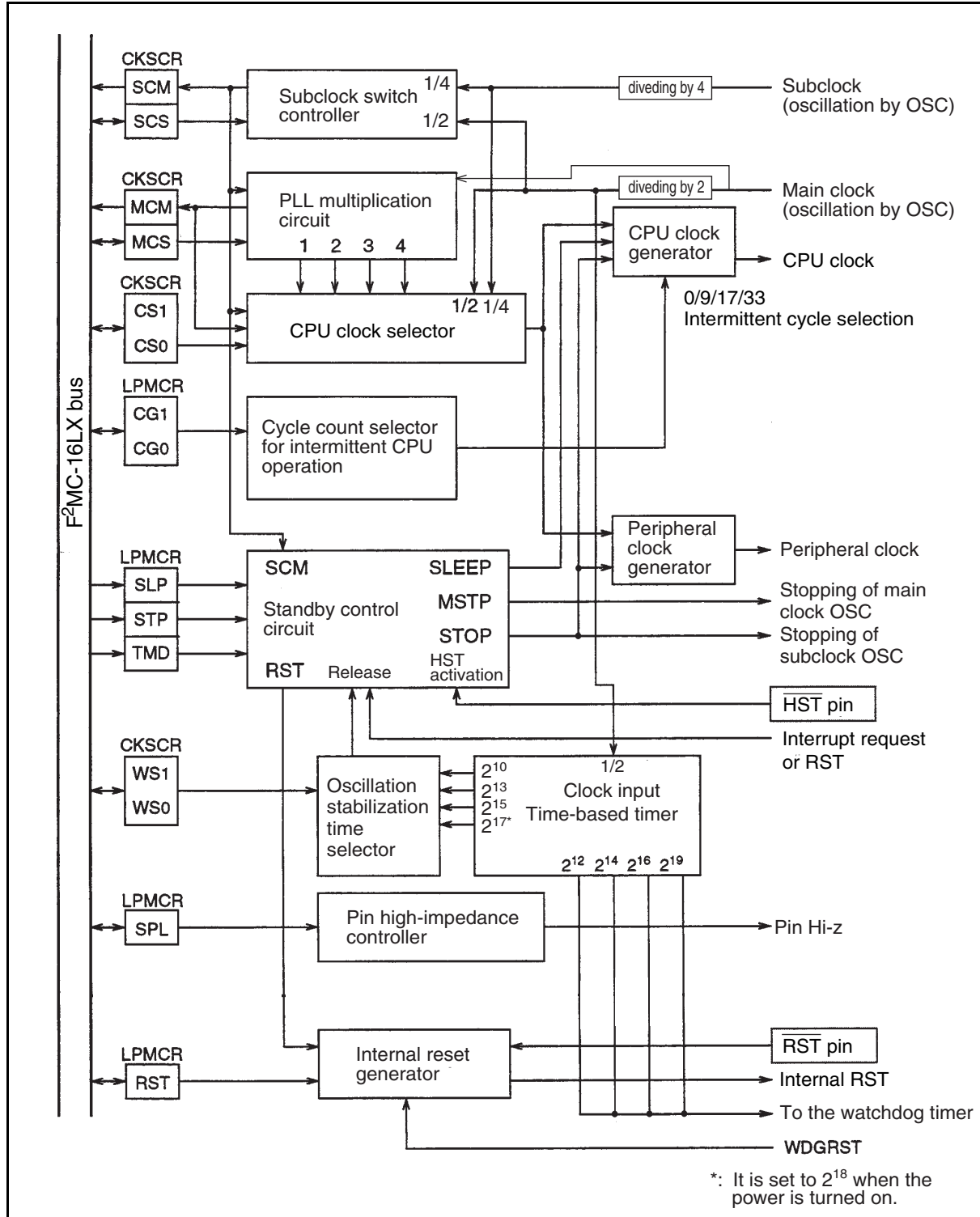
If the mode is switched to another clock mode or low power consumption mode before completion of switching, the mode may not be switched.

Figure 5.1-1 Registers in the Low-power Consumption Control Circuit

Low-power consumption mode control register									⇐ Bit No.
Address: 0000A0H	7	6	5	4	3	2	1	0	
	STP	SLP	SPL	RST	TMD	CG1	CG0	-	LPMCR
Read/write ⇒	(W)	(W)	(R/W)	(W)	(-)	(R/W)	(R/W)	(-)	
Initial value ⇒	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	
Clock selection register									⇐ Bit No.
Address: 0000A1H	15	14	13	12	11	10	9	8	
	SCM	MCM	WS1	WS0	SCS	MCS	CS1	CS0	CKSCR
Read/write ⇒	(-)	(R)	(R/W)	(R/W)	(-)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	

■ Block Diagram of the Low-power Consumption Control Circuit

Figure 5.1-2 Block Diagram of the Low-power Consumption Control Circuit and Clock Generator



5.2 Low-power Consumption Mode Control Register (LPMCR)

The low-power consumption mode control register (LPMCR) sets various types of power consumption-related operating modes together with the clock selection register.

■ Low-power Consumption Mode Control Register (LPMCR)

Figure 5.2-1 Register in the Low-power Consumption Control Circuit

Low-power consumption mode control register	7	6	5	4	3	2	1	0	↔ Bit No.
Address: 0000A0 _H	STP	SLP	SPL	RST	TMD	CG1	CG0	Reserved	LPMCR
Read/write →	(W)	(W)	(R/W)	(W)	(R/W)	(R/W)	(R/W)	(-)	
Initial value →	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

○ Notes on Accessing the low-power Consumption Mode Control Register

Writing the low-power consumption mode control register causes a transition to a low-power consumption mode (stop or sleep mode). Use an instruction listed in Table 5.2-1 "Instructions to be used to Cause A Transition to a Low-power Consumption Mode" for such a transition. Causing a transition to a low-power consumption mode using another instruction may result in a malfunction. Any instruction can be used to control a function of the low-power consumption mode control register other than the function that causes the transition to a low-power consumption mode.

Write word data in the low-power consumption mode control register at an even address. A transition to the low-power consumption mode by writing data at an odd address may result in a malfunction.

Table 5.2-1 Instructions to be used to Cause A Transition to a Low-power Consumption Mode

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

[Bit 7] STP

Writing 1 to the STP bit causes a transition to the pseudo watch mode (CKSCR:MCS = 0 & SCS=1) or the stop mode (CKSCR:MCS=1 or SCS=0). Writing 0 performs no operation. When a reset occurs or the watch or stop mode is released, this bit is cleared to 0.

This bit is a write-only bit. The read value is always 0.

[Bit 6] SLP

Writing 1 in SLP causes a transition to the sleep mode. Writing 0 performs no operation. When a reset occurs or the sleep or stop mode is released, this bit is cleared to 0. Writing 1 to the STP and SLP bits simultaneously causes a transition to the pseudo watch mode or the stop mode.

This bit is a write-only bit. The read value is always 0.

[Bit 5] SPL

When SPL is 0, the external pin levels are retained in the watch or stop mode. When it 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. This bit is a read/write bit.

[Bit 4] RST

Writing 0 in the RST bit generates the internal reset signal for three machine cycles. Writing 1 performs no operation. When this bit is read, the value is 1.

[Bit 3] TMD

Two clocks system

Writing 0 in the TMD bit causes a transition to the watch mode. Writing 1 in this bit creates no operation. When a reset occurs or the watch or stop mode is released, this bit is cleared to 0.

This bit is a write-only bit. The read value is always 1.

One clock system

Always write 1.

[Bits 2 and 1] CG1 and CG0

The CG1 and CG0 bits set the temporary stop cycle count for the intermittent CPU operation function. When a reset occurs as a result of power-on, hardware standby, or watchdog, these bits are initialized to 00 but are not initialized by a reset caused by another reset cause. These bits are read/write bits.

The intermittent CPU operation function stops the clock supplied to the CPU for the specified time and delays the start of the internal bus cycle in the following case:

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 lowered by supplying the internal peripherals with a high-speed clock.

[Bit 0] Reserved

This bit must be set to "0".

Table 5.2-2 Settings of the Low Power-Consumption Mode Control Register (CG1 and CG0 Bits)

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)

5.3 Clock Selection Register (CKSCR)

The clock selection register (CKSCR) sets and controls the CPU machine clock and sets the oscillation stabilization time required at power-on or oscillation recovery.

■ Clock Selection Register (CKSCR)

Figure 5.3-1 Clock Selection Register (CKSCR)

Clock selection register	15	14	13	12	11	10	9	8	↔ Bit No.
Address: 0000A1H	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

Two clocks system

The SCM bit indicates whether the main clock or the subclock is selected as the machine clock. When this bit is 0, it indicates that the subclock is selected. When this bit is 1, it indicates that the main clock is selected.

When the SCS bit is 0 and the SCM bit is 1, this is an indication that the machine clock is being switched from the main clock to the subclock. When the SCS bit is 1 and the SCM bit is 0, this is an indication that the machine clock is being switched from the subclock to the main clock.

One clock system

The read value is always 1.

[Bit 14] MCM

This bit indicates whether the main or the PLL clock is selected as the machine clock. When this bit is 0, it indicates that the PLL clock is selected. When this bit is 1, it indicates that the main clock is selected.

If the MCS bit is 0 and the MCM bit is 1, this is an indication of the PLL clock oscillation stabilization wait time. Also, the oscillation stabilization wait time of the PLL clock is fixed at 2^{13} main clock cycles.

[Bits 13 and 12] WS1 and WS0

The WS1 and WS0 bits set the oscillation stabilization time for the main clock to be applied after the stop or hardware standby mode is released. These bits are initialized to 11 by a power-on reset but are not initialized by a reset caused by another reset cause.

They are read/write bits.

Table 5.3-1 Settings of Clock Selection Register (WS1 and WS0 Bits)

WS1	WS0	Oscillation stabilization time (OSC oscillation: 4 MHz)
0	0	$2^{10}/F_{ch}$: About 256 μs (2^{10} OSC oscillation cycles)

Table 5.3-1 Settings of Clock Selection Register (WS1 and WS0 Bits)

WS1	WS0	Oscillation stabilization time (OSC oscillation: 4 MHz)
0	1	About 2.05 ms (2^{13} OSC oscillation cycles)
1	0	About 8.19 ms (2^{15} OSC oscillation cycles)
1	1	About 32.77 ms (2^{17} OSC oscillation cycles)(*1)

*1: Approx. 65.54 ms (2^{18} counts of source oscillation) at power-on.

[Bit 11] SCS

Two clocks system

The SCS bit indicates whether the main clock or the subclock is used as the machine clock. If this bit is 0, the subclock is selected. If 0 is written when the bit is 1, the mode is switched to the subclock mode by synchronizing with the subclock (about 130 μ s). If 1 is written when the bit is 0, the oscillation stabilization wait time of the main clock is generated and the time-base timer is cleared automatically. If SCS and MCS are both 0, SCS has priority and the subclock is selected.

One clock system

Always write 1.

[Bit 10] MCS

MCS 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-based timer to generate the oscillation stabilization time for the PLL clock. The TBOF bit in the time-based timer control register is also cleared. The oscillation stabilization time for the PLL clock is fixed to 2^{13} main clock cycles. (When the OSC oscillation frequency is 4 MHz, the oscillation stabilization time is about 2 ms.)

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.)

Note:

The oscillation stabilization wait time of the subclock (about 2 s) is generated when the power is turned on or the stop mode is canceled. Thus, if the mode is switched from the main clock mode to the subclock mode during this period, an oscillation stabilization wait time is generated.

Before writing 0 in the MCS bit when it is 1, set the TBIE bit or CPU ILM bit so that the time-based timer interrupt is masked. For eight machine cycles after 1 is written in the MCS bit, 0 may not be able to be written in this bit. Write 0 after eight machine cycles.

[Bits 9 and 8] CS1 and CS0

CS1 and CS0 select a multiplication factor for the PLL clock. They are not initialized when a reset caused by an external pin, the RST bit, or watchdog occurs or if the hardware standby mode is released but are initialized to 00 by a power-on reset.

When the MCS bit is 0, write operation is suppressed. Set the MCS bit to 1 (main clock mode), then write the CS bits.

These bits are read/write bits.

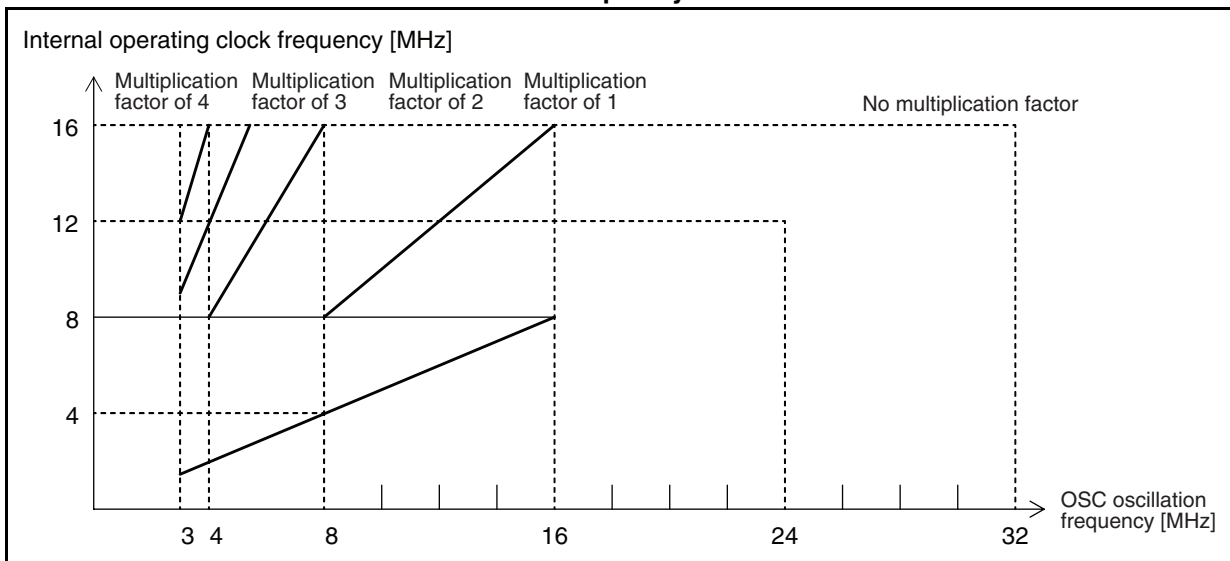
Table 5.3-2 Settings of Clock Selection Register (CS1 and CS0 Bits)

CS1	CS0	Multiplication factor	Internal operating clock (obtained by multiplying OSC oscillation by the multiplication factor)		
			When the OSC oscillation frequency is 4 MHz	When the OSC oscillation frequency is 8 MHz	When the OSC oscillation frequency is 16 MHz
0	0	1	Setting prohibited	8 MHz	16 MHz
0	1	2	8 MHz	16 MHz	Setting prohibited
1	0	3	12 MHz	Setting prohibited	Setting prohibited
1	1	4	16 MHz	Setting prohibited	Setting prohibited

Note:

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, the OSC oscillation frequency 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 this frequency cannot be specified.

Figure 5.3-2 Relationships between the OSC Oscillation Frequency and Internal Operating Clock Frequency

5.4 Operation of the Low-power Consumption Control Circuit

Table 5.4-1 "Operating States in the Low-power Consumption Mode" lists the operating states in the low-power consumption mode.

■ Operation of the Low-power Consumption Control Circuit

Table 5.4-1 Operating States in the Low-power Consumption Mode (Two clock system)

	Transition condition	Subclock oscillation	Main clock oscillation	Machine clock	CPU	Peripherals	Pins	Released by
Subclock mode*1	SCS=0 MCS=x	Operating	Stopped	Operating	Operating	Operating	Operating	Reset interrupt
Subclock sleep mode*1	SCS=0 MCS=x SLP=1	Operating	Stopped	Operating	Stopped	Operating	Operating	Reset interrupt
Main clock sleep mode	SCS=1 MCS=1 SLP=1	Operating	Operating	Operating	Stopped	Operating	Operating	Reset interrupt
PLL clock sleep mode	SCS=1 MCS=0 SLP=1	Operating	Operating	Operating	Stopped	Operating	Operating	Reset interrupt
Pseudo watch mode (SPL=0)	SCS=1 MCS=0 SLP=1	Operating	Operating	Stopped	Stopped	Stopped	Retained	Reset interrupt*2
Stop mode (SPL=1)	SCS=1 MCS=0 STP=1	Operating	Operating	Stopped	Stopped	Stopped	Hi-z	Reset interrupt*2
Watch mode (SPL=0)	SCS=x MCS=x TMD=0	Operating	Stopped	Stopped	Stopped	Stopped	Retained	Reset interrupt*3
Watch mode (SPL=1)	SCS=x MCS=x TMD=0	Operating	Stopped	Stopped	Stopped	Stopped	Hi-z	Reset interrupt*3
Stop mode (SPL=0)	MCS=1 or SCS=0 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Retained	Reset interrupt*4
Stop mode (SPL=1)	MCS=1 or SCS=0 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Hi-z	Reset interrupt*4
Hardware standby mode	$\overline{\text{HST}}=\text{L}$	Stopped	Stopped	Stopped	Stopped	Stopped	Hi-z	$\overline{\text{HST}}=\text{H}$

*1: Do not set this for the one clock system.

*2: Watch prescaler, time-base timer, and external interrupt

*3: Watch prescaler and external interrupt

*4: External interrupt

5.4 Operation of the Low-power Consumption Control Circuit

Table 5.4-2 Low Power Consumption Mode Operating Status (one clock system)

	Transition condition	Subclock oscillation	Main clock oscillation	Machine clock	CPU	Peripherals	Pins	Released by
Main sleep	SCS=1 MCS=1 SLP=1	-	Operating	Operating	Stopped	Operating	Operating	External reset interrupt
POLL sleep	SCS=1 MCS=0 SLP=1	-	Operating	Operating	Stopped	Operating	Operating	External reset interrupt
Pseudo watch (SPL=0)	SCS=1 MCS=0 STP=1	-	Operating	Stopped	Stopped	Stopped	Retained	External reset interrupt*5
Pseudo watch (SPL=1)	SCS=1 MCS=0 STP=1	-	Operating	Stopped	Stopped	Stopped	Hi-z	External reset interrupt*5
Stop (SPL=0)	SCS=1 MCS=1 STP=1	-	Stopped	Stopped	Stopped	Stopped	Retained	External reset interrupt*6
Stop (SPL=1)	SCS=1 MCS=1 STP=1	-	Stopped	Stopped	Stopped	Stopped	Hi-z	External reset interrupt*6
Hardware standby	$\overline{\text{HST}}=\text{L}$	-	Stopped	Stopped	Stopped	Stopped	Hi-z	$\overline{\text{HST}}=\text{H}$

*5: Time-base timer and external interrupt

*6: External interrupt

5.4.1 Sleep Mode

In the sleep mode, only the clock supplied to the CPU is stopped. The CPU stops, but the peripheral circuits continue operation.

■ Transition to the Sleep Mode

Writing 1 in the SLP and TMD bits of the low power-consumption mode control register and 0 in the STP bit of the same register causes the standby control circuit to enter the sleep mode.

If an interrupt request occurs when 1 is written in the SLP bit, the standby control circuit does not enter the sleep mode. If the CPU is in the interrupt disabled state, it executes the next instruction; if the CPU is in the interrupt enabled state, it immediately branches to the interrupt processing routine.

In the sleep mode, the dedicated registers, e.g., accumulator, and the internal RAM retain their contents. Even in the sleep mode, the external bus hold function operates and enters the hold state when a hold request is received.

■ Releasing the Sleep Mode

The low-power consumption control circuit cancels the sleep mode by resetting or generating an interrupt request.

○ Return by resetting

The circuit is initialized to the main clock mode by inputting a reset

○ Return by an interrupt

If a peripheral circuit or another device generates an interrupt request at an interrupt level higher than 7 in sleep mode, the low-power consumption control circuit cancels sleep mode. After sleep mode is cancelled, the interrupt is processed in the same way as are ordinary interrupts. If an interrupt is accepted by the settings of I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes interrupt processing. If an interrupt is not accepted, the CPU continues processing starting with the instruction following the one specifying sleep mode.

5.4.2 Pseudo Watch Mode

In the pseudo watch mode, all clock operations other than source clock oscillation (of the main clock and subclock), watch timer, and time base timer is stopped. Also, most chip functions stop.

Whether retaining the status of each I/O pin set immediately before the transition to the pseudo watch mode or setting each I/O pin to high impedance can be specified. Use the SLP bit of the low power-consumption mode control register (LPMCR) for this specification.

■ Transition to the Pseudo Watch Mode

Writing 1 in the SCS bit of the clock selection register (CKSCR), 0 in the MCS bit of the same register, 1 in the TMD bit of the low power-consumption mode control register, and 1 in the STP bit of the same register causes the standby control circuit to enter the pseudo watch mode.

If an interrupt request occurs when 1 is written in the STP bit, the standby control circuit does not enter the pseudo watch mode.

In the pseudo watch mode, the internal RAM and dedicated registers, e.g., accumulator, retain their contents.

■ Releasing the Pseudo Watch Mode

The standby control circuit releases the pseudo watch mode when a reset signal is input or an interrupt occurs.

○ Return by external reset

The circuit is initialized to main clock mode by an external reset.

○ Return by an interrupt

If a watch prescaler, time-base timer, or external interrupt generates an interrupt request at an interrupt level higher than 7 in pseudo watch mode, the low-power consumption control circuit cancels pseudo watch mode. After the pseudo watch mode is released, the interrupt is processed in a normal way. If an interrupt is accepted by the settings of I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes interrupt processing. If an interrupt is not accepted, the CPU continues processing starting with the next instruction before the system entered pseudo watch mode.

5.4.3 Watch Mode

In the watch mode, all clock operations other than subclock oscillation and the watch timer is stopped. Most chip functions stop.

Whether to retain the status of each I/O pin set immediately before the transition to the watch mode or to set each I/O pin to high impedance can be specified. Use the SPL bit of the low power-consumption mode control register (LPMCR) for this specification. However, the watch mode cannot be used in the one clock system.

■ Transition to the Watch Mode

Writing 0 in the TMD bit of the low power-consumption mode control register causes the standby control circuit to enter the watch mode.

If an interrupt request occurs when 1 is written in the TMD bit, the standby control circuit does not enter the watch mode.

In the watch mode, dedicated registers, e.g., accumulator, and the internal RAM retain their contents.

In the watch mode, the external bus hold function stops. If a hold request is input, it is not accepted. If a hold request is input during a transition to the watch mode, the \overline{HAK} signal may not become low with the bus set to high impedance.

■ Releasing the Watch Mode

The standby control circuit releases the watch mode when a reset signal is input or an interrupt occurs.

○ Return by resetting

If watch mode is cancelled by a reset factor, the low-power consumption control circuit first cancels watch mode, then enters the oscillation stabilization wait reset state. The reset sequence is executed after expiration of the oscillation stabilization wait time.

○ Return by an interrupt

If a watch prescaler or external interrupt generates an interrupt request at an interrupt level higher than 7 in watch mode, the low-power consumption control circuit cancels watch mode and enters the subclock mode immediately. After the low-power consumption control circuit enters subclock mode, the interrupt is processed in the same way as are ordinary interrupts. If an interrupt is accepted by the settings of I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes interrupt processing. If an interrupt is not accepted, the CPU continues processing starting with the next instruction before the system entered watch mode.

Note:

Normally, an interrupt is executed after execution of the instruction following the one specifying watch mode. If the transition to watch mode and the acceptance of the external bus hold request occur simultaneously, however, transition to interrupt processing might occur before the next instruction is executed.

5.4.4 Stop Mode

In the stop mode, all clock oscillation (subclock and main clock) is stopped and all chip functions stop. Therefore, data can be retained with the lowest power consumption.

Whether to retain the status of each I/O pin set immediately before the transition to the stop mode or to set each I/O pin to high impedance can be specified. Use the SPL bit of the low power-consumption mode control register (LPMCR) for this specification.

■ Transition to the Stop Mode

Writing 0 in the SCS bit of the clock selection register (CKSCR), 1 in the MCS bit of the same register, and 1 in the STP bit of the low power-consumption mode control register (LPMCR) causes the standby control circuit to enter the stop mode.

If an interrupt request may occur when 1 is written in the STP bit, the standby control circuit does not enter the stop mode.

In the stop mode, dedicated registers, e.g., accumulator, and the internal RAM retain their contents.

■ Releasing the Stop Mode

The standby control circuit releases the stop mode when a reset signal is input or an interrupt occurs. When returning from the stop mode, the oscillation of the operating clock has stopped, so the low-power consumption control circuit first enters the oscillation stabilization wait time, then cancels the stop mode.

○ Cancel by resetting

If stop mode is cancelled by a reset factor, the low-power consumption control circuit first cancels stop mode, then enters the oscillation stabilization wait reset state. The reset sequence is executed after expiration of the oscillation stabilization wait time.

○ Cancel by an interrupt

If an external interrupt generates an interrupt request at an interrupt level higher than 7 in stop mode, the low-power consumption control circuit cancels stop mode. After stop mode is cancelled, the interrupt is processed in the same way as are ordinary interrupts after the expiration of the oscillation stabilization wait time of the main clock. This time is specified by the oscillation stabilization wait time selection bits (WS1, WS0) of the clock selection register (CKSCR). If an interrupt is accepted by the settings of I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes interrupt processing. If an interrupt is not accepted, the CPU continues processing starting with the next instruction before the system entered stop mode.

Note:

Normally, an interrupt is executed after execution of the instruction following the one specifying stop mode. If the transition to stop mode and the acceptance of the external bus hold request occur simultaneously, however, transition to interrupt processing might occur before the next instruction is executed.

5.4.5 Hardware Standby Mode

In the hardware standby mode, when the $\overline{\text{HST}}$ 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 $\overline{\text{HST}}$ 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 the dedicated registers such as the accumulator are initialized.

■ Releasing the Hardware Standby Mode

The hardware standby mode can be released only using the $\overline{\text{HST}}$ pin.

When the $\overline{\text{HST}}$ 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 has elapsed, the standby control circuit releases the internal reset. The CPU then starts execution from the reset sequence.

5.5 Intermittent CPU Operation Function

The intermittent CPU operation function stops the clock supplied to the CPU for the specified time and delays the start of the internal bus cycle in the following case:

- When registers, internal memory (ROM, RAM, I/O, and resources), and the external bus are accessed

■ Intermittent CPU Operation Function

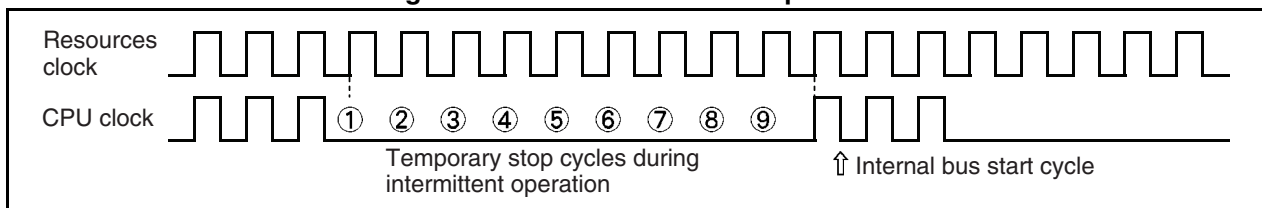
The intermittent CPU operation function can be used to perform processing with low-power consumption because the CPU execution speed is lowered by supplying the internal resources with a high-speed clock. Use the CG1 and CG0 bits to select the temporary stop cycle count for the clock supplied to the CPU.

The external bus operation itself is performed using the same clock as the resources.

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

- Add to the normal execution time the compensation value obtained by multiplying the number of accesses to registers, internal memory, internal resources, and the external bus by the temporary stop cycle count.

Figure 5.5-1 Intermittent CPU Operation



5.6 Setting the Oscillation Stabilization Time for the Main Clock

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

■ Setting the Oscillation Stabilization Time for the Main Clock

Resets other than a power-on reset do not initialize these bits. These bits are initialized to 11_B when a power-on reset occurs. For this reason, the oscillation stabilization time at power-on is about 2^{18} OSC oscillation cycles.

5.7 Switching the Machine Clock

This section explains how to switch and initialize the machine clock. However, subclocks cannot be used with the one clock system.

■ Switching between the Main and PLL Clocks

Writing to the MCS bit of the CKSCR register switches the machine clock between the main and the PLL clocks.

Rewriting the MCS bit from 1 to 0 switches the machine clock from the main clock to the PLL clock after the PLL clock oscillation stabilization time (2^{13} machine clock cycles) passes.

Rewriting the MCS bit from 0 to 1 switches the machine clock from the PLL clock to the main clock when a PLL clock edge coincides with a main clock edge (1 to 8 PLL clock cycles after).

The machine clock is not switched immediately after rewriting the MCS bit. Therefore, reference the MCM bit and check that the machine bit has been switched when operating each peripheral depending on the machine clock.

■ Switching between the Main Clock and Subclock

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

Rewriting the SCS bit from 1 to 0 switches the machine clock from the main clock to the subclock by synchronizing with subclock (about 130 μ s).

Rewriting the SCS bit from 0 to 1 switches the machine clock from the subclock to the main clock after the main clock oscillation stabilization time passes.

The machine clock is not switched immediately after rewriting the SCS bit. Therefore, reference the SCM bit and check that the machine bit has been switched when operating each peripheral depending on the machine clock.

Note:

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

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

■ Initializing the Machine Clock

The MCS and SCS bits cannot be initialized by a reset that uses the $\overline{\text{RST}}$ external reset pin or the RST bit.

The MCS and SCS bits are initialized to "1" by any other reset.

■ Initializing the Machine Clock

The MCS and SCS bits are not initialized by any reset cause from an external pin or by the RST bit. These bits are initialized to 1 by other reset causes.

Figure 5.7-1 "Transition of Clock Selection Status (1) (Two clocks system No.1)" to Figure 5.7-3 "Transition of Clock Selection Status (3) (One Clock System)" show the transition of clock selection status.

Figure 5.7-1 Transition of Clock Selection Status (1) (Two clocks system No.1)

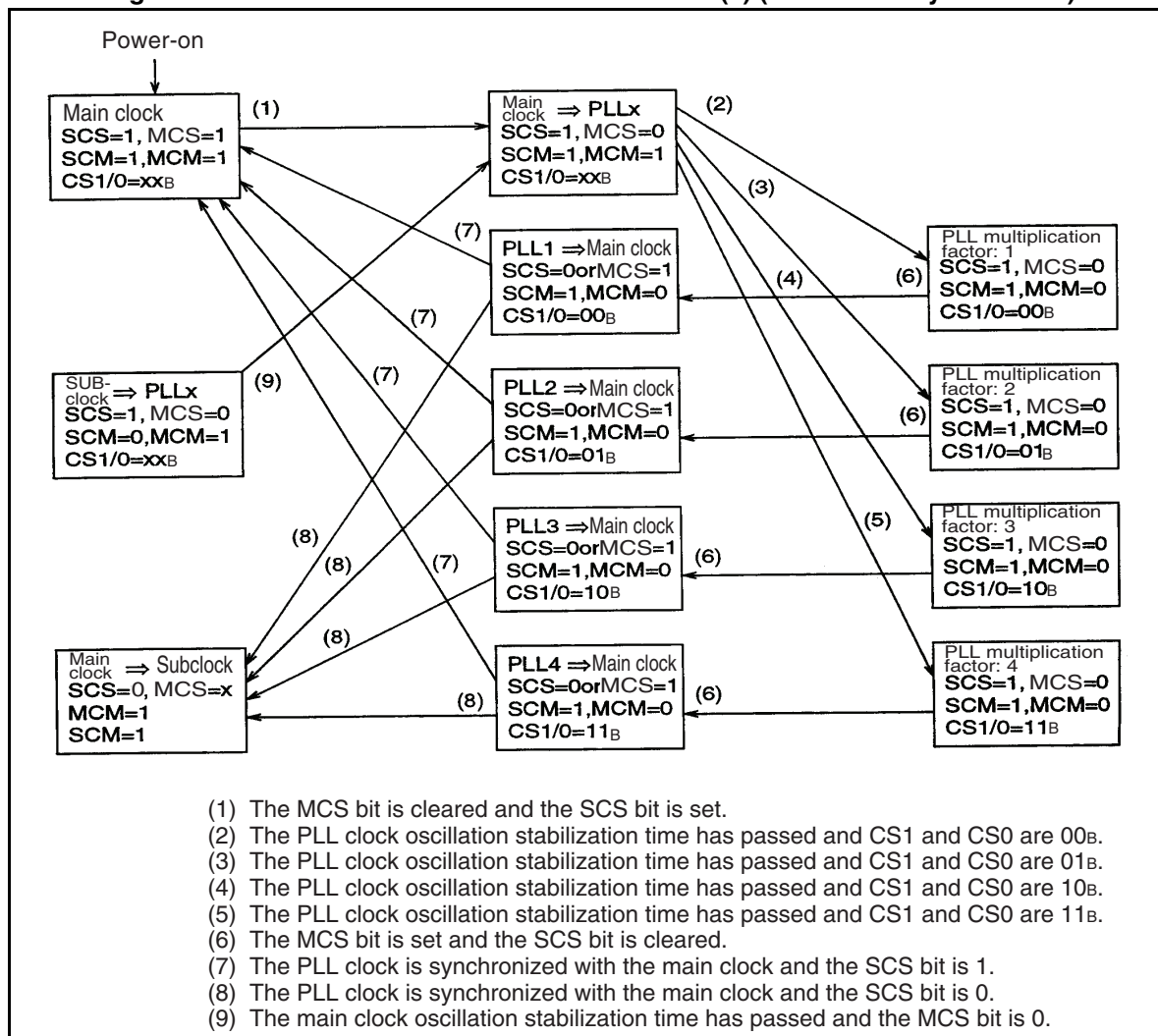


Figure 5.7-2 Transition of Clock Selection Status (2) (Two clocks system No.2)

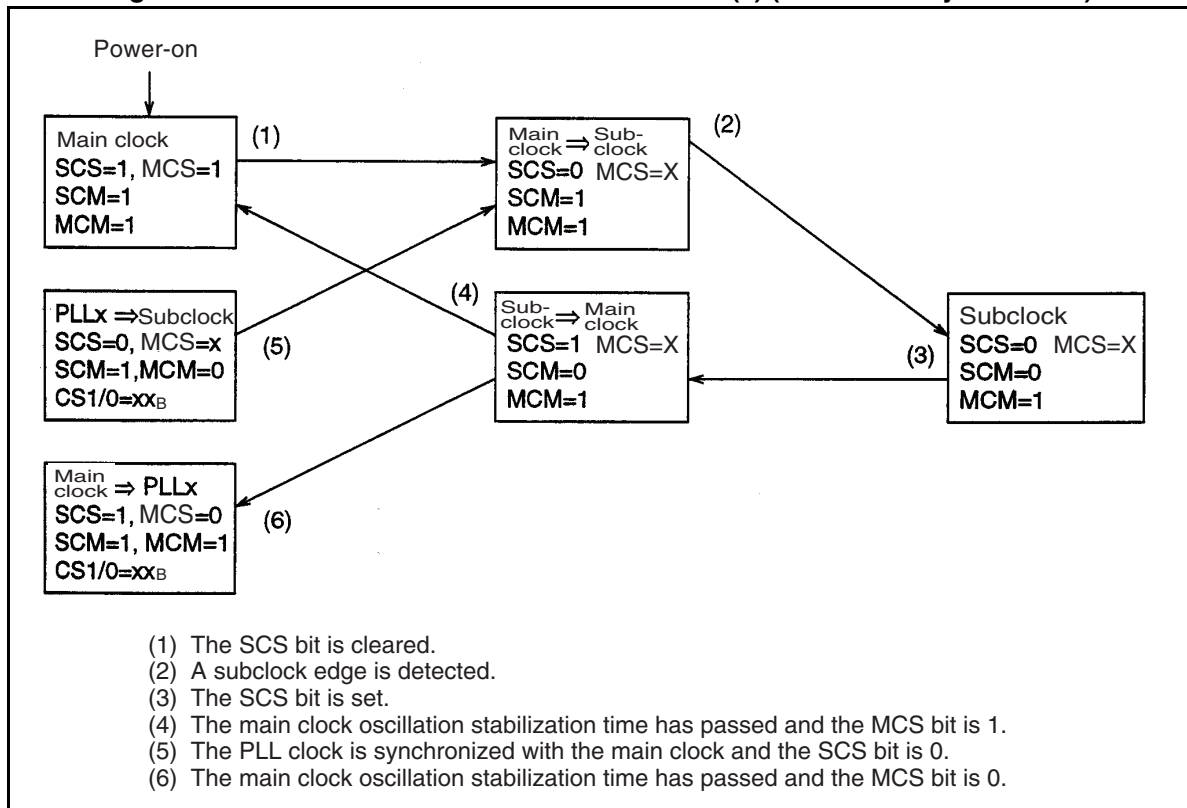
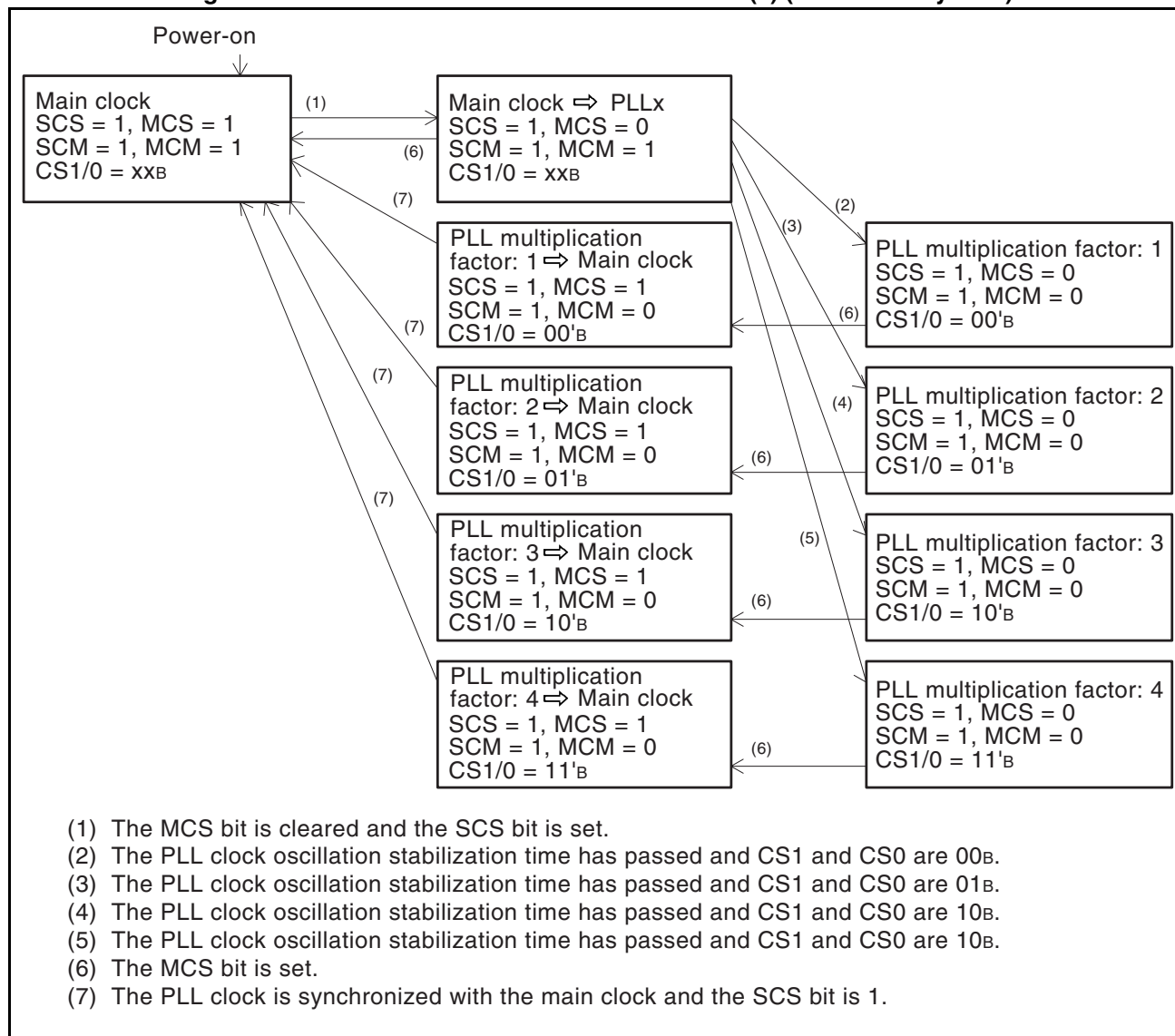


Figure 5.7-3 Transition of Clock Selection Status (3) (One Clock System)



5.8 Status Transition

Lists the conditions for status transition.

■ Status Transition

Table 5.8-1 Conditions for Status Transition (Two Clocks System)

Status before transition	Transition condition		Status after transition
Power-on	01	Main clock oscillation stabilization time has passed.	Main clock mode
Main clock oscillation stabilization	05	Main clock oscillation stabilization time has passed.	Main clock mode
Main clock mode	06	0 is written in SCS.	MS transition mode
	07	1 is written in SCS and 0 is written in MCS.	MP transition mode
	31	1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Main clock sleep mode
	32	0 is written in TMD.	Transition to watch mode with main clock
	33	1 is written in TMD and STP.	Main clock stop mode
PLL clock mode	21	0 is written in SCS.	PS transition mode
	20	1 is written in SCS and MCS.	PM transition mode
	59	1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	PLL clock sleep mode
	58	0 is written in TMD.	Transition to watch mode with PLL clock (P)
	57	1 is written in TMD and STP.	Transition to pseudo watch mode
Subclock mode	10	1 is written in SCS and MCS.	SM transition mode
	12	1 is written in SCS and MCS.	SP transition mode
	11	Reset is activated.	Main clock oscillation stabilization
	42	1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Subclock sleep mode
	43	0 is written in TMD.	Subclock mode
	44	1 is written in TMD and STP.	Subclock stop mode

Table 5.8-1 Conditions for Status Transition (Two Clocks System) (Continued)

Status before transition	Transition condition	Status after transition
PM transition mode	13 Timing to switch from PLL clock to main clock is reached.	Main clock mode
	38 1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Sleep mode for PM transition
	39 0 is written in TMD and timing to switch from PLL clock to main clock is reached.	Transition to watch mode with main clock
	40 1 is written in TMD and STP and timing to switch from PLL clock to main clock is reached.	Main clock stop mode
SM transition mode	02 Main clock oscillation stabilization time has passed.	Main clock mode
	03 Reset is activated or interrupt is generated.	Main clock oscillation stabilization
	04 0 is written in SCS.	Subclock mode
	27 1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Sleep mode for SM transition
	28 0 is written in TMD and main clock oscillation stabilization time has passed.	Watch mode with main clock
	29 1 is written in TMD and STP and main clock oscillation stabilization time has passed.	Main clock stop mode
MP transition mode	16 PLL clock oscillation stabilization time has passed.	PLL clock mode
	14 1 is written in SCS and MCS.	Main clock mode
	15 0 is written in SCS.	MS transition mode
	68 1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Sleep mode for MP transition
	70 0 is written in TMD.	Transition to watch mode with PLL clock (M)
	69 1 is written in TMD and STP.	Pseudo watch mode

Table 5.8-1 Conditions for Status Transition (Two Clocks System) (Continued)

Status before transition	Transition condition		Status after transition
SP transition mode	17	Main clock oscillation stabilization time has passed.	MP transition mode
	18	1 is written in MCS.	SM transition mode
	19	Reset is activated.	Main clock oscillation stabilization
	75	1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Sleep mode for SP transition
	76	0 is written in TMD.	Watch mode with PLL clock
	78	1 is written in TMD and STP and main clock oscillation stabilization time has passed.	Pseudo watch mode
MS transition mode	09	Timing to switch from main clock to subclock is reached.	Subclock mode
	08	Reset is activated.	Main clock mode
	51	1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Sleep mode for MS transition
	52	0 is written in TMD and timing to switch from main clock to subclock is reached.	Watch mode with subclock
	53	1 is written in TMD and STP and timing to switch from main clock to subclock is reached.	Subclock mode
PS transition mode	23	Timing to switch from PLL clock to main clock is reached.	MS transition mode
	22	1 is written in SCS.	PM transition mode
	56	1 is written in TMD, 0 is written in STP, and 1 is written in SLP.	Sleep mode for PS transition
Main clock sleep mode	26	Interrupt is generated or reset is activated.	Main clock mode
Sleep mode for SM transition	24	Main clock oscillation stabilization time has passed.	Sleep mode with main clock
	25	Interrupt is generated or reset is activated.	SM transition mode
Sleep mode for PM transition	34	PLL clock oscillation stabilization time has passed.	Main clock sleep mode
	35	Interrupt is generated or reset is activated.	PM transition mode
PLL clock sleep mode	63	Interrupt is generated or reset is activated.	PLL clock mode
Sleep mode for MP transition	66	PLL clock oscillation stabilization time has passed.	PLL clock sleep mode
	67	Interrupt is generated or reset is activated.	MP transition mode

CHAPTER 5 LOW-POWER CONSUMPTION CONTROL CIRCUIT

Table 5.8-1 Conditions for Status Transition (Two Clocks System) (Continued)

Status before transition	Transition condition		Status after transition
Sleep mode for SP transition	73	Main clock oscillation stabilization time has passed.	Sleep mode for MP transition
	74	Interrupt is generated or reset is activated.	SP transition mode
Subclock sleep mode	46	Interrupt is generated or reset is activated.	Subclock mode
Sleep mode for MS transition	49	Timing to switch from main clock to subclock is reached.	Subclock sleep mode
	50	Interrupt is generated or reset is activated.	MS transition mode
Sleep mode for PS transition	54	Timing to switch from PLL clock to main clock is reached.	Sleep mode for MS transition
	55	Interrupt is generated or reset is activated.	PS transition mode
Watch mode with main clock	30	Interrupt is generated or reset is activated.	SM transition mode
Transition to watch mode with main clock	36	Timing to switch from main clock to subclock is reached.	Watch mode with main clock
	37	Interrupt is generated or reset is activated.	Main clock mode
Watch mode with PLL clock	77	Interrupt is generated or reset is activated.	SP transition mode
Transition to watch mode with PLL clock (M)	72	Timing to switch from main clock to subclock is reached.	Watch mode with PLL clock
	71	Interrupt is generated or reset is activated.	MP transition mode
Transition to watch mode with PLL clock (P)	65	Timing to switch from PLL clock to main clock is reached.	Transition to watch mode with PLL clock (M)
	64	Interrupt is generated or reset is activated.	PLL clock mode
Watch mode with subclock	47	Interrupt is generated or reset is activated.	Subclock mode
Main clock stop mode	41	Interrupt is generated or reset is activated.	Main clock oscillation stabilization
Pseudo watch mode	62	Interrupt is generated or reset is activated.	MP transition mode
Transition to watch Pseudo with main clock	61	Timing to switch from PLL clock to main clock is reached.	Pseudo watch mode
	60	Interrupt is generated or reset is activated.	PLL clock mode
Subclock stop mode	48	Interrupt is generated.	Subclock mode
	79	Reset is activated.	Main clock oscillation stabilization

Table 5.8-1 Conditions for Status Transition (Two Clocks System) (Continued)

Status before transition	Transition condition	Status after transition
Subclock oscillation stabilization	45 Subclock oscillation stabilization time has passed.	Subclock mode
	80 Reset is activated.	Main clock oscillation stabilization

MCS: MCS bit (clock selection register) (When MCS is 0, the clock mode is selected.)

SCS: SCS bit (clock selection register) (When SCS is 0, the subclock mode is selected.)

STP: STP bit (low power-consumption mode register) (When STP is 0, the stop is selected.)

SLP: SLP bit (low power-consumption mode register) (When SLP is 0, the sleep mode is selected.)

TMD: TMD bit (low power-consumption mode register) (When TMD is 0, the watch mode is selected.)

MCM: MCM bit (clock selection register) (When MCM is 0, the PLL clock is used.)

SCM: SCM bit (clock selection register) (When SCM is 0, the subclock is used.)

SCD: Bit to stop subclock (When SCD is 1, subclock oscillation is stopped.)

MCD: Bit to stop main clock (When MCD is 1, main clock oscillation is stopped.)

PCD: Bit to stop PLL clock (When PCD is 1, PLL clock oscillation is stopped.)

Table 5.8-2 Conditions for Status Transition (One Clock System)

Status before transition	Transition condition	Status after transition
Power-on	01 Main oscillation stabilization time end	Main mode
Main oscillation stabilization	05 Main oscillation stabilization time end	Main mode
Main mode	07 SCS=1, MCS=0 write	MP transition mode
	31 TMD=1, STP=0, SLP=1 write	Main sleep
	33 TMD=1, STP=1 write	Main stop
PLL mode	20 SCS=1, MCS=1 write	PM transition mode
	59 TMD=1, STP=0, SLP=1 write	PLL sleep
	57 TMD=1, STP=1 write	Pseudo watch transition
PM transition mode	13 PLL/main switching timing wait end	Main mode
	38 TMD=1, STP=0, SLP=1 write	PM transition sleep
	40 TMD=1, STP=1 write & PLL to main switching timing wait end	Main stop
MP transition mode	16 PLL oscillation stabilization wait time end	PLL mode
	14 SCS=1, MCS=1 write	Main mode
	68 TMD=1, STP=0, SLP=1 write	MP transition sleep
	69 TMD=1, STP=1 write	Pseudo watch mode
Main sleep	26 Interrupt or reset activation	Main mode

Table 5.8-2 Conditions for Status Transition (One Clock System) (Continued)

Status before transition	Transition condition		Status after transition
PM transition sleep	34	PLL/main switching timing wait end	Main sleep
	35	Interrupt or reset activation	PM transition mode
PLL sleep	63	Interrupt or reset activation	PLL mode
MP transition sleep	66	PLL oscillation stabilization wait time end	PLL sleep
	67	Interrupt or reset activation	MP transition mode
Main stop	41	Interrupt or reset activation	Main oscillation stabilization
Pseudo watch	62	Interrupt or reset activation	MP transition mode
Pseudo watch transition	61	PLL/main switching timing wait end	Pseudo watch mode
	60	Interrupt or reset activation	PLL mode

MCS: MCS bit (clock selection register) (If MCS is 0, the clock mode is selected.)

STP: STP bit (low-power consumption mode register) (If STP is 0, the stop mode is selected.)

SLP: SLP bit (low-power consumption mode register) (If SLP is 0, the sleep mode is selected.)

MCM: MCM bit (clock selection register) (If MCM is 0, the PLL clock is operating.)

MCD: Bit to stop main clock (If MCD is 1, main clock oscillation is stopped.)

PCD: Bit to stop PLL clock (If PCD is 1, PLL clock oscillation is stopped.)

5.9 Status Transition Diagrams for Low Power-Consumption Modes

Figure 5.9-1 "Status Transition in the Low Power-Consumption Mode (Two clocks system) (1)" to Figure 5.9-7 "Status Transition in the Low Power-Consumption Mode (One Clock System) (3)" show the status transition in the low power-consumption mode.

■ Status Transition in the Low Power-Consumption Mode (Two Clocks System)

In Figure 5.9-1 "Status Transition in the Low Power-Consumption Mode (1)" to Figure 5.9-4 "Status Transition in the Low Power-Consumption Mode (4)" the events that occur at the same time are shown as if occurring in steps for easy understanding. In actual operation, each transition of status is performed momentarily. For example, if MCS is set to 1 and SLP is set to 1 simultaneously in PLL clock mode, the status transition diagram indicates the transition in which PLL clock mode first changes to PM transition mode, then changes to PM transition sleep. In actual operation, however, PLL clock mode changes to PM transition sleep immediately. Another example is the transition from the subclock sleep mode to main clock oscillation stabilization. The figures show that the subclock sleep mode changes to the subclock mode once, then changes to main clock oscillation stabilization. In actual operation, the subclock sleep mode changes to main clock oscillation stabilization momentarily.

Note:

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

Figure 5.9-1 Status Transition in the Low Power-Consumption Mode (Two clocks system) (1)

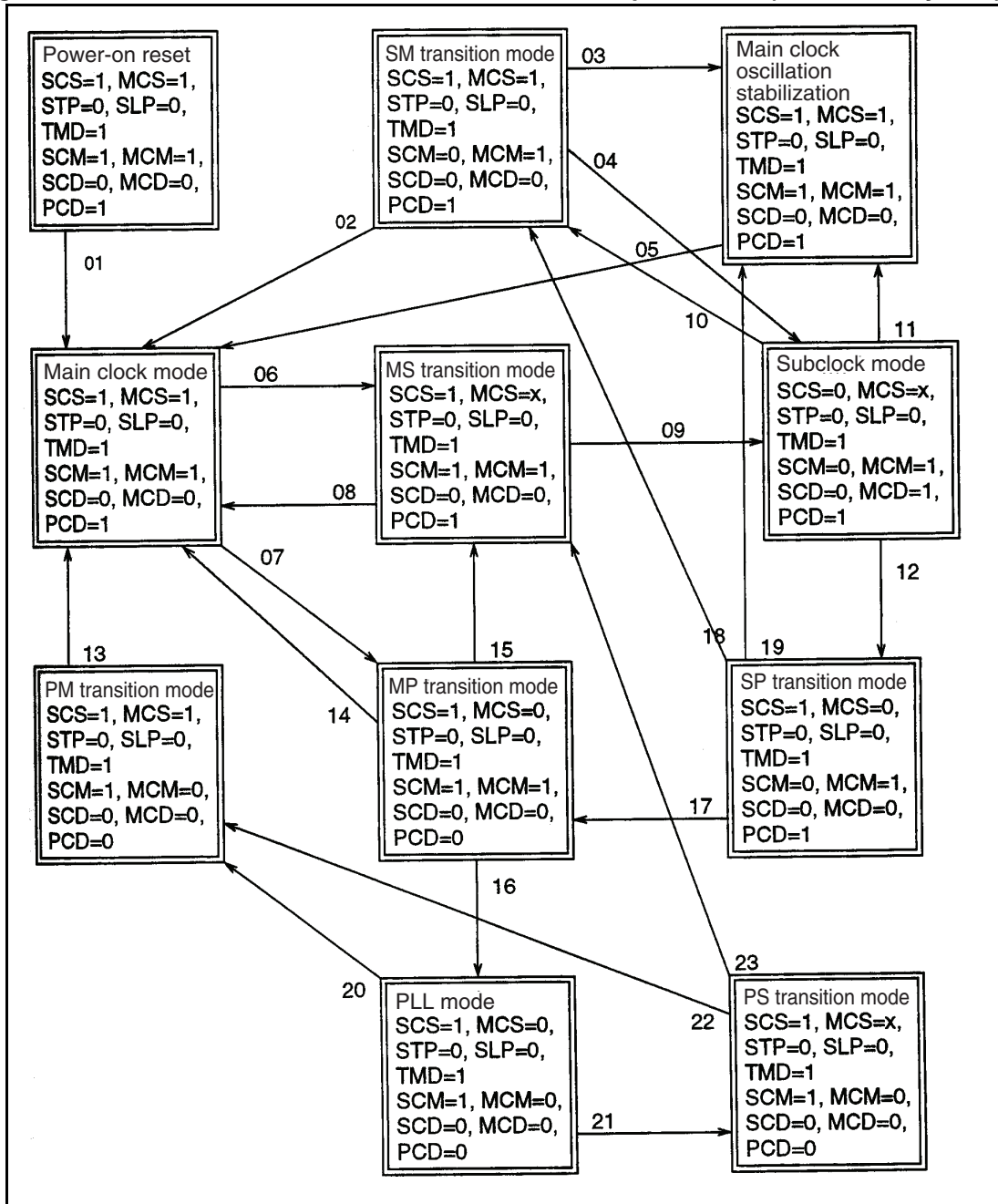


Figure 5.9-2 Status Transition in the Low Power-Consumption Mode (Two clocks system) (2)

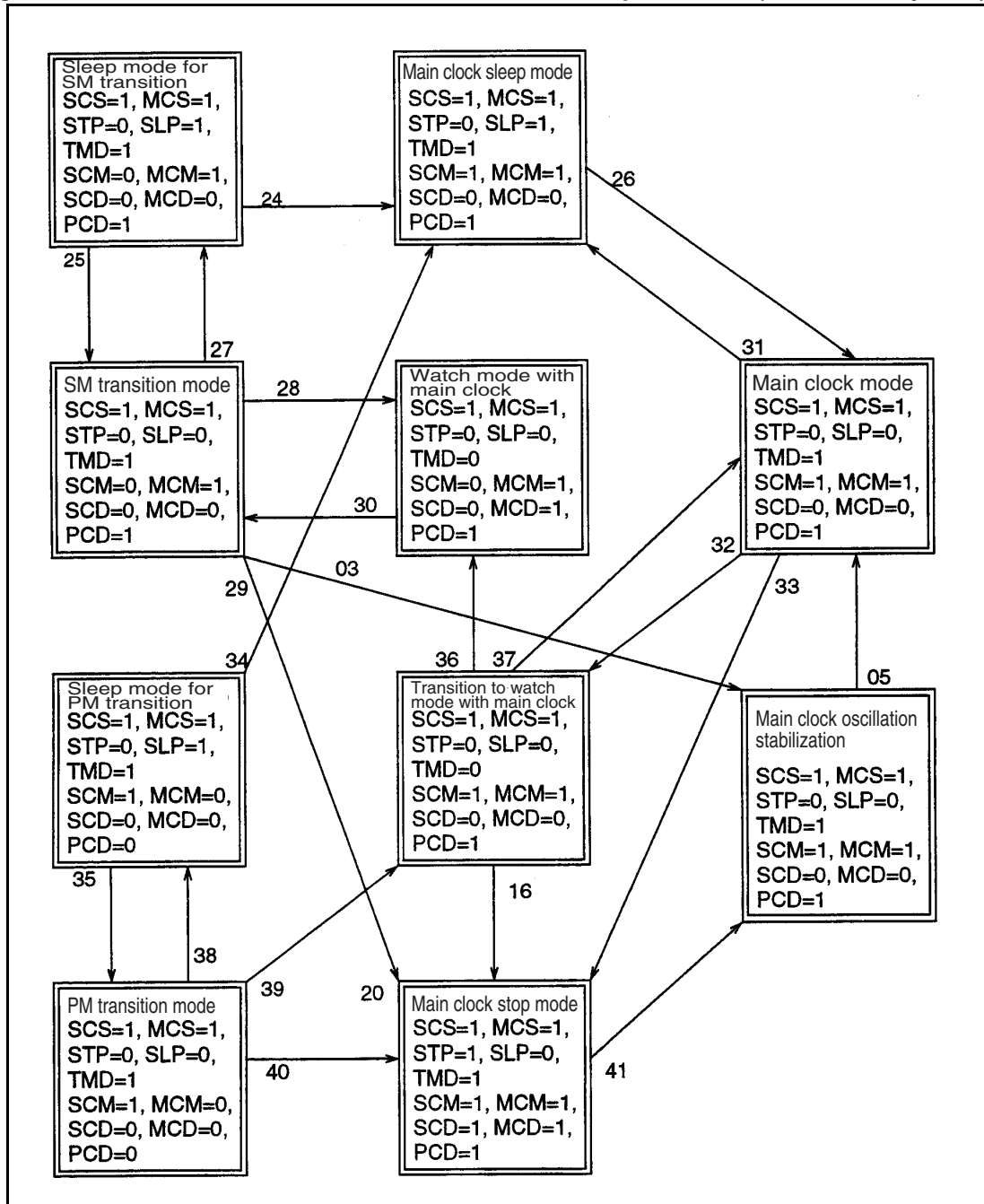


Figure 5.9-3 Status Transition in the Low Power-Consumption Mode (Two clocks system) (3)

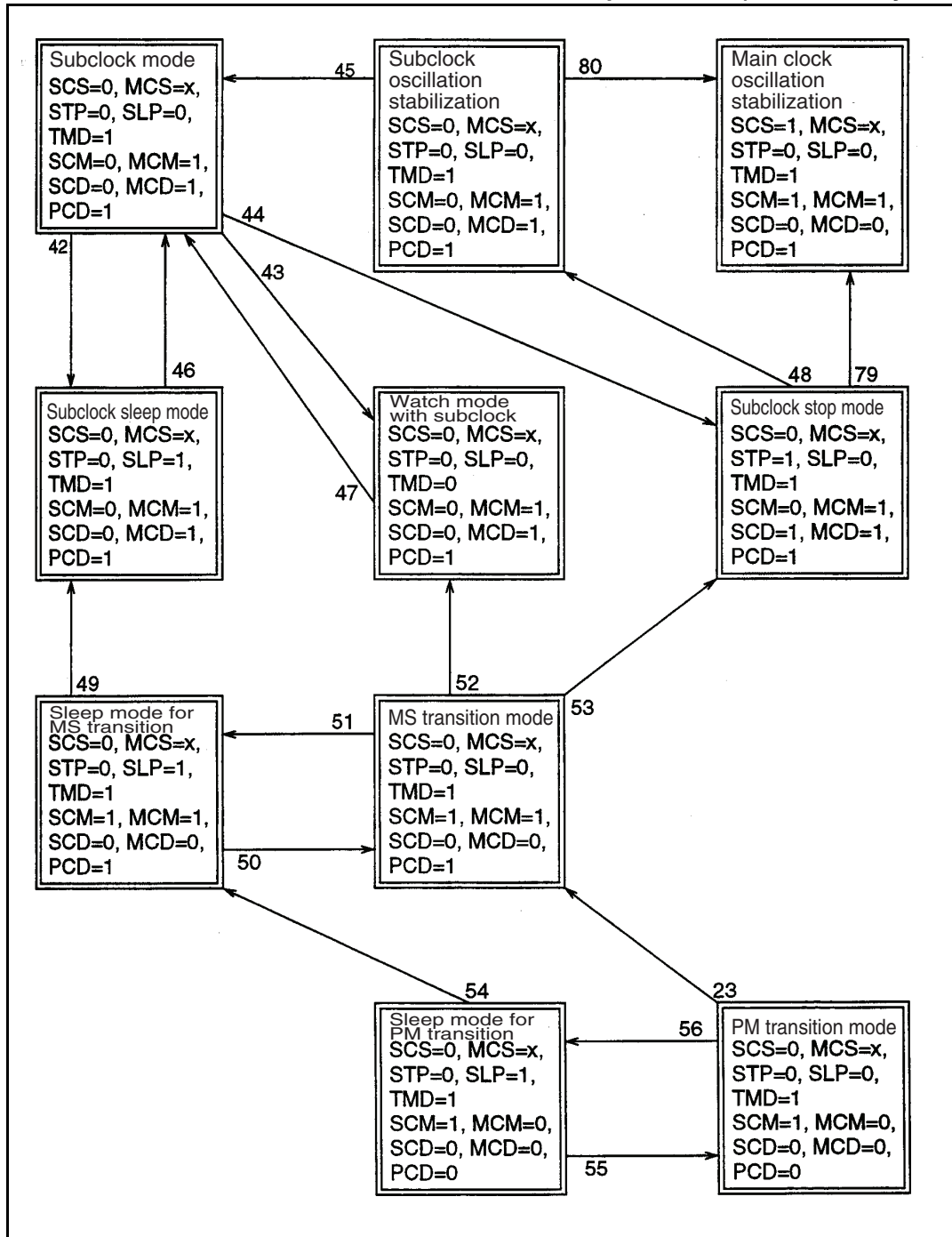
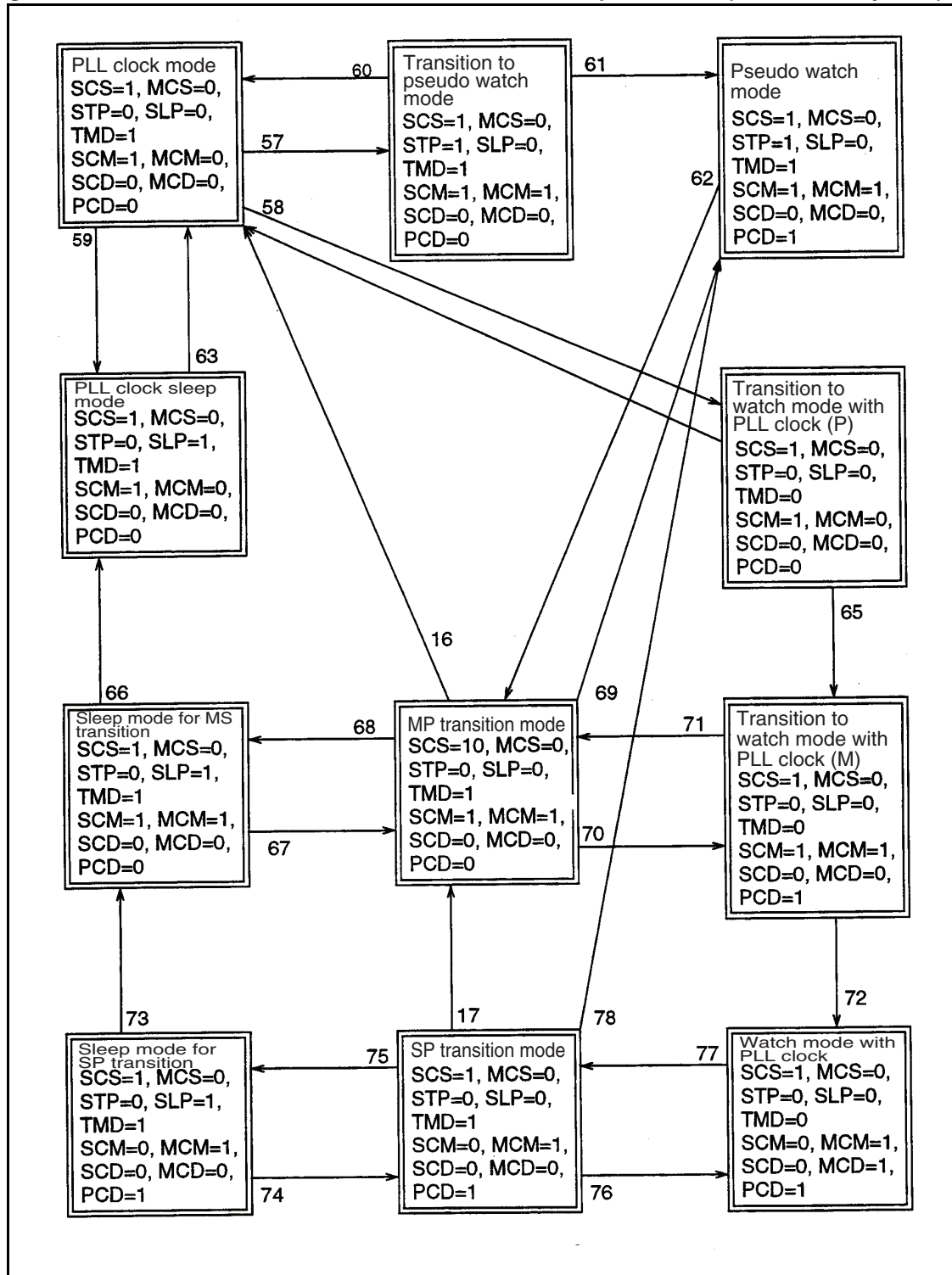


Figure 5.9-4 Status Transition in the Low Power-Consumption Mode (Two clocks system) (4)



■ Status Transition in the Low-Power Consumption Mode (One Clock System)

Figure 5.9-5 Status Transition in the Low-Power Consumption Mode (One Clock System) (1)

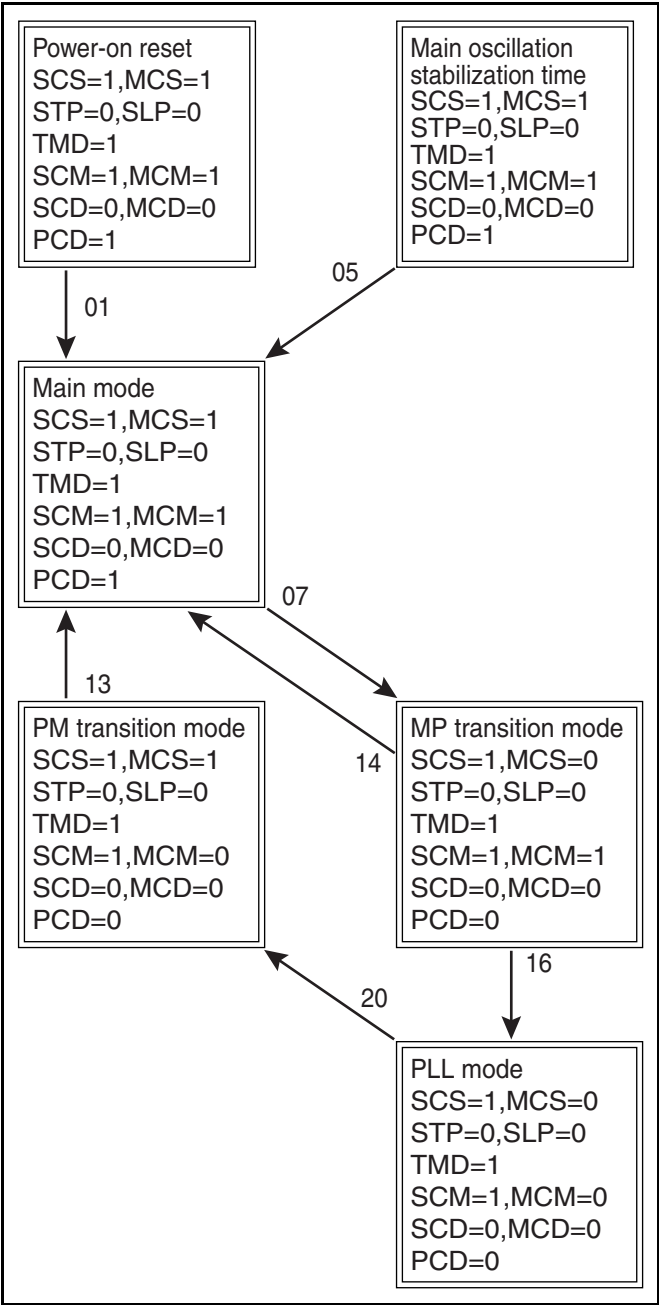


Figure 5.9-6 Status Transition in the Low-Power Consumption Mode (One Clock System) (2)

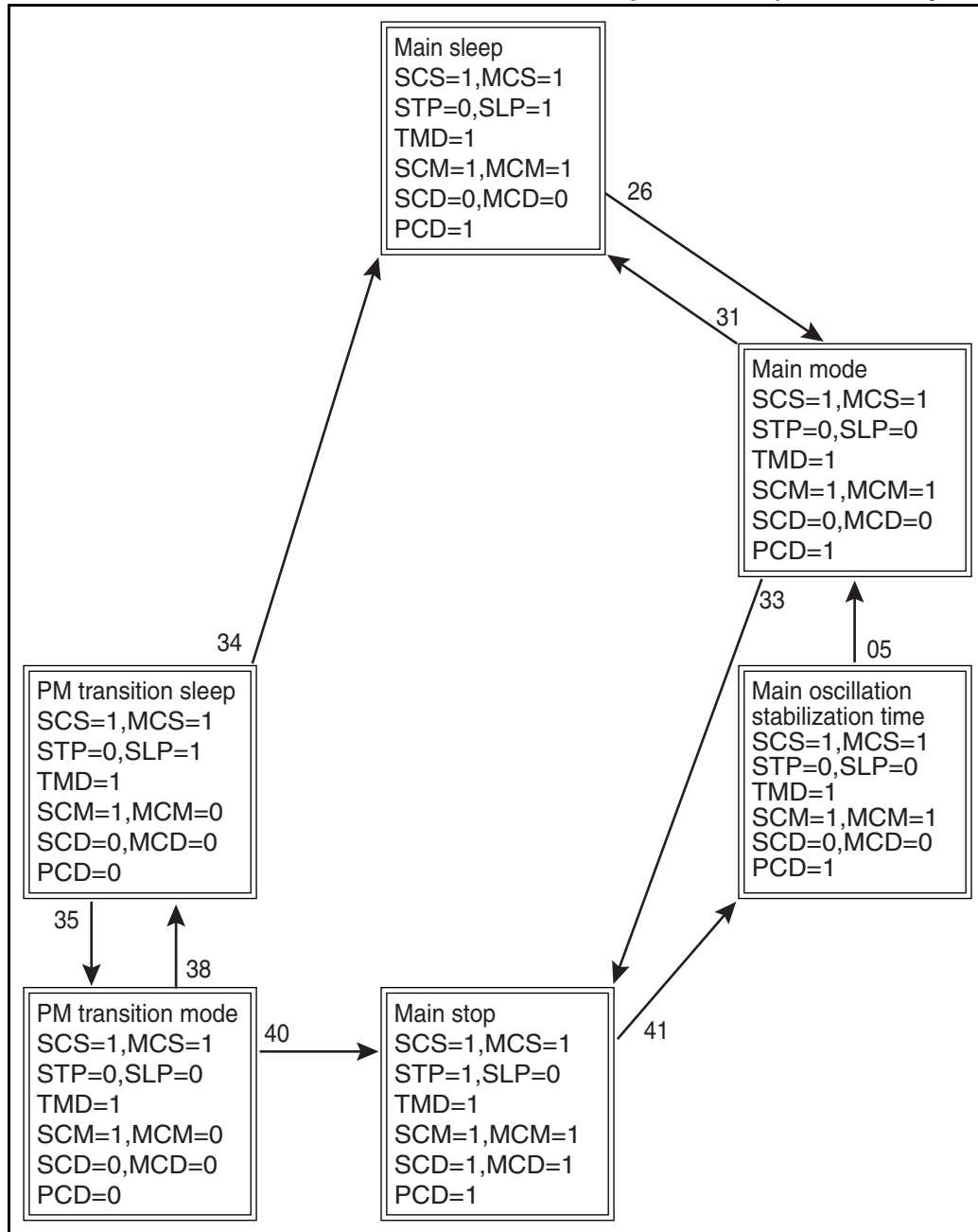
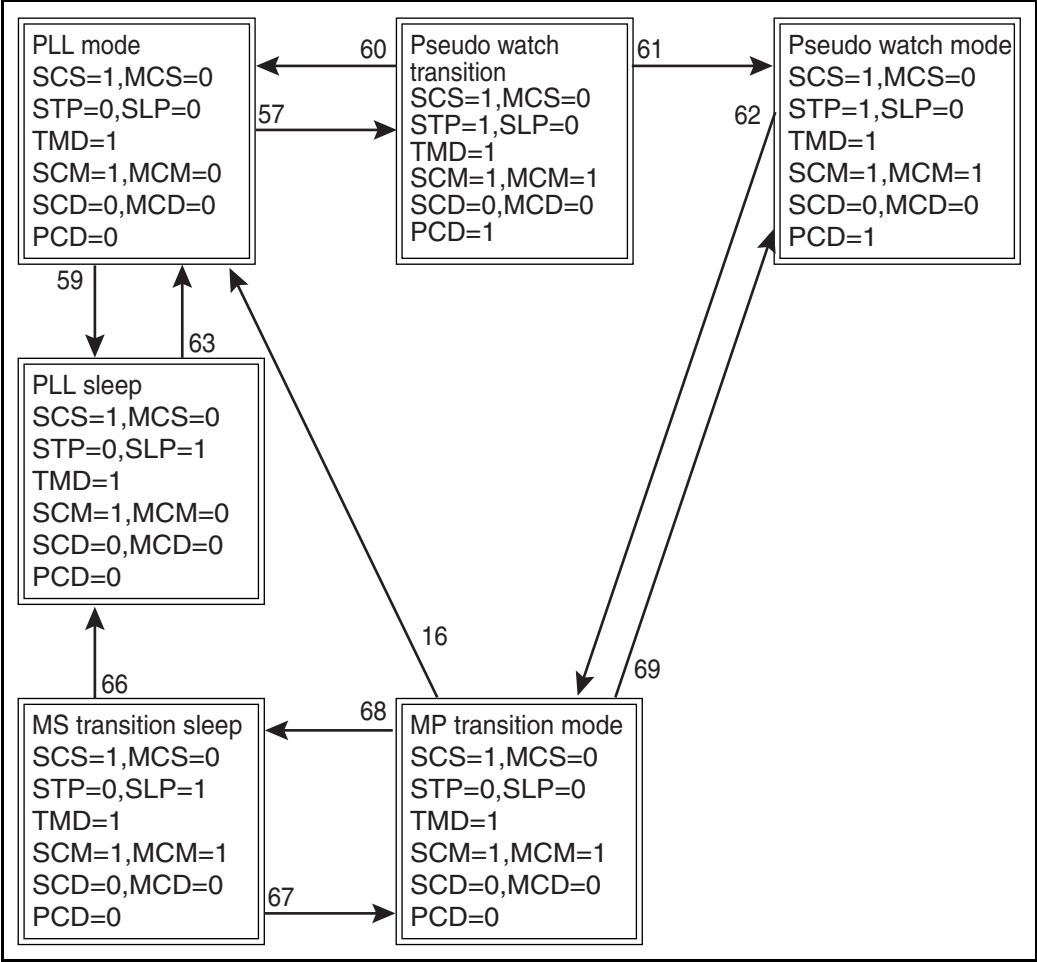


Figure 5.9-7 Status Transition in the Low-Power Consumption Mode (One Clock System) (3)



Note:

In attempting to switch the clock mode, do not attempt to switch to another clock mode or low-power consumption mode until the first switching is completed. The MCM bit of the clock selection register (CKSCR) indicates that switching is completed. If the mode is switched to another clock mode or low-power consumption mode before completion of switching, the mode may not be switched.

CHAPTER 6 MEMORY ACCESS MODES

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

- 6.1 "Memory Access Mode Overview"
- 6.2 "External Memory Access (External Bus Pin Control Circuit)"
- 6.3 "Operation of the External Memory Access Control Signals"

6.1 Memory Access Mode Overview

The F²MC-16LX provides various types of modes for the access system and access area.

■ Memory Access Mode Overview

Table 6.1-1 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
Flash memory write	-	-

○ Operating mode

In an operating mode, the device operation state is controlled. The operating mode is specified by mode setting pins (MDx) and the M1 and M0 bits of mode data. By selecting an operating mode, ordinary operation can be activated and flash memory can be written.

○ Bus mode

In a bus mode, the operations of the internal ROM and external access function are controlled. A bus mode is specified by mode setting pins (MDx) and an Mx bit in mode data. The mode setting pins (MDx) specify a bus mode for reading reset vector and mode data. An Mx bit in mode data specifies a bus mode for normal operation.

○ Access mode

In an access mode, the external data bus width is controlled. An access mode is specified by mode setting pins (MDx) and the S0 bit in mode data. Selecting an access mode specifies 8 bits or 16 bits as the external data bus width.

6.1.1 Mode Pins

Modes can be specified by setting three external pins MD2 to MD0 in combination as shown in Table 6.1-2 "Relationships between Mode Pins and Set Modes".

■ Mode Pin Settings and Corresponding Modes

Table 6.1-2 Relationships between Mode Pins and Set Modes

Mode pin setting			Mode	Reset vector access area	External data bus width	Remark
MD2	MD1	MD0				
0	0	0	External vector mode 0	External	8 bits	-
0	0	1	External vector mode 1	External	16 bits	Reset vector access with 16-bit bus width
0	1	0	Cannot be specified.			
0	1	1	Internal vector mode	Internal	(Mode data)	Controlled by mode data after reset sequence
1	0	0	Cannot be specified.			
1	0	1				
1	1	0	Flash memory serial programming	-	-	-
1	1	1	Flash memory mode	-	-	Mode for use of a parallel writer

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 0. Therefore, a 16-bit width is available in areas 0000C0_H to 0000FF_H and 002000_H to 7FFFFFF_H. To specify an 8-bit width for the areas, write 1 in IOBS and LMBS of the bus control selection register. In the external vector mode 1, the HMBS bit is set to 0 and access is performed with a 16-bit bus width.
- Data cannot be written only by setting the flash memory serial programming mode by mode pins. Other pins must be set. For details, see the example of flash memory serial programming connection.

6.1.2 Mode Data

Mode data is stored in FFFFDF_H in main storage to control CPU operation. This data is fetched during execution of a reset sequence and stored in the mode register in the device. Only the reset sequence can change the value of the mode register.

The setting of this register is valid after the reset sequence.

Set the reserved bits to 0.

■ Mode Data

Figure 6.1-1 Mode Data Configuration

Mode data	7	6	5	4	3	2	1	0	↔ Bit No.
Address:FFFFDF _H	M1	M0	Reserved	Reserved	S0	Reserved	Reserved	Reserved	

[Bits 7 and 6] M1 and M0 (bus mode setting bits)

The M1 and M0 bits specify an operating mode after termination of the reset sequence. Table 6.1-3 "Function of M1 and M0 (Bus Mode Setting Bits)" shows the relationships between the settings of M1 and M0 bits and functions.

Table 6.1-3 Function of M1 and M0 (Bus Mode Setting Bits)

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 is inhibited.

[Bit 3] S0 (access mode setting bit)

The S0 bit specifies a bus mode and access mode after termination of the reset sequence. Table 6.1-4 "Function of S0 (Access Mode Setting Bit)" shows the relationships between the settings of S0 bit and functions.

Table 6.1-4 Function of S0 (Access Mode Setting Bit)

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

6.1.3 Memory Space for Each Bus Mode

Figure 6.1-2 "Relationships between Access Areas and Physical Addresses in Each Bus Modes" shows the correspondence between access areas and physical addresses depending on the bus mode specification.

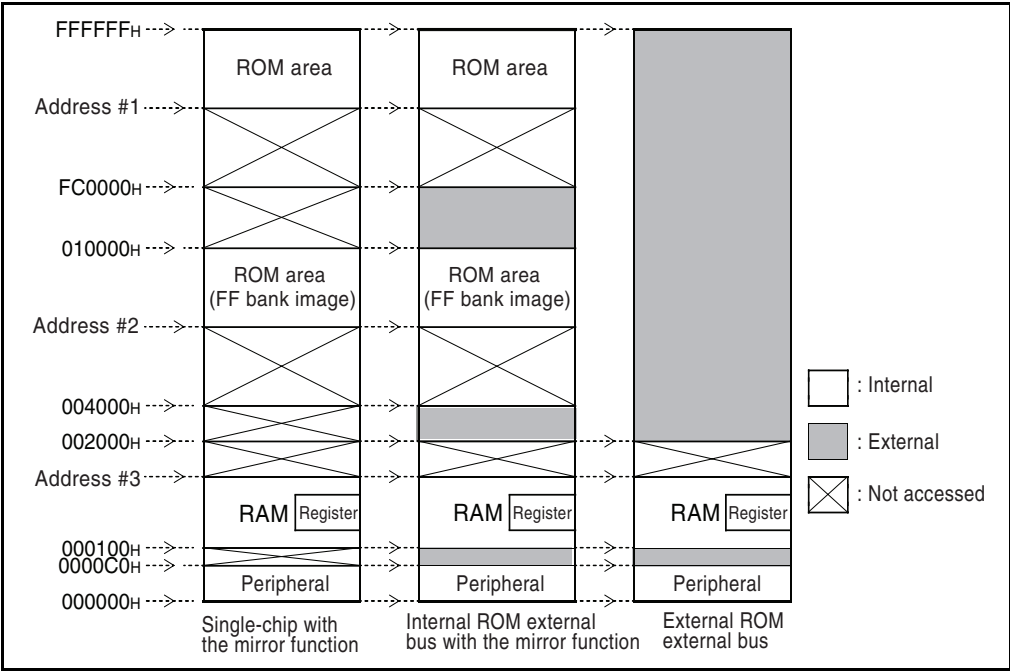
■ Memory Space for Each Bus Mode

As shown in Figure 6.1-2 "Relationships between Access Areas and Physical Addresses in Each Bus Modes" the ROM image of an FF bank can be seen in high-order bits of a 00 bank. This makes the small model of the C compiler effective. Low-order 16 bits are designed to provide the same effect so that tables in the ROM can be referenced without "far" specification in pointer declaration.

For example, if 00C000_H is accessed, the ROM contents in FFC000_H are accessed. Because the ROM area in the FF bank exceeds 48K bytes, the entire image of the FF bank cannot be stored in the 00 bank. Therefore, ROM data in FF4000_H to FFFFFFF_H can be seen in the image in 004000_H to 00FFFF_H, and so it is recommended that the ROM data table be stored in the area from FF4000_H to FFFFFFF_H.

See Chapter 23 "ADDRESS MATCH DETECTION FUNCTION" when ROM without the mirror function is selected.

Figure 6.1-2 Relationships between Access Areas and Physical Addresses in Each Bus Modes



Part number	Address #1	Address #2	Address #3
MB90583C/CA	FE00000 _H	004000 _H	001900 _H
MB90F583C/CA	FE00000 _H	004000 _H	001900 _H
MB90587C/CA	FF00000 _H	004000 _H	001100 _H
MB90V580B	(FE00000 _H)	004000 _H	001900 _H

■ Recommended Setting Sample of Memory Space for Each Bus Mode

Table 6.1-5 "Example of Recommended Setting of Mode Pins and Mode Data" shows a recommended setting sample of mode pins and mode data.

Table 6.1-5 Example of Recommended Setting of Mode Pins and Mode Data

Mode setting	MD2	MD1	MD0	M1	M0	S0
Single chip	0	1	1	0	0	x
Internal ROM, external 16-bit bus	0	1	1	0	1	1
Internal ROM, external 8-bit bus	0	1	1	0	1	0
External ROM, external 16-bit bus, vector 16-bus width	0	0	1	1	0	1
External ROM, external 8-bit bus	0	0	0	1	0	0

The signals input to and output from the external pins related to memory access modes are dependent on the mode.

Table 6.1-6 Operation of External Pins Related to Modes

Pin	Function	
	Single chip	External bus extension
		<div>8 bits</div> <div>16 bits</div>
P07 to P00	Port	AD07 to 00
P17 to P10		<div>A15 to 08</div> <div>AD15 to 08</div>
P27 to P20		A23 to 16 *
P30		ALE
P31		\overline{RD}
P32		<div>\overline{WR}</div> <div>\overline{WRL} *</div>
P33		<div>Port</div> <div>\overline{WRH} *</div>
P34		HRQ *
P35		\overline{HAK} *
P36		RDY *
P37		CLK *

*: The address high output pins, \overline{WR} , \overline{WRL} , \overline{WRH} , HRQ, \overline{HAK} , RDY, and CLK can be used as ports by function selection. For details, see Section 6.2 "External Memory Access (External Bus Pin Control Circuit)".

6.2 External Memory Access (External Bus Pin Control Circuit)

The external bus pin control circuit controls external bus pins used to expand the address/data buses of the CPU outside.

■ External Memory Access (External Bus Pin Control Circuit)

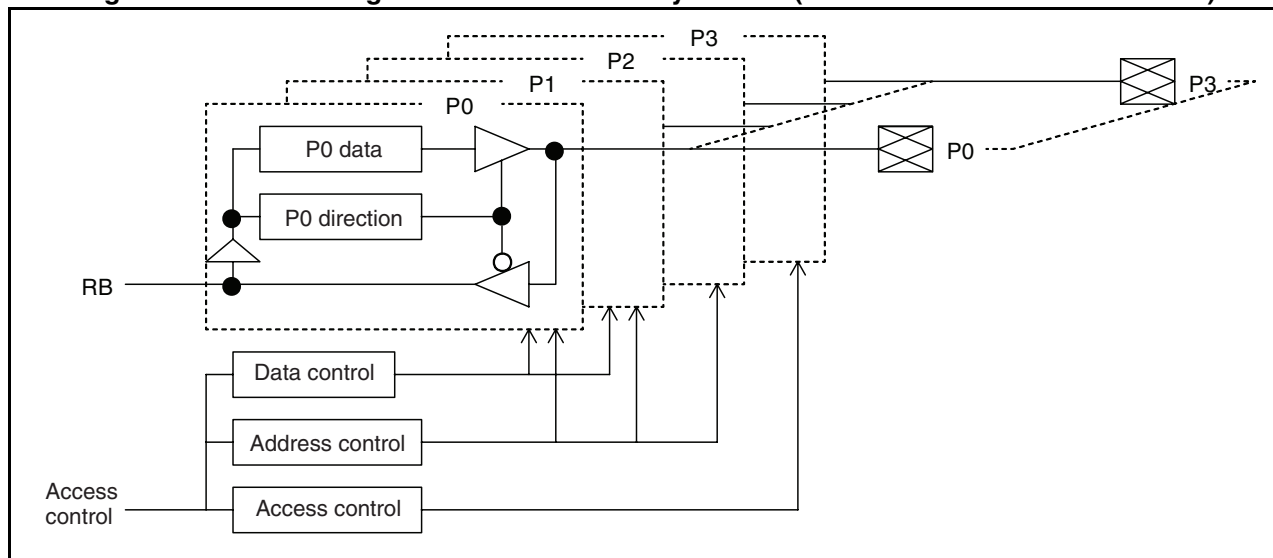
To access memory/peripheral circuits installed outside the device, the F²MC-16LX supplies the following address/data/control signals:

- CLK (P37): Machine cycle clock (KBP) output pin
- RDY (P36): External ready input pin
- $\overline{\text{WRH}}$ (P33): Write signal for high-order eight bits of the data bus
- $\overline{\text{WRL}}$ (P32): Write signal for low-order eight bits of the data bus
- $\overline{\text{RD}}$ (P31): Read signal
- ALE (P30): Address latch enable signal

■ Block Diagram of External Memory Access (External Bus Pin Control Circuit)

Figure 6.2-1 "Block Diagram of External Memory Access (External Bus Pin Control Circuit)" shows a block diagram of external memory access (external bus pin control circuit).

Figure 6.2-1 Block Diagram of External Memory Access (External Bus Pin Control Circuit)



6.2.1 Registers for External Memory Access (External Bus Pin Control Circuit)

External memory access (external bus pin control circuit) has the following three types of registers:

- Automatic ready function selection register
- External address output control register
- Bus control signal selection register

■ Registers for External Memory Access (External Bus Pin Control Circuit)

Figure 6.2-2 Registers for External Memory Access (External Bus Pin Control Circuit)

Automatic ready function selection register		15	14	13	12	11	10	9	8	⇐ Bit No.
Address:0000A5 _H		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 _H		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 _H		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)	(-)	

6.2.2 Automatic Ready Function Selection Register (ARSR)

This register sets the automatic wait time of memory access for each area at external access.

■ Automatic Ready Function Selection Register (ARSR)

Figure 6.2-3 Configuration of the Automatic Ready Function Selection Register

Automatic ready function selection register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 0000A5 _H	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 and 14] IOR1 and IOR0

The IOR1 and IOR0 bits specify an automatic wait function for external access to area 0000C0_H to 0000FF_H. IOR1 and IOR0 bits are combined as listed in Table 6.2-1 "Function of IOR1 and IOR0 (Automatic Wait Function Specification Bits)".

Table 6.2-1 Function of IOR1 and IOR0 (Automatic Wait Function Specification Bits)

IOR1	IOR0	Function
0	0	Prohibits automatic wait. [Initial value*]
0	1	Insert automatic 1-machine cycle wait at external access.
1	0	Insert automatic 2-machine cycle wait at external access.
1	1	Insert automatic 3-machine cycle wait at external access.

*: The initial value is 00_B.

[Bits 13 and 12] HMR1 and HMR0

The HMR1 and HMR0 bits specify an automatic wait function for external access to area 800000_H to FFFFFFFF_H. HMR1 and HMR0 bits are combined as listed in Table 6.2-2 "Function of HMR1 and HMR0 (Automatic Wait Function Specification Bits)".

Table 6.2-2 Function of HMR1 and HMR0 (Automatic Wait Function Specification Bits)

HMR1	HMR0	Function
0	0	Prohibits automatic wait.
0	1	Insert automatic 1-machine cycle wait at external access.
1	0	Insert automatic 2-machine cycle wait at external access.
1	1	Insert automatic 3-machine cycle wait at external access. [Initial value*]

*: The initial value is 11_B.

6.2 External Memory Access (External Bus Pin Control Circuit)

[Bits 9 and 8] LMR1 and LMR0

The LMR1 and LMR0 bits specify an automatic wait function for external access to area 002000_H to 7FFFFFF_H. LMR1 and LMR0 bits are combined as listed in Table 6.2-3 "Function of LMR1 and LMR0 (Automatic Wait Function Specification Bits)".

Table 6.2-3 Function of LMR1 and LMR0 (Automatic Wait Function Specification Bits)

LMR1	LMR0	Function
0	0	Prohibits automatic wait. [Initial value*]
0	1	Insert automatic 1-machine cycle wait at external access.
1	0	Insert automatic 2-machine cycle wait at external access.
1	1	Insert automatic 3-machine cycle wait at external access.

*: The initial value is 00_B.

6.2.3 External Address Output Control Register (HACR)

This register controls external output of address output pins (A23 to A16). Respective bits correspond to address output pins A23 to A16 and control the address output pins as shown in Figure 6.2-4 "Configuration of the External Address Output Control Register".

■ External Address Output Control Register (HACR)

Figure 6.2-4 Configuration of the External Address Output Control Register

External address output control register	7	6	5	4	3	2	1	0	↔ Bit No.
Address:0000A6H	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)	

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

All bits of this register are dedicated to writing. For reading, all bits are set to 1.

When using the bits for address output, set the port-2 data direction register (DDR2) to 0.

Table 6.2-4 Function of the External Address Output Control Register (E16 to E23 bits)

E16 to 23	Function
0	The corresponding pin acts as an address output pin (AXX). [Initial value]
1	The corresponding pin acts as an I/O port pin (PXX).

6.2.4 Bus Control Signal Selection Register (ECSR)

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

■ Bus Control Signal Selection Register (ECSR)

Figure 6.2-5 Configuration of the Bus Control Signal Selection Register

Control signal selection register	15	14	13	12	11	10	9	8	↔ Bit No.
Address:0000A7 _H	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	...
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	...
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(-)	...

[Bit 15] CKE

The CKE bit controls output of the external clock (CLK) as listed in Table 6.2-5 "Function of CKE (External Clock (CLK) Output Control Bit)".

Table 6.2-5 Function of CKE (External Clock (CLK) Output Control Bit)

CKE	Function
0	I/O port (P37) operation (Prohibits clock output.)[Initial value]
1	Enables clock signal (CLK) output.

[Bit 14] RYE

The RYE bit controls input of the external ready (RDY) as listed in Table 6.2-6 "Function of RYE (External Ready (RDY) Input Control Bit)".

Table 6.2-6 Function of RYE (External Ready (RDY) Input Control Bit)

RDY	Function
0	I/O port (P36) operation (Prohibits external RDY input.)[Initial value]
1	Enables external ready (RDY) input.

[Bit 13] HDE

The HDE bit enables input/output of hold-related pins. The HDE bit controls hold request input (HRQ) and hold acknowledge output ($\overline{\text{HAK}}$) as listed in Table 6.2-7 "Function of HDE

(Input/Output Enable Bit Of Hold Related Pins)".

Table 6.2-7 Function of HDE (Input/Output Enable Bit Of Hold Related Pins)

HDE	Function
0	I/O port (P35 and P34) operation (Prohibits hold function input/output.) [Initial value]
1	Enables input of hold request (HRQ)/output of hold acknowledge ($\overline{\text{HAK}}$).

[Bit 12] IOBS

The IOBS bit specifies a bus size for external access to the area 0000C0_H to 0000FF_H in an external 16-bit data bus mode. This bit controls the size as listed in Table 6.2-8 "Function of IOBS (Bus Size Specification Bit)".

Table 6.2-8 Function of IOBS (Bus Size Specification Bit)

IOBS	Function
0	16-bit bus access [Initial value*]
1	8-bit bus access

*: The IOBS bit is cleared to 0 by reset.

[Bit 11] HMBS

The HMBS bit specifies a bus size for external access to the area 800000_H to FFFFFFF_H in an external 16-bit data bus mode. This bit controls the size as listed in Table 6.2-9 "Function of HMBS (Bus Size Specification Bit)".

Table 6.2-9 Function of HMBS (Bus Size Specification Bit)

HMBS	Function
0	16-bit bus access [Initial value]
1	8-bit bus access

[Bit 10] WRE

The WRE bit controls output of the external write signal (in a 16-bit bus mode, the $\overline{\text{WRH}}$ and $\overline{\text{WRL}}$ pins and in an 8-bit bus mode, the $\overline{\text{WR}}$ pin) as listed in Table 6.2-10 "Function of WRE (External Write Signal Output Control Bit)".

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

Table 6.2-10 Function of WRE (External Write Signal Output Control Bit)

WRE	Function
0	I/O port (P33, P32) operation (Prohibits write signal output.) [Initial value*]
1	Enables output of the write strobe signal ($\overline{\text{WRH}}$ and $\overline{\text{WRL}}$ or only $\overline{\text{WR}}$)

*: The WRE bit is cleared to 0 by reset.

6.2 External Memory Access (External Bus Pin Control Circuit)

[Bit 9] LMBS

The LMBS bit specifies a bus size for external access to the area 002000_H to 7FFFFFF_H in an external 16-bit data bus mode. This bit controls the size as listed in Table 6.2-11 "Function of LMBS (Bus Size Specification Bit)".

Table 6.2-11 Function of LMBS (Bus Size Specification Bit)

LMBS	Function
0	16-bit bus access [Initial value*]
1	8-bit bus access

*: The LMBS bit is cleared to 0 by reset.

Note:

In a 16-bit bus mode, to enable the \overline{WRH} and \overline{WRL} functions by the WRE bit, set P33 and P32 to the input mode (set bits 3 and 2 of DDR3 to 0).

In an 8-bit bus mode, to enable the \overline{WR} function by the WRE bit, set P32 to the input mode (set bit 2 of DDR3 to 0).

When the RYE or HDE bit enables RDY or HRQ to be input respectively, the I/O port function of the port pin is validated. Set the bit corresponding to the port pin in DDR3 to 0 (input mode).

6.3 Operation of the External Memory Access Control Signals

External memory is accessed at intervals of three cycles when the ready function is not used. Eight-bit bus access in an external 16-bit bus mode enables 8-bit peripheral chips to be read and written if 8-bit peripheral chips and 16-bit peripheral chips are connected to the external bus together.

■ External Memory Access Control Signal

Because 8-bit bus access is executed using low-order eight bits of the data bus, connect an 8-bit peripheral chip to low-order eight bits of the data bus.

The access that is executed in an external 16-bit bus mode, 16-bit bus access or 8-bit bus access, is specified by the HMBS, LMBS, and IOBS of the EPCR.

In some cases, address output and ALE assert output are performed and $\overline{RD}/\overline{WR}/\overline{WRL}/\overline{WRH}$ are not asserted not to perform an actual bus operation. Do not execute access of peripheral circuit chips by the ALE signal only.

Figure 6.3-1 "Timing Chart of External Memory Access (in an External 8-Bit Bus Mode)" shows the timing chart of external memory access (in an external 8-bit bus mode). Figure 6.3-2 "Timing Chart of External Memory Access (in an External 16-Bit Bus Mode)" shows the timing chart of external memory access (in an external 16-bit bus mode).

Figure 6.3-1 Timing Chart of External Memory Access (in an External 8-Bit Bus Mode)

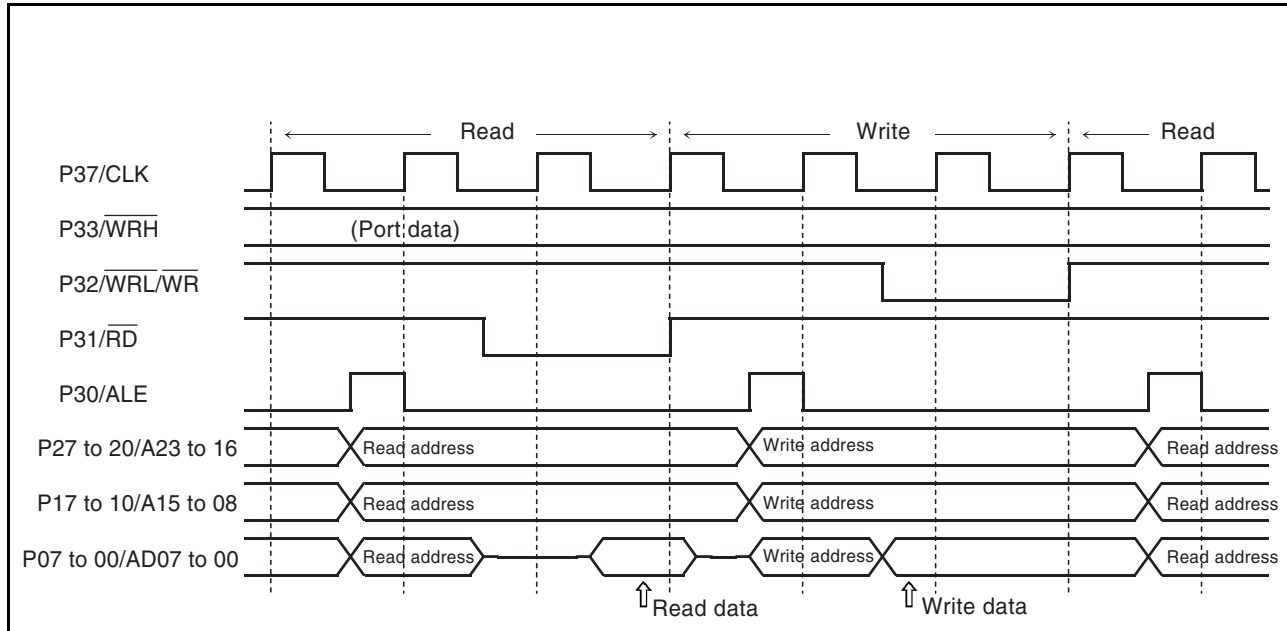
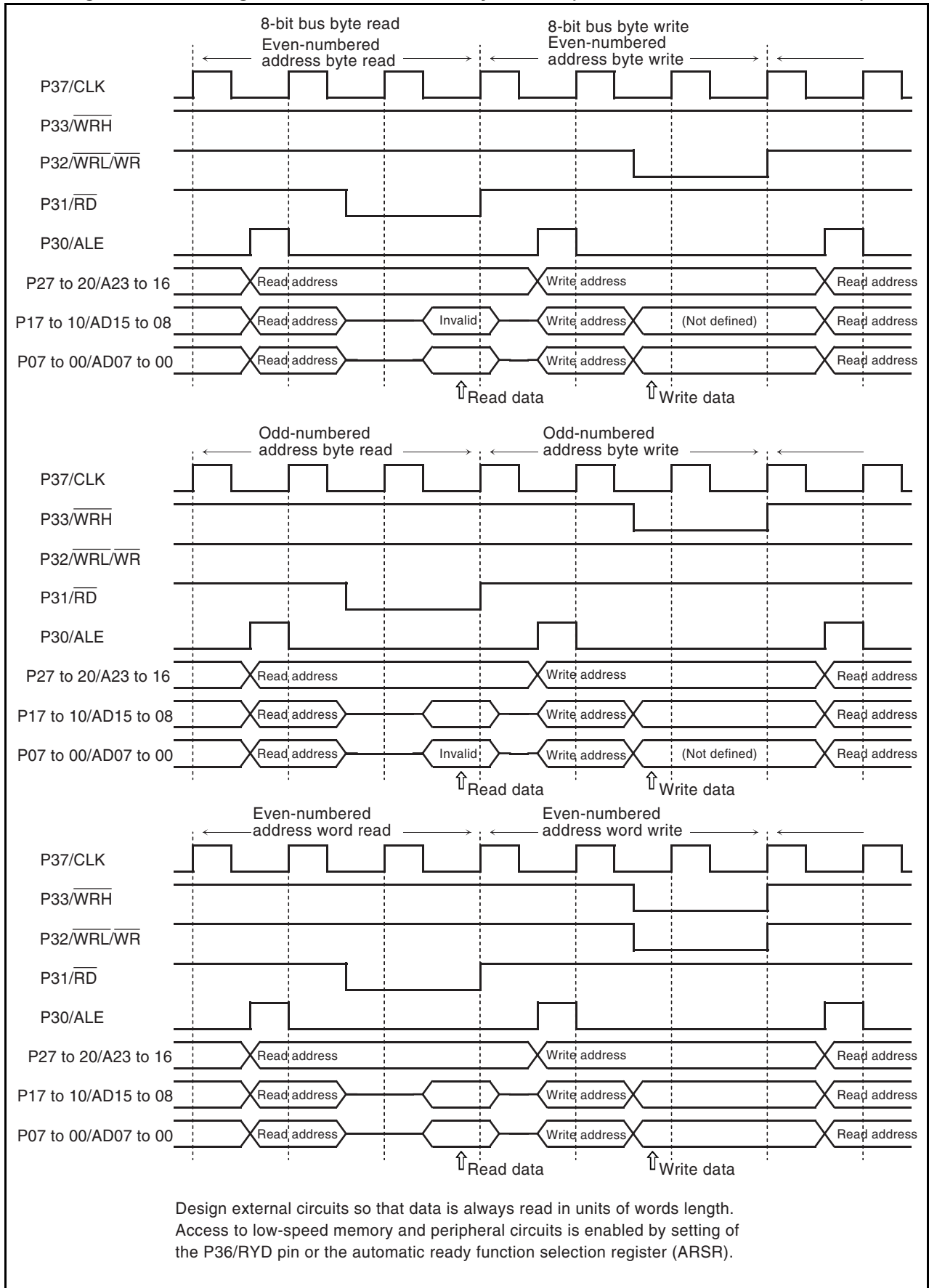


Figure 6.3-2 Timing Chart of External Memory Access (in an External 16-Bit Bus Mode)

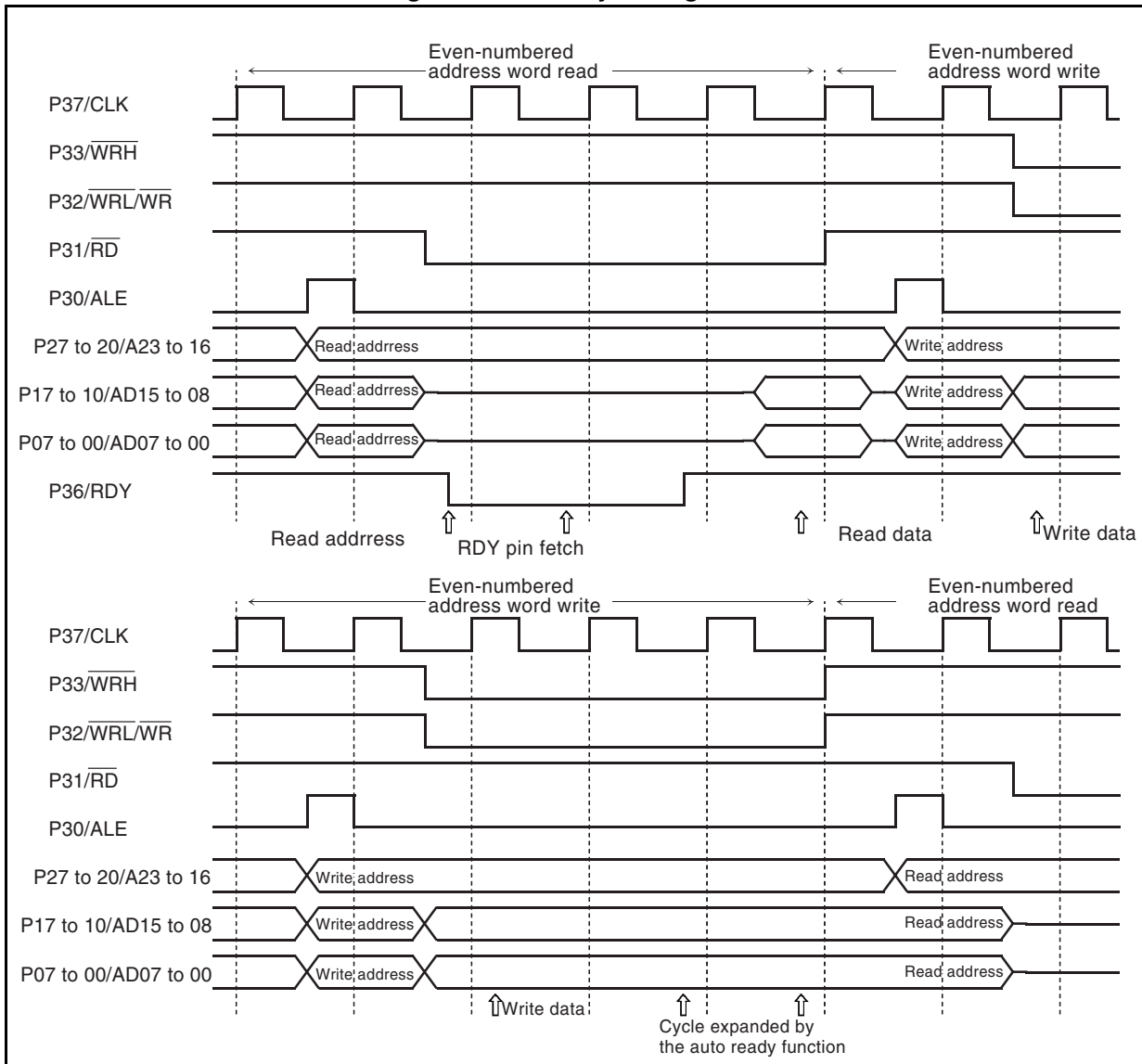
6.3.1 Ready Function

Access to low-speed memory and peripheral circuits is enabled by setting of the P36/RDY pin or the automatic ready function selection register (ARSR).

If the RYE bit of the bus control signal selection register (EPCR) is set to 1, control enters a wait cycle during access to an external area as long as the low level is input to the P36/RDY pin. Thus, an access cycle can be expanded.

■ Ready Function

Figure 6.3-3 Ready Timing Chart



6.3 Operation of the External Memory Access Control Signals

The F²MC-16LX installs two types of the auto ready function. The auto ready function for external memory can expand an access cycle by inserting one to three wait cycles automatically using no external circuit in the following cases: The external area at low-order addresses 002000_H to 7FFFFFF_H is accessed and the external area at high-order addresses 800000_H to FFFFFFF_H is accessed. This function is activated by setting the LMR1 and LMR0 bits (low-order address external area) of the ARSR and the HMR1 and HMR0 bits (high-order address external area) of the ARSR.

Additionally, the F²MC-16LX installs the auto ready function for external I/O independently from the auto ready function for external memory. This function expands an access cycle by inserting one to three wait cycles automatically with no external circuit at access to the external area in addresses 0000C0_H to 0000FF_H. This function is activated by setting the IOR1 and IOR0 bits of the ARSR.

For the auto ready function for external memory and for external I/O, the wait cycle continues as is if the RYE bit of the EPCR is set to 1 and the low level is input to the P36/RDY pin after the wait cycle applied by the auto ready function terminates.

6.3.2 Hold Function

If the HDE bit in the bus control signal selection register (EPCR) is set to 1, the external bus hold function specified by the P34/HRQ and P35/HAK bits is enabled.

■ Hold Function

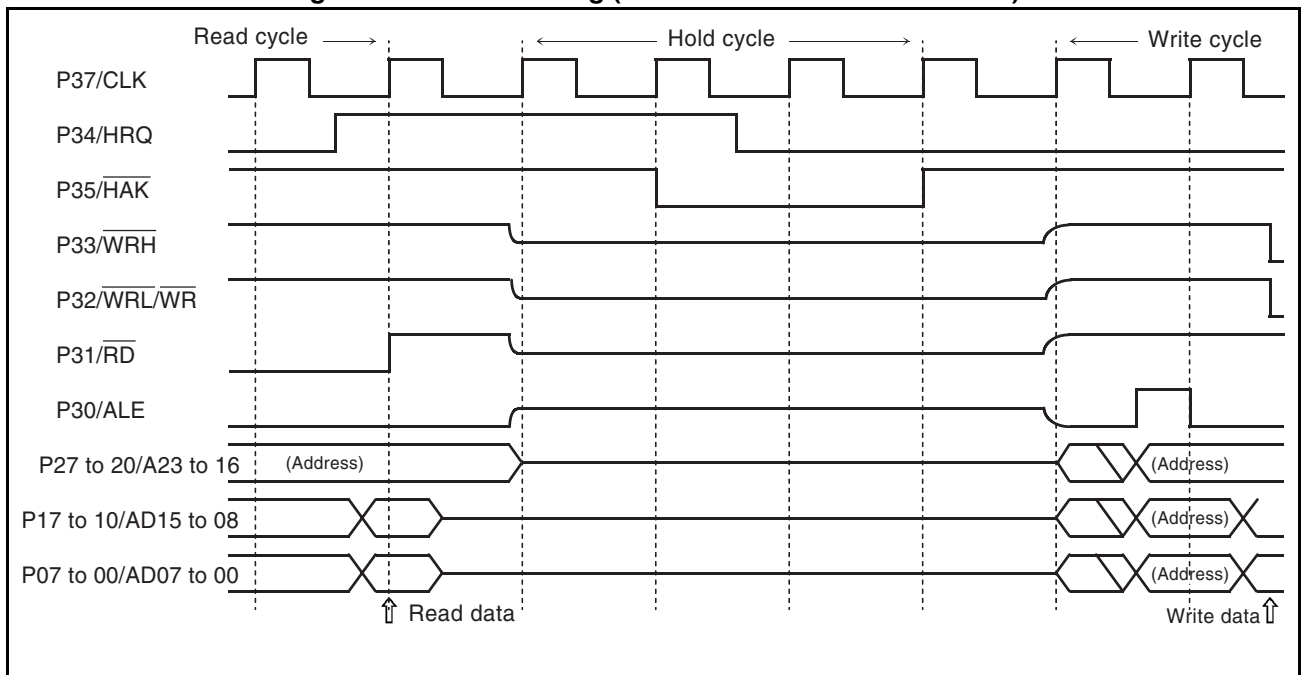
If the high level is applied to the P34/HRQ pin, the hold state is set up at termination of a CPU instruction (for a string instruction, at termination of 1-element data processing). The P35/HAK pin outputs the low level to place the following pins in a high-impedance state:

- Address output: P27/A23 to P20/A16
- Address/data I/O: P17/AD15 to P00/AD00
- Bus control signal: P30/ALE, P31/ $\overline{\text{RD}}$, P32/ $\overline{\text{WRL}}/\overline{\text{WR}}$, P33/ $\overline{\text{WRH}}$

Thus, an external bus can be used from a device external circuit. When the low level is input to the P34/HRQ pin, the P35/HAK pin outputs the high-level, thereby restoring the external pin state and restarting the CPU operation. In the stop status, hold request input is not accepted.

Figure 6.3-4 "Hold Timing (in an External Bus 16-Bit Mode)" shows the hold timing (in an external 16-bit bus mode).

Figure 6.3-4 Hold Timing (in an External Bus 16-Bit Mode)



CHAPTER 7 I/O PORTS

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

7.1 "I/O Port Overview"

7.2 "I/O Port Block Diagram"

7.3 "I/O Port Registers"

7.1 I/O Port Overview

Input or output can be specified for each pin in a port by the data direction register if the corresponding peripheral circuit is specified not to use the pin. If the data register is read while the pin is set to input, the level value of the pin is read. If the data register is read while the pin is set to output, the data register latch value is read. The same is true for read during read modify write.

■ I/O Port Overview

If the data register is read while the pin is used for control output, the level output as control output is read regardless of the value of the data direction register. When a read modify write instruction (such as a bit set instruction) is used for setting output data in the data register in advance to change input setting to output setting, note the following point: The input data, not the data register latch value, is read from the pin.

Ports 0 to A serve as input ports if the data direction register in each I/O port is 0 and as output ports if it is 1.

The MB90580C series also uses port 0 to 3 as external bus pins. Therefore, use of these ports is limited in an external bus mode.

For ports 2 and 3, some bits can be used as port pins by function selection even in an external bus mode. For details, see Section 6.2 "External Memory Access (External Bus Pin Control Circuit)".

7.2 I/O Port Block Diagram

Figure 7.2-1 "Parallel Port Block Diagram (Ports 2, 3, 7, 8, 9, and A)" to Figure 7.2-2 "Parallel Port Block Diagram (Port 4)" show the following I/O port block diagrams.

■ I/O Port Block Diagram

Figure 7.2-1 I/O Port Block Diagram (Ports 0, 1, and 6)

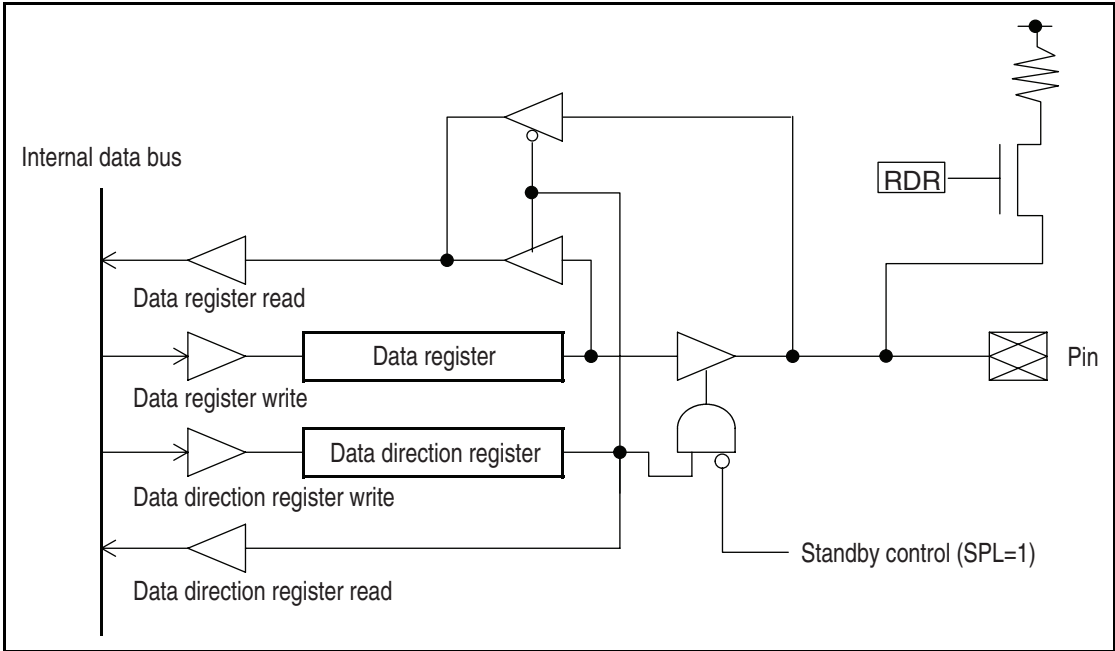


Figure 7.2-2 I/O Port Block Diagram (Ports 2, 3, 7, 8, 9, and A)

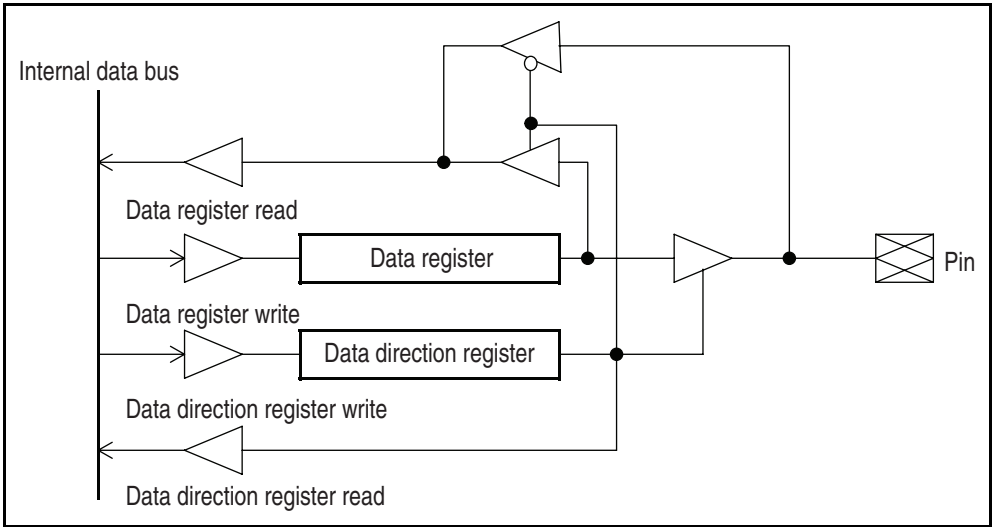


Figure 7.2-3 I/O Port Block Diagram (Port 4)

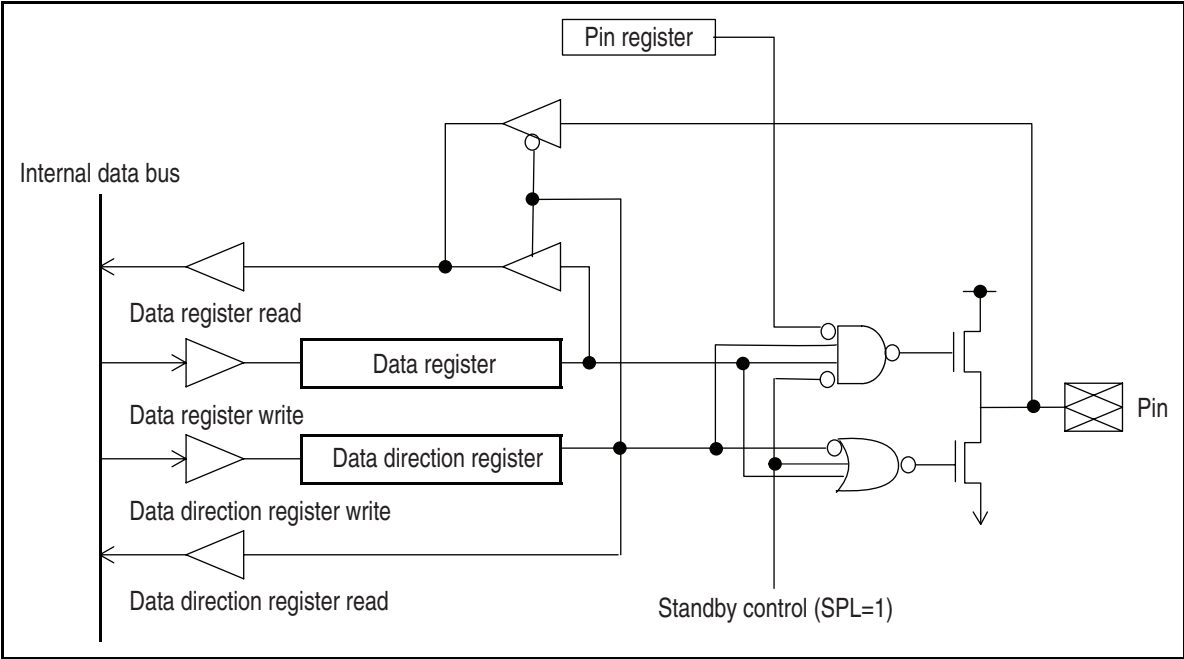
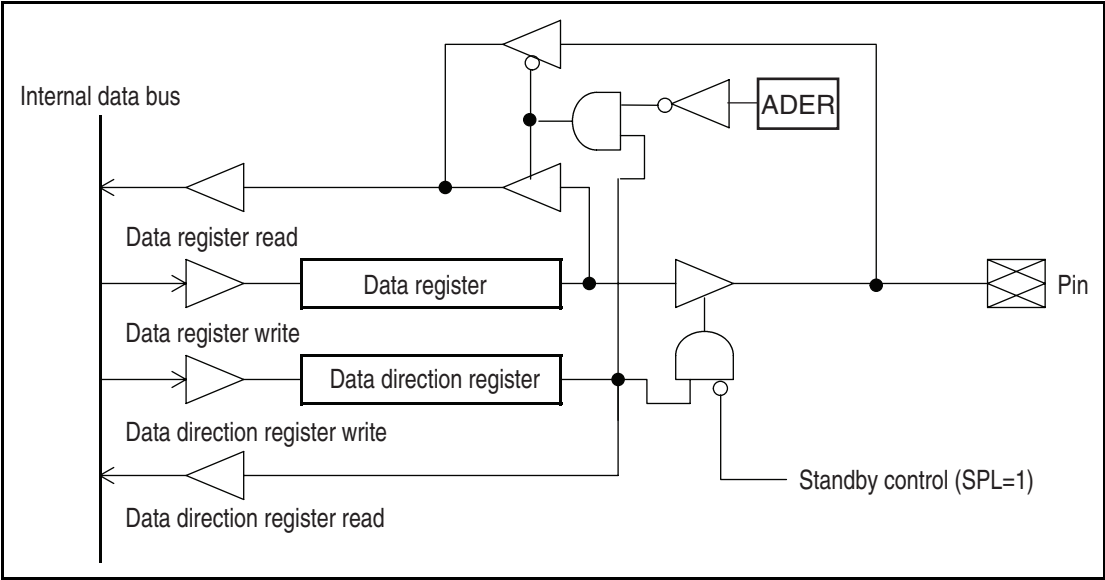


Figure 7.2-4 I/O Port Block Diagram (Port 5)



7.3 I/O Port Registers

The five I/O port registers are as follows:

- Port data registers (PDRx)
- Port data direction registers (DDRx)
- Output pin register (ODR4)
- Input resistor registers (RDR0 and RDR1)
- Analog input enable register (ADER)

■ I/O Port Registers

Figure 7.3-1 I/O Port Registers (to next page)

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00000 _H	P07	P06	P05	P04	P03	P02	P01	P00	Port 0 data register (PDR0)
Address: 00001 _H	P17	P16	P15	P14	P13	P12	P11	P10	Port 1 data register (PDR1)
Address: 00002 _H	P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address: 00003 _H	P37	P36	P35	P34	P33	P32	P31	P30	Port 3 data register (PDR3)
Address: 00004 _H	P47	P46	P45	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address: 00005 _H	P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address: 00006 _H	—	—	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address: 00007 _H	—	—	—	P74	P73	P72	P71	—	Port 7 data register (PDR7)
Address: 00008 _H	P87	P86	P85	P84	P83	P82	P81	P80	Port 8 data register (PDR8)
Address: 00009 _H	P97	P96	P95	P94	P93	P92	P91	P90	Port 9 data register (PDR9)
Address: 0000A _H	—	—	—	—	—	PA2	PA1	PA0	Port A data register (PDRA)
Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00010 _H	D07	D06	D05	D04	D03	D02	D01	D00	Port 0 data direction register (DDR0)
Address: 00011 _H	D17	D16	D15	D14	D13	D12	D11	D10	Port 1 data direction register (DDR1)
Address: 00012 _H	D27	D26	D25	D24	D23	D22	D21	D20	Port 2 data direction register (DDR2)
Address: 00013 _H	D37	D36	D35	D34	D33	D32	D31	D30	Port 3 data direction register (DDR3)
Address: 00014 _H	D47	D46	D45	D44	D43	D42	D41	D40	Port 4 data direction register (DDR4)
Address: 00015 _H	D57	D56	D55	D54	D53	D52	D51	D50	Port 5 data direction register (DDR5)
Address: 00016 _H	—	—	D65	D64	D63	D62	D61	D60	Port 6 data direction register (DDR6)
Address: 00017 _H	—	—	—	D74	D73	D72	D71	—	Port 7 data direction register (DDR7)
Address: 00018 _H	D87	D86	D85	D84	D83	D82	D81	P80	Port 8 data direction register (DDR8)
Address: 00019 _H	D97	D96	D95	D94	D93	D92	D91	D90	Port 9 data direction register (DDR9)
Address: 0001A _H	—	—	—	—	—	DA 2	DA 1	PA0	Port A data direction register (DDRA)

Bit	15	14	13	12	11	10	9	8	
Address: 00001B _H	OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	Port 4 output pin register(ODR4)
Bit	7	6	5	4	3	2	1	0	
Address: 00001C _H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 5 analog input enable register (ADER)
Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00008C _H	RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	Port 0 input resistor register (RDR0)
Address: 00008D _H	RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	Port 1 input resistor register (RDR1)
Address: 00008E _H	—	—	RD65	RD64	RD63	RD62	RD61	RD60	Port 6 input resistor register (RDR6)

7.3.1 Port Data Registers (PDRx)

Pin statuses are read by port data registers (PDRx).

■ Port Data Register (PDRx)

Figure 7.3-2 Port Data Register

	7	6	5	4	3	2	1	0	Initial value	Access
PDR0 Address: 000000 _H	P07	P06	P05	P04	P03	P02	P01	P00	XXXXXXXX _B	R/W
PDR1 Address: 000001 _H	15	14	13	12	11	10	9	8	XXXXXXXX _B	R/W
PDR2 Address: 000002 _H	7	6	5	4	3	2	1	0	XXXXXXXX _B	R/W
PDR3 Address: 000003 _H	15	14	13	12	11	10	9	8	XXXXXXXX _B	R/W
PDR4 Address: 000004 _H	7	6	5	4	3	2	1	0	XXXXXXXX _B	R/W
PDR5 Address: 000005 _H	15	14	13	12	11	10	9	8	11111111 _B	R/W
PDR6 Address: 000006 _H	7	6	5	4	3	2	1	0	--XXXXX _B	R/W
PDR7 Address: 000007 _H	15	14	13	12	11	10	9	8	---XXXX _B	R/W
PDR8 Address: 000010 _H	7	6	5	4	3	2	1	0	XXXXXXXX _B	R/W
PDR9 Address: 000009 _H	15	14	13	12	11	10	9	8	XXXXXXXX _B	R/W
PDRA Address: 00000A _H	7	6	5	4	3	2	1	0	-----XX _B	R/W

Note:

Note that the operation of input port R/W differs from that of memory R/W.

Input mode

- Read: The level of a corresponding pin is read.
- Write: An output latch is written.

Output mode

- Read: The data register latch value is read.
- Write: Output to a corresponding pin

7.3.2 Port Data Direction Registers (DDRx)

In a port data direction register (DDRx), a bit sets the I/O direction of its corresponding pin. If the bit corresponding to a port (pin) is set to 1, the port becomes an output port (pin). If the bit is set to 0, the port becomes an input port (pin).

■ Port Data Direction Register (DDRx)

Figure 7.3-3 Port Data Direction Register (DDRx)

DDRx	7	6	5	4	3	2	1	0	Initial value	Access
DDR0 Address: 000010 _H	D07	D06	D05	D04	D03	D02	D01	D00	00000000 _B	R/W
DDR1 Address: 000011 _H	15	14	13	12	11	10	9	8	00000000 _B	R/W
DDR2 Address: 000012 _H	7	6	5	4	3	2	1	0	00000000 _B	R/W
DDR3 Address: 000013 _H	15	14	13	12	11	10	9	8	00000000 _B	R/W
DDR4 Address: 000014 _H	7	6	5	4	3	2	1	0	00000000 _B	R/W
DDR5 Address: 000015 _H	15	14	13	12	11	10	9	8	00000000 _B	R/W
DDR6 Address: 000016 _H	7	6	5	4	3	2	1	0	--000000 _B	R/W
DDR7 Address: 000017 _H	15	14	13	12	11	10	9	8	---0000- _B	R/W
DDR8 Address: 000018 _H	7	6	5	4	3	2	1	0	00000000 _B	R/W
DDR9 Address: 000019 _H	15	14	13	12	11	10	9	8	00000000 _B	R/W
DDRA Address: 00001A _H	7	6	5	4	3	2	1	0	-----000 _B	R/W

A port data direction register (DDRx) controls each pin as listed in Table 7.3-1 "Function of a Port Data Direction Register (DDRx)" when the pin functions as a port pin.

Table 7.3-1 Function of a Port Data Direction Register (DDRx)

DDRx	Function
0	Input mode [initial value]
1	Output mode

Note:

When using as a resource input, specify 0 (input mode).

7.3.3 Port 4 Output Pin Register (ODR4)

The Port 4 output pin register (ODR4) controls open-drain in an output mode.

■ Port 4 Output Pin Register (ODR4)

Figure 7.3-4 Port 4 Output Pin Register (ODR4)

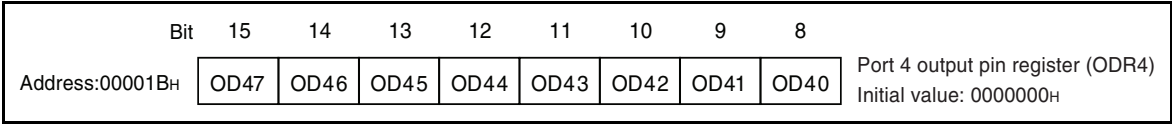


Table 7.3-2 Function of the Output Pin Register (ODR4)

ODR4	Function
0	Sets the port as a standard output port in an output mode. [Initial value]
1	Sets the port as the open-drain output port in an output mode.

Note:

This register has no meaning in an input mode. (Output Hi-z)
The data direction register (DDR) determines the I/O mode.

7.3.4 Input Pull-up Resistor Setting Registers (RDR0, RDR1, and RDR6)

An input pull-up resistor setting register (RDR0, RDR1, and RDR6) controls a pull-up resistor in an input mode.

■ Input Pull-up Resistor Setting Registers (RDR0, RDR1, and RDR6)

Figure 7.3-5 Input Pull-up Resistor Setting Registers (RDR0, RDR1, and RDR6)

Bit	7	6	5	4	3	2	1	0	
Address:00008CH	RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	Port 0 input pull-up resistor setting register (RDR0) Initial value: 00000000b
Bit	15	16	15	14	13	12	11	10	
Address:00008DH	RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	Port 1 input pull-up resistor setting register (RDR1) Initial value: 00000000b
Bit	7	6	5	4	3	2	1		
Address:00008EH	RD67	RD66	RD65	RD64	RD63	RD62	RD61	RD60	Port 6 input pull-up resistor setting register (RDR6) Initial value: 00000000b

Table 7.3-3 Function of the Input Pull-up Resistor Setting Registers (RDR0, RDR1, and RDR6)

RDR0,1,6	Function
0	Without a pull-up resistor in an input mode [initial value]
1	With a pull-up resistor in an input mode

Note:

These registers have no meaning in an output mode. (Without a pull-up resistor)

The data direction register (DDR) determines the I/O mode.

The pull-up resistor is not provided during hardware standby and stop (SPL = 1)(high impedance).

This function is prohibited in an external bus mode. Do not write this register.

7.3.5 Port 5 Analog Input Enable Register (ADER)

The Port 5 analog input enable register (ADER) controls each pin of port 5 as listed in Table 7.3-4 "Port 5 Analog Input Enable Register (ADER)".

■ Port 5 Analog Input Enable Register (ADER)

Figure 7.3-6 Port 5 Analog Input Enable Register (ADER)

Bit		7	6	5	4	3	2	1	0		
Address:00001Ch		ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 5 analog input enable register (ADER) Initial value: 11111111b	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

Table 7.3-4 Port 5 Analog Input Enable Register (ADER)

ADER	Setting
0	Port input mode
1	Analog input mode [initial value]

Note:
If a middle-level signal is input in the port input mode, an input leakage current flows. Therefore, set the analog input mode for analog input.

CHAPTER 8 TIME-BASED TIMER

This chapter describes the functions and operations of the time-based timer.

- 8.1 "Overview of the Time-Based Timer"
- 8.2 "Time-Based Timer Control Register (TBTC)"
- 8.3 "Time-Based Timer Operations"

8.1 Overview of the Time-Based Timer

The time-based timer consists of an 18-bit timer and a circuit for controlling interval interrupts and uses oscillation clocks regardless of the MCS bit in CKSCR.

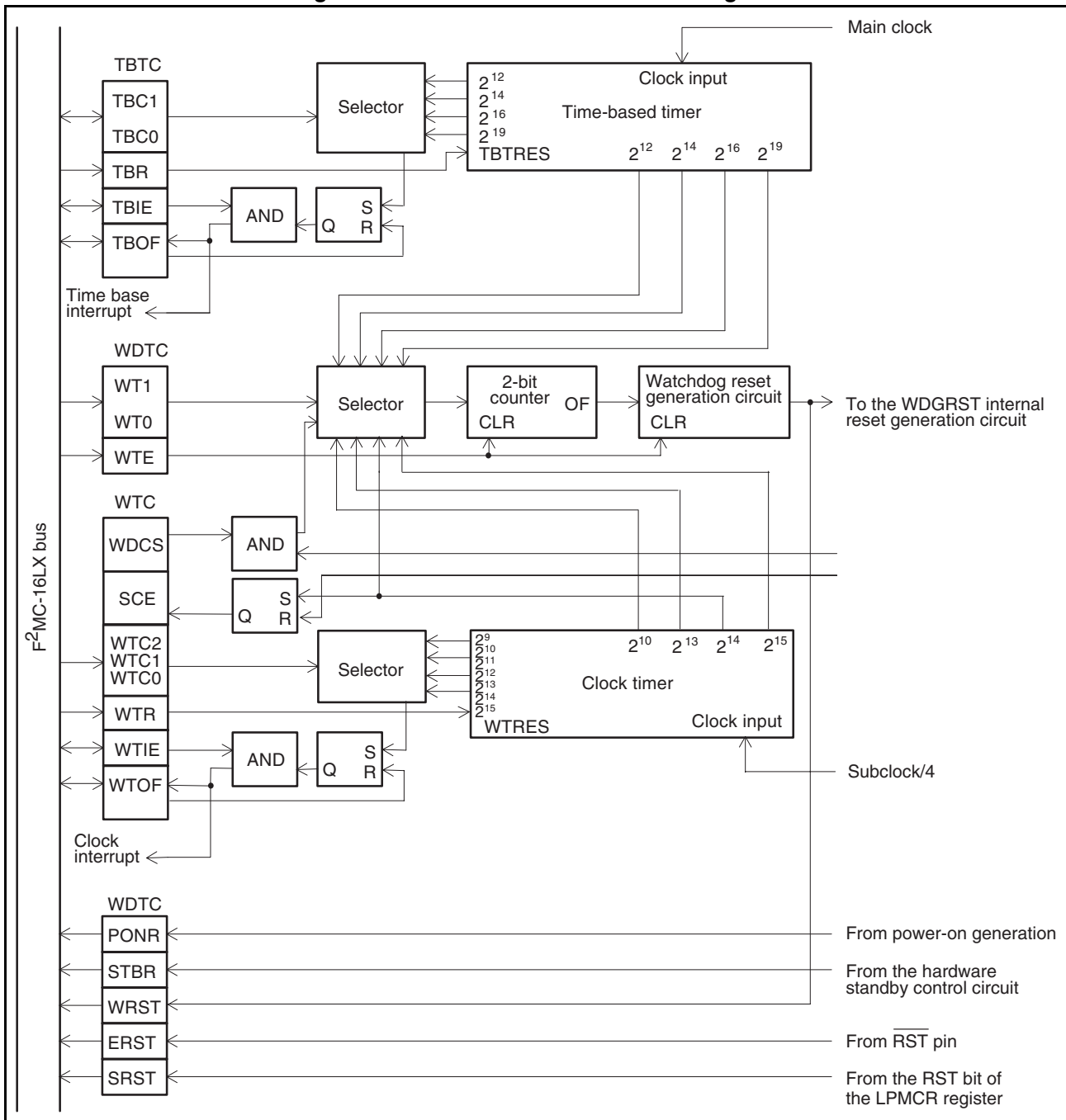
■ Time-based Timer Register

Figure 8.1-1 Time-based Timer Register

Time-based 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)	

■ Time-based Timer Block Diagram

Figure 8.1-2 Time-based Timer Block Diagram



8.2 Time-Based Timer Control Register (TBTC)

The time-based timer control register (TBTC) can control time-based timer interrupts and can clear the time base counter.

■ Time-based Timer Control Register (TBTC)

Figure 8.2-1 Configuration of the Time-based Timer Control Register (TBTC)

Time-based 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:

Because access by read modify instructions causes malfunctions, do not use the read modify instructions to obtain access.

[Bit 15] Reserved

Bit 15 is reserved. To set TBTC, this bit must be set to 1.

[Bit 12] TBIE

TBIE is used to enable an interval interrupt to be generated by the time-based timer. When this bit is 1, an interrupt is enabled. When it is 0, the interrupt is disabled. It is initialized to 0 by reset and is readable and writable.

[Bit 11] TBOF

TBOF is a flag for requesting a time-based timer interrupt. An interrupt occurs when the TBIE bit is 1 and TBOF is set to 1. TBOF is set to 1 for each interval set by the TBC1 and TBC0 bits. It is cleared by writing 0, the transition to the hardware standby mode or stop mode, or reset. Writing 1 is meaningless.

Value 1 is read during reading by read modify write instructions.

[Bit 10] TBR

The TBR bit is used to clear all bits of the time-based timer counter to 0. The time base counter is cleared by writing 0. Writing 1 is meaningless. During reading, 1 is read.

Note:

To clear the TBOF bit, mask a time-based timer interrupt by the TBIE bit or CPU ILM bit.

[Bits 9 and 8] TBC1 and TBC0

TBC1 and TBC0 bits are used to set an interval of the time-based timer.

These bits are initialized to 00 by reset and are readable and writable.

Table 8.2-1 Interval times and number of cycles for TBC1 and TBC0

TBC1	TBC0	Interval time for oscillation 4MHz	Number of oscillation clock (oscillation) cycles
0	0	1.024 ms	2^{12}
0	1	4.096 ms	2^{14}
1	0	16.384 ms	2^{16}
1	1	131.072 ms	2^{19}

8.3 Time-Based Timer Operations

The time-based timer functions as a timer for waiting for an oscillation stabilization time of a watchdog timer clock source, main clock, and PLL clock and as an interval timer for generating an interrupt in a given cycle.

■ Time-based Timer Operations

The time-based timer consists of an 18-bit counter for counting oscillation inputs that are the origin for creating machine clocks. The timer continues the count operation while oscillation is input.

The time base counter is cleared by power-on reset, a transition to the stop mode or hardware standby mode, a transition from the main clock to the PLL clock by the MCS bit in the CKSCR register, or by writing 0 in the TBR bit of the TBTC register.

The watchdog timer and interval interrupts that use time-based timer outputs are influenced by time-based timer clear.

■ Interval Interrupt Function

This function generates an interrupt in a give cycle with a carry signal of the time base counter. The TBOF flag is set for each interval time set by TBC1 and TBC0 bits in the TBTC register. This flag is set on the basis of the time the time-based timer was last cleared.

When the main clock mode changes to the PLL clock mode, the time-based timer is cleared because the time-based timer is used to wait for the oscillation stabilization of the PLL clock.

When the stop mode or hardware standby mode is entered, the TBOF flag is cleared simultaneously with mode transition because the time-based timer is used to wait for the oscillation stabilization time at return.

CHAPTER 9 WATCHDOG TIMER

This chapter describes the functions and operations of the watchdog timer.

- 9.1 "Overview of the Watchdog Timer"
- 9.2 "Watchdog Timer Control Register (WDTC)"
- 9.3 "Watchdog Timer Operations"

9.1 Overview of the Watchdog Timer

The watchdog timer consists of the 2-bit watchdog counter which uses a carry signal of the 18-bit time-based timer as a clock source, the control register, and the watchdog reset control section.

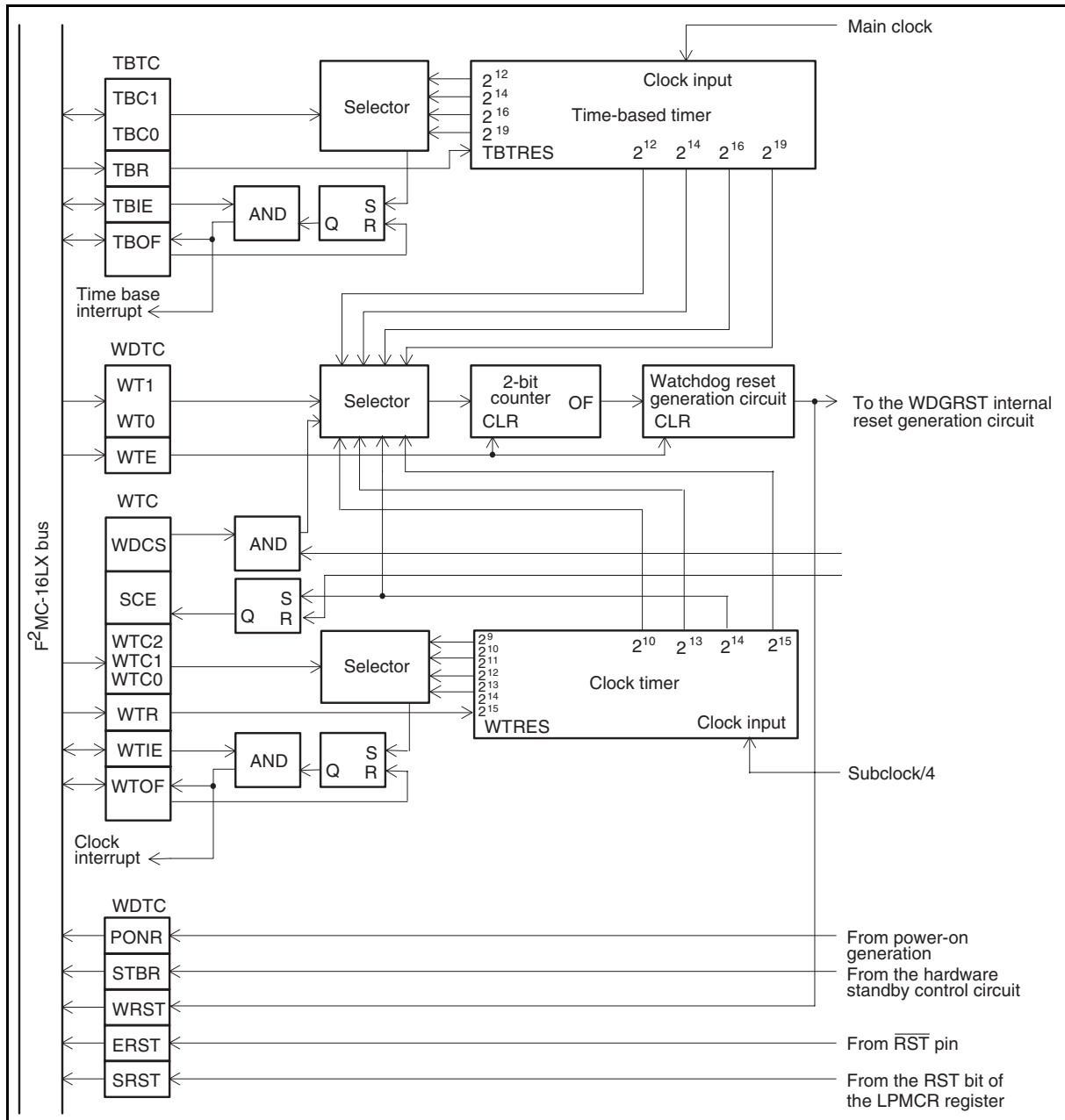
■ Watchdog Timer Register

Figure 9.1-1 Watchdog Timer Register

Watchdog control register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:0000A8H	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)	(1)	(1)	(1)	

■ Watchdog Timer Block Diagram

Figure 9.1-2 Watchdog Timer Block Diagram



9.2 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) activates and clears the watchdog timer and displays a reset cause.

■ Watchdog Timer Control Register (WDTC)

Figure 9.2-1 Watchdog Timer Control Register (WDTC)

Watchdog control register	7	6	5	4	3	2	1	0	↔ Bit No.
Address:0000A8H	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)	(1)	(1)	(1)	

Note:

Because access by read modify instructions causes malfunctions, do not use the read modify instructions to obtain access.

[Bits 7 to 3] PONR, STBR, WRST, ERST, and SRST

These bits are flags for indicating reset sources and are set by each reset as shown in Table 9.2-1 "PONR, STBR, WRST, ERST, and SRST (for Sources of Resets)". After the WDTC register read operation, all bits are cleared to 0.

The WDTC register is a read-only register. Note that because the bit contents are not predictable when they indicate a reset cause other than power-on, software programs should be designed so that bits other than the PONR bit will be ignored when the PONR bit is 1.

Table 9.2-1 PONR, STBR, WRST, ERST, and SRST (for Sources of Resets)

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

*: Retains the previous value.

[Bit 2] WTE

When the watchdog timer stops and 0 is written in WTE, the watchdog timer becomes operable. Second and subsequent 0 writings clear the watchdog timer counter. Writing 1 does not result in an operation.

The watchdog timer stops by power-on, hardware standby, or reset by the watchdog timer. Value 1 is read at reading.

[Bits 1 and 0] WT1 and WT0

Two clocks system:

WT1 and WT0 bits are used to select an interval time of the watchdog timer. Only data written during watchdog timer activation is valid. Data written at operation other than watchdog timer activation is ignored. These bits are writable only. Note that the clock to be input to the watchdog timer is selected according to the WDCS bit of the WTC register.

Table 9.2-2 "WT1 and WT0 (Interval Time Selection Bits) of Two Clocks System" lists the interval time settings by the WT1 and WT0 bits.

Table 9.2-2 WT1 and WT0 (Interval Time Selection Bits) of Two Clocks System

WDCS	WT1	WT0	Interval time (for oscillation clock frequency 4 MHz)	
			Minimum	Maximum (*)
1	0	0	About 3.58 ms	About 4.61 ms
1	0	1	About 14.33 ms	About 18.43 ms
1	1	0	About 57.23 ms	About 73.73 ms
1	1	1	About 458.75 ms	About 589.82 ms
0	0	0	About 0.457 s	About 0.576 s
0	0	1	About 3.584 s	About 4.608 s
0	1	0	About 7.168 s	About 9.216 s
0	1	1	About 14.336 s	About 18.432 s

*: The maximum values of interval time are applied when the time base counter is not reset during the watchdog timer operation.

CHAPTER 9 WATCHDOG TIMER

One clock system:

Bits WT1 and WT0 are used to select an interval time for the watchdog timer. Only data written during Watchdog timer activation is valid. Data written at other times is ignored. These bits are write only.

Table 9.2-3 "WT1 and WT0 (Interval Time Selection Bits) of One Clock System" lists the interval times set by bits WT1 and WT0.

Table 9.2-3 WT1 and WT0 (Interval Time Selection Bits) of One Clock System

WDCS	WT1	WT0	Interval time (for oscillation clock frequency 4 MHz)	
			Minimum	Maximum (*)
1	0	0	About 3.58 ms	About 4.61 ms
1	0	1	About 14.33 ms	About 18.43 ms
1	1	0	About 57.23 ms	About 73.73 ms
1	1	1	About 458.75 ms	About 589.82 ms

*: The maximum values of interval time are applied when the time base counter is not reset during the watchdog timer operation.

9.3 Watchdog Timer Operations

The watchdog timer function can detect a program crash.

The watchdog timer requests a reset if 0 is not written in the WTE bit of the WDTC register within the specified time due to a program crash.

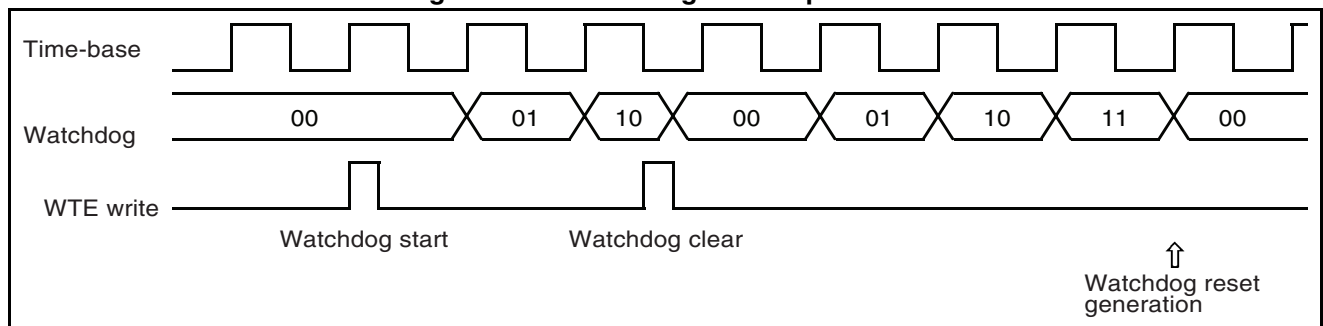
■ Activating the Watchdog Timer

The watchdog timer is activated by writing 0 in the WTE bit of the WDTC register while the watchdog timer stops. At the same time, the reset generation interval of the watchdog timer is set by WT1 and WT0 bits. Only data at this activation is valid for interval setting.

■ Preventing a Watchdog Timer Reset

When the watchdog timer is started, the 2-bit watchdog counter must be cleared periodically in the program. In effect, 0 must be periodically written in the WTE bit of the WDTC register. The watchdog counter consists of a 2-bit counter that uses a carry signal of the time-based timer as a clock source. Therefore, if the time-based timer is cleared, the watchdog reset generation time may become longer than the specified time.

Figure 9.3-1 Watchdog Timer Operations



■ Stopping the Watchdog

Once the watchdog timer is started, it is only initialized by power-on, hardware standby, or a reset with the watchdog and is put in stop status.

The watchdog counter is cleared by a reset with an external pin or software, though the watchdog function does not stop.

■ Clearing the Watchdog Timer

The watchdog timer is cleared by a write to the WTE bit, the reset generation, a transition to the sleep or stop mode, or the hold acknowledge signal.

CHAPTER 10 WATCH TIMER

**This chapter describes the functions and operations of the watch timer.
The watch timer cannot be used for the one clock system.**

10.1 "Overview of the Watch Timer"

10.2 "Watch Timer Control Register (WTC)"

10.3 "Watch Timer Operations"

10.1 Overview of the Watch Timer

The watch timer functions as a clock source for the watchdog timer, a timer to wait for subclock oscillation stabilization time, and an interval timer to generate interrupts at specified intervals.

■ Watch Timer Control Register

Figure 10.1-1 Watch Timer Control Register (WTC)

Watch timer control register									
	7	6	5	4	3	2	1	0	↔ Bit No.
Address: 0000AA _H	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)	

The diagram illustrates the internal architecture of the F2MC-16LX bus, organized into three main functional blocks: Time-based timer, Watchdog timer, and Watch timer.

Time-based timer: This block includes a **Time-based timer** unit with a **Clock input** and **TBTRES** output. It is connected to a **Selector** and an **AND** gate. The **AND** gate's output is connected to the **Q** output of a flip-flop (labeled **S** and **R**). The **Q** output is also connected to the **TBOF** output of the **TBTC** block. The **TBTC** block contains **TBC1**, **TBC0**, **TBR**, **TBIE**, and **TBOF** outputs. The **TBOF** output is connected to the **Time base interrupt** signal.

Watchdog timer: This block includes a **Watchdog reset generation circuit** with a **CLR** output. It is connected to a **Selector** and a **2-bit counter** (labeled **CLR** and **OF**). The **2-bit counter** is connected to the **CLR** output of the **Watchdog reset generation circuit**. The **Watchdog reset generation circuit** is connected to the **WDGRST** output, which is labeled **To internal reset generation circuit**.

Watch timer: This block includes a **Watch timer** unit with a **Clock input** and **WTRES** output. It is connected to a **Selector** and an **AND** gate. The **AND** gate's output is connected to the **Q** output of a flip-flop (labeled **S** and **R**). The **Q** output is also connected to the **WTOF** output of the **WTC** block. The **WTC** block contains **WDCS**, **SCE**, **WTC2**, **WTC1**, **WTC0**, **WTR**, **WTIE**, and **WTOF** outputs. The **WTOF** output is connected to the **Clock interrupt** signal.

Other components: The diagram also shows a **WDTC** block with **PONR**, **STBR**, **WRST**, **ERST**, and **SRST** outputs. These outputs are connected to various external signals: **PONR** (From power-on generation), **STBR** (From hardware standby control circuit), **WRST** (RST pin), and **ERST** (From RST bit of LPMCR register).

10.2 Watch Timer Control Register (WTC)

The watch timer control register (WTC) can select a clock signal, control interrupts and their intervals, and clear the watch timer counter.

■ Watch Timer Control Register (WTC)

Figure 10.2-1 Watch Timer Control Register (WTC)

Watch timer control register									
	7	6	5	4	3	2	1	0	↔ Bit No.
Address: 0000AA _H	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)	

[Bit 7] WDCS

The WDCS bit is used to select the clock signal from the watch timer or the clock signal from the time base timer as the input clock for the watchdog timer when the main clock or PLL clock is selected as the clock source. When this bit is 0, the clock signal from the watch timer is selected. When this bit is 1, the clock signal from the time base timer is selected. In the subclock mode, select the output of the watch timer with the WDCS bit setting to "0".

If the mode transits to the subclock mode with the WDCS bit setting to "1", the watchdog timer stops.

This bit is initialized to 1 by power-on reset.

Note:

When the WDCS bit is set to 1, the watchdog timer counter may run because time base timer output and watch timer output are asynchronous. To prevent this, when the WDCS bit is set to 1, clear the watchdog timer before and after the clock mode is changed.

[Bit 6] SCE

The SCE bit indicates the progress of subclock oscillation stabilization. When this bit is 0, subclock oscillation stabilization is in progress. The oscillation stabilization time is fixed to 2^{16} cycles (for subclock). This bit is initialized to 0 by power-on reset or when the device is stopped.

[Bit 5] WTIE

The WTIE bit is used to enable or disable the interval interrupt by the watch timer. When this bit is 1, the interval interrupt is enabled. When this bit is 0, the interval interrupt is disabled. This bit is readable and writable and is initialized to 0 by reset.

[Bit 4] WTOF

The WTOF bit is a flag to request a watch timer interrupt. An interrupt request is generated when the WTIE bit is 1 and the WTOF bit is 1. The WTOF bit is set to 1 at the interval specified by the WTC2 to WTC0 bits. The WTOF bit is cleared by writing 0 in the bit, transition to the stop or hardware standby mode, or reset. Writing 1 in this bit is ignored.

When this bit is read by a read, modify, or write instruction, the read value is always 1.

[Bit 3] WTR

The WTR bit is used to clear all bits of the watch timer counter to 0. Writing 0 in the WTR bit clears the clock counter. Writing 1 in the WTR bit is ignored. When the WTR bit is read, the read value is always 1.

[Bits 2 to 0] WTC2, WTC1, and WTC0

The WTC2, WTC1, and WTC0 bits are used to set an interval for the watch timer. Interval settings are shown in Table 10.2-1 "Selection of watch timer interval". These bits are readable and writable and are initialized to 000 by reset.

When writing a value in these bits, clear the bit 4 (WTOF).

Table 10.2-1 Selection of watch timer interval

WTC2	WTC1	WTC0	Interval (*)
0	0	0	62.5 ms
0	0	1	125.0 ms
0	1	0	250 ms
0	1	1	500 ms
1	0	0	1.0 s
1	0	1	2.000 s
1	1	0	4.000 s
1	1	1	-

*: The interval is a value with the subclock at 32 kHz.

10.3 Watch Timer Operations

The watch timer functions as a clock source for the watchdog timer, a timer to wait for subclock oscillation stabilization time, and an interval timer to generate interrupts at specified intervals.

■ Operations

The watch timer has a 15-bit counter to count the source oscillation inputs that are used to generate subclock signals. The watch timer continues the counting operation while source oscillation inputs continue. The watch timer is cleared by power-on reset, transition to the stop or hardware standby mode, or writing 0 in the WTR bit of the WTC register.

Clearance of the watch timer affects the watchdog counter and the interval interrupt function, which use watch timer output.

■ Interval Interrupt Function

The interval interrupt function generates an interrupt at specified intervals with a carry signal of the clock counter. The WTOF flag is set at each interval set by the WTC2 to WTC0 bits of the WTC register. The timing of flag setting is based on the time when the watch timer was last cleared.

When the device enters the stop or hardware standby mode, the WTOF flag is cleared at the same time as the mode transition because the watch timer is used to wait for the end of subclock oscillation stabilization time when restoring.

■ Setting Operation Clock for Watchdog Timer

The clock source of the watchdog timer can be set by the WDSCS bit in the watch timer control register (WTC). When the subclock is used for machine clock, select the watch timer output with the WDSCS bit setting to "0". If the mode transits to the subclock mode with the WDSCS bit setting to "1", the watchdog timer stops.

CHAPTER 11 PWC TIMER

This chapter describes the functions and operations of the PWC timer.

- 11.1 "Overview of the PWC Timer"
- 11.2 "PWC Timer Block Diagram"
- 11.3 "PWC Timer Registers"
- 11.4 "PWC Timer Operations"
- 11.5 "Details of Timer Mode Operation"
- 11.6 "Flowchart of Timer Mode Operation"
- 11.7 "Details of Pulse Width Measurement Mode Operation"
- 11.8 "Notes on Handling the PWC Timer"

11.1 Overview of the PWC Timer

The PWC timer (pulse-width measurement) is the multifunction 16-bit up counter with the reload function and also has a function that calculates the pulse width of the input signal.

The PWC timer consists of a 16-bit counter, an input pulse divider, a division rate control register, a count input pin, a pulse output pin, and a 16-bit control register.

■ Characteristics of PWC Timer

The PWC timer has the following characteristics:

○ Timer function

- Generates an interrupt request at the specified time interval.
- Outputs the pulse signal that is synchronized with the timer period.
- Selects the counter clock from three internal clocks.

○ Pulse-width measurement function

- Measures the time between external pulse input events.
- Selects the counter clock from three internal clocks.
- Count mode
 - H pulse width (rising edge to falling edge)/L pulse width (falling edge to rising edge)
 - Rising edge period (rising edge to rising edge)/falling edge period (falling edge to falling edge)
 - Intermediate edge count (rising or falling edge to falling or rising edge)
- Uses the 8-bit input divider to divide the input pulse by 2^2 , 2^4 , 2^6 , and 2^8 to enable period measurement.
- Generates an interrupt request at completion of count.
- Selects single count or continuous count.

The MB90580C series contains one PWC timer channel.

■ PWC Timer Operation

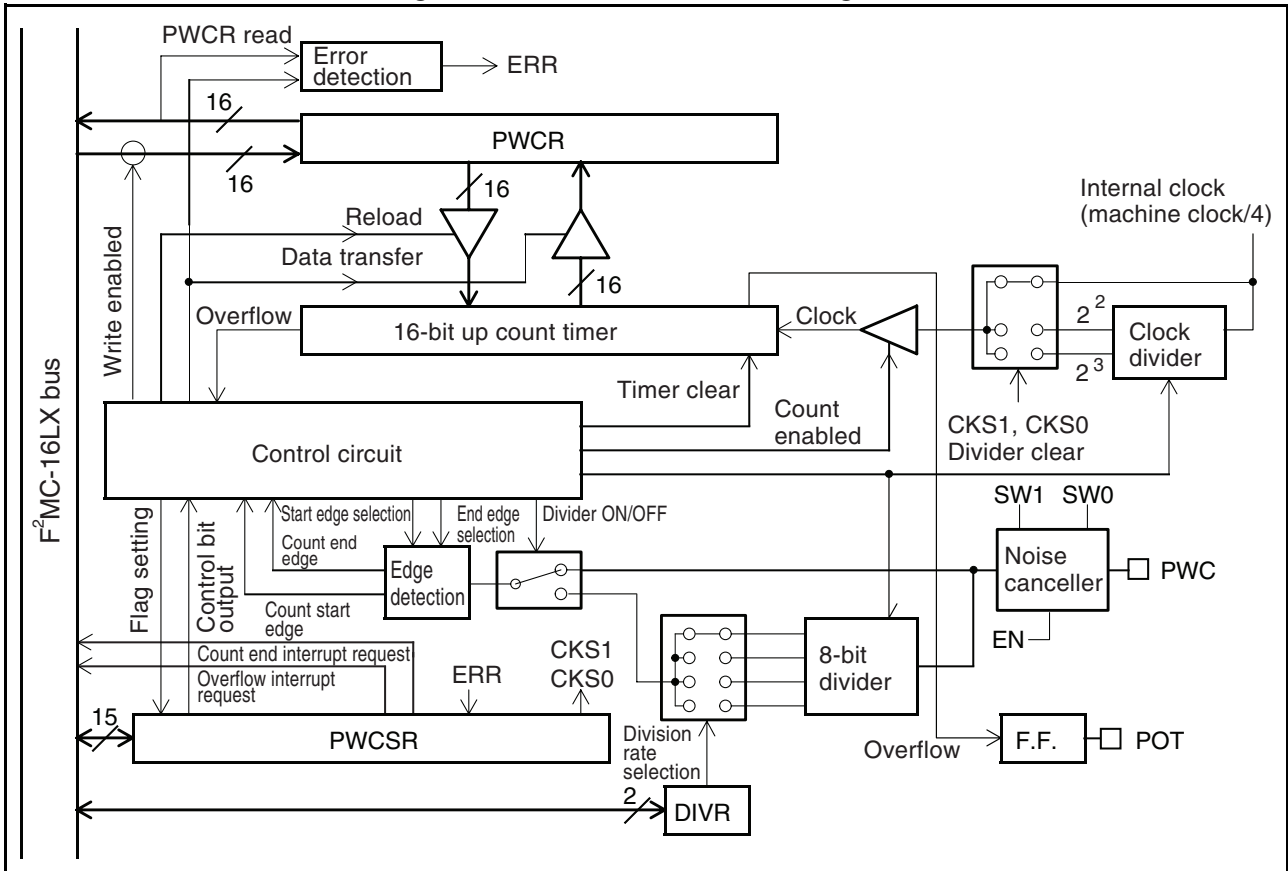
This block is a multifunction timer that is based on the 16-bit up count timer and contains a count input pin and an 8-bit input divider. The block has two main functions, a timer function and a pulse-width measurement function, both of which enable the selection of two types of count clocks.

11.2 PWC Timer Block Diagram

Figure 11.2-1 "PWC Timer Block Diagram" is the PWC timer block diagram.

■ PWC Timer Block Diagram

Figure 11.2-1 PWC Timer Block Diagram



11.3 PWC Timer Registers

This section explains the PWC timer registers.

■ PWC Timer Registers

○ PWC control status register (high-order byte)

	15	14	13	12	11	10	9	8	⇌ Bit number
Address:000055H	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	POUT	PWCSR (HIGH)
Read/write ⇌	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	
Initial value ⇌	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

○ PWC control status register (low-order byte)

	7	6	5	4	3	2	1	0	⇌ Bit number
Address:000054H	CKS1	CKS0	Reserved	Reserved	S/C	MOD2	MOD1	MOD0	PWCSR (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)	

○ PWC data buffer register (high-order byte)

	15	14	13	12	11	10	9	8	⇌ Bit number
Address:000057H									PWCR (HIGH)
Read/write ⇌	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	
Initial value ⇌	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

○ PWC data buffer register (low-order byte)

	7	6	5	4	3	2	1	0	⇌ Bit number
Address:000056H									PWCR (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)	

○ Division rate control register

	7	6	5	4	3	2	1	0	↔ Bit number
Address:000058 _H	-	-	-	-	-	-	DIV1	DIV0	DIVR
Read/write ↔	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	(R/W)	
Initial value ↔	(-)	(-)	(-)	(-)	(-)	(-)	(0)	(0)	

○ PWC noise filter register

	7	6	5	4	3	2	1	0	↔ Bit number
Address:000086 _H	-	-	-	-	-	SW1	SW0	EN	RNCR
Read/write ↔	(-)	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	
Initial value ↔	(-)	(-)	(-)	(-)	(-)	(0)	(0)	(0)	

11.3.1 PWC control status register (PWCSR)

The PWC control status register (PWCSR) controls the PWC timer operation and reads the PWC timer state.

■ PWC Control Status Register (PWCSR)

Figure 11.3-1 "PWC Control Status Register (PWCSR)" shows the register configuration of the PWC control status register (PWCSR).

Figure 11.3-1 PWC Control Status Register (PWCSR)

	15	14	13	12	11	10	9	8	⇐ Bit number
Address:000055H	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	POUT	PWCSR (HIGH)
Read/write ⇨	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

	7	6	5	4	3	2	1	0	⇐ Bit number
Address:000054H	CKS1	CKS0	Reserved	Reserved	S/C	MOD2	MOD1	MOD0	PWCSR (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)	

[Bits 15 and 14] STRT (start) and STOP (stop)

STRT (start) and STOP (stop) bits are used to start, restart, and stop the 16-bit up counte timer. When the bits are read, the timer operation status is returned. Table 11.3-1 "Operation Control Functions of STRT and STOP Bits (for write)" lists the functions of the bits to be written and Table 11.3-2 "Operation Status Indication of START and STOP Bits (for read)" lists the functions of the bits to be read.

Table 11.3-1 Operation Control Functions of STRT and STOP Bits (for write)

STRT	STOP	Operation status indication
0	0	No function. The operation is not affected.
0	1	Starts or restarts the timer (enables count). Note: The clear bit instruction can be used.
1	0	Stops the timer operation (disables count). Note: The clear bit instruction cannot be used.
1	1	No function. The operation is not affected.

Table 11.3-2 Operation Status Indication of STRT and STOP Bits (for read)

STRT	STOP	Operation status indication
0	0	Timer stop (the timer is not started or count ends.) (Initial value)
1	1	Timer count operation in progress (counting)

After reset, the bits are initialized to 00_B.

The bits can be read and written. Note that the meanings of bits depend on whether they are read or written.

A read modify write instruction always reads the bits as 11_B.

When the STRT and STOP bits are written to start and stop the timer, a bit manipulation instruction (such as bit clear instruction) can be used. However, when the operation status (which always indicates that the timer is operating, for example) is read, a bit manipulation instruction cannot be used.

[Bit 13] EDIR (EnD Interrupt Request)

The EDIR flag indicates that measurement terminated in pulse-width measurement mode. When this bit is set to enable an interrupt (bit 12: EDIE = "1"), a measurement termination interrupt request is issued.

Cause of setting	When pulse-width measurement terminates, the bit is set (PWCR contains the measurement result).
Cause of clear	Reading PWCR (measurement result) clears the bit.

Note:

In timer mode, this bit is meaningless.

After reset, the bit is initialized to 0.

The bit is read only. Writing this bit is meaningless.

[Bit 12] EDIE (EnD Interrupt Enable)

This bit is used to control a measurement termination interrupt request in pulse-width count mode as follows:

0	Disables output of a measurement termination interrupt request (when EDIR is set, the interrupt is not generated) (initial value).
1	Enables output of a measurement termination interrupt request (when EDIR is set, the interrupt is generated).

Note:

Always set 0 in timer mode.

After reset, the bit is initialized to 0.

The bit can be read and written.

[Bit 11] OVIR (Overflow Interrupt Request)

This bit is used to specify when the 16-bit up count timer overflows in the range from $FFFF_H$ to 0000_H . The operation affects all modes. When this bit is set to enable the interrupt (bit 10: $OVIE = "1"$), a timer overflow interrupt request is generated.

Cause of setting	When a timer overflow occurs ($FFFF_H$ to 0000_H), the bit is set.
Cause of clear	Writing 0 or extended intelligent I/O service clears the bit.

After reset, the bit is initialized to 0.

The bit can be read and written. However, only writing 0 is valid. Writing 1 is meaningless. A read modify write instruction always reads this bit as 1.

Note:

In H/L pulse-width count mode, do not use this bit for pulse-width time measurement.

[Bit 10] OVIE (Overflow Interrupt Enable)

OVIE is the timer overflow interrupt request control bit. This bit is used to control a timer overflow interrupt request as listed in Table 11.3-3 "OVIE (timer overflow interrupt request control bit)".

After reset, the bit is initialized to 0. The bit can be read and written.

Table 11.3-3 OVIE (timer overflow interrupt request control bit)

OVIE	Function
0	Disables output of an overflow interrupt request (when OVIR is set, the interrupt is not generated) (initial value).
1	Enables output of an overflow interrupt request (when OVIR is set, the interrupt is generated).

Note:

In the H/L pulse-width count mode, set this bit to 0.

[Bit 9] ERR (ERRor)

The ERR flag is used to execute a continuous count in the pulse-width count mode. This flag indicates that the next count has been completed before the previous count result is read from PWCR. If this state occurs, PWCR is overwritten by new count result and the previous result is lost. The count operation continues regardless of the value of this bit.

After reset, the bit is initialized to 0. The bit is read only. Writing to this bit does not change the value.

Table 11.3-4 ERR (ERRor)

Cause of setting	When the count result that has not been read is overwritten by the next result, the bit is set.
Cause of clear	Reading PWCR (measurement result) clears the bit.

[Bit 8] POUT (Pulse OUTput)

Each time the 16-bit up count timer overflows in the range from FFFF_H to 0000_H in timer mode, this bit is reversed.

In the pulse-width count mode, this bit is meaningless.

After reset, the bit is initialized to 0.

The bit can be read and written. However, the bit can be written only if the timer stops (both bit 15: STRT and bit 14: STOP are set to 0). If the bit is written during timer operation (both bit 15: STRT and bit 14: STOP are set to 1), the bit value remains unchanged.

Table 11.3-5 POUT (Pulse OUTput)

Cause of setting	When the POUT value is 0 and the timer overflows in the range from FFFF _H to 0000 _H or the timer stops and 1 is written, the bit is set.
Cause of clear	When the POUT value is 1 and the timer overflows in the range from FFFF _H to 0000 _H or the timer stops and 0 is written, the bit is cleared. The bit is also cleared by reset.

[Bits 7 and 6] CKS1 and CKS0 (Clock Select 1 and 0)

CKS1 and CKS0 bits are used to select the internal count clock. These bits are used to select the internal count clock as listed in Table 11.3-6 "CKS1 and CKS0 (internal count clock selection bits)".

After reset, the bits are initialized to 00_B. The bits can be read and written. However, 11_B cannot be set.

Table 11.3-6 CKS1 and CKS0 (internal count clock selection bits)

CKS1	CKS0	Count clock selection
0	0	Machine clock divide by 4 (0.25 μ s for machine cycle at 16 MHz) (initial value)
0	1	Machine clock divide by 16 (1.0 μ s for machine cycle at 16 MHz)
1	0	Machine clock divide by 32 (2.0 μ s for machine cycle at 16 MHz)
1	1	Setting prohibited (undefined)

Note:

After the timer is started, changing the setting is prohibited. Write these bits before the timer is started or after the timer is stopped.

[Bits 5 and 4] Reserved bits (reserved)

Bits 5 and 4 are reserved. Always write 00_B.

[Bit 3] S/C (Single/Continuous)

The S/C bit is used to select the count mode. The count mode is selected as listed in Table 11.3-7 "S/C (count mode selection bit)". After reset, the bit is initialized to 0. The bit can be read and written.

Table 11.3-7 S/C (count mode selection bit)

S/C	Count mode selection	Timer mode	Pulse-width count mode
0	Single measurement mode (initial value)	No reload (one shot)	Stop after one measurement
1	Continuous measurement mode	Reload (reload timer) Buffer register is valid	Continuous measurement: Buffer register is valid

Note:

After the timer is started, changing the setting is prohibited. Write this bit before the timer is started or after the timer is stopped.

[Bits 2, 1, and 0] MOD2, MOD1, and MOD0 (MOD2, 1, and 0)

Setting these bits enables selection of the operating mode and the pulse edge that fits the pulse-width count as listed in Table 11.3-8 "MOD2, MOD1, and MOD0 (operation mode/count edge selection bits)". After reset, these bits are initialized to 000_B. These bits can be read and written.

Table 11.3-8 MOD2, MOD1, and MOD0 (operation mode/count edge selection bits)

MOD2	MOD1	MOD0	Operation mode/count edge selection
0	0	0	Timer mode and no pulse output (initial value)
0	0	1	Timer mode and pulse output (POT pin valid): Reload mode only
0	1	0	All edge-to-edge pulse-width measurement mode (rising edge or falling edge to falling edge or rising edge)*
0	1	1	Division period measurement mode (when the input divider is used)*
1	0	0	Rising edge-to-rising edge period measurement mode (rising edge to rising edge)*
1	0	1	H pulse-width measurement mode (rising edge to falling edge)*
1	1	0	L pulse-width measurement mode (falling edge to rising edge)*
1	1	1	Falling edge-to-falling edge period measurement mode (falling edge to falling edge)*

Note:

After the timer is started, changing the setting is prohibited. Write these bits before the timer is started or after the timer is stopped.

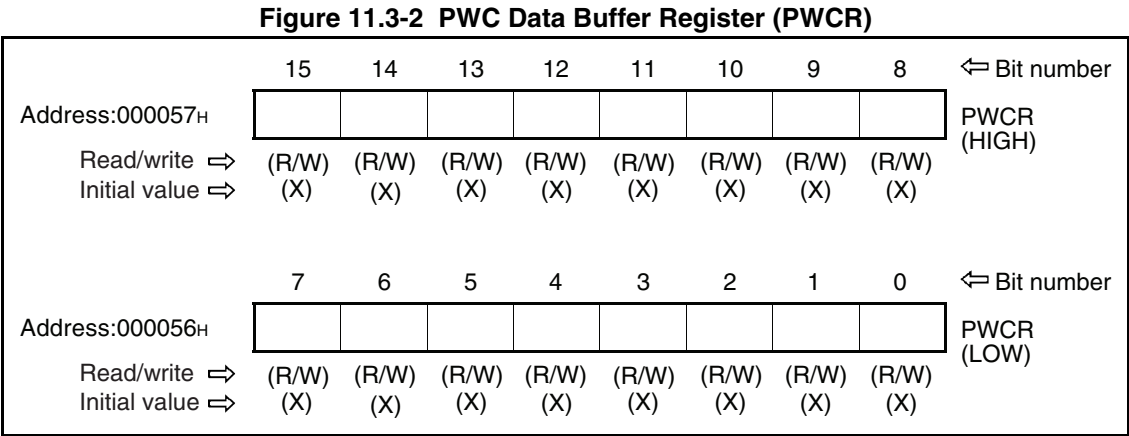
If the continuous measurement mode is set for the setting marked *, the number of edges are totaled and the divider for the internal count clock is not cleared at the end of count. In all other modes, the divider for the internal count clock is cleared at the end of the count.

11.3.2 PWC data buffer register (PWCR)

The PWC data buffer register (PWCR) has functions that depend on the operation mode of the PWC timer.

■ PWC Data Buffer Register (PWCR)

Figure 11.3-2 "PWC Data Buffer Register (PWCR)" shows the register configuration of the PWC data buffer register (PWCR).



○ **Timer mode**

In the reload timer operation mode (PWCSR [bit 3] S/C = 1), this register contains the reload value. The register can be read and written.

In the single timer operation mode (PWCSR [bit 3] S/C = 0), direct access to this register accesses the up count timer. In this mode, this register can be both read and written. However, the register is written only when the timer stops. The register can always be read and the current timer value is read.

○ **Pulse-width measurement mode (read only)**

In the continuous measurement mode (PWCSR [bit 3] S/C = 1), this register functions as the buffer register and contains the previous count result. This register is read only, and the register value remains unchanged when written.

In the single measurement mode (PWCSR [bit 3] S/C = 0), direct access to this register accesses the up count timer. In this mode, the register is also read only and the register value remains unchanged when written. The register can always be read and the current timer value is read. After the count, the register contains the count results.

Note:

To access this register, always use the word transfer instruction.

After reset, this register is initialized to 0000H.

11.3.3 Division rate control register (DIVR)

The division rate control register (DIVR) is used in the division period measurement mode (PWCSR [bits 2, 1, and 0] MOD2, 1, and 0 = 011). This register has no meaning in other modes.

■ Division Rate Control Register (DIVR)

Figure 11.3-3 "Division Rate Control Register (DIVR)" shows the register configuration of the division rate control register (DIVR).

Figure 11.3-3 Division Rate Control Register (DIVR)

	7	6	5	4	3	2	1	0	⇐ Bit number
Address:000058H	-	-	-	-	-	-	DIV1	DIV0	DIVR
Read/write ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(0)	(0)	

[Bits 1 and 0] DIV1 and DIV0

In the division range measurement mode, this register is used to divide the pulse input from the measurement pin and measure the one-period width after division. The division rate is selected as listed in Table 11.3-9 "Division Rate Selection by DIV1 And DIV0 Bits".

After reset, these bits are initialized to 00_B. These bits can be read and written.

Table 11.3-9 Division Rate Selection by DIV1 And DIV0 Bits

DIV1	DIV0	Division rate selection
0	0	2^2 = divide by 4 (initial value)
0	1	2^4 = divide by 16
1	0	2^6 = divide by 64
1	1	2^8 = divide by 256

Note:

After the timer starts, the setting cannot be changed. Write these bits before the timer has started or after the timer has stopped.

11.3.4 PWC noise filter register (RNCR)

The PWC noise filter register (RNCR) uses the PWC noise reduction circuit to reduce noise from the input signal. The high-level and low-level are detected after passing through the noise filter.

■ PWC Noise Filter Register (RNCR)

Figure 11.3-4 "PWC Noise Filter Register (RNCR)" shows the register configuration of the PWC noise filter register (RNCR).

Figure 11.3-4 PWC Noise Filter Register (RNCR)

	7	6	5	4	3	2	1	0	↔ Bit number
Address:000086H	-	-	-	-	-	SW1	SW0	EN	RNCR
Read/write ↔	(-)	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	
Initial value ↔	(-)	(-)	(-)	(-)	(-)	(0)	(0)	(0)	

The noise reduction circuit is the digital low-pass filter that is used to reduce the high-frequency component of the input signal. The PWC noise reduction circuit can be used to reduce the noise pulse width specified by the SW bits of the PWC noise filter register.

This PWC noise filter register is an 8-bit register and all bits are initialized to 0 at reset.

[Bits 2 and 1] SW1 and SW0

SW1 and SW0 are clock mode selection bits that specify the noise pulse width to be reduced. Table 11.3-10 "SW1 and SW0 (clock mode selection bits)" lists the timing when the main clock is 16 MHz.

Table 11.3-10 SW1 and SW0 (clock mode selection bits)

SW1	SW0	Noise pulse width (Minimum)
0	0	$2^8/\text{main clock}^*$ (16.0μs if the main clock* is 16 MHz)
0	1	$2^{12}/\text{main clock}^*$ (256.0μs if the main clock* is 16 MHz)
1	0	$2^{13}/\text{main clock}^*$ (512.0μs if the main clock* is 16 MHz)
1	1	$2^{14}/\text{main clock}^*$ (1.024ms if the main clock* is 16 MHz)

* : OSC oscillation

[Bit 0] EN

Using the EN bit enables this noise filter function.

Table 11.3-11 En Bit Functions

EN	Function
0	Disables the noise filter function (initial value).
1	Enables the noise filter function.

11.4 PWC Timer Operations

The PWC timer is the multifunction timer based on the 16-bit up count timer and contains the count input pin and 8-bit input divider. The block has two main functions: Timer function and pulse-width count function. Both the timer function and the pulse-width count function enable the selection of two types of count clocks.

■ Timer Function

The timer function is the up count timer that enables selection of the operation in single mode or reload mode.

When the timer is started, a timer count is performed at each count clock.

When an overflow occurs in the range from $FFFF_H$ to 0000_H , an interrupt request is issued.

If an overflow occurs, the following occurs:

- Single mode: Count is discontinued (see Figure 11.4-1 "Timer Operation (single mode)").
- Reload mode: The reload register contents are reloaded to the timer and the count is restarted (see Figure 11.4-2 "Timer Operation (reload mode)").

Figure 11.4-1 Timer Operation (single mode)

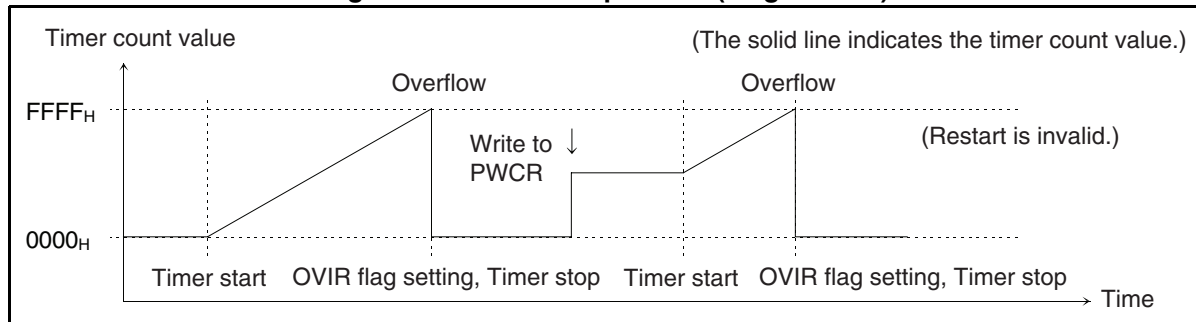
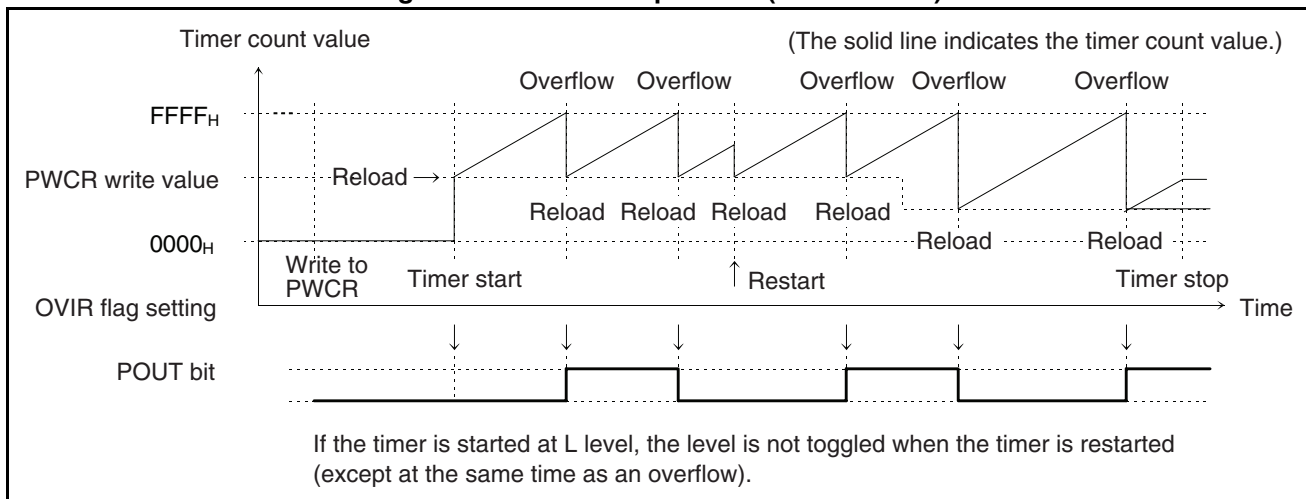


Figure 11.4-2 Timer Operation (reload mode)



■ Pulse-width Measurement Function

The pulse-width measurement function calculates the time between the specified events related to the input pulse.

When this function is activated, a count is started after the specified count start edge is input. If the counter is cleared to 0000_H, a count is started when the start edge is detected, then the stop edge is detected. The count value during this period is held in the register as the pulse width.

When the measurement terminates or an overflow occurs, an interrupt request can be generated. When the measurement is completed, the following occurs:

○ Single measurement mode

The operation is discontinued (see Figure 11.4-3 "Pulse-width Measurement Operation (single measurement mode, H-width measurement mode)").

○ Continuous measurement mode

The timer value is transferred to the buffer register and the timer is in free-run state until the next edge is input (see Figure 11.4-4 "Pulse-width Measurement Operation (continuous measurement mode, H-width measurement mode)").

Figure 11.4-3 Pulse-width Measurement Operation (single measurement mode, H-width measurement mode)

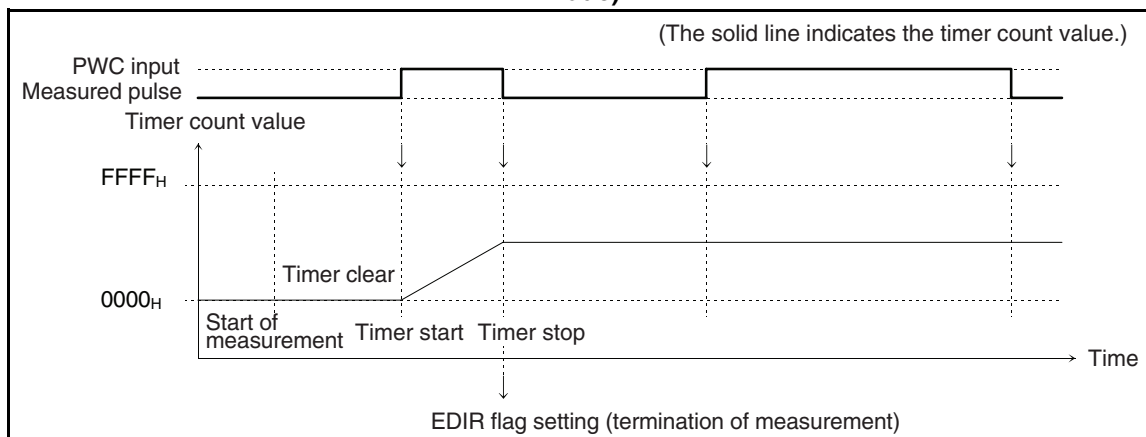
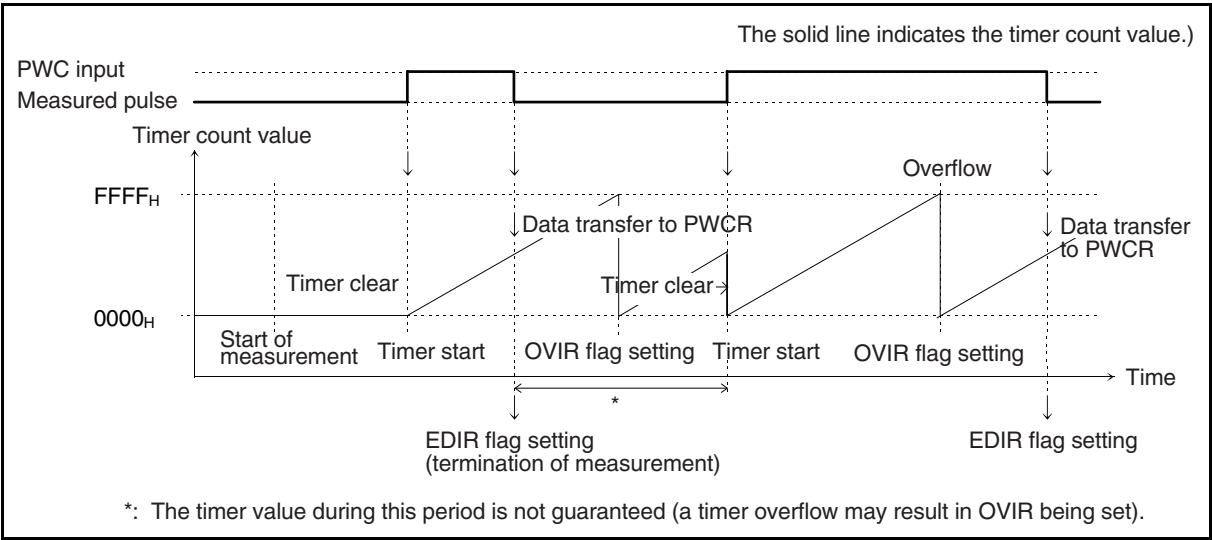


Figure 11.4-4 Pulse-width Measurement Operation (continuous measurement mode, H-width measurement mode)



11.4.1 Count clock selection

The timer count clock can be selected from the following three internal clock sources:

- Machine cycle/4
- Machine cycle/16
- Machine cycle/32

■ Count Clock Selection

Table 11.4-1 "Count Clock Selection" lists the available clock sources.

Table 11.4-1 Count Clock Selection

PWCSR/bit7, bit 6: CKS1, 0	Internal count clock selection
00 _B	Machine cycle/4 (0.25 μ s when machine cycle = 16 MHz) (initial value)
01 _B	Machine cycle/16 (1.0 μ s when machine cycle = 16 MHz)
10 _B	Machine cycle/32 (2.0 μ s when machine cycle = 16 MHz)

After the bits are set, machine cycle/4 is selected.

Note:

Before the timer is started, always select the count clock.

11.4.2 Operation mode selection

Operation modes and count modes are selected according to the setting of PWCSR.

■ Operation Mode Selection

The following registers are used to set the selection of operation modes and count modes:

○ Operation mode setting: PWCSR [bits 2, 1, and 0] MOD2, MOD1, and MOD0 bits

Select the timer mode or pulse-width measurement mode to specify control of the count operation.

○ Count mode setting: PWCSR [bit 3] S/C bit

Select single measurement or continuous measurement or reload operation or one-shot operation.

Table 11.4-2 "Operation Mode Selection" lists the operation modes selected using the mode setting bits.

Table 11.4-2 Operation Mode Selection

Operation mode			S/C	MOD2	MOD1	MOD0
Timer	One-shot timer		0	0	0	0
	Reload timer		1	0	0	0
	Setting prohibited		1	0	0	1
Pulse-width measurement	Rising edge or falling edge to rising edge to falling edge All edge-to-edge measurement	Single measurement: Buffer invalid	0	0	1	0
		Continuous measurement: Buffer valid	1	0	1	0
	Division count Divide by 1 to 256	Single measurement: Buffer invalid	0	0	1	1
		Continuous measurement: Buffer valid	1	0	1	1
	Rising edge to falling edge Rising edge-to-rising edge period measurement	Single measurement: Buffer invalid	0	1	0	0
		Continuous measurement: Buffer valid	1	1	0	0
	Rising edge to falling edge H pulse-width measurement	Single measurement: Buffer invalid	0	1	0	1
		Continuous measurement: Buffer valid	1	1	0	1

Table 11.4-2 Operation Mode Selection (Continued)

Operation mode			S/C	MOD2	MOD1	MOD0
Pulse-width measurement	Rising edge to falling edge L pulse-width measurement	Single measurement: Buffer invalid	0	1	1	0
		Continuous measurement: Buffer valid	1	1	1	0
	Rising edge to falling edge Falling edge-to-falling edge period measurement	Single measurement: Buffer invalid	0	1	1	1
		Continuous measurement: Buffer valid	1	1	1	1

After reset, the one-shot timer is selected as an initial value.

Note:

Before the timer starts, always select the operation mode.

11.4.3 Starting and stopping the timer and pulse-width measurement and clearing the timer

To start, restart, and forcibly stop the timer and pulse-width measurement, use the PWCSR bits 15 and 14 (STRT and STOP).

The 16-bit up count timer is cleared to 0000_H at reset and when the measurement start edge is detected and the count is started in the pulse-width measurement mode.

■ Starting and Stopping Timer and Pulse-width Measurement

Writing 0 to the STRT bit starts or restarts the operation, and writing 0 to the STOP bit stops the operation. However, unless the value written to the two bits is different, none of the bits executes operations. If an instruction (byte or word instruction) other than the bit manipulation instruction is being used, a value is written to the following bit combinations only.

Table 11.4-3 Functions of Start and Stop Bits

Function	STRT	STOP
Starts and restarts the timer or pulse-width measurement	0	1
Stops the timer or pulse-width measurement	1	0

If a bit manipulation instruction (clear bit instruction) is being used, the hardware automatically writes the above combination of values. The user need not know which value is to be written.

○ Operation after start

- Timer mode: The count operation is started immediately.
- Pulse-width measurement mode: Measurement is started after the measurement start edge is input. After the measurement start edge is detected, the 16-bit up count timer is cleared to 0000_H and the count is started.

○ Restarting the timer

While the timer operation continues after the timer is started in the timer mode or pulse-width measurement mode, starting the start (writing 0 to the STRT bit) is called timer restart. The operations to be executed during restart are dependent on the following modes:

- One-shot mode: The operation is not affected.
- Reload timer mode: Reload is executed and the operation is continued. If the timer is restarted when an overflow occurs, the overflow flag (OVIR) is set and the POUT bit is reversed.
- Pulse-width measurement mode: In the measurement start edge wait state, the operation is not affected. During measurement, the count stops and the timer state returns to the "measurement start edge wait" state. When the timer is restarted on termination of measurement, the measurement termination flag (EDIR) is set and the measurement results are transferred to PWCR in continuous measurement mode.

○ Stopping the timer

In one-shot timer mode or single measurement mode, measurement is automatically discontinued when the timer overflows or at the end of a count. The user need not know if the timer has stopped. However, in other modes, the timer must be stopped.

○ Checking operation state

The previously described STRT and STOP bits function as bits that indicate the operation state of the timer during a read operation. Table 11.4-4 "Functions of Operation State Indication Bits" lists the contents of the indicated values.

Table 11.4-4 Functions of Operation State Indication Bits

STRT	STOP	Operation state
0	0	Timer is stopping (except measurement start edge wait state). The bits indicate that the timer has not started or a measurement has terminated.
1	1	Measurement start edge wait state or timer count operation

During a read operation, both the STRT bit and the STOP bit have the same value. However, during a read operation using the read modify write instruction (such as bit manipulation instruction), the values of the bits are always 11_B. Do not use this instruction to read the values of the bits.

■ Clearing the Timer

In the following cases, the 16-bit up count timer is cleared to 0000_H.

- During reset
- When a count has started after the count start edge is detected in the pulse-width measurement mode

11.5 Details of Timer Mode Operation

The timer mode includes the one-shot operation mode and reload operation mode.

■ One-shot Operation Mode

When the timer is started in this mode, a count is incremented at each count clock. The timer automatically stops when an overflow occurs from FFFF_H to 0000_H .

If PWCR is set before the timer has started, the count is started from this set value. After overflow, the set value is deleted and the current count value remains in PWCR.

Although bit 8 (POUT) of PWCSR is inverted when an overflow occurs, its value is not output from the pin in this mode. This is also true when pulse output mode is specified.

■ Reload Operation Mode

When the timer is started in this mode, the reload value in PWCR is set in the timer and the count is incremented at each count clock. If an overflow occurs when the timer counts FFFF_H to 0000_H , the reload value in PWCR is set in the timer again, the POUT bit (bit 8) of PWCSR is reversed, and the count operation is repeated. The timer does not stop until a value is written to the STOP bit of PWCSR to stop the timer or it is reset.

The reload value set in PWCR before the timer is started is stored during a count. When the timer is started or restarted and an overflow occurs, the reload value is always set in the timer. If the value that is set during a count is to be changed, a new reload value becomes valid when the next overflow occurs or the timer is restarted.

■ Timer Value and Reload Value

In one-shot operation mode, direct access to PWCR accesses the up-count timer. When a value is written to PWCR, the value is written directly to the timer. When PWCR is read during a count operation, the current timer value is read. If the value is set in PWCR before the timer is started, the timer starts a count from the specified value.

In reload operation mode, the up-count timer cannot be accessed and PWCR functions as a reload register (stores the reload value). When the timer is started or restarted and an overflow occurs, the value written to PWCR is always set in the timer. When PWCR is read, the stored reload value is read.

The PWCR value and timer value are undefined if the timer is set in one-shot mode after the operation is discontinued in reload mode. Therefore, always set the values before the timer is used.

The PWCR value is undefined if the timer is set in reload mode after the operation is forcibly discontinued in one-shot mode. Therefore, always set the value before the timer is used.

■ Interrupt Request Generation

During operation in timer mode, an overflow enables the generation of an interrupt request. If the increment of a timer count causes an overflow, the overflow flag is set, an overflow interrupt request is enabled, and an interrupt request is generated.

■ Timer Period

If the timer is started in one-shot mode after 0000_H is set in PWCR, a timer overflow occurs and the count is discontinued if the count exceeds 65536. The following formula is used to calculate the time from start to stop of the timer.

$$T_1 = (65536 - n_1) \times t \quad \left\{ \begin{array}{l} T_1 \cdots \text{Time from start to stop } (\mu\text{s}) \\ n_1 \cdots \text{Timer value set in PWCR when the timer is started} \\ t \cdots \text{Count clock period } (\mu\text{s}) \end{array} \right.$$

If the timer is started after 0000_H is set in PWCR, a timer overflow occurs every time the count exceeds 65536. The following formula is used to calculate the reload period and the POT pin output pulse period.

$$\begin{array}{l} T_R = (65536 - n_R) \times t \\ T_{\text{POT}} = T_R \times 2 \end{array} \quad \left\{ \begin{array}{l} T_R \cdots \text{Reload period (overflow period) } (\mu\text{s}) \\ T_{\text{POT}} \cdots \text{POT pin output pulse period } (\mu\text{s}) \\ n_R \cdots \text{Reload value stored in PWCR} \\ t \cdots \text{Count clock period } (\mu\text{s}) \end{array} \right.$$

■ Count clock and Maximum Period

In timer mode, when 0000_H is set in PWCR, the maximum period results.

Table 11.5-1 "Count Clock and Period" lists the count clock period and maximum timer period corresponding to the machine cycle (indicated by ϕ in the table) at 16 MHz.

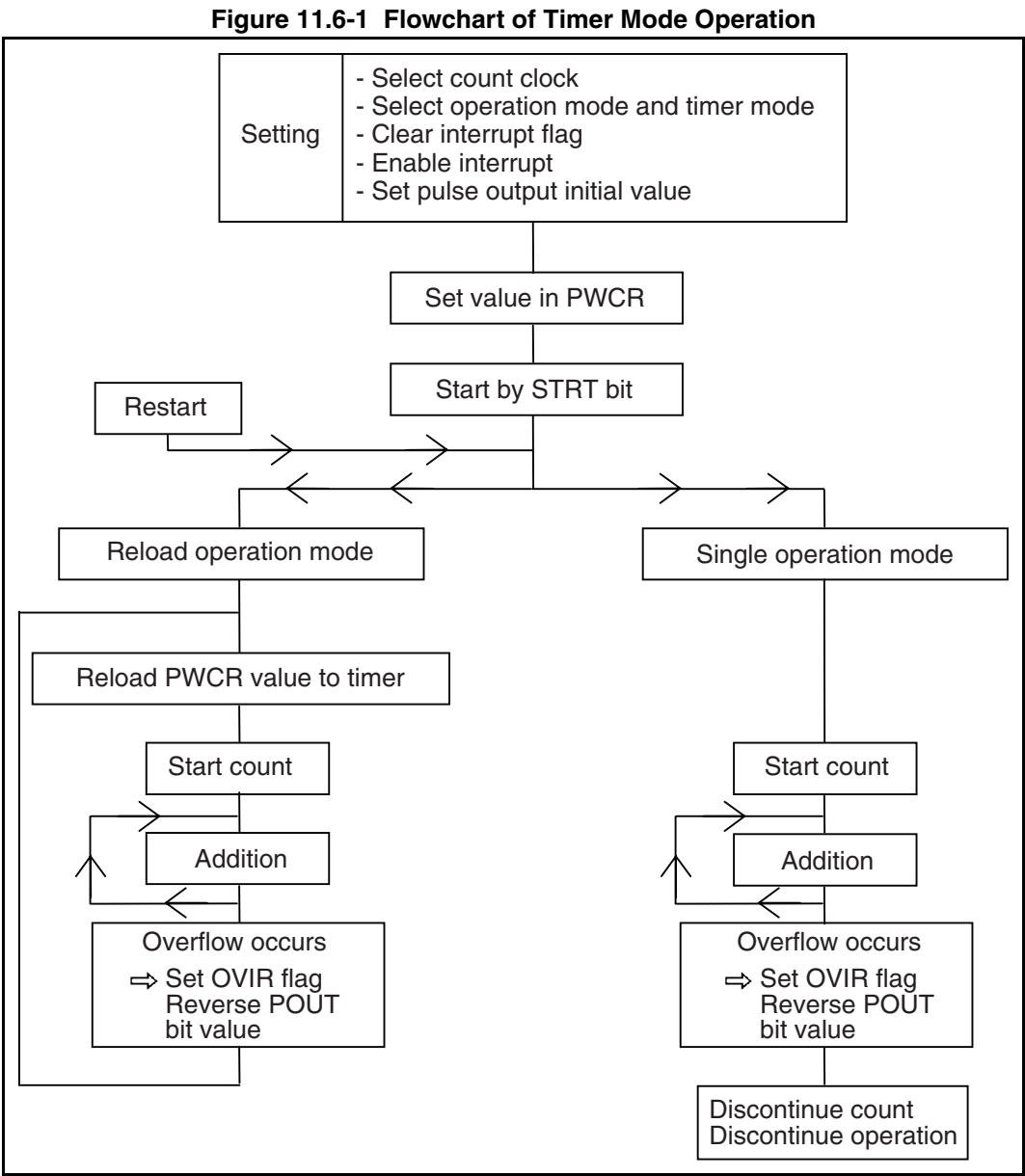
Table 11.5-1 Count Clock and Period

Count clock selection	When CKS1, 0=00 _B ($\phi/4$)	When CKS1, 0=01 _B ($\phi/16$)	When CKS1, 0=10 _B ($\phi/32$)
Count clock period	0.25 μs	1 μs	2 μs
Maximum timer period	16.38 ms	65.5 ms	131.1 ms

11.6 Flowchart of Timer Mode Operation

Figure 11.6-1 "Flowchart of Timer Mode Operation" is the flowchart of timer mode operation.

■ Flowchart of Timer Mode Operation



11.7 Details of Pulse Width Measurement Mode Operation

The signal for pulse-width measurement is input from the PWC pin.

The pulse-width measurement mode includes the single measurement mode in which the count is performed only once and continuous measurement mode in which the pulse width is continuously measured.

■ Single Measurement Mode and Continuous Measurement Mode

The differences between the single measurement mode and continuous measurement mode are as follows:

○ Single Measurement Mode

When the first count end edge is input, the timer discontinues the count, the count end flag (EDIR) of PWCSR is set, and the subsequent measurement is not performed. However, if a timer restart is also specified, the timer state changes to measurement start edge wait state.

○ Continuous Measurement Mode

[H/L pulse-width measurement mode]

When the count end edge is input, the count end flag (EDIR) of PWCSR is set, the timer count result is transferred to PWCR, and the timer may continue incrementing the count in a free-run state. When the next count start edge is input, the timer is cleared to 0000_H and the pulse-width count is started.

Note:

When the count end edge is input and the timer enters a free-run state, the timer may overflow and the OVIR flag may be set. In the H/L pulse-width measurement mode, do not use the OVIR flag to measure the pulse-width time.

[All edge-to-edge pulse-width measurement mode, division period measurement mode, rising edge-to-rising edge measurement mode, and falling edge-to-falling edge measurement mode]

When the count end edge (count start edge) is input, the count end flag (EDIR) of PWCSR is set, the timer count result is transferred to PWCR, the timer is cleared to 0000_H, and the count is restarted.

■ Measurement Result Data

Handling of the measurement result, timer value, and PWCR function varies with the single measurement mode and continuous measurement mode as follows:

○ Single measurement mode

When PWCR is read during timer operation, the current timer value is read.

When PWCR is read after termination of measurement, the measurement results are read.

○ Continuous measurement mode

At termination of measurement, the timer measurement results are transferred to PWCR.

When PWCR is read, the previous measurement results are read. While measurement is in progress, the previous measurement results are stored in PWCR. During measurement, the timer value cannot be read.

In continuous measurement mode, unless the previous measurement results are read before completion of the next measurement, a new measurement result overwrites the existing value. The error flag (ERR) of PWCSR is set. When PWCR is read, the error flag (ERR) is cleared automatically.

■ Minimum Input Pulse Width

The pulse must be input to the pulse-width count input pin (PWC) longer than the following minimum input pulse width.

Pulse width: 2 machine cycles (0.125 μs or more for the machine clock at 16 MHz)

However, the input pulse that is shorter than the above specification may also be recognized as a valid pulse.

■ Calculating Pulse Width/period

The pulse width or pulse period of the measurement object is calculated based on the count result read from PWCR at the end of a count as follows.

$T_w = n \times t / D_{IV} (\mu s)$

T_w

n

t

D_{IV}

Measured pulse width or pulse period (μs)

Measurement result contained in PWC

Count clock period (μs)

Division rate set in the division rate register (DIVR)
(a value of 1 is used in a mode other than the division count mode.)

■ Pulse Width/period Measurement Range

The range of the pulse width/period that can be measured depends on the count clock and division rate of an input divider.

Table 11.7-1 "Pulse Width Measurement Range" lists the measurement range for the machine cycle (indicated by ϕ) at 16 MHz.

Table 11.7-1 Pulse Width Measurement Range

Division rate	DIV1, 0	CKS1, 0=00 _B ($\phi/4$)	CKS1, 0=01 _B ($\phi/16$)	CKS1, 0=10 _B ($\phi/32$)
No division	-	0.125 μ s to 16.38 ms [0.25 μ s]	0.125 μ s to 65.5 ms [1.0 μ s]	0.125 μ s to 131 ms [2.0 μ s]
Divide-by 4	00 _B	0.125 μ s to 4.10 ms [62.5 ns]	0.125 μ s to 16.38 ms [0.25 μ s]	0.125 μ s to 32.75 ms [500 ns]
Divide-by 16	01 _B	0.125 μ s to 1024 μ s [15.6 ns]	0.125 μ s to 4.10 ms [62.5 ns]	0.125 μ s to 8.19 ms [125 ns]
Divide-by 64	10 _B	0.125 μ s to 256 μ s [3.91 ns]	0.125 μ s to 1024 μ s [15.6 ns]	0.125 μ s to 2.048 ms [31.25 ns]
Divide-by 256	11 _B	0.125 μ s to 64 μ s [0.98 ns]	0.125 μ s to 256 μ s [3.91 ns]	0.125 μ s to 512 ms [7.81 ns]

Note:

The number in [] indicates the resolution per bit.

■ Interrupt Request Generation

In the pulse-width measurement mode, the following two interrupt requests can be generated:

○ **Timer overflow interrupt request**

If an overflow occurs during a count, the overflow flag is set. When the overflow interrupt request is enabled, an interrupt request is generated.

○ **Measurement termination interrupt request**

When the measurement termination edge is detected, the count end flag (EDIR) of PWCSR is set. If the measurement termination interrupt is enabled, an interrupt request is generated.

The measurement termination flag (EDIR) is automatically cleared when PWCR is read.

11.7.1 Measurement mode and measurement operation

The measurement mode can be selected from five different modes. This mode is used to determine the component of the input pulse to be measured. The mode can be used to divide the input pulse at the specified division rate and measure the resulting period to accurately measure the width of a high-frequency pulse.

■ Measurement Mode and Measurement Operation

Table 11.7-2 "Measurement Mode Operation" lists measurement mode operations.

Table 11.7-2 Measurement Mode Operation

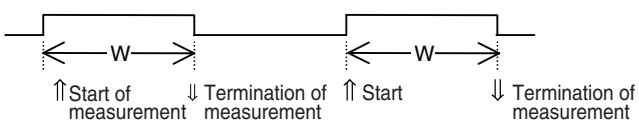
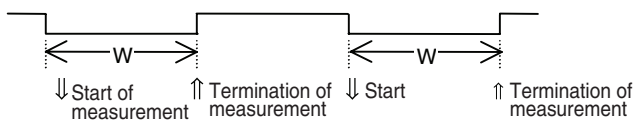
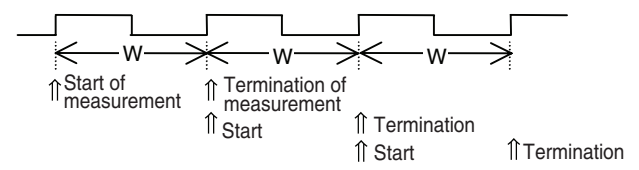
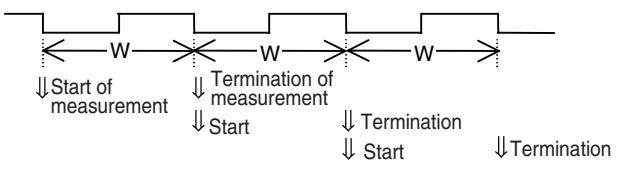
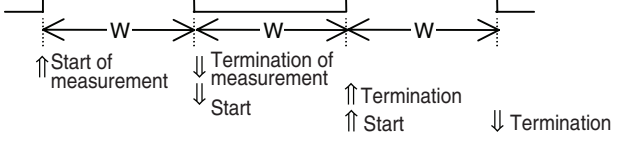
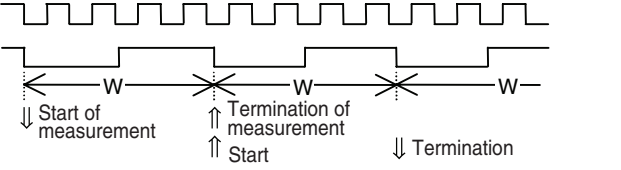
Measurement mode	MOD2	MOD1	MOD0	Measurement operation
H-pulse-width measurement	1	0	1	 <p>The H-period width is measured.</p> <p>Start of measurement: When the rising edge is detected</p> <p>Termination of measurement: When the falling edge is detected</p>
L pulse-width measurement	1	1	0	 <p>The L period width is measured.</p> <p>Start of measurement: When the falling edge is detected</p> <p>Termination of measurement: When the rising edge is detected</p>
Rising edge-to-rising edge period measurement	1	0	0	 <p>The rising edge-to-rising edge time is measured.</p> <p>Start of measurement: When the rising edge is detected</p>

Table 11.7-2 Measurement Mode Operation (Continued)

Measurement mode	MOD2	MOD1	MOD0	Measurement operation
Falling edge-to-falling edge period measurement	1	1	1	 <p>The falling edge-to-falling edge time is measured.</p> <p>Start of measurement: When the falling edge is detected</p> <p>Termination of measurement: When the falling edge is detected</p>
All edge pulse-width measurement	0	1	0	 <p>The width between continuous input edges is measured.</p> <p>Start of measurement: When the edge is detected</p> <p>Termination of measurement: When the edge is detected</p>
Division measurement	0	1	1	 <p>(Divided by 4 in the above example.)</p> <p>The input pulse is divided by the division rate set in the division rate register (DIVR), and the measurement period is obtained as a result.</p> <p>Start of measurement: The falling edge is detected after the operation is started.</p> <p>Termination of measurement: One period of division signal ends.</p>

W: Pulse width being measured

In all modes, the timer does not start count during the period from the start of measurement to input of measurement start edge. After the measurement start edge is input, the timer is cleared to 0000_H, and the count is incremented at each count clock until the measurement termination edge is input.

When the measurement termination edge is input, the following operations are executed:

1. The count end flag (EDIR) of PWCSR is set.
2. The timer stops count operation (except if the timer is restarted at the same time or continuous measurement mode of the H/L pulse-width measurement is used).
3. Continuous measurement mode: The timer value (measurement result) is transferred to PWCR.
4. Single measurement mode: Measurement is terminated (except if the timer is restarted at the same time).

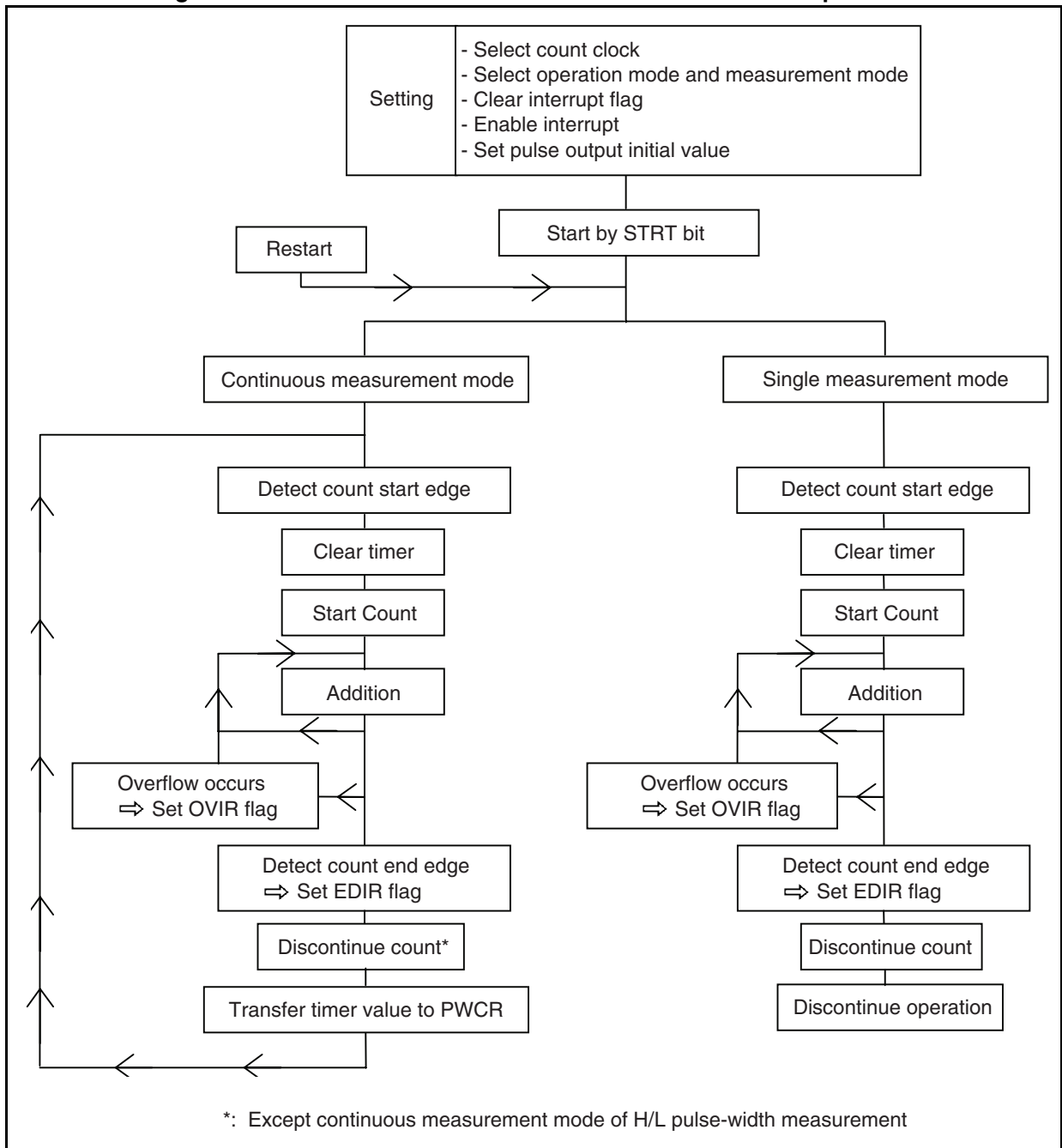
If all edge-to-edge pulse-width measurement, period measurement, falling edge-to-falling edge period measurement, or rising edge-to-edge period measurement is done in continuous measurement mode, the termination edge becomes the next measurement start edge.

11.7.2 Flowchart of pulse-width measurement operation

Figure 11.7-1 "Flowchart of Pulse-width Measurement Mode Operation" is the flowchart of pulse-width measurement mode operation.

■ Flowchart of Pulse-width Measurement Operation

Figure 11.7-1 Flowchart of Pulse-width Measurement Mode Operation



11.8 Notes on Handling the PWC Timer

Note the following contents for handling the PWC timer:

- Register value change
 - Measurement termination flag in timer mode
 - STRT and STOP bits of PWCSR
 - Timer clear
 - Clock selection bits
 - PWCR and timer values when mode is changed
 - Minimum input pulse width
 - Division period measurement mode
 - Restart during operation
 - Pulse-width measurement mode using continuous measurement mode
-

■ Register Value Change

Changing the following PWCSR bit values is prohibited during timer operation. The bit values are changed only before the timer is started or after the operation is discontinued.

[Bits 7 and 6] CKS1 and CKS0: Clock selection bits

[Bit 3] S/C: Measurement mode (single or continuous) selection bit

[Bits 2, 1, and 0] MOD2, MOD1, and MOD0: Operation mode and measurement edge selection bits

Note that the value of pulse output level indication bit (POUT: bit 8) remains unchanged even if the bit is written during timer operation.

Changing the DIVR value is prohibited during timer operation. Change the DIVR value before the timer is started or after the operation has stopped.

■ Measurement Termination Flag in Timer Mode

In timer mode, the value of the measurement termination interrupt request flag (EDIR) of PWCSR is insignificant. Therefore, always set 0 in the count end interrupt request (EDIE) enable bit of PWCSR.

■ STRT and STOP Bits of PWCSR

Note that these two bits are dependent on whether they are read or written (see the details of registers).

Note that a read modify write instruction always reads the bits as 11_B. A bit manipulation instruction cannot be used to read the operation state.

However, a bit manipulation instruction (bit clear instruction) can be used to start or stop the timer by writing the STRT or STOP bit.

■ Timer Clear

In the pulse-width measurement mode, the measurement start edge causes the timer to be cleared, and the previous timer data is insignificant.

■ Clock Selection Bits

Setting 11_B in clock selection bits (bits 7 and 6: CKS1 and CKS0) of PWCSR is prohibited.

■ PWCR and Timer Values when the Mode is Changed

The PWCR and timer values are determined when the timer is set in the one-shot mode after the operation is terminated in reload timer mode. Therefore, always set the values after the timer is used.

The PWCR value is undefined if the timer is set in reload timer mode after the operation is discontinued in the one-shot mode. Therefore, always set the value before the timer is used.

To change the mode from pulse-width measurement mode to timer mode, always set the value in PWCR before the timer has started.

■ Minimum Input Pulse Width

Pulse input to the pulse width measurement input pin is controlled as follows:

○ Minimum pulse width

Divide-by 2 of machine cycle (0.125 μ s or more for the machine cycle at 16 MHz)

○ Maximum input frequency

Divide-by 4 of machine cycle (4 MHz or less for the machine cycle at 16 MHz)

If a pulse width smaller than the above or a frequency larger than the above is input, the timer operation is not guaranteed. A noise violating the above constraint and appearing in the input signal must be reduced.

■ Division Period Measurement Mode

When division period measurement mode is used in pulse-width measurement mode, the input pulse is divided. Note that the pulse width calculated from the count result becomes a mean.

■ Restart During Operation

If the timer is restarted after the count operation starts, the following operation may occur according to the timing:

○ If the timer is restarted when an overflow occurs in reload timer mode

The timer is restarted but the overflow flag (OVIR) is set and the POUT bit is reversed (that is, the same operation as the normal overflow is executed).

○ If the timer is restarted when the measurement termination edge is detected in one-shot pulse-width measurement mode

The timer is restarted and enters measurement start edge wait state but the measurement termination flag (EDIR) is also set.

- **If the timer is restarted when the measurement termination edge is detected in continuous pulse-width measurement mode**

The timer is restarted and enters the measurement start edge wait state, the count termination flag (EDIR) is set, and the measurement results are transferred to PWCR.

To restart the timer during operation, note the above flag operations to generate interrupts and exercise other controls.

■ **Pulse-width Measurement Mode Using Continuous Measurement Mode**

During continuous measurement in this mode, the division circuit for an internal count clock is not cleared, and the number of edges smaller than the count clock is added to the count result.

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 "16-Bit I/O Timer Block Diagram"

12.3 "16-Bit I/O Timer Registers"

12.4 "16-Bit Free-Run Timer Operations"

12.5 "16-Bit Output Compare Operations"

12.6 "16-Bit Input Capture Operations"

12.1 Overview of the 16-Bit I/O Timer

The 16-bit I/O timer consists of one 16-bit free-run timer, two output compares, and four input captures.

Using this function enables two independent waveforms to be output based on the 16-bit free-run timer and also enables an input pulse width and external clock cycle to be measured.

■ 16-bit Free-run Timer (x 1)

The 16-bit free-run timer consists of the 16-bit up counter, control register, and prescaler. An output value of this timer counter is used as the basic time (base timer) of the input capture and output compare.

○ Counter operation clock (selectable from four types)

Four types of internal clocks: $\phi/4$, $\phi/16$, $\phi/64$, $\phi/256$

ϕ : Machine clock

○ Interrupt

An interrupt can be generated by an overflow of a counter value of the 16-bit free-run timer or a compare match with compare register 0. (The compare match requires mode setting.)

○ Counter value

An interrupt can be generated if a counter value of the 16-bit free-run timer overflows or a match with compare register 0 occurs (a compare match can be used according to mode setting).

○ Initialization

A counter value can be initialized to '0000_H' at reset, software clear, or match with compare register 0.

■ Output Compare (x 2)

An output compare module consists of two 16-bit compare registers, compare output latch, and control register. When a 16-bit free-run timer value matches a compare register value, the output level is reversed and an interrupt can be generated.

○ Two compare registers are operated independently.

- Output pin and interrupt flag corresponding to each compare register

○ An output pin can be controlled by pairing two compare registers.

- The polarity of the output pin can be reversed using two compare registers.

○ An initial value of the output pin can be set.

○ An interrupt can be generated by a compare match.

■ Input Capture (x 4)

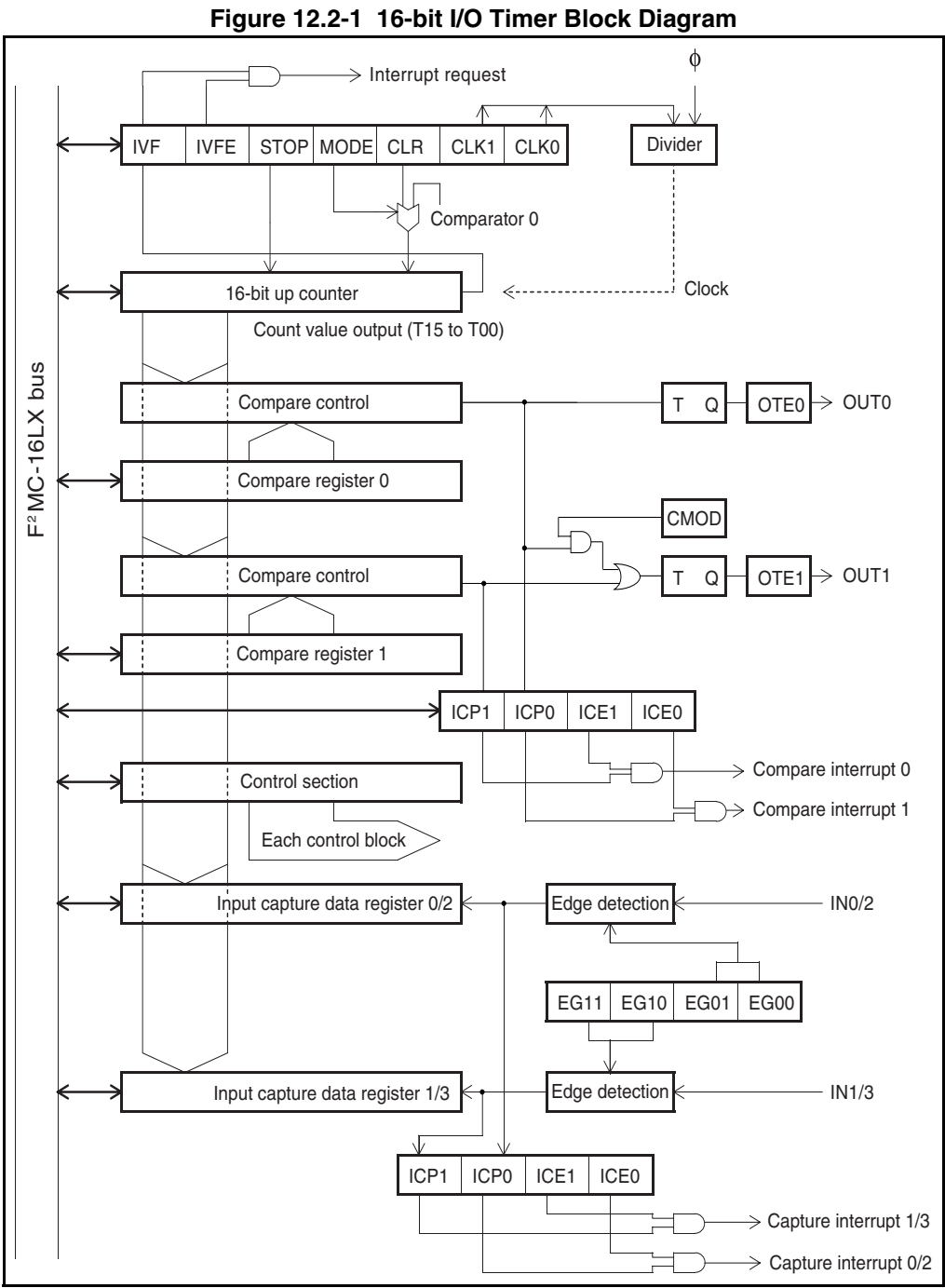
An input capture module consists of the capture register and control register corresponding to four independent external input pins. The value of the 16-bit free-run timer can be stored in the capture register. An interrupt is also generated upon detection of an edge of the signal input from an external pin.

- **An edge of an external input signal can be selected.**
 - Selectable from a rising edge, falling edge, or both edges.
- **Four input captures can operate independently.**
- **An interrupt can be generated by a valid edge of an external input signal.**
 - The extended intelligent I/O service can be activated by an interrupt of the input capture.

12.2 16-Bit I/O Timer Block Diagram

Figure 12.2-1 "16-bit I/O Timer Block Diagram" shows the 16-bit I/O timer block diagram.

■ 16-bit I/O Timer Block Diagram



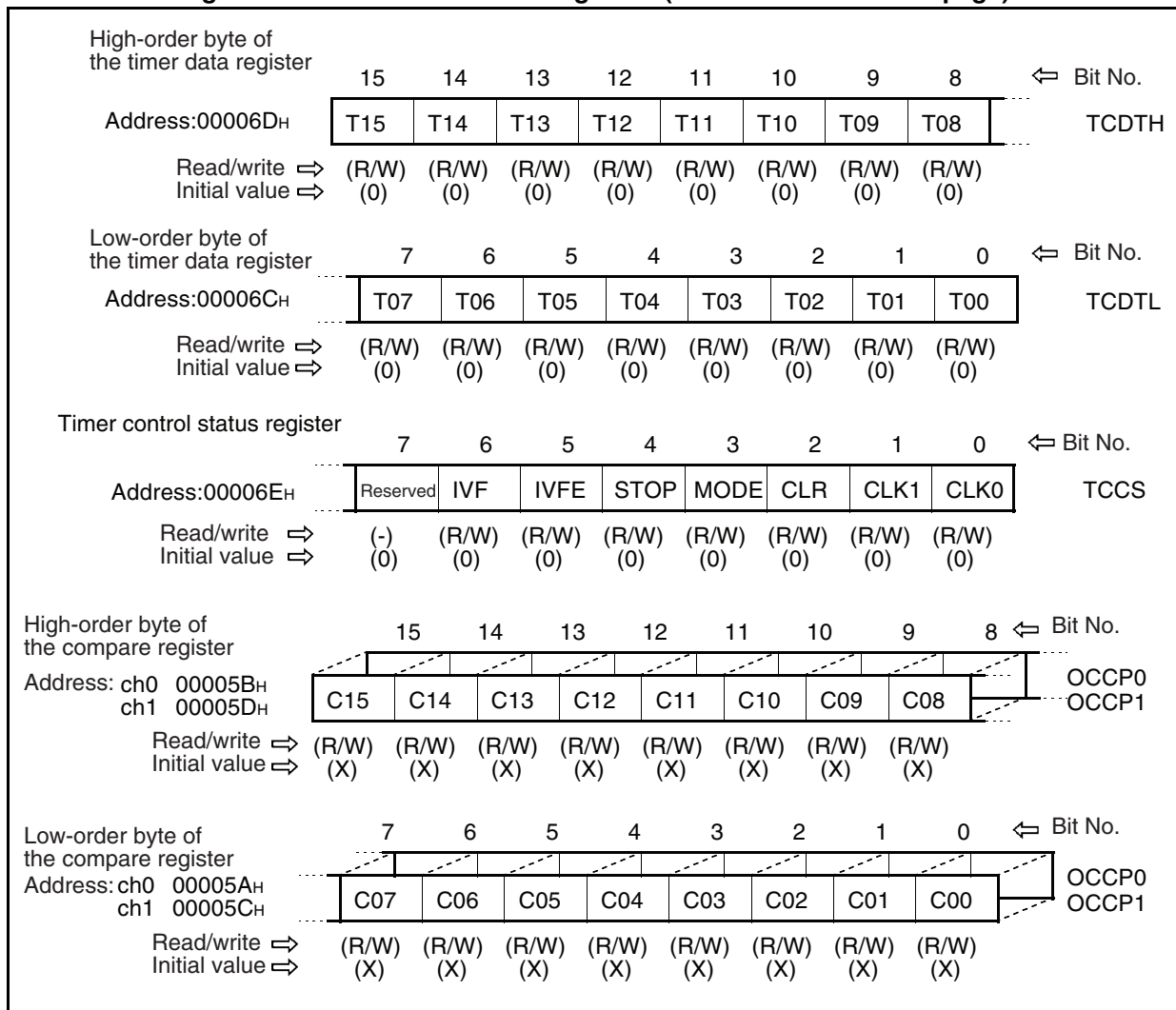
12.3 16-Bit I/O Timer Registers

The following six 16-bit I/O timer registers are supported:

- Timer data register (TCDTH and TCDTL)
- Timer control status register (TCCS)
- Compare register (OCCP0 and OCCP1)
- Compare control status register (OCS0 and OCS1)
- Input capture register (IPCP0 to IPCP3)
- Control status register (ICS01 and ICS23)

■ 16-bit I/O Timer Registers

Figure 12.3-1 16-bit I/O Timer Registers (continued on the next page)



Compare control status register 1		15	14	13	12	11	10	9	8	⇐ Bit No.
Address: ch1 00005F _H		—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1
Read/write ⇨	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	(0)	
Compare control status register 0		7	6	5	4	3	2	1	0	⇐ Bit No.
Address: ch0 00005E _H		IOP1	IOP0	IOE1	IOE0	—	—	CST1	CST0	OCS0
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)	
High-order byte of the input capture register		15	14	13	12	11	10	9	8	⇐ Bit No.
Address: ch0 000061 _H ch1 000063 _H ch2 000065 _H ch3 000067 _H		CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	High-order IPCP0 High-order IPCP1 High-order IPCP2 High-order IPCP3
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
Low-order byte of the input capture register		7	6	5	4	3	2	1	0	⇐ Bit No.
Address: ch0 000060 _H ch1 000062 _H ch2 000064 _H ch3 000066 _H		CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	Low-order IPCP0 Low-order IPCP1 Low-order IPCP2 Low-order IPCP3
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
High-order byte of the input capture control status register 01		7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 000068 _H		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)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Low-order byte of the input capture control status register 23		7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 00006A _H		ICP3	ICP2	ICE3	ICE2	EG31	EG30	EG21	EG20	ICS23
Read/write ⇨	(R/W)	(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)	(0)	

12.3.1 16-bit Free-run Timer

The following two 16-bit free-run timer registers are supported:

- Timer Data register (TCDTH and TCDTL)
- Timer Control status register (TCCS)

■ Timer Data Register (TCDTH and TCDTL)

The data register can read a count value of the 16-bit free-run timer. The counter value is cleared to 0000_H at reset. A timer value can be set by a write to this register and this write operation must be performed during stop (STOP = 1) status.

The 16-bit free-run timer is initialized by the following sources:

- By a reset
- By a clear bit (CLR) of the control status register
- By a match between the compare register 0 of the output compare and a timer counter value. (The mode setting is required.)

Figure 12.3-2 Timer Data register

High-order byte of the timer data register								
	15	14	13	12	11	10	9	8 ⇐ Bit No.
Address:00006D _H	T15	T14	T13	T12	T11	T10	T09	T08 TCDT
Read/write ⇐	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value ⇐	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
Low-order byte of the timer data register								
	7	6	5	4	3	2	1	0 ⇐ Bit No.
Address:00006C _H	T07	T06	T05	T04	T03	T02	T01	T00 TCDT
Read/write ⇐	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value ⇐	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

Note:

This register requires word accesses.

■ Timer Control Status Register (TCCS)

Figure 12.3-3 Control Status Register

Timer control status register									⇐ Bit No.
Address:00006EH	7	6	5	4	3	2	1	0	TCCS
	Reserved	IVF	IVFE	STOP	MODE	CLR	CLK1	CLK0	
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)	(0)	

[Bit 7] Reserved bit

Bit7 is reserved. Set this bit to 0.

[Bit 6] IVF

IVF is an interrupt request flag of the 16-bit free-run timer.

If the 16-bit free-run timer causes an overflow or the counter is cleared with a match with compare register 0 through mode setting, this bit is set to 1. An interrupt is generated if the IVFE bit (bit 5) is set.

This bit is cleared by writing 0. Writing 1 is meaningless. Value 1 can be read by a read modify instruction.

Table 12.3-1 Function of IVF (Interrupt Request Flag)

IVF	Function
0	No interrupt request (initial value)
1	Interrupt request

[Bit 5] IVFE

IVFE is an interrupt enable bit of the 16-bit free-run timer.

When this bit is 1, if the IVF bit (bit 6) is set to 1, an interrupt occurs.

Table 12.3-2 Function of IVFE (Interrupt Enable Bit)

IVFE	Function
0	Interrupt disabled (initial value)
1	Interrupt enabled

[Bit 4] STOP

The STOP bit is used to stop the 16-bit free-run timer count.

When 1 is written, the timer count stops. When 0 is written, the timer count starts.

Table 12.3-3 Function of STOP (Count Stop Bit)

STOP	Function
0	Count enabled (operation) (initial value)
1	Count disabled (stop)

Note:

If the 16-bit free-run timer count stops, the output compare operation also stops.

[Bit 3] MODE

The MODE bit is used to set the initialization condition of the 16-bit free-run timer.

If this bit is 0, a counter value can be initialized by the reset or CLR bit (bit 2). If it is 1, the counter value can be initialized by the reset, CLR bit (bit 2), or a match with compare register 0 of the output compare.

Table 12.3-4 Function of MODE (Initialization Condition Setting Bit)

MODE	Function
0	Initialization by the reset or clear bit (initial value)
1	Initialization by the reset, clear bit, or match with compare register 0

Note:

The counter value is initialized at the change point of the counter value.

[Bit 2] CLR

The CLR bit is used to initialize an active 16-bit free-run timer value to 0000_H. When 1 is written, the counter value is initialized to 0000_H. Writing 0 is meaningless. Value 0 is always read. The counter value is initialized at the count value change point.

Table 12.3-5 Function of CLR (Initialization Bit)

CLR	Function
0	Meaningless (initial value)
1	Initializes a counter value to 0000 _H .

Note:

To initialize a counter value while the timer is stopped, write 0000_H in the data register.

[Bits 1 and 0] CLK1 and CLK0

CLK1 and CLK0 bits are used to select a count clock of the 16-bit free-run timer. A clock is updated immediately after the values are written to these bits. Before the values are written to these bits, always stop an output compare operation and input capture operation.

Table 12.3-6 CLK1 and CLK0 (Count Clock Selection Bits)

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	$\phi/256$	$16\ \mu\text{s}$	$32\ \mu\text{s}$	$64\ \mu\text{s}$	$256\ \mu\text{s}$

ϕ = machine clock

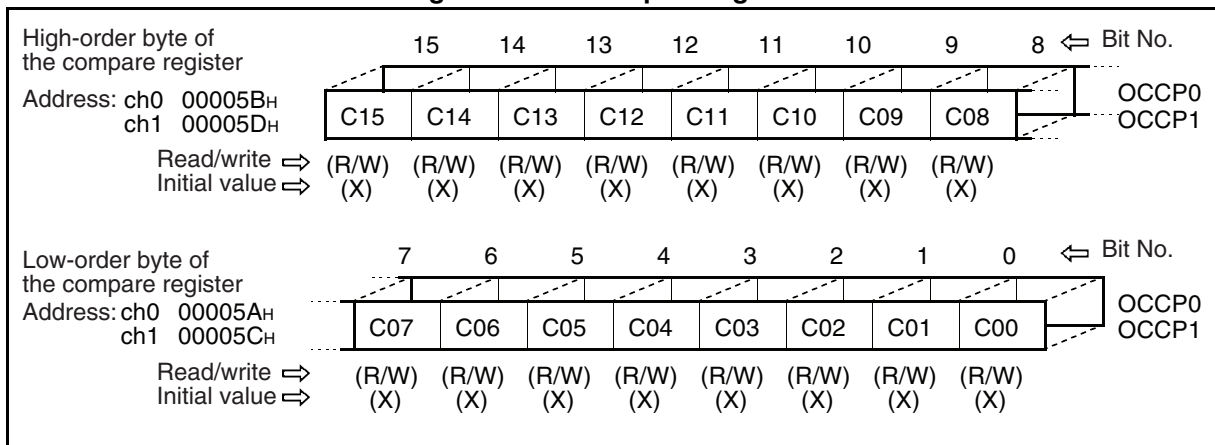
12.3.2 Output Compare

The output compare has the following two registers:

- Compare register (OCCP0 and OCCP1)
- Control status register (OCS0 and OCS1)

■ Compare Register (OCCP0 and OCCP1)

Figure 12.3-4 Compare registers



Note:

This register requires word accesses.

The compare register is a 16-bit register used to make a comparison with the 16-bit free-run timer. An initial value of a register value is undefined. Therefore, set the initial value, then allow the activation. When this register value matches a 16-bit free-run timer value, a compare signal is generated to set the output compare interrupt flag. If output is enabled, the output level associated with the compare register is reversed.

■ Control Status Register (OCS0 to OCS1)

Figure 12.3-5 Control Status Register

Compare control status register 1	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: ch1 00005FH	—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1
Read/write ⇐	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇐	(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	
Compare control status register 0	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: ch0 00005EH	IOP1	IOP0	IOE1	IOE0	—	—	CST1	CST0	OCS0
Read/write ⇐	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇐	(0)	(0)	(0)	(0)	(-)	(-)	(0)	(0)	

[Bits 15 to 13] Unused bits**[Bit 12] CMOD**

CMOD switches the pin output level reverse operation mode for a compare match when pin output is allowed (OTE1 = 1 or OTE0 = 1).

○ **For CMOD = 0 (initial value)**

When CMOD is 0 (initial value), the output level of the pin corresponding to the compare register is reversed.

- OUT0: Reverses the level by a match with compare register 0.
- OUT1: Reverses the level by a match with compare register 1.

○ **For CMOD = 1**

When CMOD is 1, the output level of the pin (OUT0) corresponding to the compare register 0 is reversed in the same way as when CMOD is 0. However, the output level of the pin (OUT1) corresponding to the compare register 1 is reversed by either a match with compare register 0 or a match with compare register 1. If values of compare registers 0 and 1 are equal, the operation is the same as when one compare register is used.

- OUT0: Reverses the level by a match with compare register 0.
- OUT1: Reverses the level by either a match with compare register 0 or a match with compare register 1.

[Bits 11 and 10] OTE1 and OTE0

OTE1 and OTE0 bits enable the pin output of the output compare. The initial value of these bits is 0.

Table 12.3-7 Function of OTE1 and OTE0 (Pin Output Enable Bits)

OTE1, 0	Function
0	Operates as a general-purpose port. [Initial value]
1	Enables the output compare pin output.

Note:

OTE1 corresponds to output compare 1 and OTE0 to output compare 0.

[Bits 9 and 8] OTD1 and OTD0

OTD1 and OTD0 bits are used to change the pin output level when the pin output of the output compare is enabled. The initial value of the compare pin output is 0. Before writing, stop the compare operation. At reading, an output compare pin output value can be read.

Table 12.3-8 Function of OTD1 and OTD0 (Pin Output Level Change Bits)

OTD1, OTD0	Function
0	Sets the compare pin output to 0. [Initial value]
1	Sets the compare pin output to 1.

Note:

OTD1 corresponds to output compare 1 and OTD0 corresponds to output compare 0.

[Bits 7 and 6] ICP1 and ICP0

ICP1 and ICP0 are output compare interrupt flags. These bits are set to 1 when the compare register matches a 16-bit free-run timer value. When interrupt request bits (ICE1 and ICE0) are enabled, if ICP1 and ICP0 bits are set, an output compare interrupt occurs.

These bits are cleared by writing 0. Writing 1 is meaningless. Value 1 can be read by read modify instructions.

Table 12.3-9 Function of ICP1 and ICP0 (Output Compare Interrupt Bits)

ICP1, ICP0	Function
0	No compare match [initial value]
1	Compare match

Note:

ICP1 corresponds to output compare 1 and ICP0 to output compare 0.

[Bits 5 and 4] ICE1 and ICE0

ICE1 and ICE0 are output compare interrupt enable bits. When these bits are 1, if the interrupt flags (ICP0 and ICP1) are set, an output compare interrupt occurs.

Table 12.3-10 Function of ICE1 and ICE0 (Output Compare Interrupt Enable Bits)

ICE1, ICE0	Function
0	Disables an output compare interrupt. [Initial value]
1	Enables an output compare interrupt.

Note:

ICE1 corresponds to output compare 1 and ICE0 to output compare 0.

[Bits 3 and 2] Unused bits

[Bits 1 and 0] CST1 and CST0

CST1 and CST0 are used to enable the match operation with the 16-bit free-run timer.

Table 12.3-11 CST1 and CST0 (for Enabling the Match Operation with the 16-bit Free-run Timer)

CST1, CST0	Setting
0	Disables compare operation. [Initial value]
1	Enables compare operation.
<ul style="list-style-type: none">Before enabling the compare operation, set a compare register value. <p>Note: CST1 corresponds to output compare 1 and CST0 to output compare 0.</p>	

Note:

The output compare is synchronized with clocks of the 16-bit free-run timer. Therefore, the compare operation stops when the 16-bit free-run timer stops.

12.3.3 Input Capture

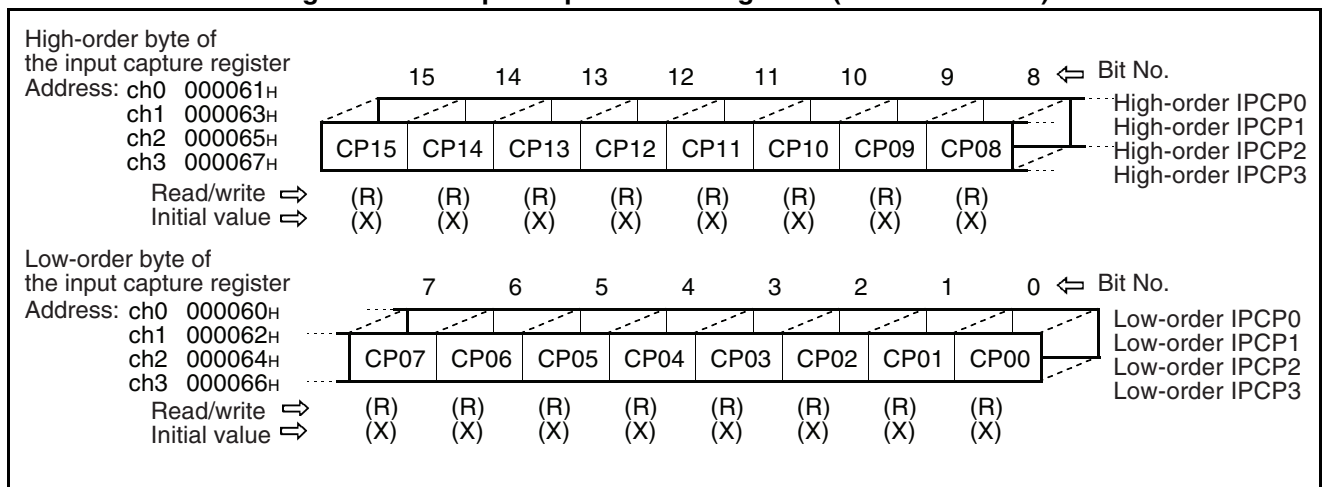
The input capture has the following two registers:

- Input capture data register (IPCP0 to IPCP3)
- Control status register (ICS23 and ICS01)

■ Input Capture Data Register (IPCP0 to IPCP3)

Input capture data registers (IPCP0 to IPCP3) are used to retain 16-bit free-run timer values when a valid edge of the corresponding external pin input waveform is detected.

Figure 12.3-6 Input Capture Data Registers (IPCP0 to IPCP3)

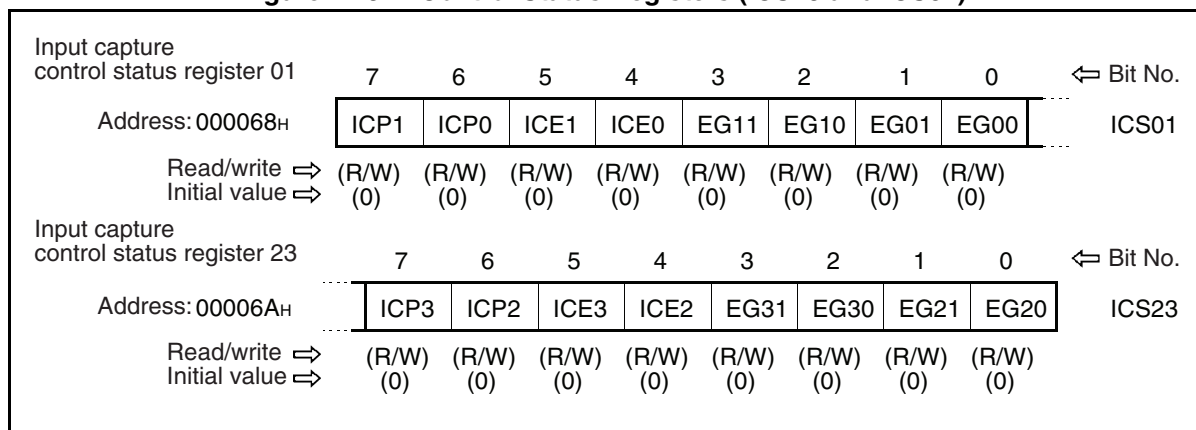


Note:

Input capture data registers (IPCP0 to IPCP3) require word access. Writing is not possible.

■ Control Status Registers (ICS23 and ICS01)

Figure 12.3-7 Control Status Registers (ICS23 and ICS01)



Note:

Input capture control status register (ICS23 and ICS01) register requires byte access.

[Bits 7, and 6] ICPx (x: channel number)

ICPx is an input capture interrupt flag.

When the valid edge of an external input pin is detected, this bit is set to 1.

When the corresponding interrupt enable bit ICEx is set, an interrupt can be generated by detecting the valid edge.

This bit is cleared by writing 0. Writing 1 is meaningless. Value 1 can be read by read modify write instructions.

Table 12.3-12 Function of ICPx (Input Capture Interrupt Flag)

ICPx	Function
0	No valid edge detection (initial value)
1	Valid edge detection

[Bits 5, and 4] ICEx (x: channel number)

ICEx is an input capture interrupt enable bit. When this bit is 1, if the corresponding interrupt flag ICPx is set, an input capture interrupt occurs.





Table 12.3-13 Function of ICEx (Input Capture Interrupt Enable Bit)

ICEx	Function
0	Interrupt disabled (initial value)
1	Interrupt enabled

[Bits 3, 2, 1, and 0] EGx1 and EGx0 (x: channel number)

EGx1 and EGx0 bits specify the valid edge polarity of the external input. These bits can also allow an input capture operation.

Table 12.3-14 Function of EGx1 and EGx0 (for Specifying a Valid Edge Polarity of the External Output)

EGx1	EGx0	Edge detection polarity
0	0	No edge detection (stop) (initial value)
0	1	Rising edge detection 
1	0	Falling edge detection 
1	1	Both-edge detection  

12.4 16-Bit Free-Run Timer Operations

The 16-bit free-run timer starts counting from counter value 0000_H after the reset is released. This counter value is used as the reference time of the 16-bit output compare and the 16-bit input capture.

■ 16-bit Free-run Timer Operations

A counter value is cleared under the following conditions:

- An overflow occurs.
- A match occurs with the output compare register 0 value. (Mode setting is required.)
- Value 1 is written in the CLR bit of the TCCS register during operation.
- 0000_H is written in the TCDC register during stop.
- Reset

An interrupt can occur when an overflow occurs or when the counter is cleared by a match with the compare register 0 value. (The compare match interrupt requires mode setting.)

Figure 12.4-1 Clearing the Counter by an Overflow

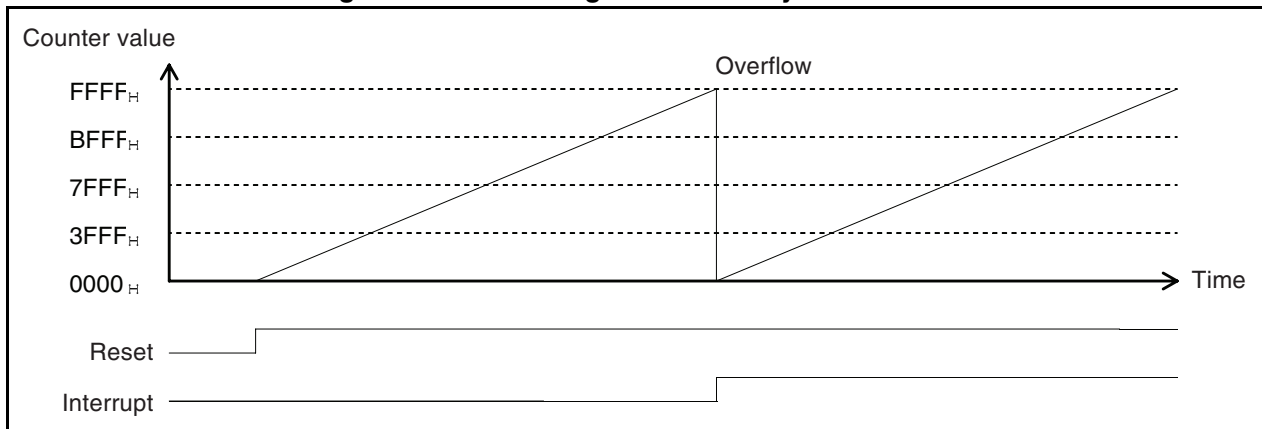
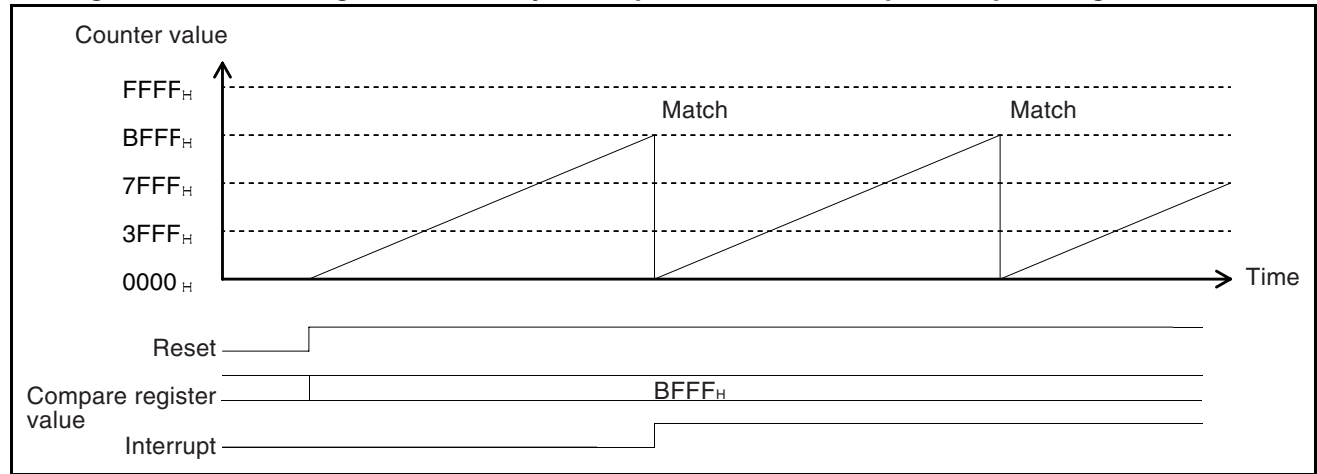


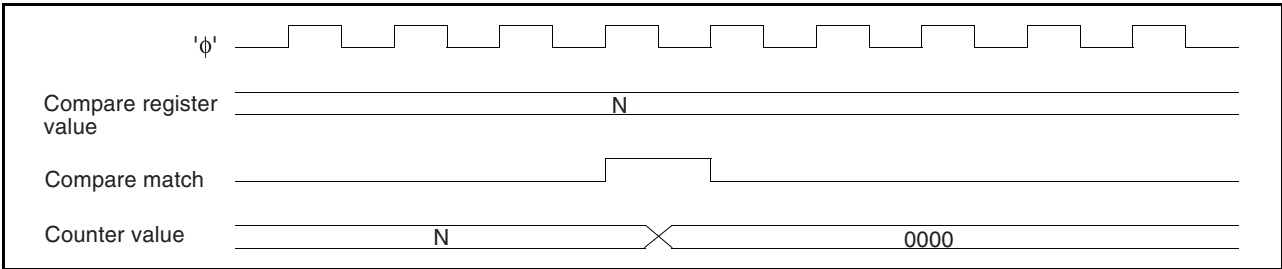
Figure 12.4-2 Clearing the Counter by a Compare Match with output Compare Register 0 Value



■ 16-bit Free-run timer Count Timing

Counter clear can be executed by reset, software clear, or match with compare register 0. The counter is cleared by reset or software clear. The counter is also cleared by match with compare register 0.

Figure 12.4-3 Clear Timing of the Free-run Timer (Match With Compare Register 0)

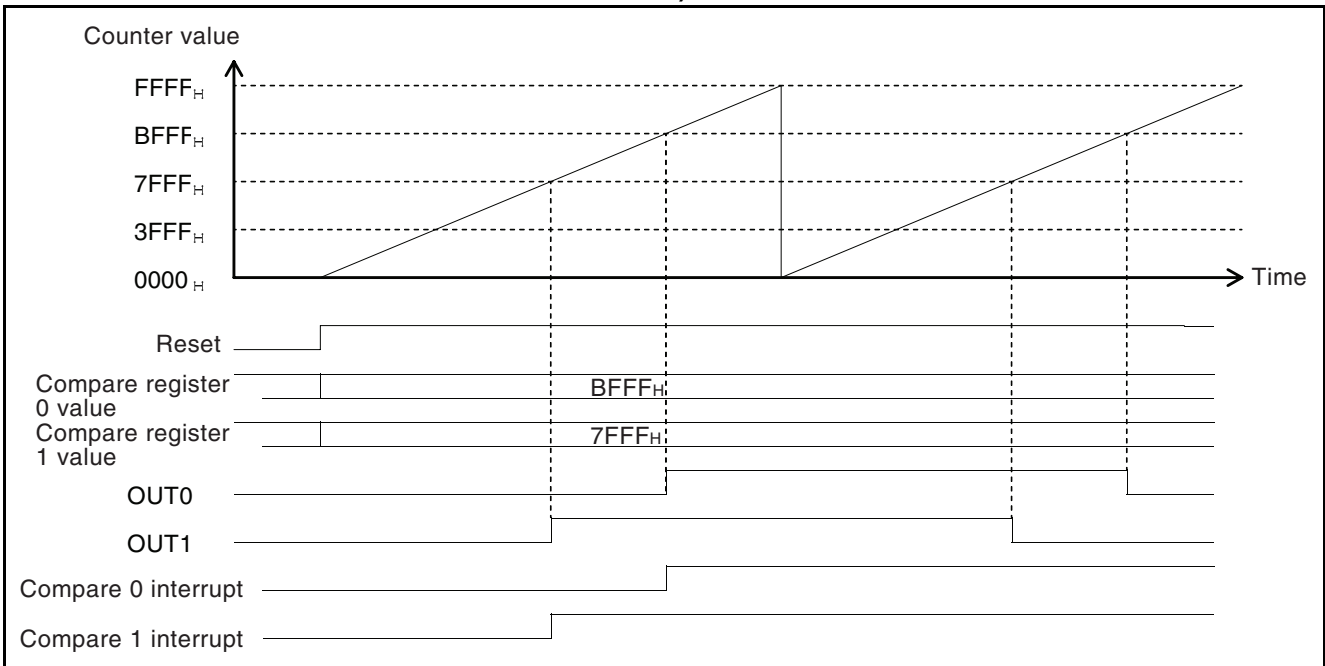


12.5 16-Bit Output Compare Operations

The 16-bit output compare compares the specified compare register value with a 16-bit free-run timer value. When a match occurs, it can set the interrupt request flag and reverse the output level.

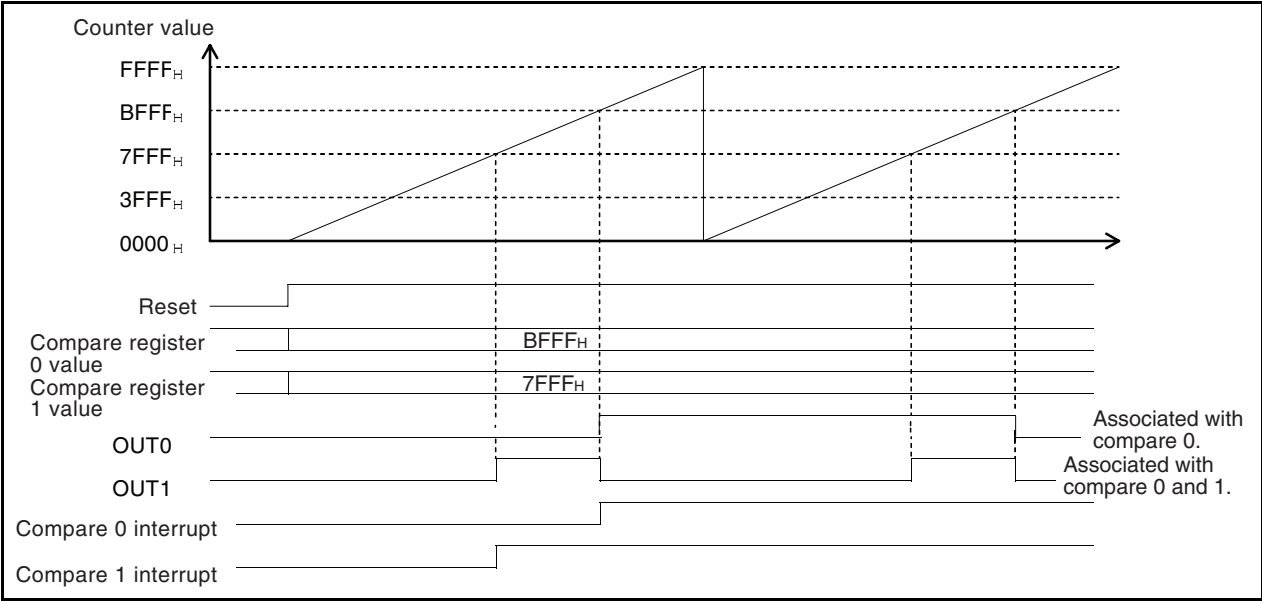
■ 16-bit Output Compare Operations

Figure 12.5-1 Example of an Output Waveform when Compare Registers 0 and 1 are Used (When CMOD = 0)



As shown in Figure 12.5-2 "Example of an Output Waveform when Compare Registers 0 and 1 are Used (Initial value of output is 0, CMOD=1)" the output level can be changed by using both compare registers 0 and 1.

Figure 12.5-2 Example of an Output Waveform when Compare Registers 0 and 1 are Used (Initial value of output is 0, CMOD = 1)



■ 16-bit Output Compare Timing

When a free-run timer value matches the specified compare register value, the output compare can reverse the output value by generating a compare match signal and can then generate an interrupt.

When a compare match occurs, the output reverse timing is executed synchronously with the count timing of the counter.

Figure 12.5-3 Compare Operation at Compare Register Rewriting

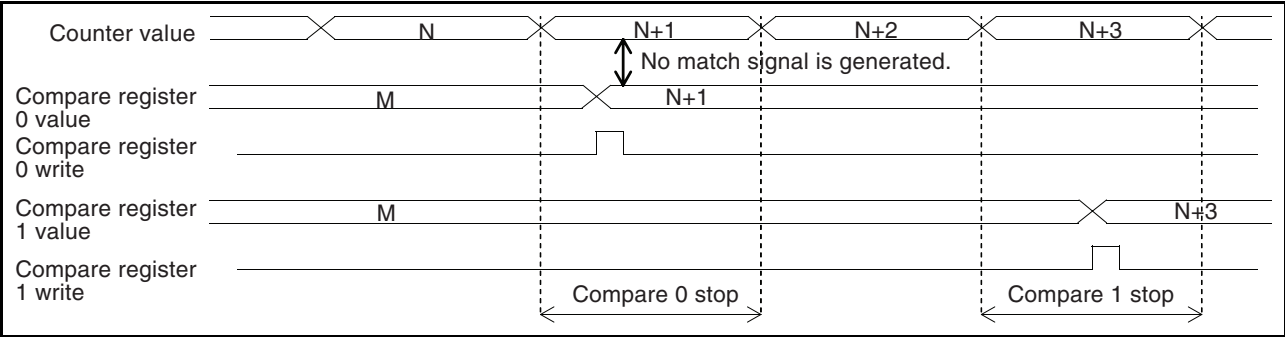


Figure 12.5-4 Interrupt Timing

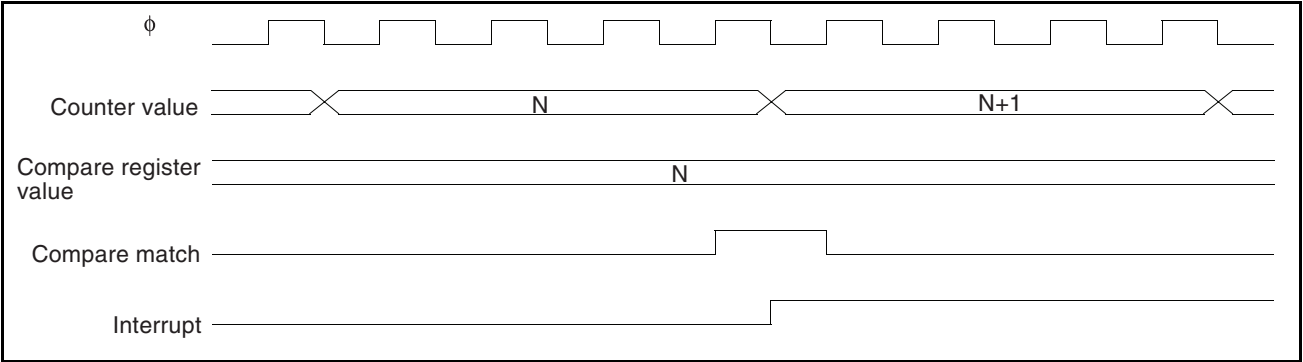
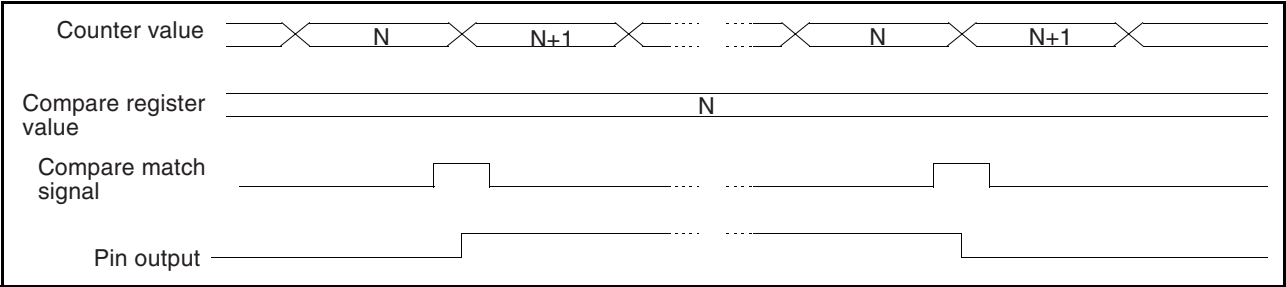


Figure 12.5-5 Output Pin Change Timing

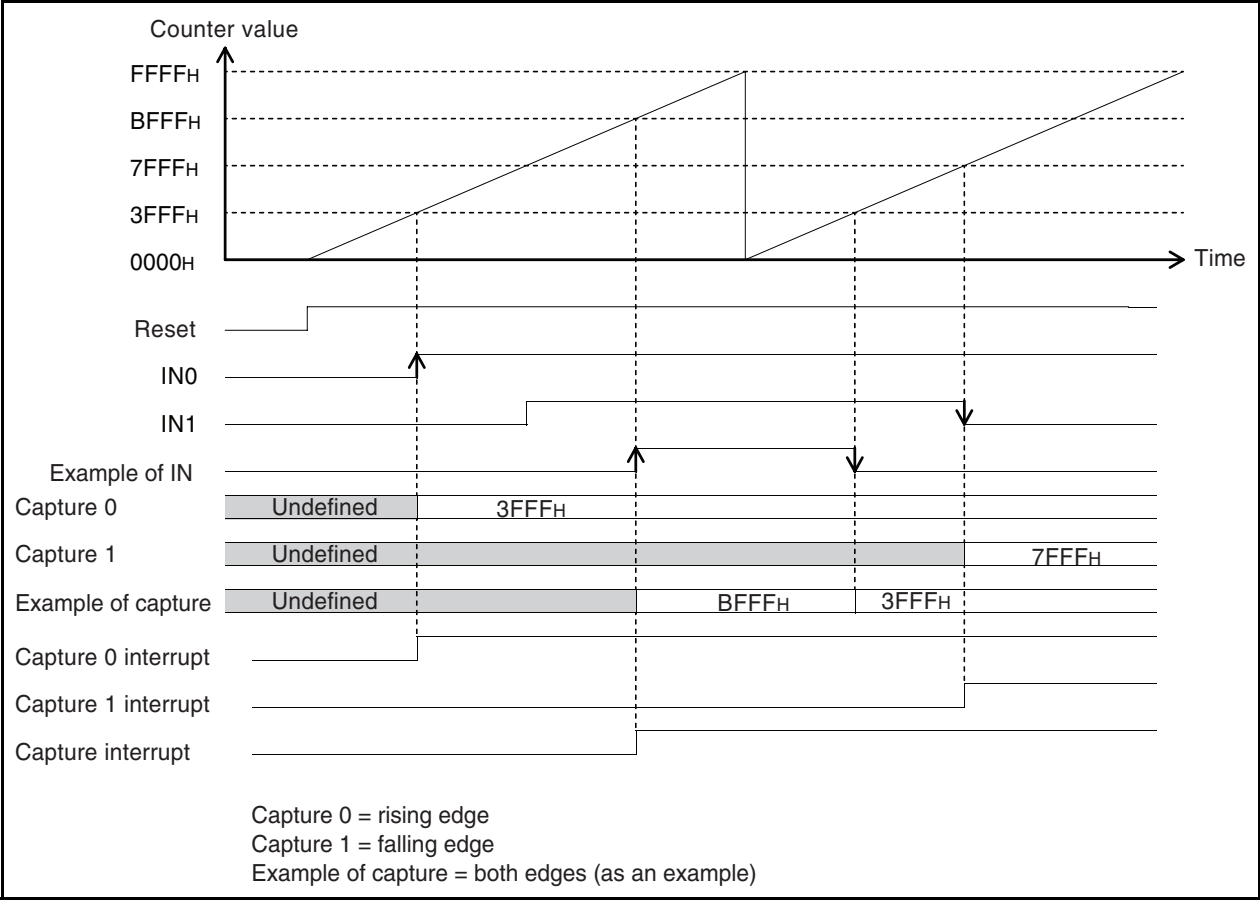


12.6 16-Bit Input Capture Operations

When detecting the specified valid edge, the 16-bit input capture can take a 16-bit free-run timer value in the capture register to generate an interrupt.

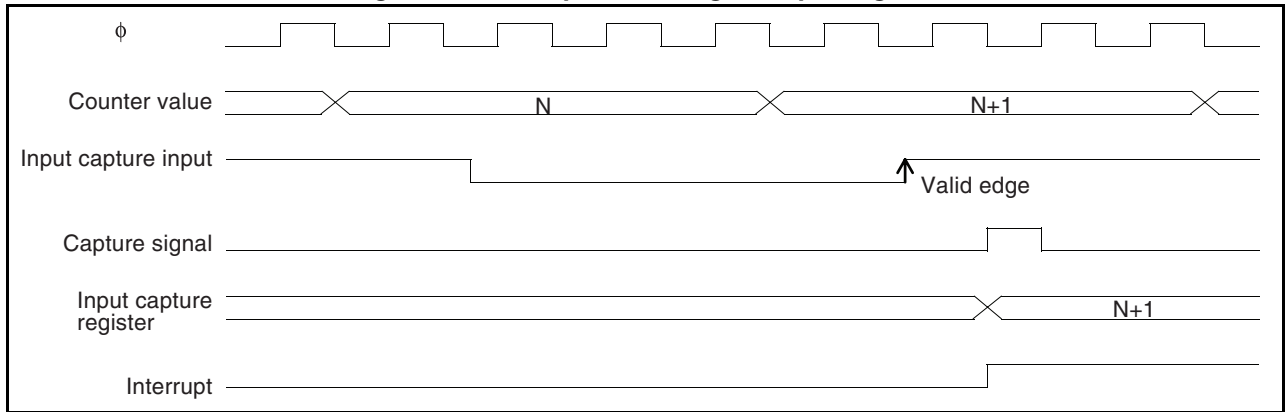
■ 16-bit Input Capture Operations

Figure 12.6-1 Example of Input Capture Take-in Timings



■ Input Capture Input Timing

Figure 12.6-2 Capture Timing for Input Signals



CHAPTER 13 16-BIT RELOAD TIMER (WITH THE EVENT COUNT FUNCTION)

This chapter gives an overview of the 16-bit reload timer (with the event count function) and explains its functions.

- 13.1 "Overview of the 16-Bit Reload Timer (with the Event Count Function)"
- 13.2 "Registers of the 16-Bit Reload Timer (with the Event Count Function)"
- 13.3 "Clock Operations"
- 13.4 "Underflow Operation"
- 13.5 "I/O Pin Functions (for the Internal Clock Mode)"
- 13.6 "Counter Operation Statuses"

13.1 Overview of the 16-Bit Reload Timer (with the Event Count Function)

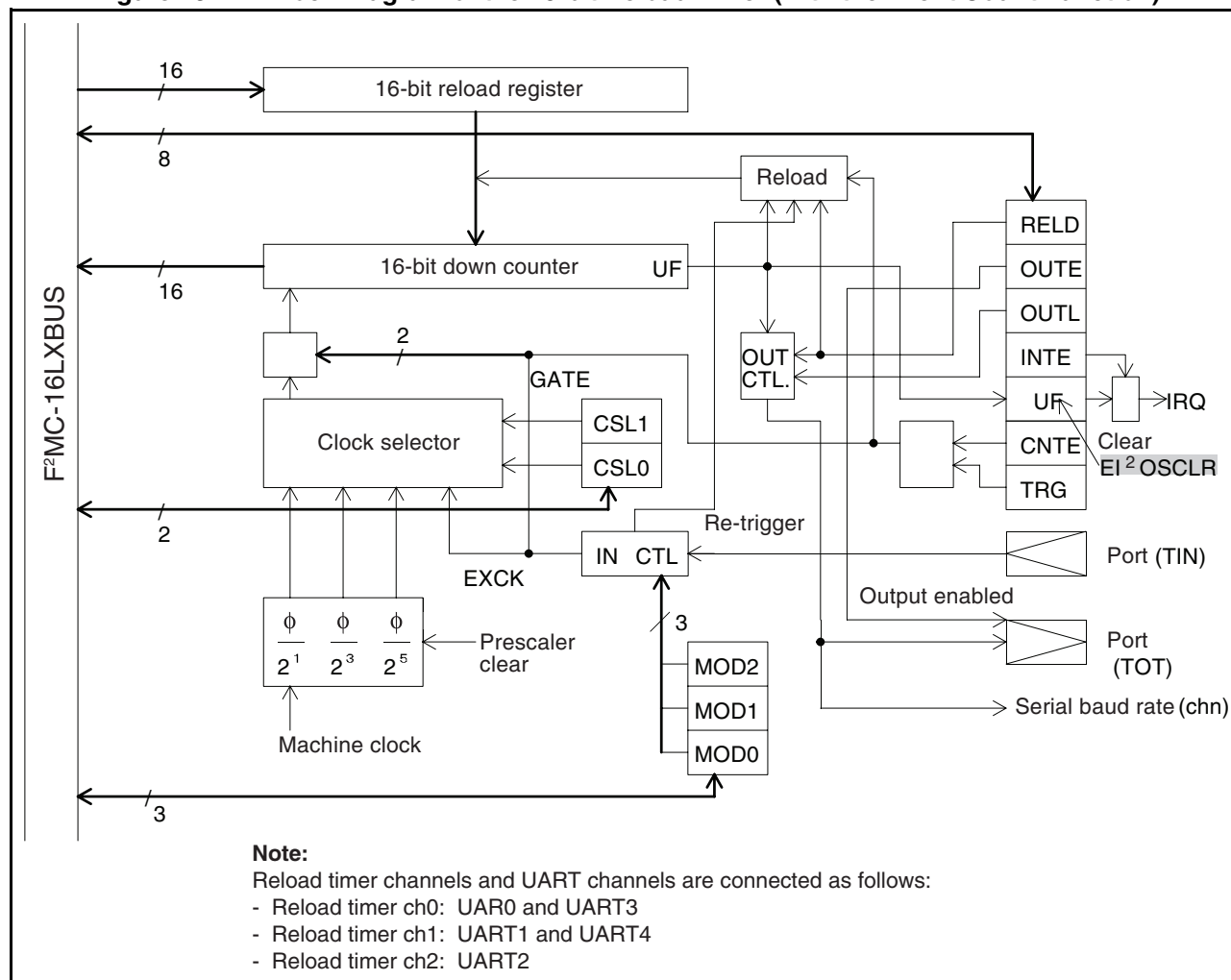
The 16-bit reload timer has three channels consisting of a 16-bit down counter, a 16-bit reload register, a input pin (TIN), an output pin (TOT), and a control register. An input clock can be selected from one external clock and three internal clocks.

■ Overview of the 16-bit Reload Timer (with the Event Count Function)

In the reload mode, a toggle output waveform is output to the output pin (TOT). In the one-shot mode, a rectangular wave is output to indicate that the count is ongoing. The input pin (TIN) becomes an even input in the event count mode and can be used as a trigger or gate input in the internal clock mode.

■ Block Diagram of the 16-bit Reload Timer (with the Event Count Function)

Figure 13.1-1 Block Diagram of the 16-bit Reload Timer (with the Event Count Function)



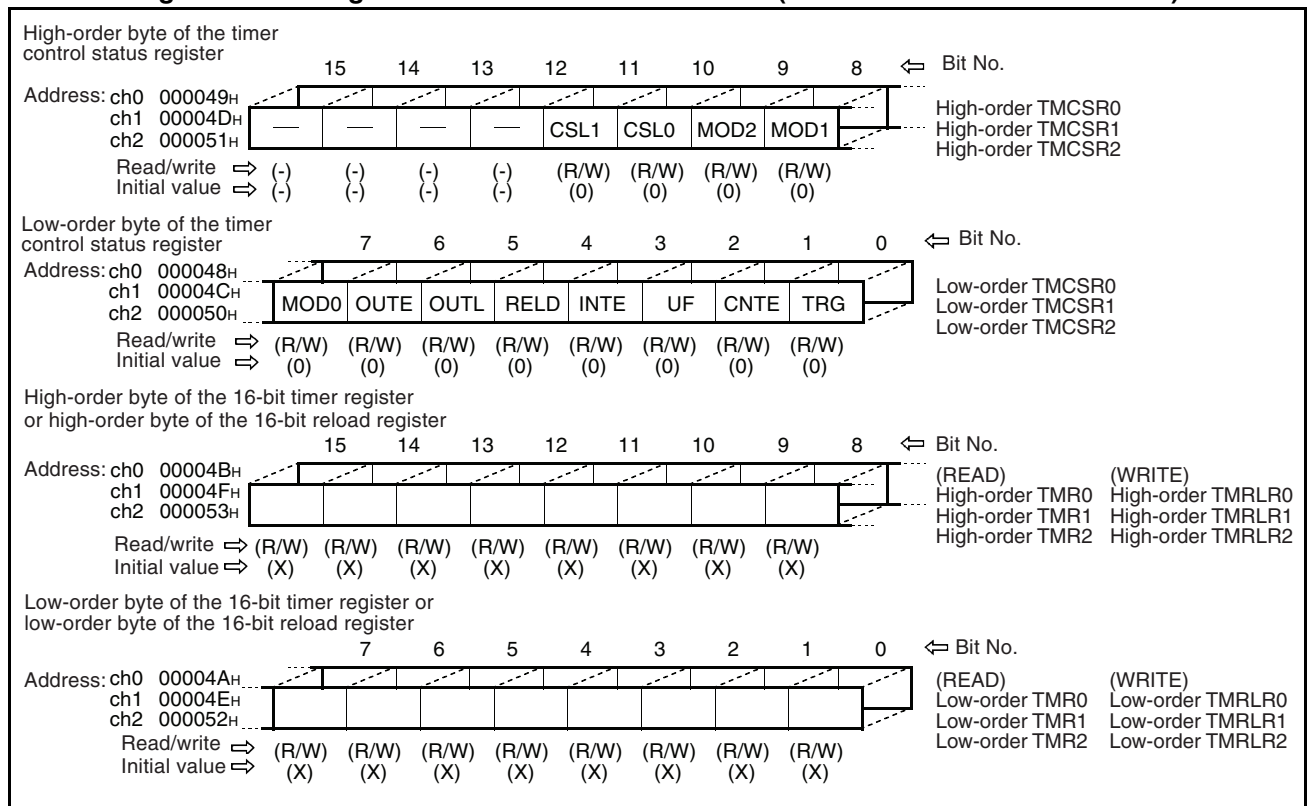
13.2 Registers of the 16-Bit Reload Timer (with the Event Count Function)

The 16-bit reload timer (with the event count function) has the following four types of registers:

- High-order byte of the timer control status register
- Low-order byte of the timer control status register
- High-order byte of the 16-bit timer register or high-order byte of the 16-bit reload register
- Low-order byte of the 16-bit timer register or low-order byte of the 16-bit reload register

■ Registers of the 16-bit Reload Timer (with the Event Count Function)

Figure 13.2-1 Registers of the 16-bit Reload Timer (with the Event Count Function)

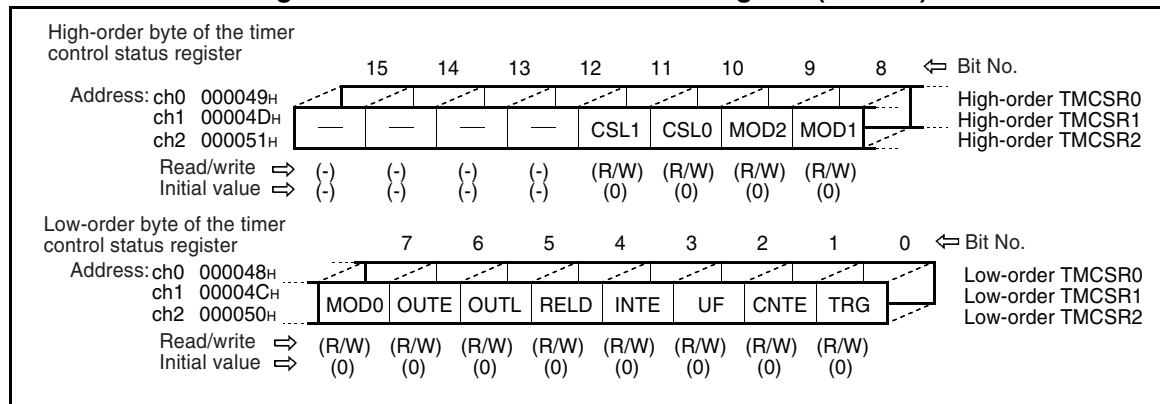


13.2.1 Timer Control Status Register (TMCSR)

The timer control status register (TMCSR) controls 16-bit timer operation modes and interrupts.

■ Timer Control Status Register (TMCSR)

Figure 13.2-2 Timer Control Status Register (TMCSR)



Note:

Rewrite a bit other than the UF, CNTE, and TRG bits when CNTE is 0.

[Bits 11 and 10] CSL1 and CSL0

CSL1 and CSL0 bits are used to select a count clock. The following table lists the clock sources to be selected:

Table 13.2-1 Function of CSL1 and CSL0 (Count clock select bit)

CSL1	CSL0	Clock source (machine cycle $\phi = 16$ MHz)
0	0	$\phi/2^1$ (0.125 μ s) [Initial value]
0	1	$\phi/2^3$ (0.5 μ s)
1	0	$\phi/2^5$ (2.0 μ s)
1	1	External event count mode

[Bits 9, 8, and 7] MOD2, MOD1, and MOD0

MOD2, MOD1, and MOD0 bits are used to set an operation mode and an I/O pin function.

When MOD2 = 0, the input pin functions as trigger input. If a valid edge is input, the contents of the reload register are loaded to the counter and the count operation continues. When MOD2 = 1, the gate counter mode is entered. The input pin (TIN) becomes a gate input and the count operation continues only while the valid level is input.

The MOD1 and MOD0 bits are used to set a pin function in each mode.

Table 13.2-2 Functions of MOD2, MOD1, and MOD0 (for Setting an Operation Mode and I/O Pin Function)

Mode	MOD2	MOD1	MOD0	I/O pin function	Valid edge, level
Internal clock mode (CSL0, 1=00 _B , 01 _B , 10 _B)	0	0	0	Trigger disabled	-
	0	0	1	Trigger input	Rising edge
	0	1	0		Falling edge
	0	1	1		Both edges
	1	x	0	Gate input	L level
	1	x	1		H level
Event count mode (CSL0, 1=11 _B)	x	0	0	-	-
		0	1	Trigger input	Rising edge
		1	0		Falling edge
		1	1		Both edges

x: Any value

[Bit 6] OUTE

The OUTE bit enables the output.

- When this bit is 0, the TOT pin is used as the general-purpose port.
- When this bit is 1, the TOT pin is used as the timer output pin

[Bit 5] OUTL

The OUTL bit sets the output level of the TOT pin.

[Bit 4] RELD (Reload)

The RELD bit enables the reload operation.

- When this bit is 0, a one-shot operation mode is entered. The count operation stops with the underflow of the counter value from 0000_H to FFFF_H.
- When this bit is 1, a reload mode is entered. An underflow of the counter value from 0000_H to FFFF_H occurs and, at the same time, the contents of the reload register are loaded to the counter. The count operation continues.

Table 13.2-3 Function of RELD (reload operation enable bit)

OUTE	RELD	OUTL	Output waveform
0	x	x	General-purpose port
1	0	0	Output of H level rectangular wave during counting
1	0	1	Output of L level rectangular wave during counting
1	1	0	Toggle output. L level at start of count
1	1	1	Toggle output. H level at start of count

x: Any value

[Bit 3] INTE (INTerrupt Enable)

The INTE bit enables a timer interrupt request. When the INTE bit is set to 1, an interrupt request is generated even if the UF bit is changed to 1. When the INTE bit is set to 0, an interrupt request is not generated even if the UF bit is changed to 1.

Table 13.2-4 Function of INTE (for Enabling a Timer Interrupt Request)

INTE	Function
0	Interrupt disabled
1	Interrupt enabled

[Bit 2] UF (UnderFlow)

The UF bit is a timer interrupt request flag.

This bit is set to 1 with an underflow of the count value from 0000_H to FFFF_H.

This bit is cleared by writing 0 or by the extended intelligent I/O service. Writing 1 in this bit is meaningless. Value 1 is read at reading by read modify write instructions.

[Bit 1] CNTE (CouNT Enable)

The CNTE bit enables timer counting.

When 1 is written in this bit, an activation trigger wait status is entered. Writing 0 stops the count operation.

[Bit 0] TRG (TRiG)

The TRG bit triggers software.

A software trigger is provided by writing 1, and the contents of the reload register are loaded to the counter. The count operation starts.

Writing 0 is meaningless. Value 0 is always read. The trigger input by this register is valid only when CNTE is 1. No operation occurs when CNTE is 0.

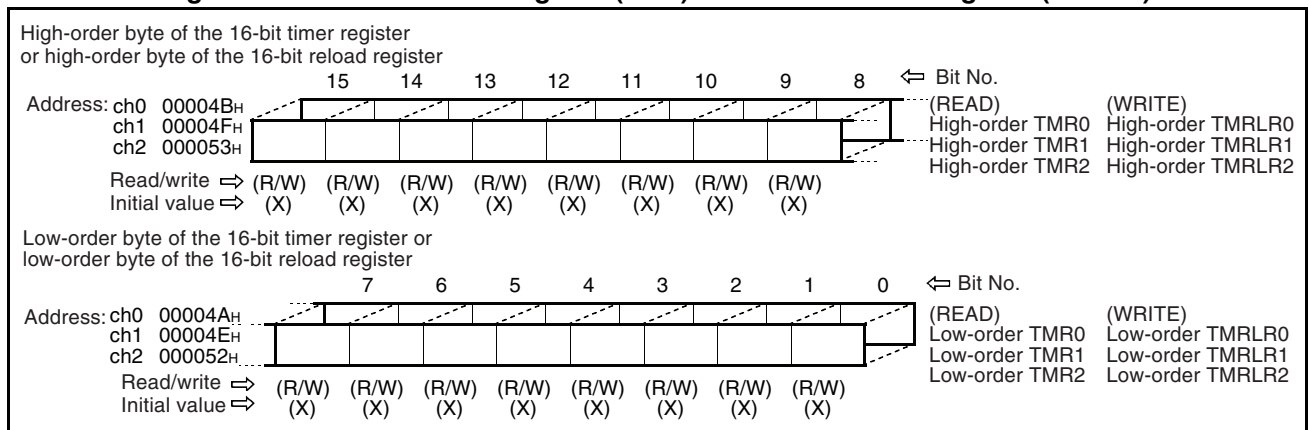
13.2.2 16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)

The 16-bit timer register (TMR) (at reading) can read a count value of the 16-bit timer. The initial value is undefined.

The 16-bit reload register (TMRLR) (at writing) is used to retain the initial value of the count. The initial value is undefined.

■ 16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)

Figure 13.2-3 16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)



Note:

Word accesses are required for the 16-bit timer register (TMR) and 16-bit reload register (TMRLR).

13.3 Clock Operations

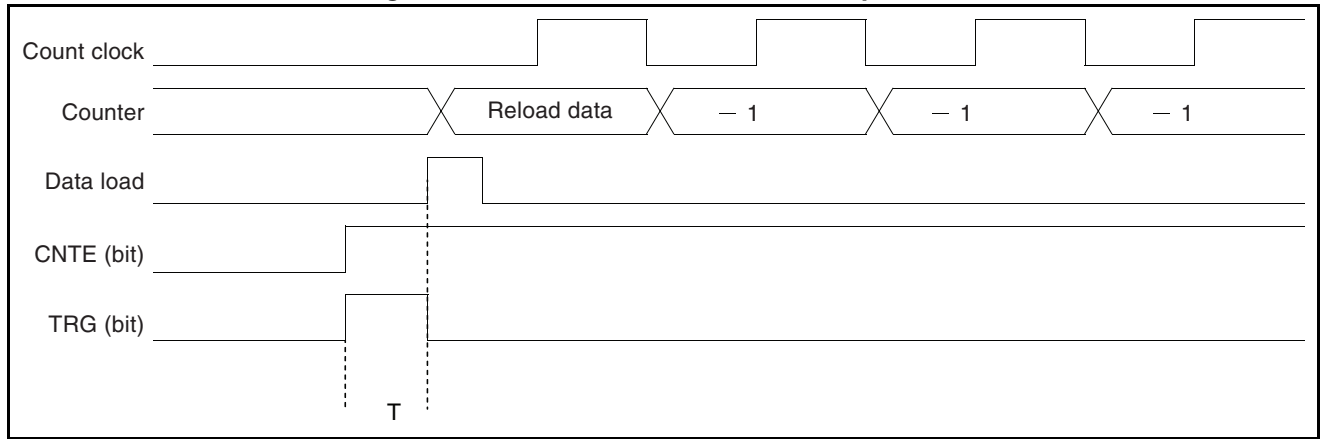
When the timer is operated with a divide-by clock of internal clocks, the divide-by clock can be selected, as a clock source, from 2^1 , 2^3 , and 2^5 divide-by clocks of the machine clock. The external input pin can be used as the trigger or gate input by the register setting.

■ Internal Clock Operations

To start the count operation the moment the count is enabled, write 1 in both the CNTE and TRG bits of the control register. The trigger input by the TRG bit is always valid when the timer is in the activation status (CNTE = 1) regardless of an operation mode.

The time of T (T: machine cycle) is required from when the trigger of the counter start is input until data of the reload register is loaded to the counter.

Figure 13.3-1 Counter Activation and Operation



■ External Event Count

When an external clock is selected, the TIN pin becomes an external event input pin to count the valid edge set by the register. Input a pulse width of at least $4 \times T$ (T: machine cycle) to the TIN pin.

13.4 Underflow Operation

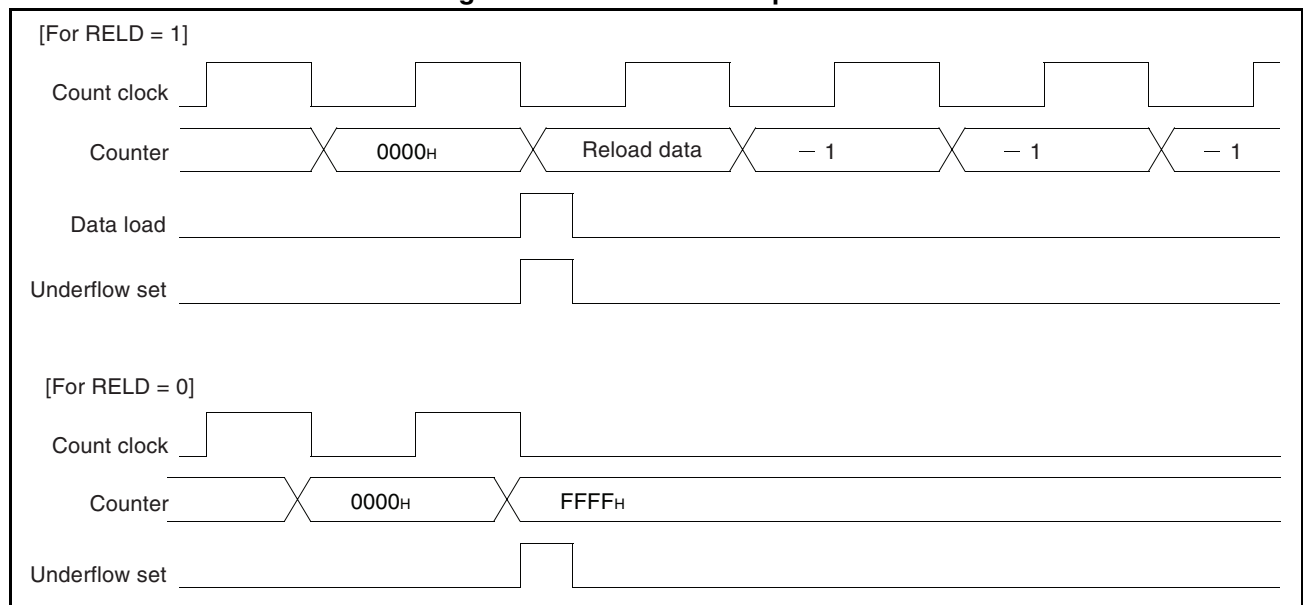
The 16-bit reload timer (with the event count function) defines the following case as an underflow: when a counter value changes 0000_H to FFFF_H.

Therefore, an underflow occurs with [reload register setting value + 1].

■ Underflow Operation

If an underflow occurs when the RELD bit of the control register is 1, the contents of the reload register are loaded to the counter to continue the count operation. When the RELD bit of the control register is 0, the counter stops with FFFF_H. If an underflow occurs, the UF bit of the control register is set. At this time, if the INTE bit is 1, an interrupt request is generated.

Figure 13.4-1 Underflow Operation



■ Extended Intelligent I/O Service (EI²OS) Function and Interrupts

This timer supports EI²OS. When an underflow occurs, EI²OS can be activated. This product allows EI²OS to be used with both timers. However, because the timer with three channels (ch0-2) is connected to the same interrupt control register (ICRx) in the interrupt controller, ch0 to ch2 cannot be allocated to different EI²OS services. It both timers have different interrupt vectors, they can be allocated to two different interrupt services. However, as previously described, because ch0 to ch2 share the interrupt control register, the same interrupt level applies to three channels.

13.5 I/O Pin Functions (for the Internal Clock Mode)

If an internal clock is selected as a clock source, the TIN pin can be used as either a trigger or gate input.

The output polarity can be set by the OUTL bit of the register. In the reload mode, the TOT pin functions as the toggle output, which is reversed by an underflow. In the one-shot mode, the TOT pin functions as the pulse output, which indicates that the count is ongoing.

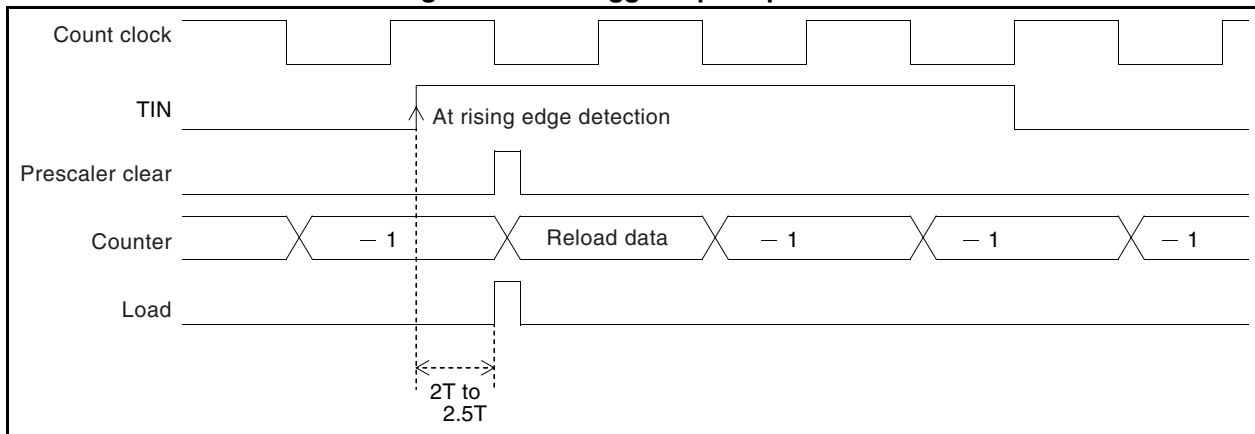
■ Input Pin Function (for the Internal Clock Mode)

If an internal clock is selected as the clock source, the TIN pin can be used as either a trigger or gate input.

If the TIN pin is used as the trigger input, when an active edge is input as shown in Figure 13.5-1 "Trigger Input Operation" the contents of the reload register are loaded to the counter. After the internal prescaler is cleared, the count operation starts.

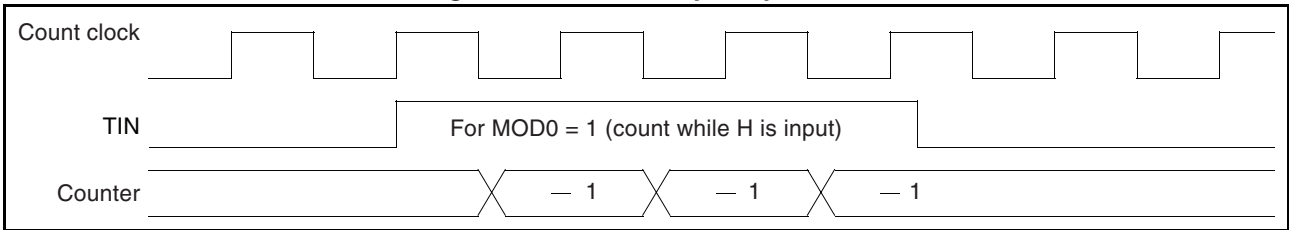
Input a pulse width of at least $2 \times T$ (T: machine cycle) to TIN.

Figure 13.5-1 Trigger Input Operation



If the TIN pin is used as the gate input, the count occurs only while the active level set by the MOD0 bit of the control register is input from the TIN pin, as shown in Figure 13.5-2 "Gate Input Operation". At this time, the count clock continues without stopping. The software trigger in the gate mode is possible regardless of the gate level. Input a pulse width of at least $2 \times T$ (T: machine cycle) to the TIN pin.

Figure 13.5-2 Gate Input Operation



■ Output Pin Function

The output polarity can be set by the OUTL bit of the register. In the reload mode, the TOT pin functions as the toggle output, which is reversed by an underflow. In the one-shot mode, the TOT pin functions as the pulse output, which indicates that the count is ongoing.

Assume that OUTL is 0. In this case, for the toggle output, the initial value is 0. For the one-shot pulse output, 1 is output to indicate that the count is ongoing. When OUTL is set to 1, the output waveform is reversed.

Figure 13.5-3 Output Pin Function (1)

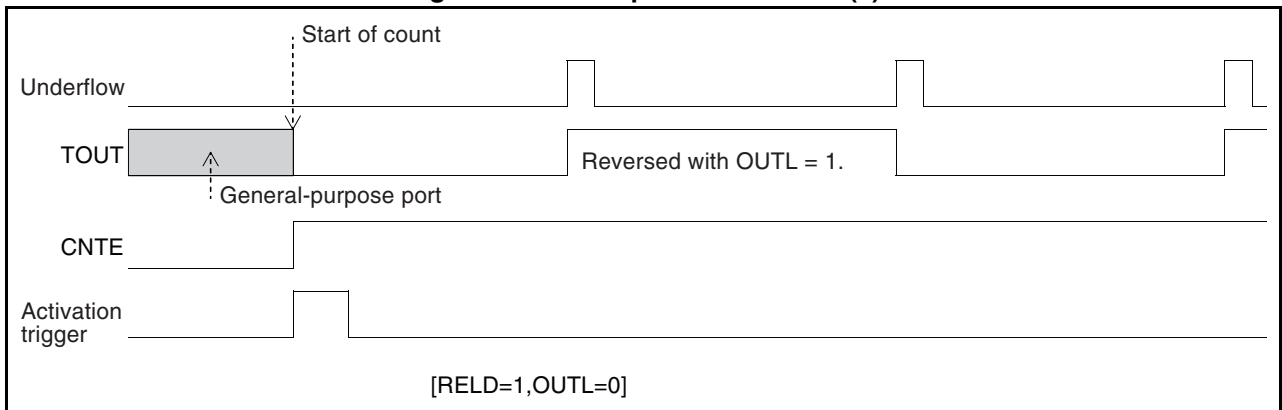
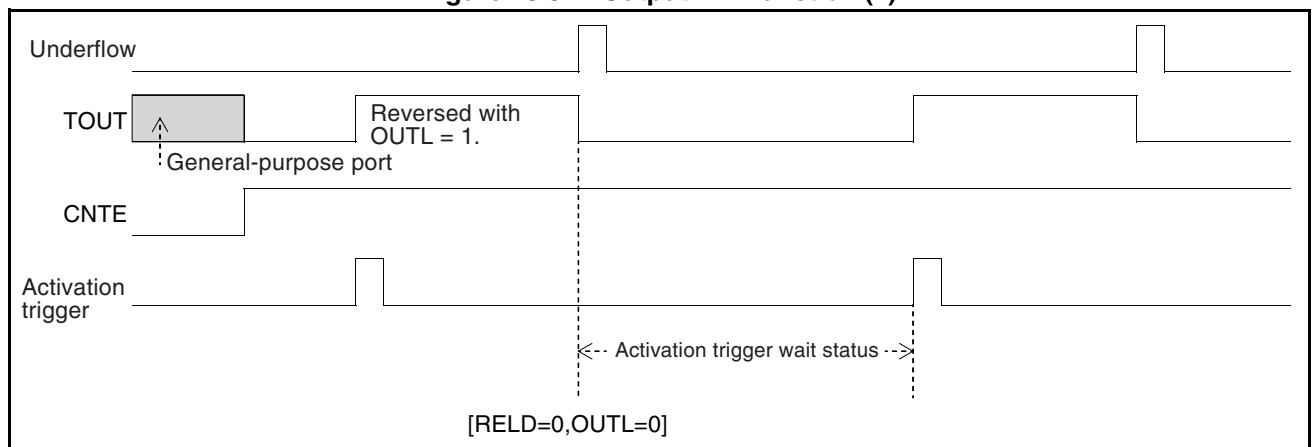


Figure 13.5-4 Output Pin Function (2)



13.6 Counter Operation Statuses

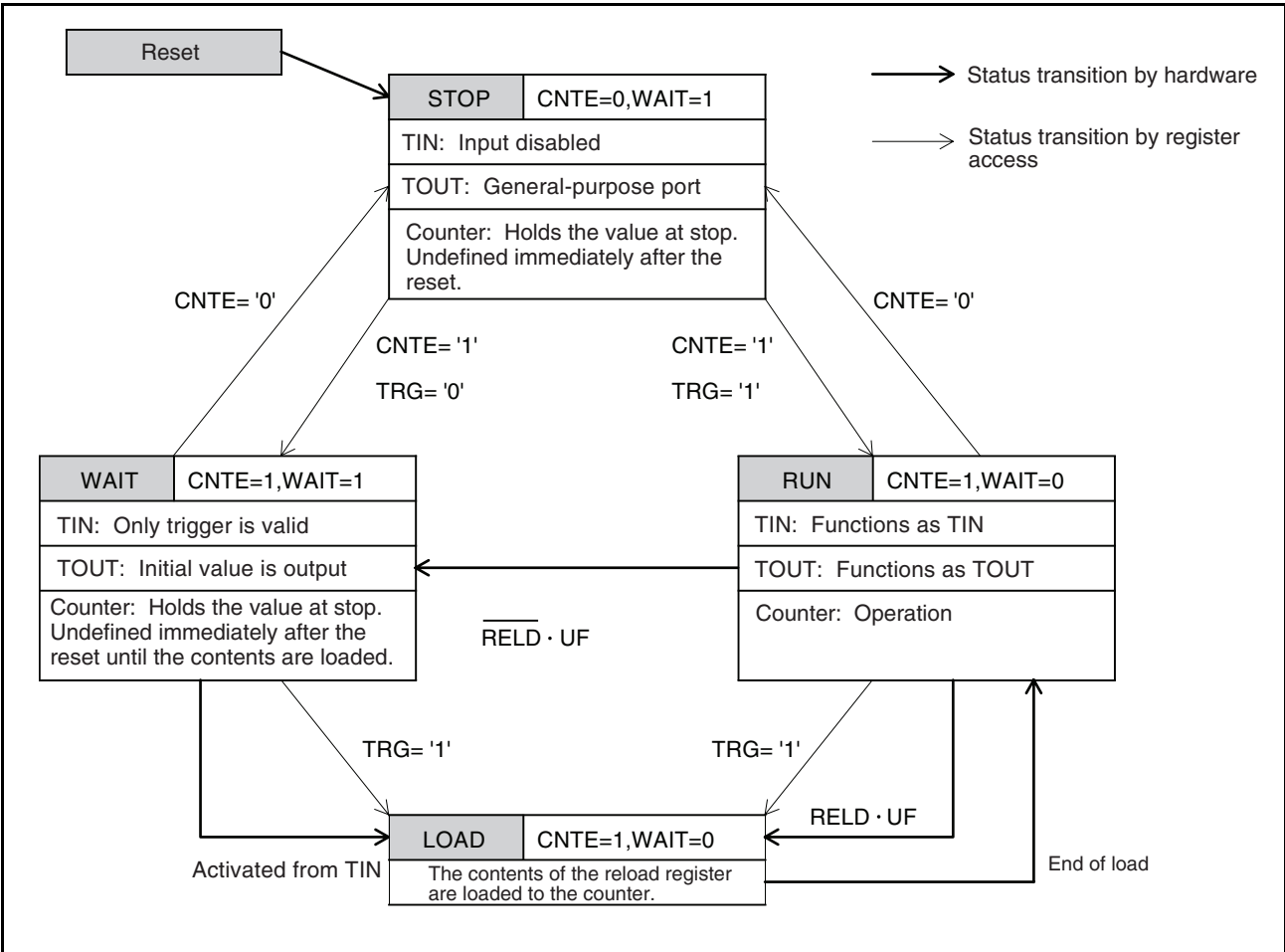
A counter status is determined by the CNTE bit of the control register and the WAIT signal of the internal signal. The following three statuses can be set:

- Stop status (STOP status) by CNTE = 0 and WAIT = 1
- Activation trigger wait status (WAIT status) by CNTE = 1 and WAIT = 1
- Operation status (RUN status) by CNTE = 1 and WAIT = 0

■ Counter Operation Statuses

Figure 13.6-1 "Counter Status Transition" shows the counter operation statuses.

Figure 13.6-1 Counter Status Transition



CHAPTER 14 8/16-BIT PPG

This chapter describes the function and operation of the 8/16-bit PPG.

- 14.1 "Overview of the 8/16-Bit PPG"
- 14.2 "Block Diagrams of the 8/16-Bit PPG"
- 14.3 "Registers in the 8/16-Bit PPG"
- 14.4 "8/16-Bit PPG Operation"

14.1 Overview of the 8/16-Bit PPG

The 8/16-bit PPG is an 8/16-bit reload timer module. Based on timer operation, the module performs pulse output control to allow PPG output.

■ Overview of the 8/16-bit PPG

The 8/16-bit PPG consists of two 8-bit down counters, four 8-bit reload registers, one 16-bit control registers, two external pulse output pins, and two interrupt outputs. The PPG provides the following functions:

- **8-bit PPG output in 2-channel independent operation mode (8-bit PPG 2-ch mode)**

Allows 2-channel independent PPG output operation.

- **16-bit PPG output operation mode (16-bit PPG 1-ch mode)**

Allows 2-channel 16-bit PPG output operation.

- **Pair 8-bit-prescaler + 8-bit-PPG mode**

Allows 8-bit PPG output operation at the specified interval by applying ch0 output to the ch1 clock input.

- **PPG output operation**

Outputs a pulse waveform with an arbitrary cycle period and duty cycle. The PPG can also be used as a D/A converter when a circuit is connected externally.

14.2 Block Diagrams of the 8/16-Bit PPG

Figure 14.2-1 "Block Diagram of the 8/16-bit PPG (ch0)" shows a block diagram of the 8/16-bit PPG (ch0), and Figure 14.2-2 "Block Diagram of the 8/16-bit PPG (ch1)" shows a block diagram of the 8/16-bit PPG (ch1).

■ Block Diagrams of the 8/16-bit PPG

Figure 14.2-1 Block Diagram of the 8/16-bit PPG (ch0)

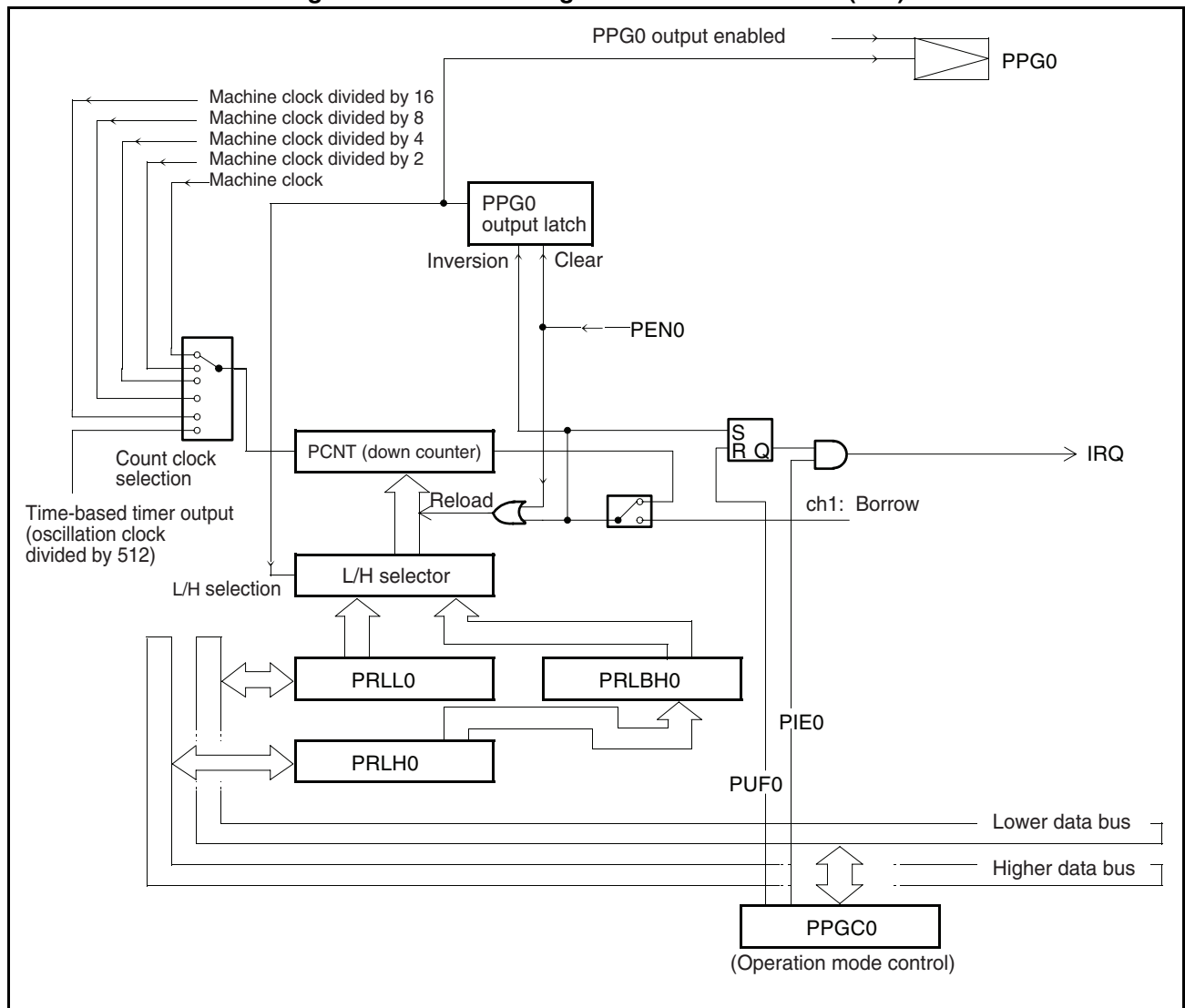
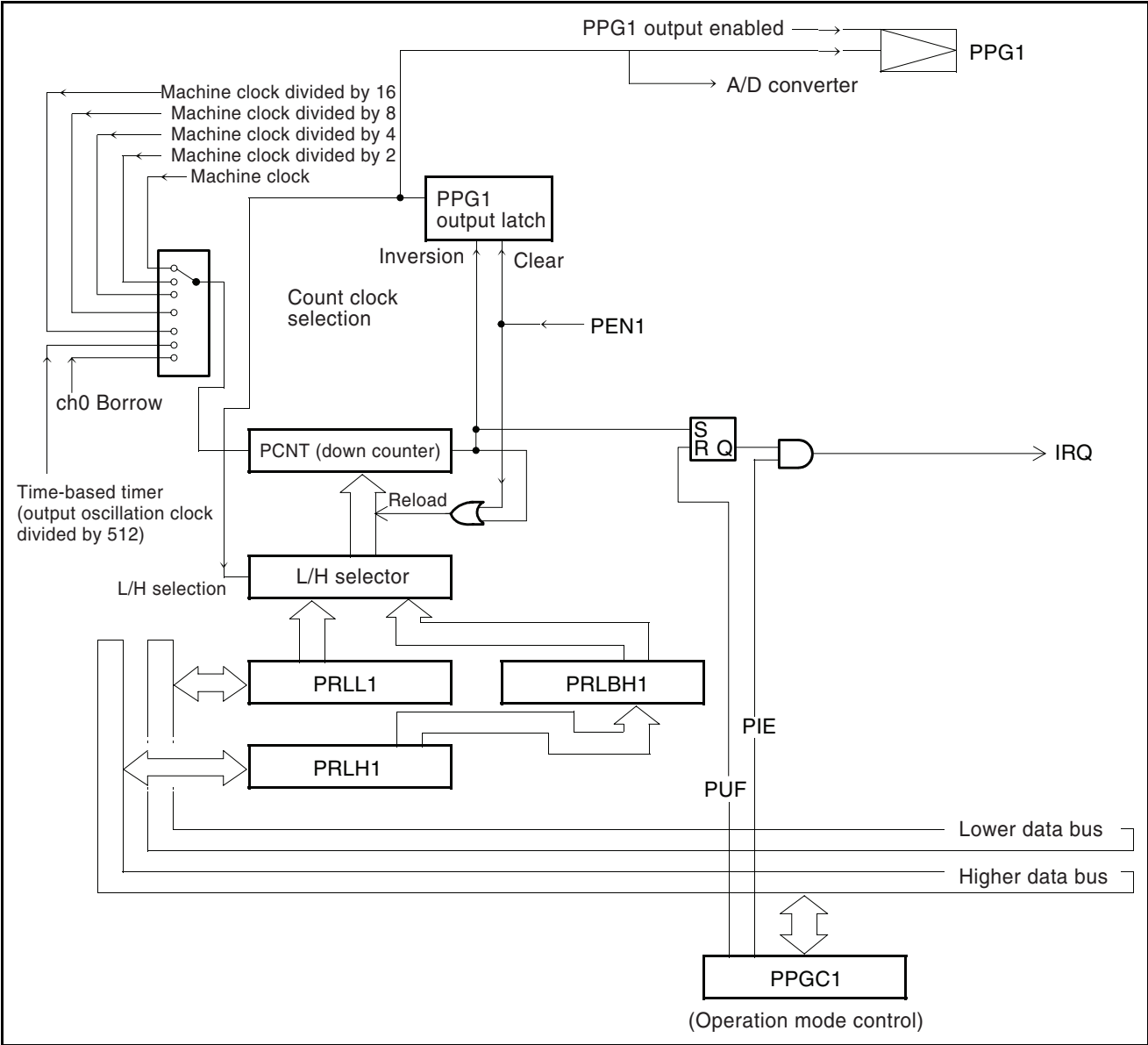


Figure 14.2-2 Block Diagram of the 8/16-bit PPG (ch1)



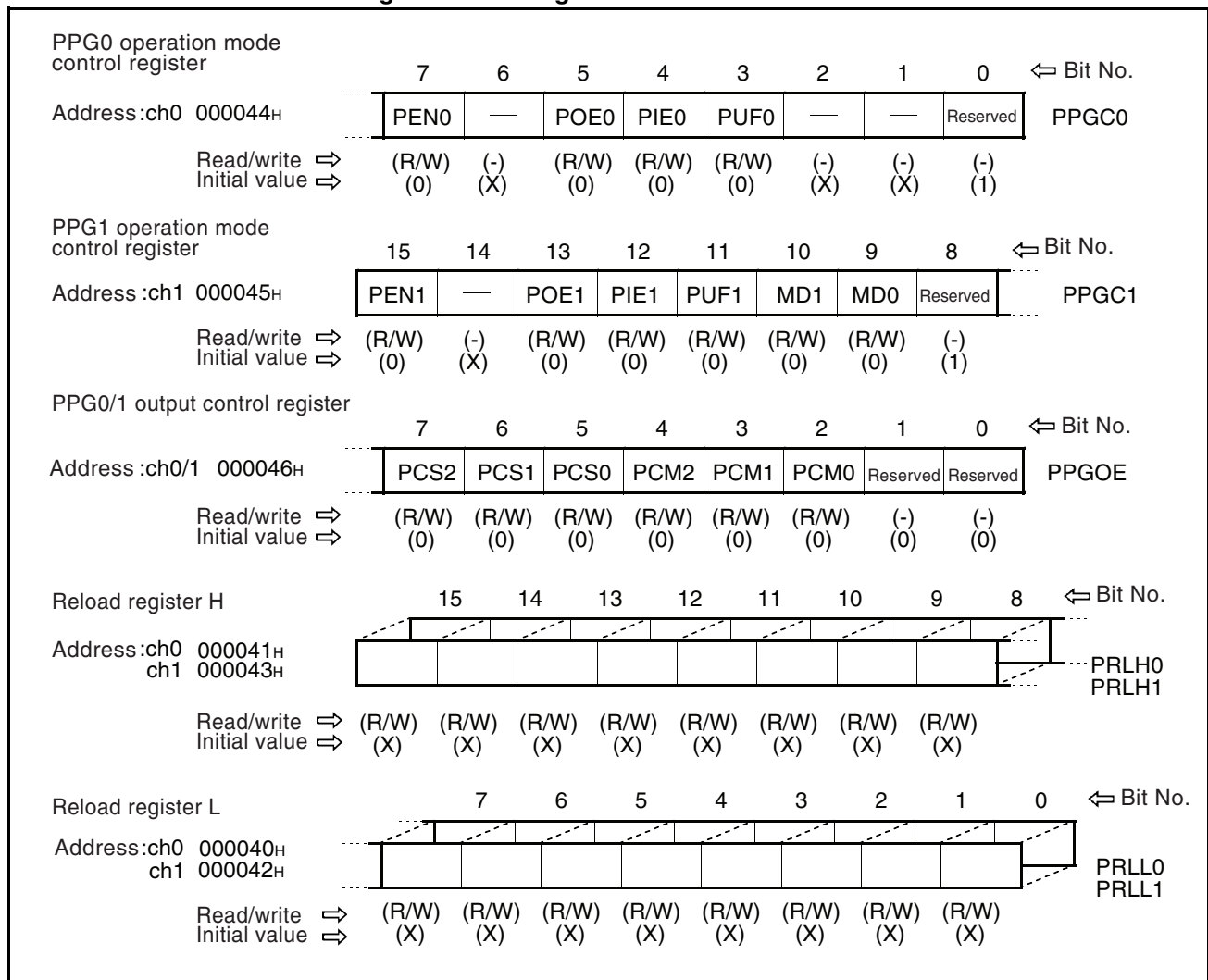
14.3 Registers in the 8/16-Bit PPG

The registers in the 8/16-bit PPG are classified into the following three types:

- PPG0/1 operation mode control register
- PPG0/1 output control register
- Reload register H/L

■ Registers in the 8/16-bit PPG

Figure 14.3-1 Registers in the 8/16-bit PPG



14.3.1 PPG0 Operation Mode Control Register (PPGC0)

The PPG0 operation mode control register (PPGC0) is used for selection of the operation mode of the 8/16-bit PPG, pin output control, count clock selection, and trigger control.

■ PPG0 Operation Mode Control Register (PPGC0)

Figure 14.3-2 PPG0 Operation Mode Control Register (PPGC0)

PPG0 operation mode control register									⇌ Bit No.
Address:ch0 000044H									PPGC0
	7	6	5	4	3	2	1	0	
	PEN0	—	POE0	PIE0	PUF0	—	—	Reserved	
Read/write ⇌	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(-)	(-)	(-)	
Initial value ⇌	(0)	(X)	(0)	(0)	(0)	(X)	(X)	(1)	

[Bit 7] PEN0 (PPg ENable)

The PEN0 bit selects the start of PPG operation and the operation mode of the PPG as shown in Table 14.3-1 "PEN0 (Operation Enable Bit) Function". Writing 1 to this bit causes the PPG to start counting. This bit is initialized to 0 at reset. This bit can be read and written.

Table 14.3-1 PEN0 (Operation Enable Bit) Function

PEN0	Function
0	Operation stopped (low level held output) [initial value]
1	PPG operation enabled

[Bit 5] POE0 (PPg Output Enable)

The POE0 bit controls the pulse output external pin PPG0 as shown in Table 14.3-2 "POE0 (PPG0 Pin Output Enable Bit) Function". This bit is initialized to 0 at reset. This bit can be read and written.

Table 14.3-2 POE0 (PPG0 Pin Output Enable Bit) Function

POE0	Function
0	General-purpose port pin (pulse output disabled) [initial value]
1	PPG0 = pulse output pin (pulse output enabled)

[Bit 4] PIE0 (PPG Interrupt Enable)

The PIE0 bit enables or disables PPG interrupts as shown in Table 14.3-3 "PIE0 (PPG Interrupt Enable Bit) Function".

If this bit is 1, an interrupt request is issued when PUF0 is set to 1.

If this bit is 0, no interrupt request is issued.

This bit is initialized to 0 at reset. This bit can be read and written.

Table 14.3-3 PIE0 (PPG Interrupt Enable Bit) Function

PIE0	Function
0	Interrupt disabled [initial value]
1	Interrupt enabled

Note:

8-/16-bit PPG timer 0 (PPG0) and 16-bit reload timer share the interrupt control register (ICR07). To use EI²OS with the 16-bit reload timer, disable PPG0 interrupt (PIE bit = 0).

[Bit 3] PUF0 (PPG Underflow Flag)

The PUF0 bit controls the PPG counter underflow bit as listed in Table 14.3-4 "PUF0 (PPG Counter Underflow Bit) Function".

Table 14.3-4 PUF0 (PPG Counter Underflow Bit) Function

PUF0	Function
0	PPG counter underflow is not detected [initial value]
1	PPG counter underflow was detected

In the 8-bit PPG 2-ch mode and the 8-bit prescaler + 8-bit PPG mode, an underflow occurred when the ch0 counter value changes from 00_H to FF_H sets the bit to 1. In the 16-bit PPG 1-ch mode, an underflow caused when the ch1/ch0 counter value changes from 0000_H to FFFF_H sets the bit to 1. This bit is set to 0 by writing 0 to this bit.

Writing 1 to this bit has no meaning.

At read in read-modify-write operation, 1 is read from this bit.

This bit is initialized to 0 at reset. This bit can be read and written.

[Bit 0] Reserved bit

Bit 0 is a reserved bit. Whenever setting PPGC0, be sure to set this bit to 1.

14.3.2 PPG1 Operation Mode Control Register (PPGC1)

The PPG1 operation mode control register (PPGC1) is used for selection of the operation mode of the 8/16-bit PPG, pin output control, count clock selection, and trigger control.

■ PPG1 Operation Mode Control Register (PPGC1)

Figure 14.3-3 PPG1 Operation Mode Control Register (PPGC1)

PPG1 operation mode control register Address :ch1 000045H	15	14	13	12	11	10	9	8	⇐ Bit No.
	PEN1	—	POE1	PIE1	PUF1	MD1	MD0	Reserved	PPGC1
	Read/write ⇒ Initial value ⇒	(R/W) (0)	(-) (X)	(R/W) (0)	(R/W) (0)	(R/W) (0)	(R/W) (0)	(-) (1)	

[Bit 15] PEN1 (PPg ENable)

The PEN1 bit selects the start of PPG operation and the operation mode of the PPG as shown in Table 14.3-5 "Operation Enable Bit (PEN1) Function". Writing 1 to this bit causes the PWM to start counting.

This bit is initialized to 0 at reset. This bit can be read and written.

Table 14.3-5 Operation Enable Bit (PEN1) Function

PEN1	Function
0	Operation stopped (low level held output) [initial value]
1	PPG operation enabled

[Bit 13] POE1 (PPg Output Enable)

The POE1 bit controls the pulse output external pin PPG1 as shown in Table 14.3-6 "POE1 (PPG1 Pin Output Enable Bit) Function".

This bit is initialized to 0 at reset. This bit can be read and written.

Table 14.3-6 POE1 (PPG1 Pin Output Enable Bit) Function

POE1	Function
0	General-purpose port pin (pulse output disabled) [initial value]
1	PPG1 serving as a pulse output pin (pulse output enabled)

[Bit 12] PIE1 (PPG Interrupt Enable)

The PIE1 bit enables or disables PPG interrupts as shown in Table 14.3-7 "PIE1 (PPG Interrupt Enable Bit) Function". If this bit is 1, an interrupt request is issued when PUF1 is set to 1. If this bit is 0, no interrupt request is issued.

A reset initializes this bit to 0. This bit can be read from and written to.

Table 14.3-7 PIE1 (PPG Interrupt Enable Bit) Function

PIE1	Function
0	Interrupt disabled [initial value]
1	Interrupt enabled

Note:

8-/16-bit PPG timer 1 (PPG1) and UART0 transmission completion share the interrupt control register (ICR11). To use EI²OS with UART0 transmission completion, disable the PPG1 interrupt (PIE bit = 0).

[Bit 11] PUF1 (PPG Underflow Flag)

PUF1 controls the PPG counter underflow bit as listed in Table 14.3-8 "PUF1 (PPG Counter Underflow Bit) Function".

In the 8-bit PPG 2-ch mode and the 8-bit prescaler + 8-bit PPG mode, an underflow occurred when the ch1 counter value changes from 00_H to FF_H sets the bit to 1. In the 16-bit PPG 1-ch mode, an underflow caused when the ch1/ch0 counter value changes from 0000_H to FFFF_H sets the bit to 1. This bit is set to 0 by writing 0 to this bit. Writing 1 to this bit has no meaning. At read in read-modify-write operation, 1 is read from this bit.

A reset initializes this bit to 0. This bit can be read from and written to.

Table 14.3-8 PUF1 (PPG Counter Underflow Bit) Function

PUF1	Function
0	PPG counter underflow is not detected [initial value]
1	PPG counter underflow was detected

[Bits 10 and 9] MD1,MD0 (PPG count Mode)

The MD1 and MD0 bits select the operation mode of the PPG timer as shown in Table 14.3-9 "MD1 and MD0 (Operation Mode Selection Bits) Function".

A reset initializes these bits to 00_B.

These bits can be read from and written to.

Table 14.3-9 MD1 and MD0 (Operation Mode Selection Bits) Function

MD1	MD0	Operation mode [initial value]
0	0	8-bit PPG 2-ch mode
0	1	8-bit prescaler + 8-bit PPG 1-ch mode
1	0	Reserved (setting inhibited)
1	1	16-bit PPG 1-ch mode

Note:

Do not set these bits to 10_B.

When setting these bits to 01_B, do not set the PEN0 bit of PPGC0 to 01_B and the PEN1 bit of PPGC1 to 01_B. It is recommended that the PEN0 and PEN1 bits be set to 11_B or 00_B at the same time.

When setting these bits to 11_B, rewrite PPGC0/PPGC1 by a word transfer to set the PEN0 and PEN1 bits to 11_B or 00_B at the same time.

[Bit 8] Reserved bit

Bit 8 is a reserved bit. Whenever setting PPGC1, be sure to set this bit to 1.

14.3.3 PPG0/1 Output Pin Control Register (PPGOE)

The PPG0/1 output pin control register (PPGOE) is an 8-bit control register for 8/16-bit PPG pin output control.

■ PPG0/1 Output Pin Control Register (PPGOE)

Figure 14.3-4 PPG0/1 Output Pin Control Register (PPGOE)

PPG0/1 output control register								
	7	6	5	4	3	2	1	0 ⇐ Bit No.
Address :ch0/1 000046H	PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	Reserved	Reserved PPGOE
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

[Bits 7 to 5] PCS2 to PCS0 (PPg Count Select)

PCS2 to PCS0 select the operation clock for the ch1 down counter as shown in Table 14.3-10 "PCS2 to PCS0 (Count Clock Selection Bits) Function". These bits are initialized to 000_B at reset. These bits can be read and written.

Table 14.3-10 PCS2 to PCS0 (Count Clock Selection Bits) Function

PCS2	PCS1	PCS0	Operation mode
0	0	0	Machine clock (62.5 ns, machine clock at 16 MHz)
0	0	1	Machine clock/2 (125 ns, machine clock at 16 MHz)
0	1	0	Machine clock/4 (250 ns, machine clock at 16 MHz)
0	1	1	Machine clock/8 (500 ns, machine clock at 16 MHz)
1	0	0	Machine clock/16 (1 μs, machine clock at 16 MHz)
1	1	1	Clock input from time-based timer (128 μs, source oscillation at 4 MHz)

Note:

In the 8-bit prescaler + 8-bit PPG mode and in the 16-bit PPG 1-ch mode, the PPG for ch1 operates with the count clock signal received from ch0. Therefore, the settings on PCS bits are ignored.

[Bits 4 to 2] PCM2 to PCM0 (PPg Count Mode)

The PCM2 to PCM0 bits select the operation clock of the ch0 down counter as shown in Table 14.3-11 "PCM2 to PCM0 (Count Clock Selection Bits) Function". These bits are initialized to 000_B at reset. These bits can be read and written.

Table 14.3-11 PCM2 to PCM0 (Count Clock Selection Bits) Function

PCM2	PCM1	PCM0	Operation mode
0	0	0	Machine clock (62.5 ns, machine clock at 16 MHz)
0	0	1	Machine clock/2 (125 ns, machine clock at 16 MHz)
0	1	0	Machine clock/4 (250 ns, machine clock at 16 MHz)
0	1	1	Machine clock/8 (500 ns, machine clock at 16 MHz)
1	0	0	Machine clock/16 (1 μ s, machine clock at 16 MHz)
1	1	1	Clock input from time-based timer (128 μ s, source oscillation at 4 MHz)

[Bits 1 and 0] Reserved bits

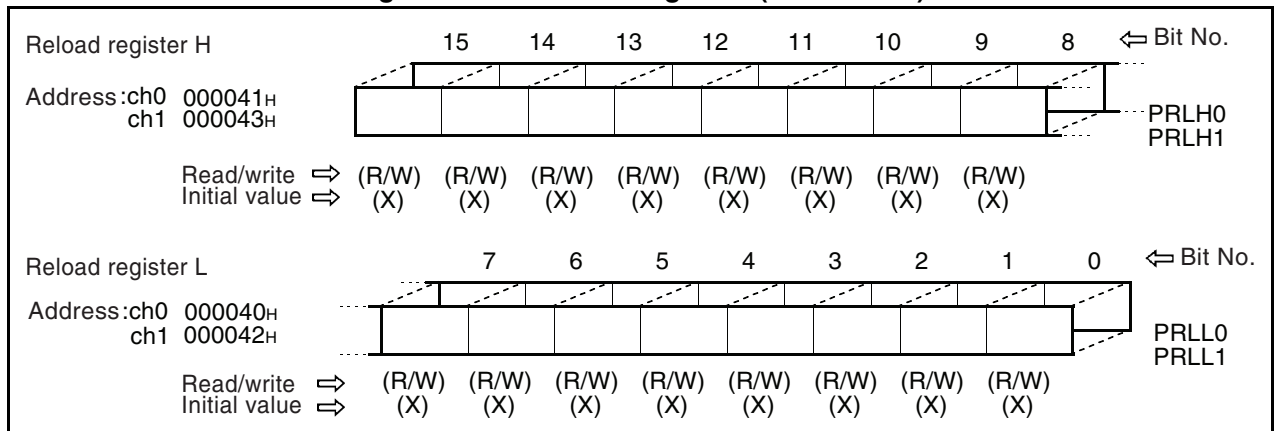
Bits 1 and 0 are reserved bits. Whenever setting PPGE, be sure to set these bits to 0.

14.3.4 Reload Registers (PRLH/PRLH)

The reload registers (PRLH/PRLH), each consisting of 8 bits, hold a value to be reloaded to the down counter PCNT.

■ Reload Registers (PRLH/PRLH)

Figure 14.3-5 Reload Registers (PRLH/PRLH)



The reload registers (PRLH/PRLH) have the function shown in Table 14.3-12 "Reload Registers (PRLH/PRLH)". Each register can be read from and written to.

Table 14.3-12 Reload Registers (PRLH/PRLH)

Register name	Function
PRLH	Holds the reload value for lower data.
PRLH	Holds the reload value for higher data.

Note:

In the 8-bit prescaler + 8-bit PPG mode, setting different values in PRLH and PRLH for ch1 may vary the PPG waveform on ch1 from cycle to cycle. Therefore, it is recommended that the same value be set in PRLH and PRLH for ch0.

14.4 8/16-Bit PPG Operation

The 8/16-bit PPG contains two 8-bit PPG units and can operate in the 8-bit 2-ch mode and in two other operation modes, including the 8-bit prescaler + 8-bit PPG mode and the 16-bit PPG 1-ch mode, where the two PPG units interact.



■ 8/16-bit PPG Operation

For each of the 8-bit PPG units, two 8-bit reload registers (PRLH and PRLH) are provided for the lower and higher data. The value for the lower data and the value for the higher data written in these registers are alternately reloaded into the 8-bit down counter (PCNT), which counts down on each clock pulse. At a reload operation performed when a borrow is generated in the counter, the pin output (PPG) value is inverted. This operation allows the pin output (PPG) to output a pulse signal with low-level and high-level widths corresponding to the reload register values.

Operation is started and restarted by writing an appropriate register bit.

The relationship between reload operation and pulse output is shown in Table 14.4-1 "Relationship between Reload Operation and Pulse Output".

Table 14.4-1 Relationship between Reload Operation and Pulse Output

Reloading	Pin output change
PRLH --> PCNT	PPG0/1 [0 --> 1]  Rising edge
PRLH --> PCNT	PPG0/1 [1 --> 0]  Falling edge

When the bit 4 (PIE0) bit of the PPGC0 register is 1 and when the bit 12 (PIE1) bit of the PPGC1 register is 1, a borrow from 00_H to FF_H in each counter (a borrow from 0000_H to FFFF_H in the 16-bit PPG mode) causes an interrupt request to be output.

■ 8/16-bit PPG Interrupt

An interrupt of the 8/16-bit PPG becomes active when the counter counts out the reloaded value, generating a borrow.

In the 8-bit PPG 2-ch mode and in the 8-bit prescaler + 8-bit PPG mode, a borrow into each counter causes a relevant interrupt request. In the 16-bit PPG mode, a borrow into the 16-bit counter sets the PUF0 bit and PUF1 bit at the same time. It is recommended that only one of the PIE0 and PIE1 bits be enabled to determine a single interrupt source. It is also recommended that the interrupt source be cleared by resetting the PUF0 bit and PUF1 bit at the same time.

■ Initial Values In Hardware Components

A reset initializes hardware components of the 8/16-bit PPG as follows:

○ Registers

- PPGC0 --> 0X000XX1_B
- PPGC1 --> 0X000001_B
- PPGOE --> 00000000_B

○ Pulse output

- PPG0 --> "L"
- PPG1 --> "L"
- PE0 --> PPG0 output disabled
- PE1 --> PPG1 output disabled

○ Interrupt request

- IRQ0 --> "L"
- IRQ1 --> "L"

Hardware components other than the above are not initialized.

14.4.1 8/16-bit PPG Operation Modes

The following three types of operation modes are available with the 8/16-bit PPG:

- 8-bit 2-ch mode
 - 8-bit prescaler + 8-bit PPG mode
 - 16-bit PPG 1-ch mode
-

■ 8/16-bit PPG Operation Modes

○ 8-bit 2-ch mode

In the 2-ch independent mode, two channels of 8-bit PPG unit operate independently of each other.

The PPG0 pin is connected to the PPG output of ch0, and the PPG1 pin is connected to the PPG output of ch1.

○ 8-bit prescaler + 8-bit PPG mode

In the 8-bit prescaler + 8-bit PPG mode, an 8-bit PPG waveform can be output by using ch0 as an 8-bit prescaler and counting ch1 with ch0 borrow output.

The PPG0 pin is connected to the prescaler output of ch0, and the PPG1 pin is connected to the PPG output of ch1.

○ 16-bit PPG 1-ch mode

In 16-bit PPG 1-ch mode, ch0 and ch1 are connected and used as 16-bit PPG. The PPG0 pin and PPG1 pin are both connected to 16-bit PPG output.

14.4.2 PPG Output Operation

8-/16-bit PPG on ch0 is activated and starts counting when 1 is written to bit 7 (PEN0) of the PPGC0 (PWM operation mode control) register. PPG on ch1 is also activated and starts counting when 1 is written to bit 15 (PEN1) of the PPGC1 register. After an operation is started, a counting operation is stopped by writing 0 to bit 7 of PPGC0 or bit 15 of PPGC1. When the counting operation stops, the pulse output is held at the low-level.

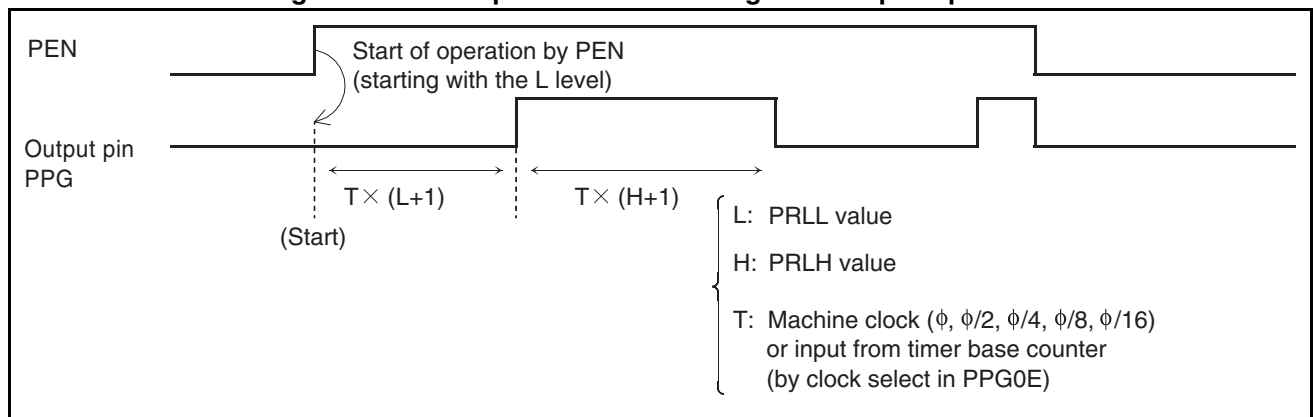
■ PPG Output Operation

In PPG output operation, observe the following two precautions:

- In the 8-bit prescaler + 8-bit PPG mode, do not enable ch1 operation while ch0 is in the stopped state.
- In the 16-bit PPG mode, always perform start and stop control by manipulating bit 7 (PEN0) of the PPGC0 register and bit 15 (PEN1) of the PPGC1 register at the same time.

Figure 14.4-1 "Output Waveform During PPG Output Operation" shows PPG output operation. During PPG operation, a pulse waveform is output successively at any frequency and any duty ratio (the ratio of the time the pulse wave is at H level to the time the pulse wave is at L level). Once the PPG starts outputting a pulse waveform, it does not stop until the operation is set to stop.

Figure 14.4-1 Output Waveform During PPG Output Operation



■ Relationship between the Reloaded Value and Pulse Width

The pulse width of the output pulse signal is obtained by multiplying the value written in the reload register + 1 by the count clock cycle. Note that when the reload register value is 00_H during 8-bit PPG operation and when the reload register value is 0000_H during 16-bit PPG operation, the pulse width equals one count clock cycle. When the reload register value is FF_H during 8-bit PPG operation, the pulse width equals 256 count clock cycles. Similarly, when the reload register value is FFFF_H during 16-bit PPG operation, the pulse width equals 65536 count clock cycles. The following expressions are for calculating a pulse width:

$$Pl = T \times (L + 1)$$

$$Ph = T \times (H + 1)$$

L: PRLl value

H: PRLH value

T: Input clock cycle

Ph: High-level pulse width

Pl: Low-level pulse width

14.4.3 Selecting a Count Clock

The count clock used for 8-/16-bit PPG operation is supplied from the machine clock and the time-based counter, and six types of count clock inputs can be selected.

Bits 4 to 2 (PCM2 to PCM0) of the PPG0E register are used to select the clock for ch0, and bits 7 to 5 (PCS2 to PCS0) of the PPG0E register are used to select the clock for ch1. The clock signal is selected from the clocks produced by dividing the machine clock by 16 to 1 and the clocks input from the time-based counter.

In the 8-bit prescaler + 8-bit PPG mode or 16-bit PPG mode, however, the value of bit 14 (PCS1) of register PPGC1 is invalid. Since PPG on ch1 receives the count clock from ch0, the register becomes invalid.

■ Notes on Selecting a Count Clock

When the time-based timer input is used, the first count cycle when PPG operation is triggered and the first count cycle after PPG operation is stopped may be shifted. In addition, when the time-based counter is cleared during operation of this module, a cycle shift may occur.

In the 8-bit prescaler + 8-bit PPG mode, when ch0 is operating and ch1 is in the stopped state, starting ch1 may shift the first count cycle.

14.4.4 Controlling Pulse Output on Pins

The pulse output generated by operating this module can be output on external pin PPG0/PPG1.

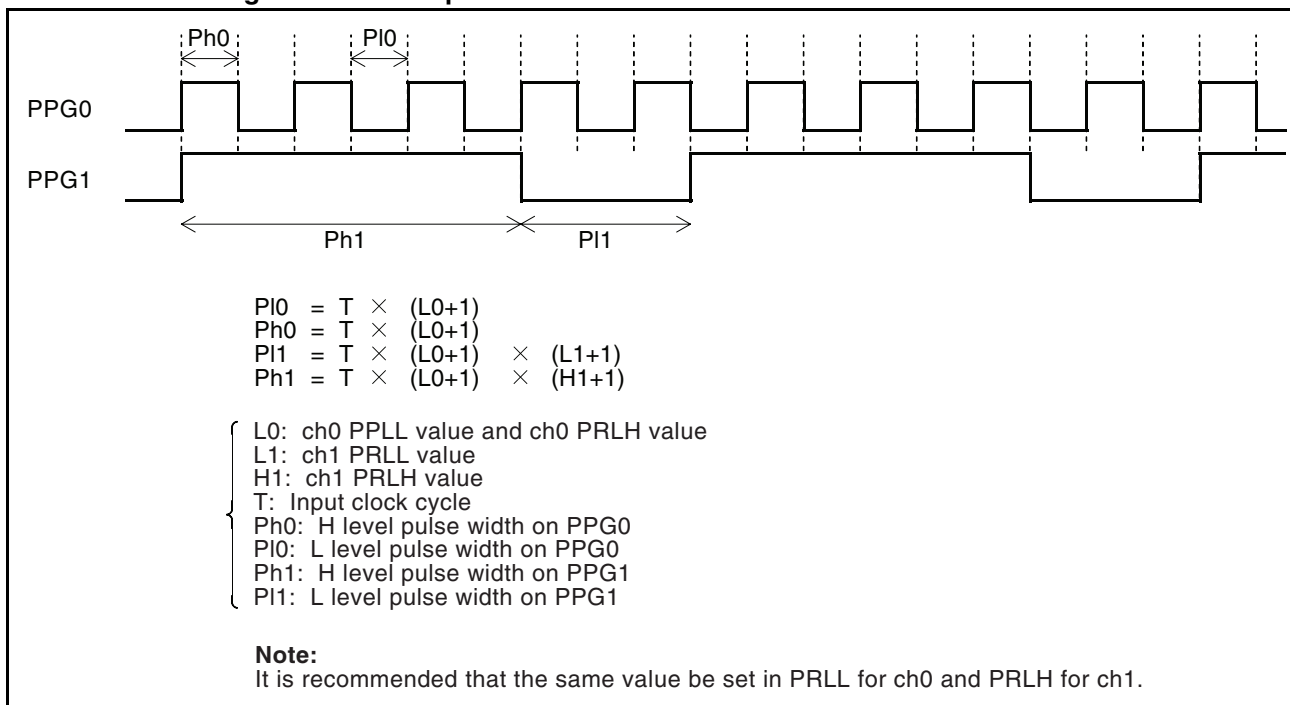
To output the pulse from the external pins, write 1 to the bits corresponding to the pins. Bit 5 (POE0) of the PPGC0 register is used for the PPG0 pin and bit 13 (POE1) of the PPGC1 register is used for the PPG1 pin. When 0 (initial value) is written to these bits, the pulse output does not appear on the external pins. These pins function as a general-purpose port.

■ Controlling Pulse Output on Pins

In the 16-bit PPG 1-ch mode, the same waveform is output on PPG0 and PPG1, so the same output can be obtained by enabling the output of either external pin.

In the 8-bit prescaler + 8-bit PPG mode, a toggle waveform from the 8-bit prescaler is output on PPG0, and a waveform from the 8-bit PPG is output on PPG1. Figure 14.4-2 "Output Waveforms in 8-bit Prescaler + 8-bit PPG Mode" gives an example of output waveforms in this mode.

Figure 14.4-2 Output Waveforms in 8-bit Prescaler + 8-bit PPG Mode

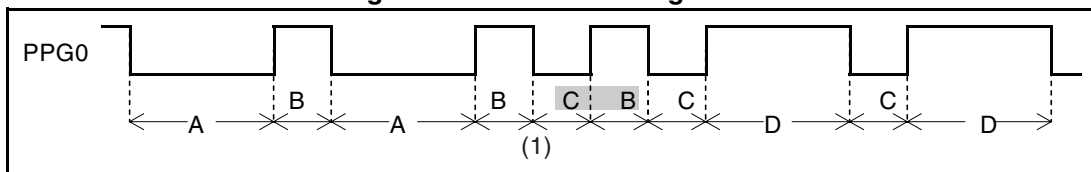


14.4.5 Write Timing for the Reload Registers

In all modes except the 16-bit PPG 1 mode, it is recommended that a word transfer instruction be used to write to the reload registers PRL and PRLH. If a byte transfer instruction is used twice to write data in these registers, an output with an unpredictable pulse width may result, depending on the write timing.

■ Write Timing for the Reload Registers

Figure 14.4-3 Write Timing Chart

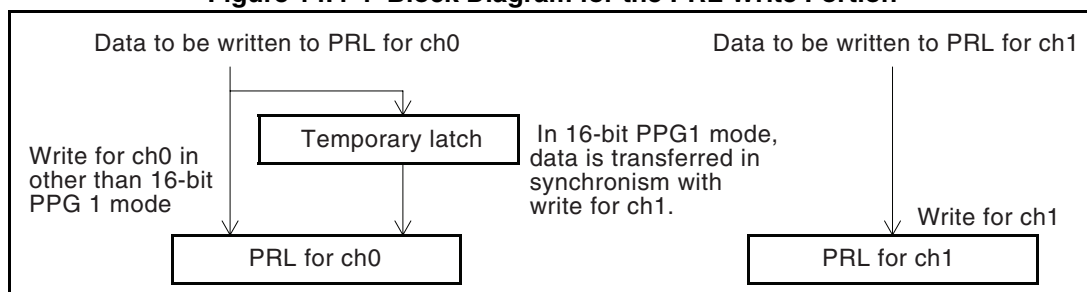


In the above timing chart, suppose that the PRL content is rewritten from A to C before (1), and that after (1), the PRLH content is rewritten from B to D. In such a case, a low for count C and a high for count B are output only once because at point (1), the PRL values include C in PRL and B in PRLH.

Similarly, in the 16-bit PPG mode, perform long-word transfer to write data in PRL for ch0 and ch1, or perform word transfer to write data in PRL in order from ch0 to ch1. In this mode, a write to PRL for ch0 is performed temporarily. After a write to PRL for ch1 has been performed, a write to PRL for ch0 is performed.

In modes other than the 16-bit PPG mode, a write to PRL for ch0 and ch1 can be performed separately.

Figure 14.4-4 Block Diagram for the PRL Write Portion



CHAPTER 15 DTP/EXTERNAL INTERRUPT CIRCUIT

This chapter describes the function and operation of the DTP/external interrupt circuit.

- 15.1 "Overview of the DTP/External Interrupt Circuit"
- 15.2 "Registers in the DTP/External Interrupt Circuit"
- 15.3 "Operation of DTP/External Interrupt Circuit"
- 15.4 "Notes on Using the DTP/External Interrupt Circuit"

15.1 Overview of the DTP/External Interrupt Circuit

The data transfer peripheral (DTP)/external interrupt circuit is placed between peripheral devices and the F²MC-16LX CPU. The DTP/external interrupt circuit receives DMA requests or interrupt requests issued from external peripheral devices and posts these requests to the F²MC-16LX CPU to initiate extended intelligent I/O service or interrupt processing. For extended intelligent I/O service, two request levels H and L are selectable. For external interrupt requests, four request levels including H, L, a rising edge, and a falling edge are selectable.

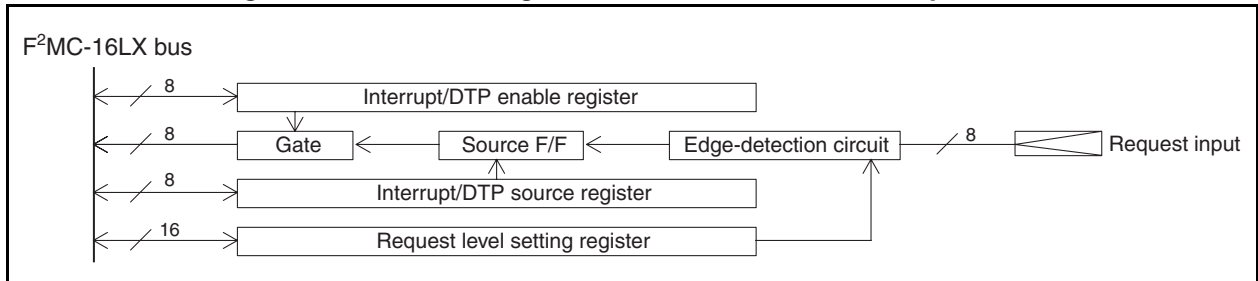
■ Registers in the DTP/External Interrupt Circuit

Figure 15.1-1 Registers in the DTP/External Interrupt Circuit

Interrupt/DTP enable register									⇐ Bit No.	
	7	6	5	4	3	2	1	0		
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)		
Interrupt/DTP source register									⇐ Bit No.	
	15	14	13	12	11	10	9	8		
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)		
Low-order byte of the request level setting register									⇐ Bit No.	
	7	6	5	4	3	2	1	0		
Address:000032H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	ELVR (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)		
High-order byte of the request level setting register									⇐ Bit No.	
	15	14	13	12	11	10	9	8		
Address:000033H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	ELVR (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)		

■ Block diagram of the DTP/external interrupt circuit

Figure 15.1-2 Block Diagram of the DTP/External Interrupt Circuit



15.2 Registers in the DTP/External Interrupt Circuit

The ENIR register determines whether to use device pins as DTP/external interrupt request inputs to initiate the function of issuing a request to the interrupt controller.

■ DTP/External Interrupt Enable Register (ENIR)

Figure 15.2-1 DTP/External Interrupt Enable Register (ENIR)

DTP/external interrupt enable register									↔ Bit No.
Address:000030H	7	6	5	4	3	2	1	0	ENIR
	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Read/write ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Initial value ⇒									

When a bit in the DTP/external interrupt enable register (ENIR) is set to 1, the corresponding pin is used as an DTP/external interrupt request input to initiate the function of issuing a request to the interrupt controller. For a pin corresponding to a bit set to 0, an DTP/external interrupt request input source is held, but no request is issued to the interrupt controller.

■ DTP/External Interrupt Source Register (EIRR)

Figure 15.2-2 DTP/External Interrupt Source Register (EIRR)

DTP/external interrupt source register									↔ Bit No.
	15	14	13	12	11	10	9	8	
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)	

When the EIRR register is read from, it indicates that a corresponding DTP/external interrupt request is present. When the register is written to, the flip-flop content indicating the request is cleared. When 1 is read from a bit in this register, it indicates that an DTP/external interrupt request is present on the pin corresponding to the bit.

When 0 is written to this register, the request flip-flop of the corresponding bit is cleared. Writing 1 to this register causes no operation. At read in read-modify-write operation, 1 is read.

Note:

If more than one external interrupt request output is enabled (EN7 to EN0 of ENIR are set to 1), clear to 0 only the bit for which the CPU accepted an interrupt (any of bits EN7 to EN0 that are set to 1). Do not clear the other bits without a valid reason.

■ Request Level Setting Register (ELVR)

Figure 15.2-3 Request Level Setting Register (ELVR)

Low-order byte of the request level setting register								⇐ Bit No.	
	7	6	5	4	3	2	1	0	
Address:000032H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	ELVR (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)	
High-order byte of the request level setting register								⇐ Bit No.	
	15	14	13	12	11	10	9	8	
Address:000033H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	ELVR (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)	

The ELVR register is used to select request detection. Two bits are assigned to each pin and correspond to each other as listed in Table 15.2-1 "Operation of the Request Level Setting Register (ELVR)". If a request must be detected according to the level, the value of the register is kept even if the register is cleared while the input is active.

Table 15.2-1 Operation of the Request Level Setting Register (ELVR)

LBx	LAx	Operation
0	0	Request detected by low level
0	1	Request detected by high level
1	0	Request detected by rising edge
1	1	Request detected by falling edge

15.3 Operation of DTP/External Interrupt Circuit

If the request set in the ELVR register is input to the corresponding pin after an external interrupt request is set, the external interrupt issues an interrupt request signal to the interrupt controller.

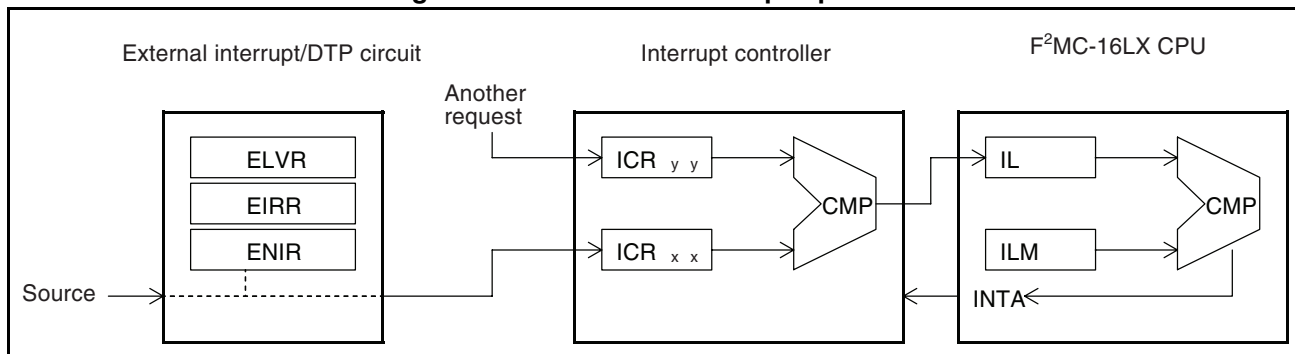
The DTP operation sequence is almost the same as external interrupt operation. In particular, the operation steps until the CPU starts the hardware interrupt processing microprogram are exactly the same.

■ External Interrupt Operation

If the request set in the ELVR register is input to the corresponding pin after an external interrupt request is set, the external interrupt issues an interrupt request signal to the interrupt controller. When the priority of the interrupts generated concurrently is identified in the interrupt controller and the interrupt from this resource has the highest priority, the interrupt controller issues an interrupt request to F²MC-16LX CPU.

The F²MC-16LX CPU compares the ILM bit of the CCR register in the CPU with the interrupt request. If the request level is higher than the ILM bit setting, the CPU starts the hardware interrupt processing microprogram when the currently executed instruction terminates.

Figure 15.3-1 External Interrupt Operation



With the hardware interrupt processing microprogram, the CPU reads ISE bit information from the interrupt controller to confirm that the request is a request for interrupt processing, then passes control to the interrupt processing microprogram. The interrupt processing microprogram reads the interrupt vector area, issues an interrupt acknowledge signal to the interrupt controller, generates a jump destination address of a macro instruction from a vector, transfers the address to the program counter, and then executes a user-defined interrupt processing program.

■ DTP Operation

When the extended intelligent I/O service is activated, the user program first sets the address of a register allocated in the range from 000000_H to 0000FF_H in the I/O address pointer in the extended intelligent I/O service descriptor. The user program then sets the start address of the memory buffer in the buffer address pointer.

The DTP operation sequence is almost the same as external interrupt operation. In particular, the operation steps until the CPU starts the hardware interrupt processing microprogram are exactly the same. For DTP, the content of the ISE bit the CPU reads within the hardware

interrupt processing microprogram indicates DTP, so control is passed to the microprogram for processing extended intelligent I/O service. When the extended intelligent I/O service is initiated, a read or write signal is sent to an addressed external peripheral device to perform a transfer with this chip. The external peripheral device must cancel the interrupt request issued for this chip within three machine cycles after the start of the transfer. When the transfer terminates, an operation such as descriptor update is performed. The interrupt controller then generates a signal for clearing the source of the transfer. When receiving the signal for clearing the transfer source, this resource clears the flip-flop that holds the source and is made ready for another request from a pin.

Refer to the MB90500 programming manual for details on extended intelligent I/O service processing.

Figure 15.3-2 Timing for Canceling an External Interrupt request when DTP Operation Terminates

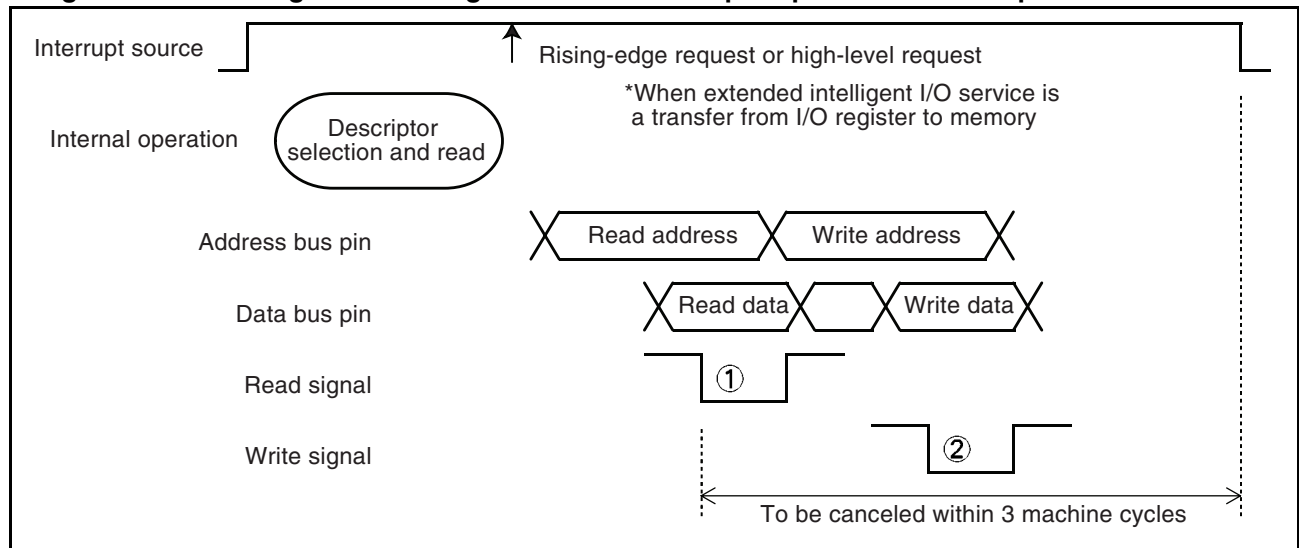
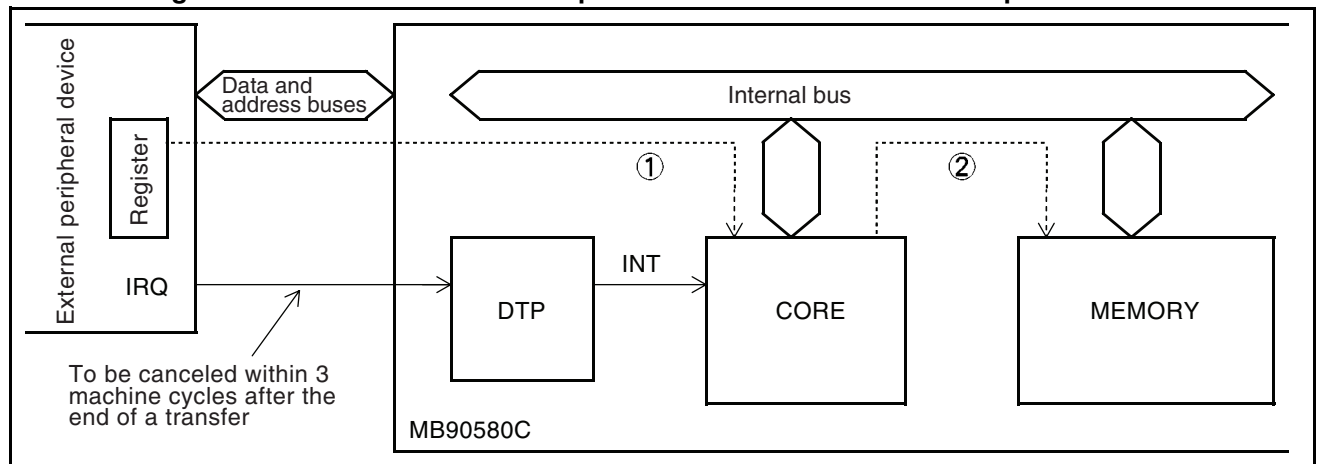


Figure 15.3-3 Schematic of a Sample Interface with an External Peripheral Device

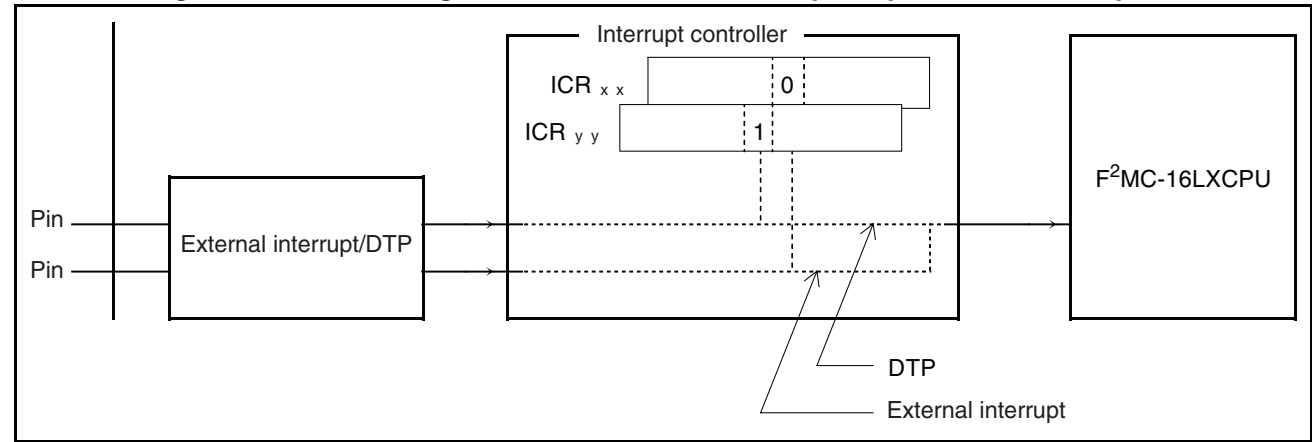


■ Switching between an external interrupt request and a DTP request

Switching between an external interrupt request and a DTP request is performed by setting the ISE bit of an ICR register for this resource. ICR registers are in the interrupt controller.

A separate ICR register is assigned to each pin. When the ISE bit of the ICR for a pin is set to 1, the pin functions as a DTP request. When the ISE bit is set to 0, the pin functions as an external interrupt request.

Figure 15.3-4 Switching between an External Interrupt Request and DTP Request



15.4 Notes on Using the DTP/External Interrupt Circuit

When using the DTP/external interrupt circuit, special care must be taken regarding the following four points:

- Conditions of peripheral devices connected externally when DTP is used
 - Return from the standby state
 - DTP/external interrupt circuit operation procedure
 - External interrupt request level
-

■ Conditions of Peripheral Devices Connected Externally when DTP is Used

External peripheral devices that DTP can support must automatically clear the request when transfer is performed. In addition, unless a peripheral device can cancel its transfer request within three machine cycles after the start of transfer operation, this resource assumes that another transfer request has occurred.

■ Return from the Standby State

When using an external interrupt to perform a return from the standby state of the clock stopped mode, use a H level request as the input request. Using a L level request may cause a malfunction. Using an edge request does not cause a return from the standby state of the clock stopped mode.

■ DTP/External Interrupt Circuit Operation Procedure

When setting registers in the DTP/external interrupt circuit, proceed as follows.

1. Disable a target bit in the enable register.
2. Set target bits in the request level setting register.
3. Clear a target bit in the source register.
4. Enable the target bit in the enable register. (Note that in steps 3 and 4, a word may be written to the register at a time.)

Before setting registers in this resource, always disable the enable register. Also, before enabling the enable register, always clear the source register to prevent an interrupt source from being generated by mistake during register setting or in the interrupt enable state.

■ External Interrupt Request Level

- For an edge-detected request, the pulse width must be at least three machine cycles to detect the input of an edge.
- For a level-detected request input, even if an external request is input and is canceled later, the request to the interrupt controller is kept active while the interrupt request is enable (ENIR : EN=1).

To cancel the request to the interrupt controller, clear the interrupt request flag bit (EIRR : ER).

Figure 15.4-1 Clearing the Interrupt Request Flag Bit (EIRR : ER) during Level Setting

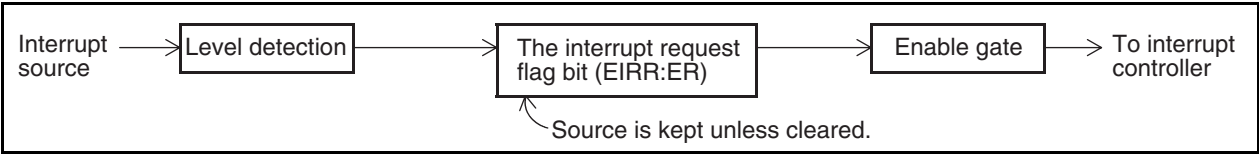
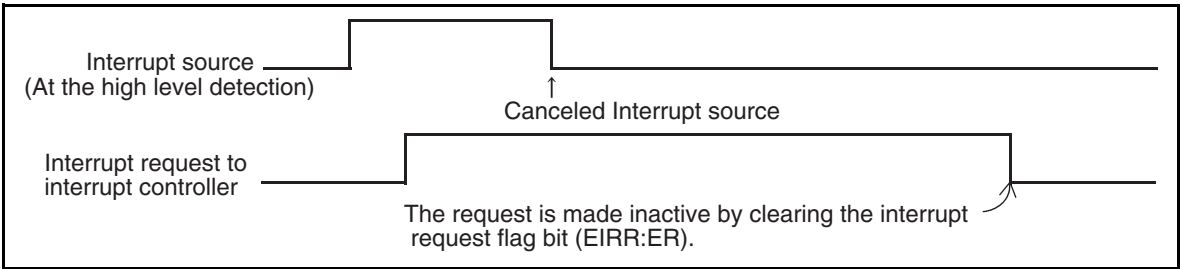


Figure 15.4-2 Interrupt Source and Interrupt Request to the Interrupt Controller when an Interrupt is Enabled



CHAPTER 16 DELAYED INTERRUPT GENERATING MODULE

This chapter describes the function and operation of the delayed interrupt generating module.

16.1 "Overview of the Delayed Interrupt Generating Module"

16.2 "Operation of the Delayed Interrupt Generating Module"

16.1 Overview of the Delayed Interrupt Generating Module

The delayed interrupt generating module generates an interrupt for task switching. With this module, an interrupt request to the F²MC-16LX CPU can be generated and canceled by software.

■ Register in the Delayed Interrupt Generating Module (DIRR)

The delayed interrupt source generation/cancel register (DIRR) controls generation and cancellation of a delayed interrupt request. Writing 1 to this register generates a delayed interrupt request, and writing 0 cancels a delayed interrupt request.

After a reset, the register is in the source canceled state.

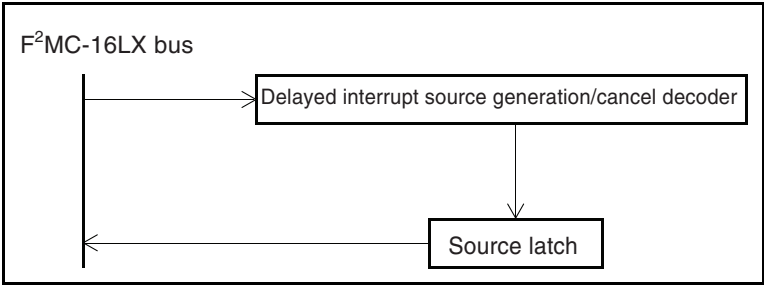
Either 0 or 1 may be written to the unused bit area. For future expansion, it is recommended that the set bit and clear bit instructions be used to access this register.

Figure 16.1-1 Delayed Interrupt Source Generation/Cancel Register (DIRR)

Delayed interrupt source generation/cancel register								⇐ Bit No.	
	15	14	13	12	11	10	9	8	
Address:00009FH	—	—	—	—	—	—	—	R0	DIRR
Read/write ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

■ Block Diagram of the Delayed Interrupt Generating Module

Figure 16.1-2 Block Diagram of the Delayed Interrupt Generating Module



16.2 Operation of the Delayed Interrupt Generating Module

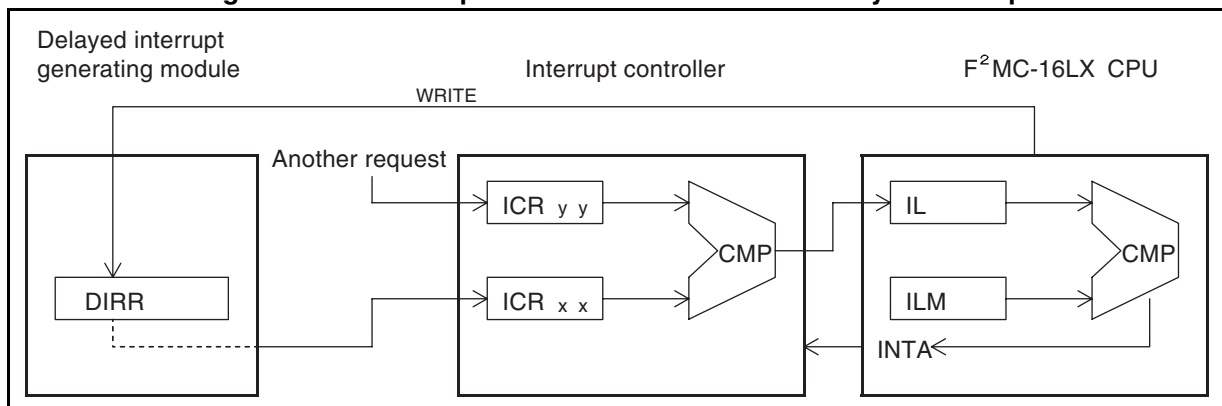
When software causes the CPU to write 1 to a bit of the DIRR register, the request latch in the delayed interrupt generating module is set, issuing an interrupt request to the interrupt controller.

When this interrupt has higher priority than the other interrupt requests, or when there is no other interrupt request, the interrupt controller issues the interrupt request to the F²MC-16LX CPU.

■ Operation of the Delayed Interrupt Generating Module

The F²MC-16LX CPU compares the ILM bit of the CCR register in the CPU with an interrupt request. If the request level is higher than the ILM bit setting, the CPU initiates the hardware interrupt processing microprogram when the currently executed instruction terminates. As a result, the interrupt processing routine for this interrupt is executed.

Figure 16.2-1 Description of the Generation of a Delayed Interrupt



When 0 is written to an appropriate bit of the DIRR register by the interrupt processing routine, the interrupt source is cleared, and task switching is performed at the same time.

■ Note on Use of the Delayed Interrupt Request Latch

The delayed interrupt request latch is set by writing 1 to an appropriate bit in the DIRR register, and the latch is cleared by writing 0 to the same bit. Therefore, software must be created so that a source can be cleared within the interrupt processing routine. Otherwise, when control is returned from interrupt processing, the interrupt processing is started again.

CHAPTER 17 A/D CONVERTER

This chapter describes the functions and provides an overview of the A/D converter.

17.1 "Overview of the A/D Converter"

17.2 "A/D Converter Block Diagram"

17.3 "Resisters of the A/D Converter"

17.4 "Operation of A/D Converter"

17.5 "Conversion Data Protection Function"

17.1 Overview of the A/D Converter

The A/D converter converts analog input voltage into a digital value.

■ Overview of the A/D Converter

The A/D converter has the following features:

- **Conversion time**

Minimum 34.7 μ s per channel (in machine cycle at 12 MHz)

- **Adoption of RC type successive approximation conversion format with a sample and hold circuit**

- **Resolution of 8 or 10 bits**

- **Analog input to be program-selected from eight channels**

- Single conversion mode: One channel is selected and converted.
- Scan conversion mode: The voltage of continuous multiple channels is converted. Up to eight channels can be programmed.
- Continuous conversion mode: The voltage of the specified channels is repeatedly converted.
- Pause conversion mode: After the voltage of one channel is converted, the converter pauses and waits for the next activation (enables synchronization with the start of conversion).

- **Interrupt request at completion of A/D conversion**

At completion of A/D conversion, an interrupt request of A/D conversion completion for the CPU can be generated. The generation of this interrupt enables the start of EI²OS and the transfer of A/D conversion result data to memory. Therefore, the A/D converter is suitable for continuous processing.

- **Selection of a startup cause from software, external trigger (falling edge), and timer (rising edge)**

■ Cautions on using the A/D converter

To start the A/D converter using an external trigger or internal timer, use A/D startup cause bits STS1 and STS0 in the ADCS2 register to set the startup. At this time, confirm that the value entered by the external trigger or internal timer is inactive. If it is active, the A/D converter may start immediately.

Set STS1 and STS0 when ADTG = 1 (input) and internal timer (timer 2) = 0 (output).

Always set the bit of the ADER register that corresponds to the pin used for analog input to 1.

Figure 17.1-1 Setting of pin used for analog input

Bit No.	15	14	13	12	11	10	9	8	
Address:00001C	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Initial value
Read/write ⇔	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	11111111 _B

Each pin of port 5 is controlled as follows:

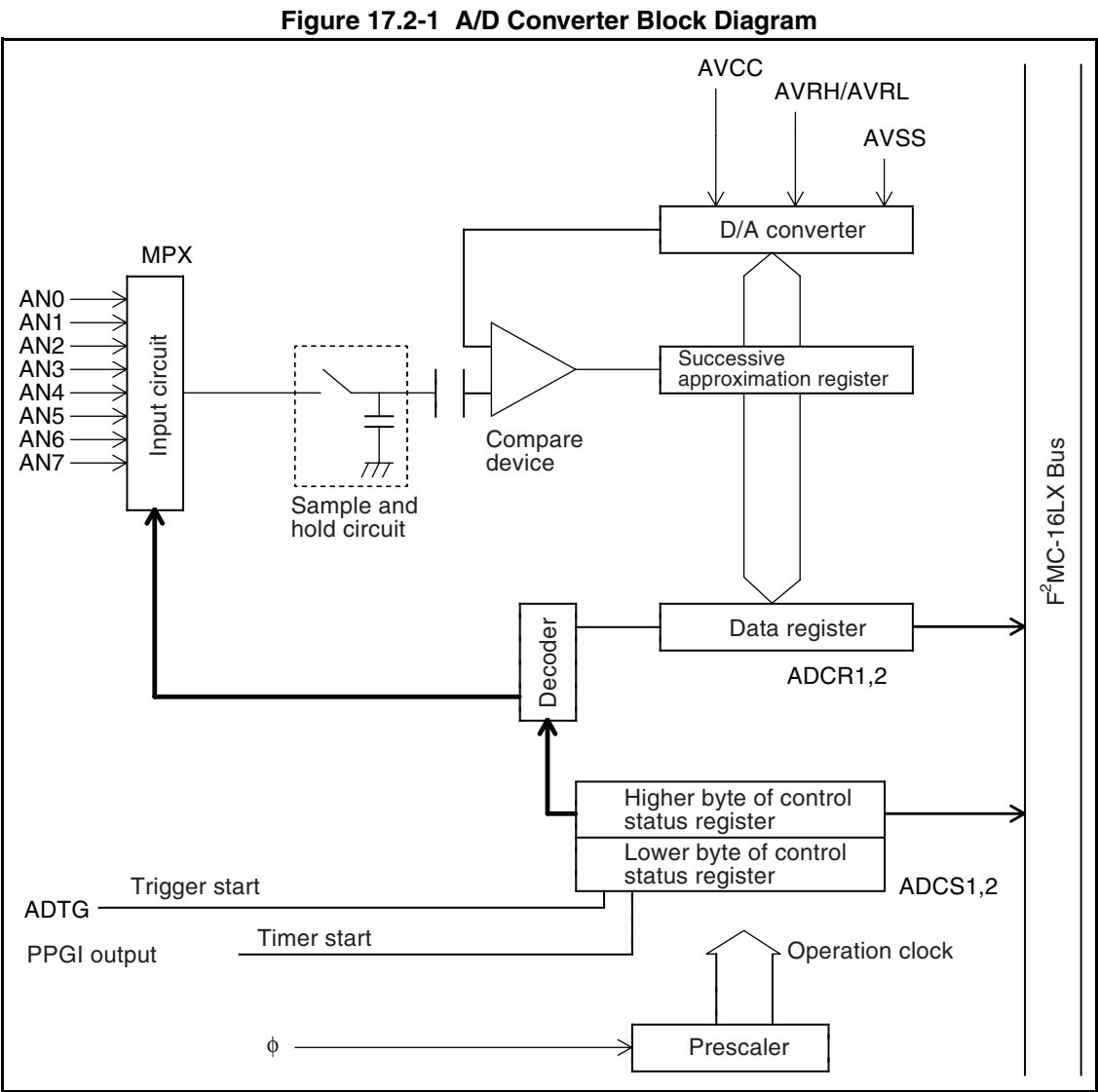
- 0: Port input mode
- 1: Analog input mode

The pin is set to 1 at reset.

17.2 A/D Converter Block Diagram

Figure 17.2-1 "A/D Converter Block Diagram" is the A/D converter block diagram.

■ A/D Converter Block Diagram



17.3 Registers of the A/D Converter

Figure 17.3-1 "Registers of the A/D Converter" shows the registers of the A/D converter.

■ Registers of the A/D Converter

Figure 17.3-1 Registers of the A/D Converter

Higher byte of the control status register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:000037H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS2
Read/write ⇐	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(-)	
Initial value ⇐	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Lower byte of the control status register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:000036H	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 byte of the data register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:000039H	SELB	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 byte of the data register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:000038H	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)	

17.3.1 Control Status Registers (ADCS1 and ADCS2)

The control status registers (ADCS1 and ADCS2) control the A/D converter and display its status.

■ Control Status Registers (ADCS1 and ADCS2)

Do not rewrite data to ADCS1 and ADCS2 during A/D conversion.

Figure 17.3-2 Control Status Registers (ADCS1 and ADCS2)

Higher byte of the control status register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:000037H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS2
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(-)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Lower byte of the control status register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:000036H	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 data to ADCS1 during A/D conversion.

[Bit 15] BUSY (Busy flag and stop)

○ **At read:**

This bit is used to indicate the operation of the A/D converter. This bit is set at activation of A/D conversion and cleared at termination.

○ **At write:**

If 0 is written to this bit during A/D operation, the A/D conversion is forced to be stopped during successive and pause mode.

1 cannot be written to the operation indication bit. In the read/modify/write (RMW) instruction, 1 is read. In single mode, this bit is cleared at termination of A/D conversion. In successive and pause modes, this bit is not cleared until the converter pauses by writing 0. This bit is initialized to 0 at reset.

Note:

Do not perform forcible stop and activation with software at the same time (when 0 is written to the BUSY bit, do not write 1 to the STRT bit).

[Bit 14] INT (Interrupt)

INT is used to display data and is set when the conversion data is written to ADCR.

If this bit is set when the Bit 5 (INTE) bit is 1, an interrupt request is generated. If the EI²OS activation is allowed, the EI²OS is activated. Writing 1 is meaningless.

This setting is cleared by writing 0 in this bit or with an EI²OS interrupt clear signal. This bit is initialized to 0 at reset.

Note:

When clearing this bit by writing 0, confirm that the A/D converter is not under operation.

[Bit 13] INTE (INTerrupt Enable)

The INTE bit specifies whether or not to allow an interrupt by terminating conversion.

Set this bit when using the EI²OS. The EI²OS is activated when an interrupt request is generated. This bit is initialized to 0 at reset.

Table 17.3-1 Functions of INTE (interrupt enable or disable specification bit)

INTE	Function
0	Interrupt prohibited [Initial value]
1	Interrupt allowed

[Bit 12] PAUS (a/d converter PAUSE)

This bit is set when conversion by the A/D is temporarily stopped.

Because there is only one register to store A/D conversion result, when successive conversion is performed, the previous data is destroyed unless it is transferred with the EI²OS.

To avoid this problem, new data cannot be stored until the contents of the data register is transferred with the EI²OS. During this transmission, conversion by the A/D pauses. The A/D converter resumes conversion upon completion of transmission with the EI²OS. This bit is initialized to 0 at reset.

Note:

This bit is valid only when the EI²OS is used. See the conversion data protection function in the operation description.

[Bits 11 and 10] STS1 and STS0 (Start Source select)

An A/D activation cause is selected by setting the STS1 and STS0 bits.

In mode in which multiple activation causes are selected, the A/D converter is activated by the first-selected cause. When switching the conversion activation causes during A/D operation, confirm that the target activation cause is not selected, because the activation cause is changed upon rewriting.

The external pin trigger detects a falling edge.

If this bit is rewritten and external pin trigger activation is selected when the external trigger input level is L, the A/D converter may be activated.

When the timer is selected, output of 16-bit reload timer 1 (PPG1) is selected.

Table 17.3-2 Functions of STS1 and STS0 (A/D Startup Cause Selecting Bits)

STS1	STS0	Function
0	0	Activation with software [Initial value]
0	1	Activation with an external pin trigger and software
1	0	Activation with a timer and software
1	1	Activation with an external pin trigger, timer, and software

[Bit 9] STRT (StaRT)

The A/D is activated by writing 1 to the STRT bit. For reactivation, write 1 again.

In pause mode, the A/D is not activated because of its operational function. This bit is initialized to 0 at reset.

The byte/word instructions read "1". The read-modify-write type instructions read "0".

Note:

Do not perform forcible stop and activation with software at the same time (when 1 is written to the STRT bit, do not write 0 to the BUSY bit).

[Bit 8] Reserved bit

Bit 8 is reserved. When setting ADCS1, always set 0.

[Bits 7 and 6] MD1 and MD0 (a/d converter MoDe set)

The MD1 and MD0 bits are used to set the operation mode of the A/D converter.

Table 17.3-3 Operation Modes of MD1 and MD0

MD1	MD0	Operation mode
0	0	Single mode, reactivation during operation is always possible [Initial value]
0	1	Single mode, reactivation during operation is not possible
1	0	Successive mode, reactivation during operation is not possible
1	1	Pause mode, reactivation during operation is not possible

○ Single mode

Performs A/D conversions successively from the setting channels between ANS2 and ANS0 to the setting channels between ANE2 and ANE0. The conversion pauses after each conversion.

○ Successive mode

Performs A/D conversions repeatedly from the setting channels between ANS2 and ANS0 to the setting channels between ANE2 and ANE0.

○ Pause mode

Performs A/D conversions by one channel from the setting channels between ANS2 and ANS0 to the setting channel between ANE2 and ANE0 and pauses. Conversion is resumed by the generation of an activation cause. These bits are initialized to 00 at reset.

Note:

- If A/D conversion is activated in successive or pause mode, the conversion is repeated until it is stopped by the BUSY bit.
- The conversion is stopped by writing 0 to the BUSY bit.
- Reactivation is not possible during single, successive, and pause modes and this is true for activation by timer, external trigger, or software.

[Bits 5, 4, and 3] ANS2, ANS1, and ANS0 (Analog Start channel set)

The ANS2, ANS1, and ANS0 bits set the start channel of A/D conversion.

When the A/D converter is activated, A/D conversion starts from the channel selected by this bit.

Table 17.3-4 Start Channel of the ANS2, ANS1, and ANS0 Bits

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

Note:

When this bit group is read, conversion channel numbers can be read during A/D conversion. While A/D conversion is stopped, however, the previously A/D converted channel numbers are read. Even if values are set in this register, the previously A/D converted values are read instead of the set values until A/D conversion is started. The values read by these bits are the previous conversion channel numbers until A/D conversion is started. These bits are initialized to 000_B at reset.

[Bits 2, 1, and 0] ANE2, ANE1, and ANE0 (ANalog End channel set)

The end channel of A/D conversion is set by ANE2, ANE1, and ANE0 bits.

Table 17.3-5 End Channel of ANE2, ANE1, and ANE0 bits

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

Note:

- Setting the same channel as ANE2 to ANE0 and ANS2 to ANS0 selects one-channel conversion (single conversion).
- In successive mode or pause mode, the conversion returns to the start channel specified by ANS2 to ANS0 at completion of conversion of the channels specified by ANE2 to ANE0.
- If the ANS value is smaller than the ANE value, conversion starts from the ANS channel. When channels up to channel 7 are converted, the conversion returns to channel 0 and

CHAPTER 17 A/D CONVERTER

channels up to ANE channel are converted.

- These bits are initialized to '000_B' at reset.

Example:

Channel setting ANS = 6ch and ANE = 3ch in single mode

Conversion is performed in the following sequence: 6ch to 7ch to 0ch to 1ch to 2ch to 3 ch.

17.3.2 Data Register (ADCR1 and ADCR2)

In the data register (ADCR1 and ADCR2), resolution is selected and machine cycle is set.

■ Data Registers (ADCR1 and ADCR2)

Figure 17.3-3 Data Registers (ADCR1 and ADCR2)

Higher byte of the data register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:000039H	SELB	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 byte of the data register	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:000038H	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)	

Note:

The value read by ADCR2 is undefined.

[Bit 15] SELB

The SELB bit is used to select a resolution of 8 or 10 bits.

Table 17.3-6 Functions of SELB

SELB	Resolution
0	10-bit
1	8-bit

[Bits 14 and 13] ST1 and ST0 (Sampling Time)

The ST1 and ST0 bits set the machine cycle number at sampling.

Table 17.3-7 ST1 and ST0 (Machine Cycle Setting Bits at Sampling)

ST1	ST0	Machine cycle at sampling	Sampling Time
0	0	64 machine cycle	4 μs/machine clock 16 MHz
0	1	Reserved	
1	0	Reserved	
1	1	4096 machine cycle	256 μs/machine clock 16 MHz

[Bits 12 and 11] CT1 and CT0 (Compare Time)

The CT1 and CT0 bits set the machine cycle number at compare.

Table 17.3-8 CT1 and CT0 (Machine Cycle Number Setting Bits at Compare)

CT1	CT0	Machine cycle at compare	Compare time
0	0	176 machine cycle	22 μ s/machine clock 8 MHz
0	1	352 machine cycle	22 μ s/machine clock 16 MHz
1	0	Reserved	
1	1	Reserved	

Note:

To set these bits to '00_B', set the machine clock to up to 8 MHz.

[Bits 9 to 0] D9 to D0

D9 to D0 are A/D conversion store registers and digital values of the conversion result are stored.

Values in this register are updated whenever a conversion is completed. Usually, the last value of conversion is stored. The registers support the conversion data protection function. See Section 17.4 "Operation of A/D Converter".

These registers are undefined at reset.

Note:

Do not write data to this register during A/D operation.

17.4 Operation of A/D Converter

The A/D converter is operated using a successive approximation method and has 8- or 10-bits resolution. Because the A/D converter has only one register to store conversion results (8- or 10-bits), the conversion data registers (ADCR0) are updated upon completing conversion. For this reason, the A/D converter alone is not suitable for successive conversion. Therefore, conversion by transferring conversion data to memory using the EI²OS function is recommended.

■ Single Mode

In single mode, the A/D converter sequentially converts analog outputs set by ANS and ANE bits and terminates operation when the conversion of the end channel set by the ANE bit is completed.

If the start channel and end channel are the same (ANS = ANE), only the channel specified by ANS is converted.

[Example]

ANS = 000_B, ANE = 011_B

Start --> AN0 --> AN1 --> AN2 --> AN3 --> End

ANS = 010_B, ANE = 010_B

Start --> AN2 --> End

■ Successive Mode

In successive mode, the A/D converter sequentially converts analog outputs set by ANS and ANE bits, returns to analog outputs by ANS, and continues operation when the conversion of the end channel set by the ANE bit is completed.

If the start channel and end channel are the same (ANS = ANE), one-channel conversion specified by ANS is repeated.

[Example]

ANS = 000_B, ANE = 011_B

Start --> AN0 --> AN1 --> AN2 --> AN3 --> AN0 --> Repeat

ANS = 010_B, ANE = 010_B

Start --> AN2 --> AN2 --> AN2 --> Repeat

Conversion in successive mode is continued until 0 is written to the BUSY bit. (Writing 0 to the BUSY bit, forced stop of operation)

Note that the conversion of data is not completed if the operation is stopped forcibly (In this case, the previous data whose conversion is completed is stored in the conversion register).

■ Pause Mode

In pause mode, the A/D converter sequentially converts analog inputs set by the ANS and ANE bits. However, its operation pauses upon conversion of one-channel. To release pausing, reactivate the converter.

When the conversion of an end channel set by the ANE bit is completed, the converter returns to the analog input by ANS and continues A/D conversion.

If the start channel and end channel are the same (ANS = ANE), the one-channels specified by ANS are converted.

[Example]

ANS = 000_B, ANE = 011_B

Start --> AN0 --> Stop --> Activate --> AN1 --> Stop --> Activate --> AN2 --> Stop --> Activate --> AN3 --> Stop --> Activate --> AN0 --> Repeat

ANS = 010_B, ANE = 010_B

Start --> AN2 --> Stop --> Activate --> AN2 --> Stop --> Activate --> AN2 --> Repeat

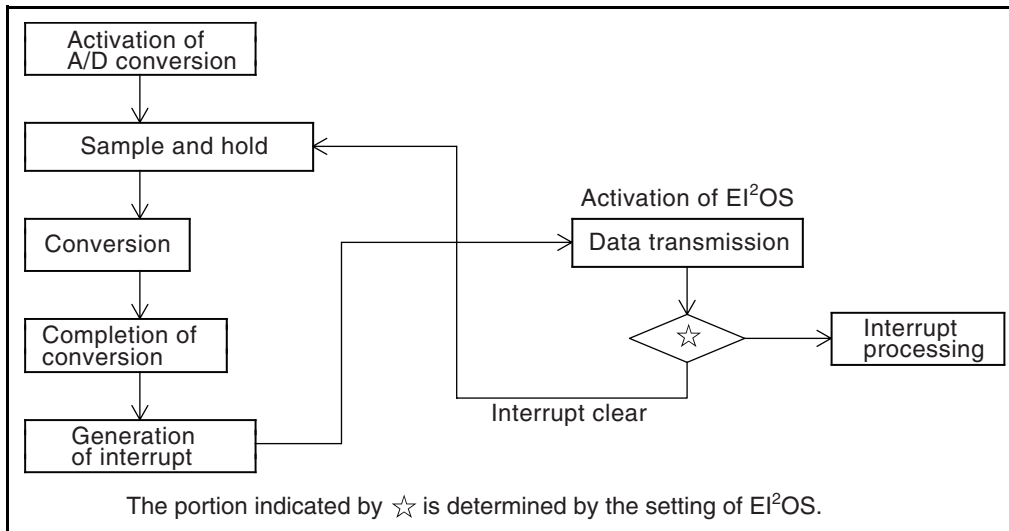
Only activation causes set by the STS1 and STS0 bits are used in this mode.

Using this mode, conversions can be started synchronously.

■ Conversion with the EI²OS

Figure 17.4-1 "Example of Flow from Activation of A/D Conversion to Conversion Data Transfer (Successive Mode)" shows an example of the flow from activation of A/D conversion to conversion data transfer (successive mode).

Figure 17.4-1 Example of Flow from Activation of A/D Conversion to Conversion Data Transfer (Successive Mode)



17.4.1 Example of EI²OS Activation in Single Mode

In single mode, the EI²OS is activated in the following procedure:

- Terminate after converting analog input (AN1 to AN3)
- Transfer conversion data to the addresses 200H to 205H sequentially
- Activate with software
- Maximum interrupt level

■ Example of EI²OS Activation in Single Mode

Table 17.4-1 Example of EI²OS Activation in Single Mode

Setting item	Example of programming	Description of operation
Setting of EI ² OS	MOV ICR00, #08 _H	Sets the maximum interrupt, activates EI ² OS at interrupt, and sets the descriptor address.
	MOV BAPL, #00 _H	Specifies the transfer destination address of conversion data.
	MOV BAPM, #02 _H	
	MOV BAPH, #00 _H	
	MOV ISCS, #18 _H	Specifies word data transfer. After transfer, increments the transfer destination address. Transfers data from I/O to memory and does not terminate transfer by a request from the resource.
	MOV IOA, #38 _H	Sets the transfer source address (result register of A/D converter).
	MOV DCT, #03 _H	Transfers data with EI ² OS three times. The number of data transfers is the same as the number of conversions.
Setting of A/D converter	MOV ADCS0, #0B _H	Specifies single mode, start channel AN1, and end channel AN3.
	MOV ADCS1, #A2 _H	Specifies activation with software and start of A/D conversion.
Interrupt sequence	RETI	Specifies return from the interrupt.

ICR00: Interrupt control register

BAPL: Low-order byte of the buffer address pointer

BAPM: Middle-order byte of the buffer address pointer

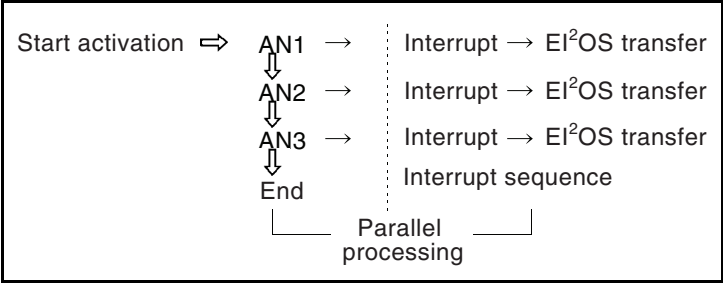
BAPH: high-order byte of the buffer address pointer

ISCS: EI²OS status register

IOA: I/O address register

DCT: Data counter

Figure 17.4-2 Example of EI²OS Activation in Single Mode



17.4.2 Example of EI²OS Activation in Successive Mode

In successive mode, the EI²OS is activated in the following manner:

- Obtain two pieces of conversion data for each channel by converting analog inputs (AN3 to AN5).
- Transfer conversion data to the addresses 600H to 60BH sequentially
- Start conversion with external edge input
- Use maximum interrupt level

■ Example of EI²OS Activation in Successive Mode

Table 17.4-2 Example of EI²OS Activation in Successive Mode

Setting item	Example of programming	Description of operation
Setting of EI ² OS	MOV ICR00, #08 _H	Sets the maximum interrupt, activates EI ² OS at interrupt, and sets the descriptor address.
	MOV BAPL, #00 _H	Specifies the transfer destination address of conversion data.
	MOV BAPM, #06 _H	
	MOV BAPH, #00 _H	
	MOV ISCS, #18 _H	Specifies word data transfer. After transfer, increments the transfer destination address. Transfers data from I/O to memory, and does not terminate transfer by a request from the resource.
	MOV I/OA, #38 _H	Sets the transfer source address (result register of A/D converter).
	MOV DCT, #06 _H	Transfers data with EI ² OS six times. Transfers data of 3 channels x 2 statements.
Setting of A/D converter	MOV ADCS0, #9D _H	Specifies successive mode, start channel AN3, and end channel AN5.
	MOV ADCS1, #A4 _H	Specifies activation with external edge and start of A/D conversion.
EI ² OS termination interrupt sequence	MOV ADCS1, #00 _H	Specifies return from the interrupt.
	RETI	-

ICR00: Interrupt control register

BAPL: Low-order byte of the buffer address pointer

BAPM: Middle-order byte of the buffer address pointer

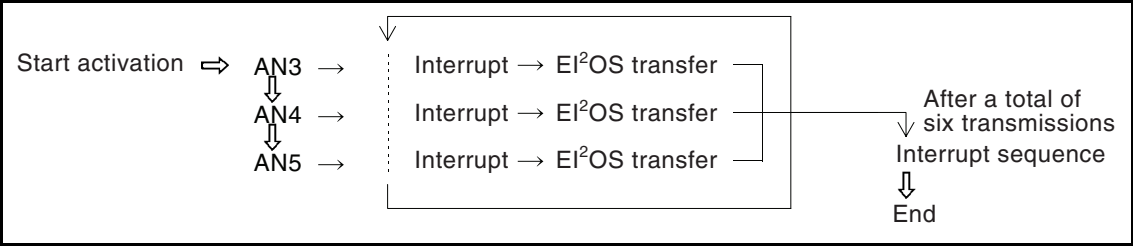
BAPH: high-order byte of the buffer address pointer

ISCS: EI²OS status register

IOA: I/O address register

DCT: Data counter

Figure 17.4-3 Example of EI²OS Activation in Successive Mode



17.4.3 Example of EI²OS Activation in Pause Mode

In pause mode, the EI²OS is activated in the following manner:

- Converts analog input (AN3) 12 times in a certain interval
- Transfer conversion data to the addresses 600H to 617H sequentially
- Start conversion with external edge input
- Use maximum interrupt level

■ Example of EI²OS Activation in Pause Mode

Table 17.4-3 Example of EI²OS Activation in Pause Mode

Items set	Example of programming	Description of operation
Setting of EI ² OS	MOV ICR00, #08 _H	Setting of maximum interrupt, activation of EI ² OS at interrupt, and setting of descriptor address
	MOV BAPL, #00 _H	Destination address of conversion data
	MOV BAPM, #06 _H	
	MOV BAPH, #00 _H	
	MOV ISCS, #08 _H	Transfers word data and increments destination address after transmission. Transfer from I/O to memory, and does not terminate by a request from a resource.
	MOV I/OA, #38 _H	Destination address
	MOV DCT, #0C _H	Transfers data 12 times with EI ² OS.
Setting of A/D converter	MOV ADCS0, #0B _H	Pause mode, start channel AN3, end channel AN3 (one-channel conversion)
	MOV ADCS1, #A4 _H	Activation with external edge, start of A/D converter
Interrupt sequence	MOV ADCS1, #00 _H	Recover from interrupt
	RETI	

ICR00: Interrupt control register

BAPL: Low-order byte of the buffer address pointer

BAPM: Middle-order byte of the buffer address pointer

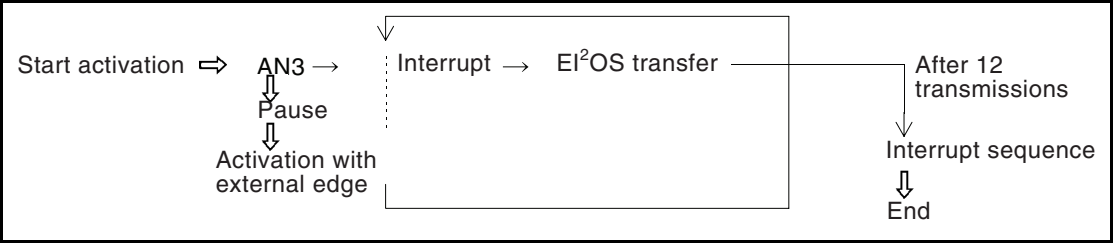
BAPH: high-order byte of the buffer address pointer

ISCS: EI²OS status register

IOA: I/O address register

DCT: Data counter

Figure 17.4-4 Example of EI²OS Activation in Pause Mode



17.5 Conversion Data Protection Function

This A/D converter has the conversion data protection function and features the ability to perform successive conversion using the EI²OS and to secure multiple pieces of data.

■ Conversion Data Protection Function

Because there is only one conversion data register, when successive A/D conversion is performed, the conversion data is stored upon completion of each conversion and the previous data is lost. To protect the previous data, this A/D converter pauses without storing new conversion data unless the previous data has been transferred to memory by the EI²OS.

The A/D converter is released from the pause when the previous data is transferred to memory by the EI²OS.

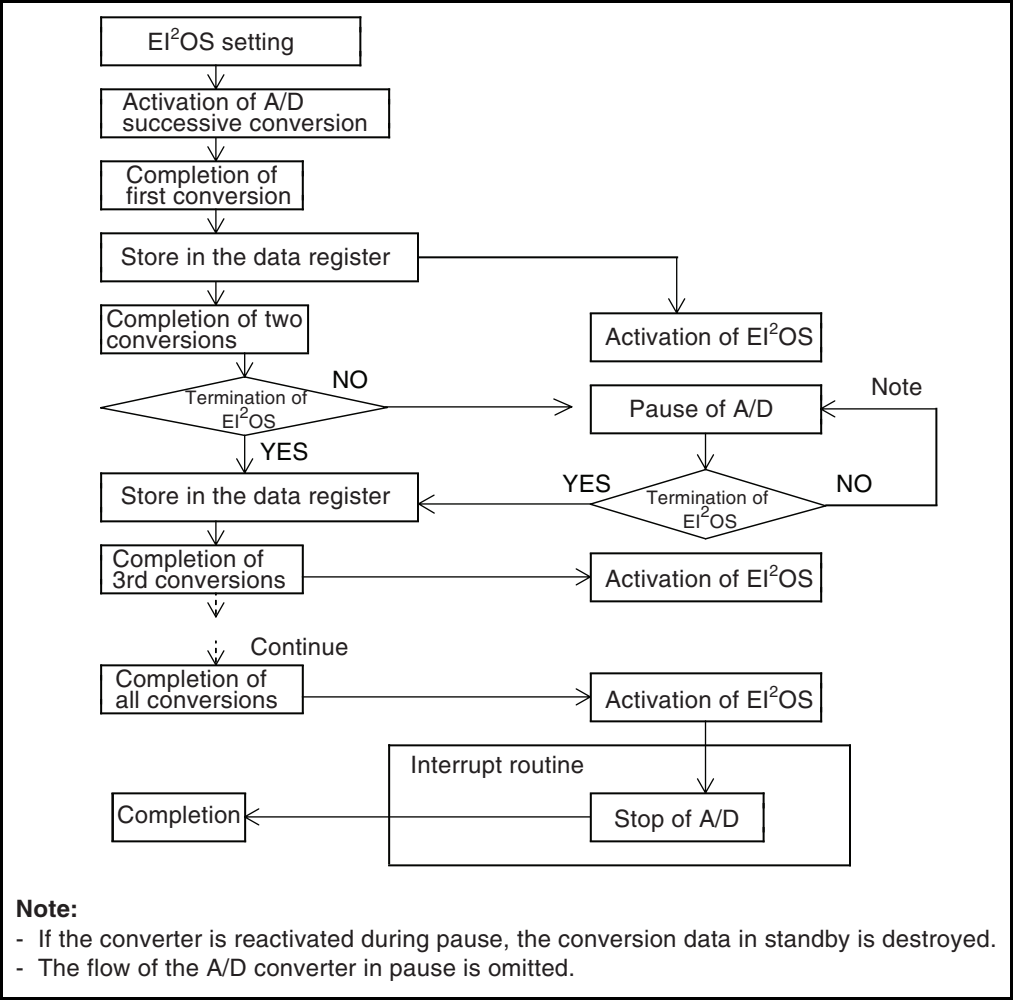
The A/D converter continues its operation without pause so long as the previous data is transferred to memory.

Note:

- This function is applied to the INT and INTE bits in the ADCS2 register.
- The data protection function works only when the interrupt is allowed (INTE = 1).
- This function does not work when the interrupt is not allowed (INTE = 0). Therefore, in successive A/D conversion, new conversion data is stored successively in the register, destroying previous data.
- Also, the INT bit is not cleared if the EI²OS is not used when the interrupt is allowed (INTE = 1). In this case, the data protection function works and the A/D suspends the conversion. The suspension is released by clearing the INT bit in the interrupt sequence.
- If the interrupt is disabled when the A/D converter pauses during EI²OS operation, the A/D converter is restarted and the contents of conversion data register may be changed before transfer.

Also, the standby data is destroyed if the A/D is restarted during a suspension (pause).

Figure 17.5-1 Data Protection Function Flow (when EI²OS is Used)



CHAPTER 18 D/A CONVERTER

This chapter explains the functions and operation of the D/A converter.

18.1 "Overview of D/A Converter"

18.2 "D/A Converter Registers"

18.3 "Operation of D/A Converter"

18.1 Overview of D/A Converter

This block is the R-2R type D/A converter with a resolution of 8 bits. The D/A converter has two channels.

Output control can be executed for two channels using the D/A control registers individually.

■ D/A converter registers

D/A converter registers are as follows.

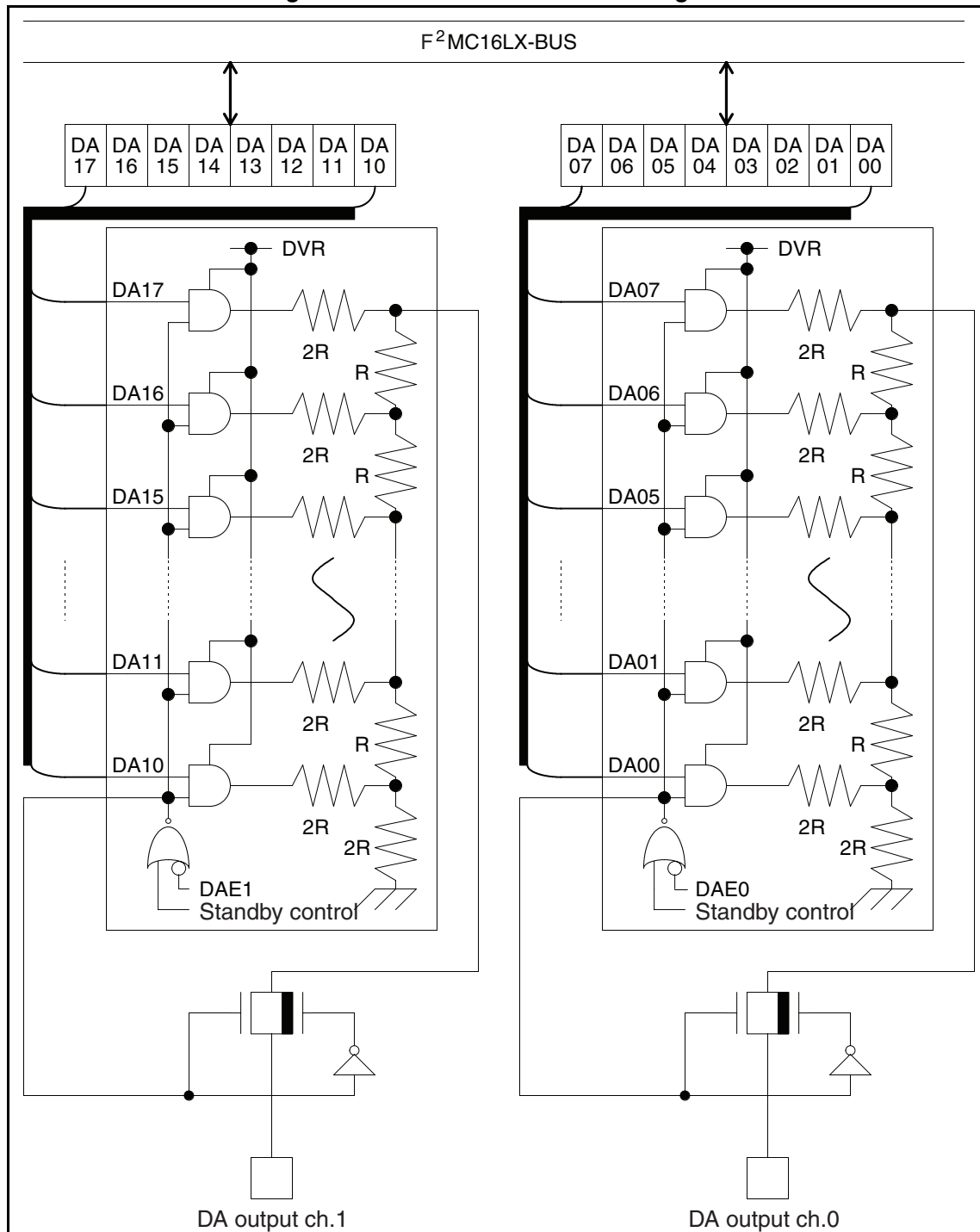
Figure 18.1-1 D/A converter registers

D/A converter data register 1								
Bit	15	14	13	12	11	10	9	8
Address:00003B _H	DA17	DA16	DA15	DA14	DA13	DA12	DA11	DA10
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)
DAT1								
D/A converter data register 0								
Bit	7	6	5	4	3	2	1	0
Address:00003A _H	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00
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)
DAT0								
D/A control register 1								
Bit	15	14	13	12	11	10	9	8
Address:00003D _H	-	-	-	-	-	-	-	DAE1
Read/write →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)
Initial value →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)
DACR1								
D/A control register 0								
Bit	7	6	5	4	3	2	1	0
Address:00003C _H	-	-	-	-	-	-	-	DAE0
Read/write →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)
Initial value →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)
DACR0								

■ D/A converter block diagram

Figure 18.1-2 "D/A converter block diagram" is the D/A converter block diagram.

Figure 18.1-2 D/A converter block diagram



18.2 D/A Converter Registers

The D/A converter has the following two types of registers:

- D/A converter data registers (DAT0 and DAT1)
- D/A control registers (DACR0 and DACR1)

■ D/A converter data registers (DAT0 and DAT1)

Figure 18.2-1 "D/A converter registers (DAT0 and DAT1)" shows the register configuration of D/A converter registers (DAT0 and DAT1).

Figure 18.2-1 D/A converter data registers (DAT0 and DAT1)

D/A converter data register 1								
Bit	15	14	13	12	11	10	9	8
Address:00003Bh	DA17	DA16	DA15	DA14	DA13	DA12	DA11	DA10
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)
DAT1								
D/A converter data register 0								
Bit	7	6	5	4	3	2	1	0
Address:00003Ah	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00
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)
DAT0								

[Bits 15 to 8] DA17 to DA10

These bits are used to set the output voltage of D/A converter channel 1.

These bits are not initialized at reset. These bits can be read and written.

[Bits 7 to 0] DA07 to DA00

These bits are used to set the output voltage of D/A converter channel 0.

These bits are not initialized at reset. These bits can be read and written.

■ D/A control registers (DACR0 and DACR1)

Figure 18.2-2 "D/A control registers (DACR0 and DACR1)" shows the register configuration of D/A control registers (DACR0 and DACR1).

Figure 18.2-2 D/A control registers (DACR0 and DACR1)

D/A control register 1								
Bit	15	14	13	12	11	10	9	8
Address:00003D _H	-	-	-	-	-	-	-	DAE1
Read/write →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)
Initial value →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)
D/A control register 0								
Bit	7	6	5	4	3	2	1	0
Address:00003C _H	-	-	-	-	-	-	-	DAE0
Read/write →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)
Initial value →	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)

[Bits 8 and 0] DAE1 and DAE0

These bits are used to specify whether D/A converter output is enabled or disabled. DAE1 controls channel 1 and DAE0 controls channel 0.

Writing 1 to these bits enables D/A output. Setting 0 disables D/A output.

These bits are initialized to 0 at reset. These bits can be read and written.

18.3 Operation of D/A Converter

To start D/A output, set 1 in the enable bit for the corresponding D/A output channel belonging to the D/A control register (DACR).

■ Operation of D/A converter

If D/A output is disabled, the analog switch inserted to the output of each D/A converter channel in series is turned off. In the D/A converter, the bit is cleared to 0 and the direct-current path is shut off. The above is also true in the stop mode.

The output voltage of the D/A converter ranges from 0 V to $255/256 \times \text{DVR}$. To change the output voltage range, adjust the DVR voltage externally.

The D/A converter output does not have the internal buffer amplifier. The analog switch ($= 100 \Omega$) is inserted to the output in series. To apply load to the output externally, estimate a sufficient stabilising time.

Table 18.3-1 "Theoretical values of output voltage of the D/A converter" lists the theoretical values of output voltage of the D/A converter.

Table 18.3-1 Theoretical values of output voltage of the D/A converter

Value written to DA07 to DA00 and DA17 to DA10	Theoretical value of output voltage
00 _H	$0/256 \times \text{DVR} (=0 \text{ V})$
01 _H	$1/256 \times \text{DVR}$
02 _H	$2/256 \times \text{DVR}$
:	:
FD _H	$253/256 \times \text{DVR}$
FE _H	$254/256 \times \text{DVR}$
FF _H	$255/256 \times \text{DVR}$

CHAPTER 19 COMMUNICATION PRESCALER REGISTER

This chapter describes the functions and overview of the communication prescaler register.

The output of the communication prescaler is used by UART.

19.1 "Overview of Communication Prescaler Register"

19.2 "Operation of Communication Prescaler Register"

19.1 Overview of Communication Prescaler Register

The communication prescaler register (clock division control register) controls the machine clock division. It is designed to assure a constant baud rate for various machine clocks as specified by the user.

The output of the communication prescaler is used by the UART.

■ Clock division control register (CDCR)

Figure 19.1-1 Clock division control registers [0 to 4]

Clock division control registers 0 to 4									
Address:00002CH	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:00002EH	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	CDCR0
Address:000034H									CDCR1
Address:000087H									CDCR2
Address:00008FH									CDCR3
Read/write ⇒	(R/W)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	CDCR4
Initial value ⇒	(0)	(-)	(-)	(-)	(1)	(1)	(1)	(1)	

[Bit 15] MD (Machine clock divide moDe select)

MD is an operation enable bit of the communication prescaler.

Table 19.1-1 Function of MD (Machine clock divide moDe select) Bit

MD	Function
0	Communication prescaler stops. [Initial value]
1	Communication prescaler operates.

[Bits 11, 10, 9, 8] DIV3 to DIV 0 (DIVide 3 to 0)

DIV3 to DIV0 decide a machine clock dividing ratio.

Table 19.1-2 Function of DIV3 to DIV0 Bits

DIV3	DIV2	DIV1	DIV0	Dividing ratio
1	1	1	1	Do not use [Initial value]
1	1	1	0	Divide by 2
1	1	0	1	Divide by 3
1	1	0	0	Divide by 4
1	0	1	1	Divide by 5
1	0	1	0	Divide by 6
1	0	0	1	Divide by 7
1	0	0	0	Divide by 8

Note:

- In actual use, set the above bits to something other than 1111_B.
- When changing the dividing ratio, wait for two cycles as the clock stabilizing time before starting communication.

Table 19.1-3 Correspondence between Communication Prescaler and Channels in UART

Communication prescaler setting	Communication prescaler output
CDCR 0	UART 0
CDCR 1	UART 1
CDCR 2	UART 2
CDCR 3	UART 3
CDCR 4	UART 4

19.2 Operation of Communication Prescaler Register

Set the clock division control register as listed in Table 19.2-1 "Operation of Communication Prescaler" depending on the machine clock Φ used. See Chapter 20 "UART" for details.

■ Operation of communication prescaler

Table 19.2-1 Operation of Communication Prescaler

Machine clock ϕ	div	DIV3	DIV2	DIV1	DIV0	ϕ/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
12MHz	3	1	1	0	1	
16MHz	4	1	1	0	0	

Confirm that ϕ divided by div does not exceed 4.25 MHz if setting a machine clock and div different than the above.

CHAPTER 20 UART

This chapter describes the UART functions and operations.

20.1 "Overview of UART"

20.2 "UART Block Diagram"

20.3 "UART Registers"

20.4 "UART Operations"

20.5 "Application of UART (During Operation in Mode 1)"

20.1 Overview of UART

The UART is a serial I/O port for asynchronous (start-stop synchronous) communication or CLK-synchronous communication.

■ Features of UART

The UART has the following features:

- Full duplex double buffer
- Asynchronous (start-stop synchronous) communication and CLK-synchronous communication
- Support of multiprocessor
- Built-in dedicated baud rate generator

Table 20.1-1 Baud Rate

Operation	Baud rate (*)
Asynchronous	31250/9615/4808/2404/1202 bps
CLK-synchronous	2 M/1 M/500 K/250 K/125 K/62.5 Kbps

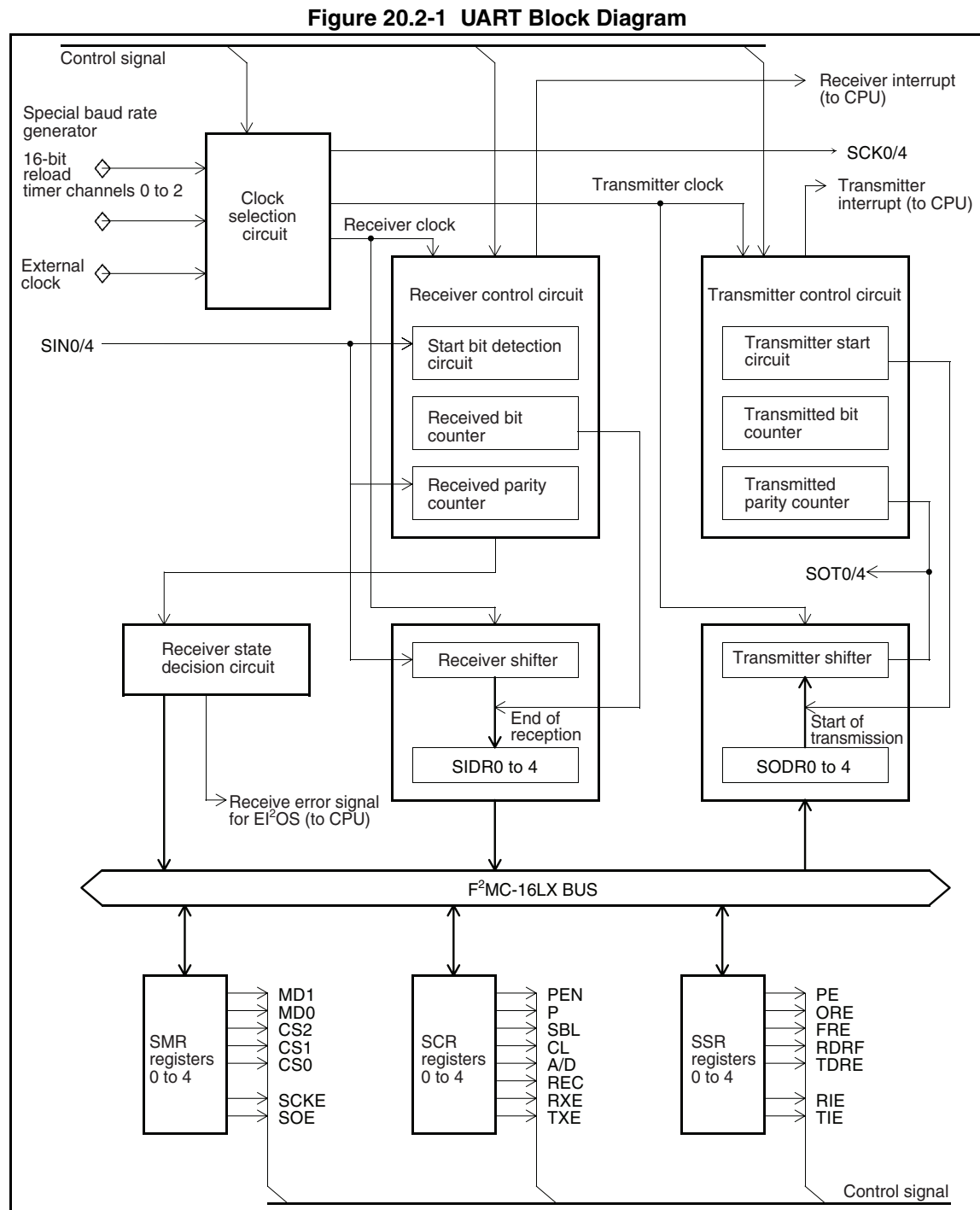
*: Values when internal machine clocks are 6, 8, 10, 12 and 16 MHz.

- Free baud rate setting by external clocks
- Error detection function (parity, framing, and overrun)
- Transfer signal of HRz code

20.2 UART Block Diagram

Figure 20.2-1 "UART Block Diagram" shows a UART block diagram.

■ UART Block Diagram



20.3 UART Registers

The following four types of UART registers are available:

- Serial mode register
- Serial control register
- Serial input register/serial output register
- Serial status register

■ UART Registers

Figure 20.3-1 UART Registers

Serial mode register									
Address:000020H									
Address:000024H									
Address:000028H									
Address:000082H									
Address:000088H									
	7	6	5	4	3	2	1	0	⇐ Bit No.
	MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE	SMR0 to 4
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)	
Serial control register									
Address:000021H									
Address:000025H									
Address:000029H									
Address:000083H									
Address:000089H									
	15	14	13	12	11	10	9	8	⇐ Bit No.
	PEN	P	SBL	CL	A/D	REC	RXE	TXE	SCR0 to 4
Read/write ⇐	(R)	(R)	(R)	(R)	(R)	(R/W)	(R/W)	(R/W)	
Initial value ⇐	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	
Serial input register/serial output register									
Address:000022H									
Address:000026H									
Address:00002AH									
Address:000084H									
Address:00008AH									
	7	6	5	4	3	2	1	0	⇐ Bit No.
	D7	D6	D5	D4	D3	D2	D1	D0	SIDR0 to 4 (read) SODR0 to 4 (write)
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)	
Serial status register									
Address:000023H									
Address:000027H									
Address:00002BH									
Address:000085H									
Address:00008BH									
	15	14	13	12	11	10	9	8	⇐ Bit No.
	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE	SSR0 to 4
Read/write ⇐	(R)	(R)	(R)	(R)	(R)	(-)	(R/W)	(R/W)	
Initial value ⇐	(0)	(0)	(0)	(0)	(1)	(-)	(0)	(0)	

20.3.1 Serial Mode Register (SMR0 to 4)

The SMR0 to 4 register specifies a UART operating mode.

Set an operating mode when the register is stopping. Do not write anything to the register during operation.

■ Serial Mode Register (SMR0 to 4)

Figure 20.3-2 Configuration of Serial Mode Register (SMR)

Serial mode register								
Address:000020H								
Address:000024H								
Address:000028H								
Address:000082H								
Address:000088H								
	7	6	5	4	3	2	1	0 ⇐ Bit No.
	MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE
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)

[Bits 7 and 6] MD1 and MD0 (MoDe select)

MD1 and MD0 bits select a UART operating mode.

Table 20.3-1 MD0 and MD1 (Operating Mode Selecting Bits)

MD1	MD0	Mode	Operating mode
0	0	0	Asynchronous normal mode [Initial value]
0	1	1	Asynchronous multiprocessor mode
1	0	2	CLK-asynchronous mode
1	1	-	Do not use

Note:

The synchronous mode (multiprocessor) of mode 1 is a mode in which multiple slave CPUs are connected to one host CPU.

The peripherals are not capable of identifying a receiver data format. Therefore, only the master multiprocessor mode is supported.

Also, the parity check function is not available, so set the PEN of the SCR register to 0.

[Bits 5 to 3] CS2, CS1, and CS0 (Clock Select)

CS2 to CS0 bits select a baud rate clock source.

If the communication prescaler is selected, a baud rate is also determined simultaneously.

Table 20.3-2 CS0 to CS2 (Baud Rate Clock Source Selecting Bits)

CS2	CS1	CS0	Clock input
000 _B to 100 _B			Special baud rate generator
1	0	1	Reserved
1	1	0	Internal timer (16-bit reload timer 0)
1	1	1	External clock

Note:

If the internal timer is selected, the MB90580C series selects three 16-bit reload timers.

The UART channels and reload timer channels are as follows:

- UART 0ch: Reload timer 0ch
- UART 1ch: Reload timer 1ch
- UART 2ch: Reload timer 2ch
- UART 3ch: Reload timer 0ch
- UART 4ch: Reload timer 1ch

[Bit 2] Reserved

Bit 2 is a reserved bit. Set the bit to 0.

[Bit 1] SCKE (SCK Enable)

The SCKE bit specifies whether to use the SCK terminal as a clock input terminal or clock output terminal when communicating in the CLK-synchronous mode (mode 2).

Table 20.3-3 Function of SCKE (SCLK Enable) Bit

SCKE	Function
0	Uses the SCK terminal as a clock input terminal. [Initial value]
1	Uses the SCK terminal as a clock output terminal.

Note:

An external clock source must be used when the SCK terminal is used as a clock input terminal.

The UART channels correspond to the serial clock I/O terminals as follows:

- UART 0ch: SCK0 terminal
- UART 1ch: SCK1 terminal
- UART 2ch: SCK2 terminal
- UART 3ch: SCK3 terminal
- UART 4ch: SCK4 terminal

[Bit 0] SOE (Serial Output Enable)

The SOE bit specifies whether the external terminal is used as a serial output terminal (SOT) or an I/O port terminal.

Table 20.3-4 Function of SOE (Serial Output Enable) Bit

SOE	Function
0	Uses the external terminal as a general-purpose I/O port terminal. [Initial value]
1	Uses the external terminal as a serial data output terminal (SOT).

Note:

The UART channels correspond to the serial data output terminals as follows:

- UART 0ch: SOT0 terminal
- UART 1ch: SOT1 terminal
- UART 2ch: SOT2 terminal
- UART 3ch: SOT3 terminal
- UART 4ch: SOT4 terminal

20.3.2 Serial Control Register (SCR0 to 4)

The serial control register (SCR0 to 4) controls a transfer protocol for serial communication.

■ Serial Control Register (SCR0 to 4)

Figure 20.3-3 Configuration of Serial Control Register (SCR)

Serial control register								
Address:000021 _H								
Address:000025 _H								
Address:000029 _H								
Address:000083 _H								
Address:000089 _H								
	15	14	13	12	11	10	9	8 ⇐ Bit No.
	PEN	P	SBL	CL	A/D	REC	RXE	TXE
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)	(1)	(0)	(0)

[Bit 15] PEN (Parity Enable)

The PEN bit specifies whether to add a parity bit when establishing serial data communication.

Table 20.3-5 Function of PEN (Parity Enable) Bit

PEN	Function
0	No parity [Initial value]
1	Parity

Note:

A parity bit can be added only in the normal mode (mode 0) of asynchronous (start-stop synchronous) communication modes, not in multiprocessor mode (mode 1) and CLK-synchronous communication mode (mode 2).

[Bit 14] P (Parity)

The P bit specifies even or odd parity when adding a parity bit for data communication.

Table 20.3-6 P (Event/Odd Parity Specification Bit)

P	Function
0	Even parity [Initial value]
1	Odd parity

[Bit 13] SBL (Stop Bit Length)

The SBL bit specifies the length of a stop bit, which is a frame end mark used in asynchronous communication.

Table 20.3-7 SBL (Stop Bit Length Specification Bit)

SBL	Function
0	1 stop bit [Initial value]
1	2 stop bits

[Bit 12] CL (Character Length)

The CL bit specifies the data length of each frame to be transmitted or received.

Table 20.3-8 CL (Transmitter or Receiver Data Length Specification Bit)

CL	Function
0	7-bit data [Initial value]
1	8-bit data

Note:

7-bit data can only be processed in the normal synchronous communication mode (mode 0). In multiprocessor mode (mode 1) or CLK-synchronous communication mode (mode 2), specify 8-bit data.

[Bit 11] A/D (Address/Data)

The A/D bit specifies a data format of a frame to be transmitted in the multiprocessor mode (mode 1) of asynchronous communication modes.

Table 20.3-9 Function of A/D (Address/Data) Bit

A/D	Function
0	Data frame [Initial value]
1	Address frame

[Bit 10] REC (Receiver Error Clear)

Writing 0 to this bit clears the error flags (PE, ORE, and FRE) of the SSR register. Writing 1 is invalid. The read value is always 1.

[Bit 9] RXE (Receiver Enable)

The RXE bit controls a UART receive operation.

Table 20.3-10 Function of RXE (Receiver Enable) Bit

RXE	Function
0	Disables a receive operation. [Initial value]
1	Enables a receive operation.

Note:

If the RXE is set to 0 during the receive operation (while inputting data into the receiver shift register), the receive operation is stopped when the receiving of the frame is completed and the receiver data is stored in the receiver data buffer SDR register.

[Bit 8] TXE (Transmitter Enable)

The TXE bit controls a UART transmit operation.

Table 20.3-11 Function of TXE (Transmitter Enable) Bit

TXE	Function
0	Disables a transmit operation. [Initial value]
1	Enables a transmit operation.

Note:

If the TXE is set to 0 during the transmit operation (while outputting data from the transmit register), the transmit operation is stopped after all data is output from the transmitter data buffer SODR register.

20.3.3 Serial Input Data Register (SIDR0 to 4) and Serial Output Data Register (SODR0 to 4)

The serial input data register (SIDR0 to 4) and serial output data register (SODR0 to 4) are receiver and transmitter data buffer registers.

■ Configuration of Serial Input Data Register (SIDR0 to 4) and Serial Output Data Register (SODR0 to 4)

If SIDR or SODR data is 7 bits long, the high-order 1 bit (D7) becomes invalid. Write data into the SODR register when the TDRE of the SSR register is 1.

Figure 20.3-4 Configuration of Serial Input Data Register (SIDR0 to 4) and Serial Output Data Register (SODR0 to 4)

Serial input register/serial output register								
Address:000022H								
Address:000026H								
Address:00002AH	7	6	5	4	3	2	1	0
Address:000084H								
Address:00008AH								
	D7	D6	D5	D4	D3	D2	D1	D0
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)

⇨ Bit No.

SIDR0 to 4 (read)
SODR0 to 4 (write)

Note:

Writing data at this address means writing data into the SODR register; reading data at this address means reading data from the SIDR register.

20.3.4 Serial Status Register (SSR0 to 4)

The serial status register (SSR0 to 4) is comprised of flags that represent the UART operating status.

Serial Status Register (SSR0 to 4)

Figure 20.3-5 Configuration of Serial Status Register (SSR0 to 4)

Serial status register							
Address:000023H							
Address:000027H							
Address:00002BH							
Address:000085H							
Address:00008BH							
	15	14	13	12	11	10	9 8 ⇐ Bit No.
	PE	ORE	FRE	RDRF	TDRE	—	RIE TIE
	SSR0 to 4						
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)

The PE bit is an interrupt request flag that is set when a parity error occurs during the receive operation. To clear the flag set once, write 0 into the REC bit (bit 10) of the SCR register.

If the bit is set, data in the SISR register becomes invalid.

Table 20.3-12 Function of PE (Parity Error) Bit

PE	Function
0	No parity error [Initial value]
1	Parity error occurs.

[Bit 14] ORE (Over Run Error)

The ORE bit is an interrupt request flag that is set when an overrun error occurs during the receive operation. To clear the flag set once, write 0 into the REC bit (bit 10) of the SCR register.

If the bit is set, data in the SISR register becomes invalid.

Table 20.3-13 Function of ORE (Over Run Error)

ORE	Function
0	No overrun error [Initial value]
1	Overrun error occurs.

[Bit 13] FRE (FRaming Error)

The FRE bit is an interrupt request flag that is set when a framing error occurs during the receive operation. To clear the flag set once, write 0 into the REC bit (bit 10) of the SCR register. If the bit is set, data in the SISR register becomes invalid.

Table 20.3-14 Function of FRE (FRaming Error)

FRE	Function
0	No framing error [Initial value]
1	Framing error occurs.

[Bit 12] RDRF (Receiver Data Register Full)

The RDRF bit is an interrupt request flag indicating that the SDR register is full with receiver data. The bit is set when receiver data is loaded into the SDR register and automatically cleared when the data is read from the SDR register.

Table 20.3-15 Function of RDRF (Receiver Data Register Full)

SDR	Function
0	Register is not full with receiver data. [Initial value]
1	Register is full with receiver data.

[Bit 11] TDRE (Transmitter Data Register Empty)

The TDRE bit is an interrupt request flag indicating that transmitter data can be written into the SODR register. Once the data is written into the SODR register, the bit is cleared. When the written data is loaded into the transmitter shifter and transfer is started, the bit is set again, indicating the next transmitter data can be written.

Table 20.3-16 Function of TDRE (Transmitter Data Register Empty)

TDRE	Function
0	Transmitter data cannot be written.
1	Transmitter data can be written. [Initial value]

[Bit 10] Empty bit**[Bit 9] RIE (Receiver Interrupt Enable)**

The RIE bit controls a receiver interrupt.

Table 20.3-17 Function of RIE (Receiver Interrupt Enable)

RIE	Function
0	Disables an interrupt. [Initial value]
1	Enables an interrupt.

Note:

Receiver interrupt sources include the occurrence of a PF, ORE, or FRE error and normal reception by RDRF.

[Bit 8] TIE (Transmitter Interrupt Enable)

The TIE bit controls a transmitter interrupt.

Table 20.3-18 Function of TIE (Transmitter Interrupt Enable)

TIE	Function
0	Disables an interrupt. [Initial value]
1	Enables an interrupt.

Note:

Transmitter interrupt sources include a request to transmit by TDRE.

20.4 UART Operations

The UART has operating modes as shown in Table 20.4-1 "UART Operating Modes" that can be switched by setting a value into the SMR and SCR registers.

■ UART Operations

Table 20.4-1 UART Operating Modes

Mode	Parity	Data length	Operating mode	Stop bit length
0	Yes/No	7	Asynchronous normal mode	1 or 2 bits
	Yes/No	8		
1	No	8 + 1	Asynchronous (start-stop synchronous) multiprocessor mode	
2	No	8	CLK-synchronous mode	-

Note:

The stop bit length in asynchronous (start-stop synchronous) mode can be specified only for a transmit operation. A receive operation is always 1 bit long. The UART cannot operate in any mode other than the above. Do not attempt a setting.

In the CLK-synchronous mode, start and stop bits are added to the data byte.

Set a communication mode while the UART is stopping; otherwise, data received or transmitted during the mode setting cannot be guaranteed.

■ Extended Intelligent I/O Services (EI²OS)

For EI²OS, see Section 3.6 "Expanded Intelligent I/O Service (EI²OS)".

20.4.1 UART Clock Selection

The following three kinds of UART clocks can be selected:

- Special baud rate generator
- Internal timer
- External clock

■ Communication Prescaler

Table 20.4-2 "Baud Rates (Asynchronous (Start-Stop Synchronous) Mode)" and Table 20.4-3 "Baud Rates (CLK-synchronous Mode)" list the baud rates when the special baud rate generator is selected. See Chapter 19 "COMMUNICATION PRESCALER REGISTER" for details on the formula in the tables.

Table 20.4-2 Baud Rates (Asynchronous (Start-Stop Synchronous) Mode)

CS2	CS1	CS0	$\phi/\text{div} = 2 \text{ MHz}$	$\phi/\text{div} = 4 \text{ MHz}$	Expression for calculation
0	0	0	9615 bps	19230 bps	$(\phi/\text{div})/(8 \times 13 \times 2)$
0	0	1	4808 bps	9615 bps	$(\phi/\text{div})/(8 \times 13 \times 2^2)$
0	1	0	2404 bps	4808 bps	$(\phi/\text{div})/(8 \times 13 \times 2^3)$
0	1	1	1202 bps	2404 bps	$(\phi/\text{div})/(8 \times 13 \times 2^4)$
1	0	0	31250 bps	62500 bps	$(\phi/\text{div})/2^6$

ϕ : Machine clock div: Setting of communication prescaler

Table 20.4-3 Baud Rates (CLK-synchronous Mode)

CS2	CS1	CS0	$\phi/\text{div} = 2 \text{ MHz}$	$\phi/\text{div} = 4 \text{ MHz}$	Expression for calculation
0	0	0	1 Mbps	2 Mbps	$(\phi/\text{div})/2$
0	0	1	500 Kbps	1 Mbps	$(\phi/\text{div})/2^2$
0	1	0	250 Kbps	500 Kbps	$(\phi/\text{div})/2^3$
0	1	1	125 Kbps	250 Kbps	$(\phi/\text{div})/2^4$
1	0	0	62.5 Kbps	125 Kbps	$(\phi/\text{div})/2^5$

ϕ : Machine clock div: Setting of communication prescaler

■ Internal timer

If the internal timer is selected by setting CS2 to CS0 bits of the SMR register to 110_B, the 16-bit timer (timer 0 to 2) is operated in reload mode. The expressions for calculating a baud rate at this time are as follows:

Asynchronous (start-stop synchronous) $(\phi/N) / (16 \times 2 \times (n + 1))$

CLK-synchronous $(\phi/N) / (2 \times (n + 1))$

○ ϕ : **Machine clock**

ϕ indicates the internal operation frequency of the microcomputer. See Section 5.7 "Switching Machine Clock" for details.

○ **N**

N indicates the dividing ratio of the count clock source (TMCSR: CSL1 and CSL0) of the internal 16-bit reload timer.

Example: $N = 8$ when CSL1 and CSL0 = '01_B'

See Section 13.2.1 "Timer control status register (TMCSR)" for details.

○ **n**

n indicates the reload value of the internal 16-bit reload timer.

Table 20.4-4 "Baud Rates and Reload Values" lists the relationship between a baud rate and reload value (decimal) when the machine clock is taken as 7.3728 MHz.

Table 20.4-4 Baud Rates and Reload Values

Baud rate	Reload value	
	N = 2 (Machine clock divided by 2)	N = 2 ³ (Machine clock divided by 8)
38400	2	-
19200	5	-
9600	11	2
4800	23	5
2400	47	11
1200	95	23
600	191	47
300	383	95

If the internal timer (16-bit reload timers 0 to 2) is selected as a baud rate clock source, the outputs (TOT0 to TOT2) of 16-bit reload timers 0 to 2 have already been connected inside this controller. Therefore, external terminals TOT0 to TOT2 of 16-bit reload timers 0 to 2 do not have to be externally connected to external clock input terminals (SCK0 to SCK4) of UART. The output terminal of 16-bit reload timers 0 to 2 can also be used as an I/O port terminal unless otherwise used.

■ External Clock

If the external clock is selected by setting CS2 to CS0 bits of the SMR register to 111_B, the baud rates are as follows on the assumption that the frequency is f:

Asynchronous (start-stop synchronous): $f/16$

CLK-synchronous: f

However, f is up to 2 MHz.

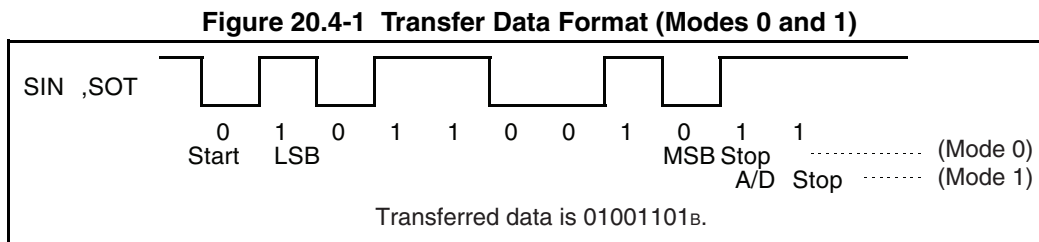
20.4.2 Asynchronous (Start-stop Synchronous) Mode

In the asynchronous (start-stop synchronous) mode, transfer data always begins with a start bit (L level data) and ends with a stop bit (H level data).

The receiver operation is controlled by the SCR register, and the transmitter operation is controlled by the SODR register.

■ Transfer Data Format

The UART handles only data of a NRZ (non return to zero) format. Figure 20.4-1 "Transfer Data Format (Modes 0 and 1)" shows the transfer data format.



Transfer data always begins with a start bit (L level data), is transmitted in the data bit length specified by the LSB first and ends with a stop bit (H level data). If the external clock is selected, always enter a clock.

In the normal mode (mode 0), the data length can be set to 7 or 8 bits. In the multiprocessor mode (mode 1), however, the data length must be set to 8 bits. Also, no parity bit can be added in the multiprocessor mode. Instead, an A/D bit is always added to data.

■ Receiver Operation

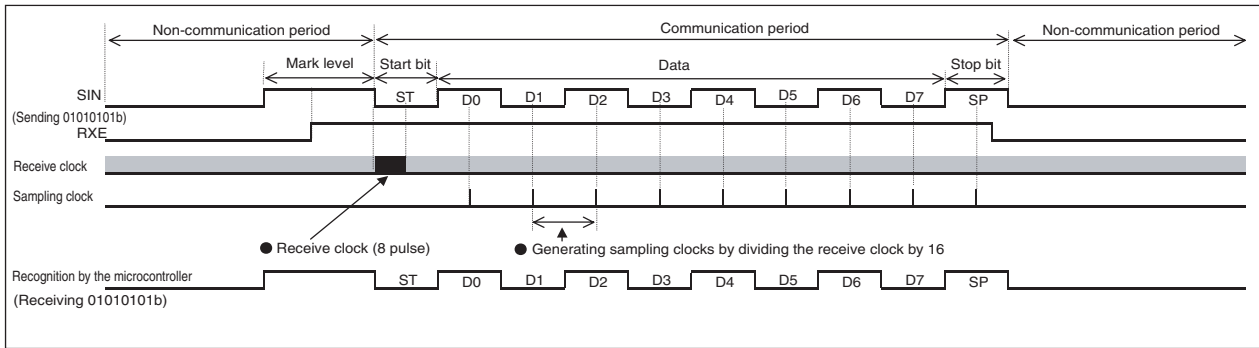
If the RXE bit (bit 9) of the SCR register is 1, a receiver operation is always performed. Once a start bit is detected, one-frame data is received according to the data format specified by that register. If an error occurs at the end of one-frame data reception, an error flag is set and the RDRF flag (bit 12) of the SSR register is then set. If the RIE bit (bit 9) of the SSR register is set to 1 simultaneously, a receiver interrupt occurs to the CPU. The flags of the SSR register are checked. If the receiver is normal, data is read from the SIDR register; if an error occurs, take corrective actions accordingly. The RDRF flag is cleared by reading data from the SIDR register.

■ Detecting the start bit

Implement the following settings to detect the start bit:

- Set the communication line level to H (attach the mark level) before the communication period.
- Specify reception permission (RXE=H) while the communication line level is H (mark level).
- Do not specify reception permission (RXE=H) for periods other than the communication period (without mark level). Otherwise, data is not received correctly.
- After the stop bit is detected (the RDRF flag is set to 1), specify reception inhibition (RXE=L) while the communication line level is H (mark level).

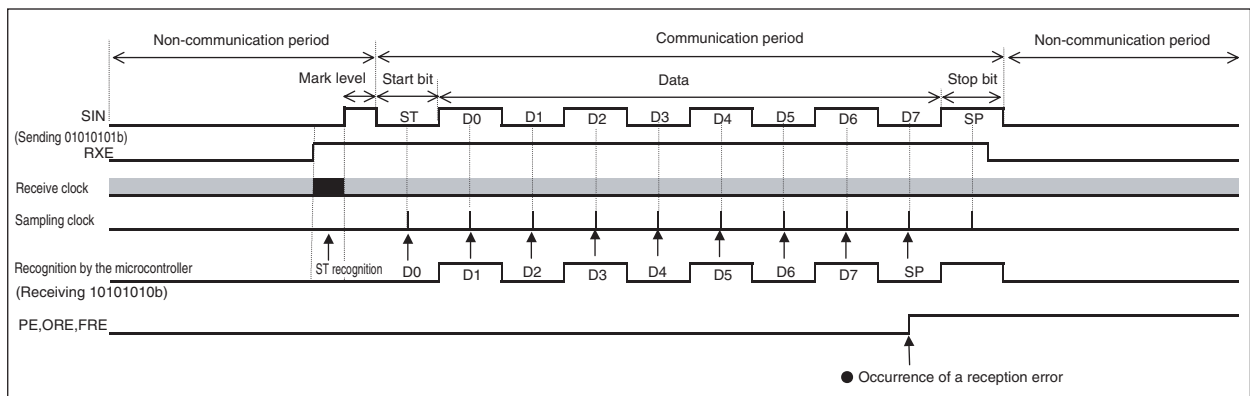
Figure 20.4-2 Normal Operation



Note that specifying reception permission at the timing shown below obstructs the correct recognition of the input data (SIN) by the microcontroller.

- Example of operation if reception permission (RXE=H) is specified while the communication line level is L.

Figure 20.4-3 Abnormal Operation



■ Transmitter Operation

When 1 is set in the TXE bit (bit 8) of the SSR register, transmitter data is written into the SODR register. When 1 is then written to the TXE bit (bit 8) of the SCR register, the data is transmitted.

When the data set in the SODR register is loaded into the transmitter shift register and begins to be transmitted, the TDRE flag is set again and the next transmitter data can be set. If the TIE bit (bit 8) of the SSR register is set to 1 at this time, a transmitter interrupt occurs to the CPU to request the SODR register to set transmitter data.

The TDRE flag is cleared once when the data is set to the SODR register.

20.4.3 CLK-synchronous Mode

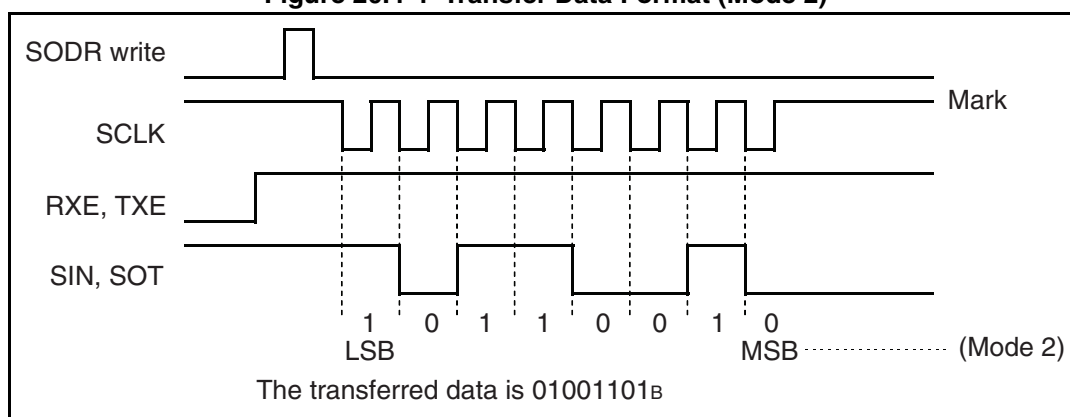
In the CLK-synchronous mode, a synchronous clock for receiving data is generated automatically if the internal clock is selected. A one-byte clock must be supplied if the external clock is selected.

The start of communication is controlled by the SODR register, and the termination of communication is controlled by the SSR register.

■ Transfer Data Format

The UART handles only data in NRZ (non return to zero) format. Figure 20.4-4 "Transfer Data Format (Mode 2)" shows the relationship between the transmitter or receiver clock and data.

Figure 20.4-4 Transfer Data Format (Mode 2)



If the internal clock (special baud rate generator or internal reload timer) is selected, a synchronous clock for receiving data is automatically generated when data is transmitted.

If the external clock is selected, the transmitter data buffer SODR register of the transmitter UART is checked for data (TDRE flag is 0) and then a one-byte clock must be supplied accurately. Set the mark level to H before and after transmission.

Only eight-bit data is valid and no parity bit can be added to data. No errors other than overrun errors are detected because neither start bit nor stop bit is provided.

■ Values set in registers in the CLK-synchronous mode

The settings of the control registers used in the CLK-synchronous mode are shown below.

Table 20.4-5 Settings of Control Registers Used in CLK-synchronous Mode

Register name	Bit name	Setting
SMR register	MD1, MD0	10
	CS2, CS1, CS0	Specifies clock input.
	SCKE	1 for special baud rate generator or internal timer and 0 for external clock
	SOE	1 for transmit operation and 0 for receive-only operation
SCR register	PEN	0
	P, SBL, A/D	Invalid
	CL	1
	REC	0 (for initialization)
	RXE, TXE	Set at least either of the bits to 1.
SSR register	RIE	1 if using an interrupt; otherwise, 0.
	TIE	0

■ Start of Communication

Communication is started by writing data into the SODR register. Even in the case of a receive-only operation, it is always necessary to write temporary transmitter data into the SODR register.

■ End of Communication

The end of communication can be confirmed by the fact that the RDRF flag of the SSR register is changed to 1.

Check the ORE bit of the SSR register to see if communication has been completed normally.

20.4.4 Occurrence of Interrupt and Flag Setting Timing

The UART has five flags and two interrupt sources. The five flags are PE, ORE, FRE, RDRF, and TDRE. One of the two interrupt sources is for the receiver, and the other is for the transmitter.

■ Five Flags (PE, ORE, FRE, RDRF, and TDRE) and Two Interrupt Sources

○ PE (Parity error), ORE (Over run error), FRE (Framing error)

The PE, ORE, and FRE flags are set when a relevant error occurs during transmission, and cleared when 0 is written to the REC bit of the SCR register.

○ RDRF

This flag is set when receiver data is loaded into the SIDR register, and cleared when data is read from the SIDR register. However, mode 1 is not provided with a parity detection function, and mode 2 is not provided with a parity detection function and a framing error detection function.

○ TDRE

This flag is set when the SODR register is empty and ready to write data, and cleared when data is written into the SODR register.

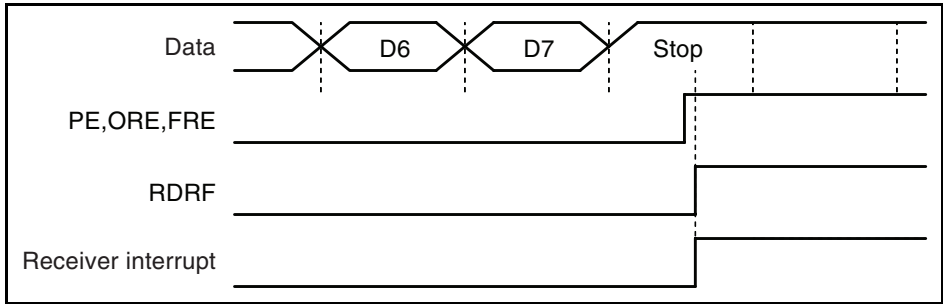
The two interrupt sources are for both receiver and transmitter. During the receive operation, PE, ORE, FRE, or RDRF issues an interrupt request; during the transmit operation, TDRE issues an interrupt request.

■ Interrupt Flag Setting Timing in Operating Modes

○ During receiver operation in mode 0

The PE, ORE, FRE, or RDRF flag is set when receive transfer is finished and the last stop bit is detected, and an interrupt request to the CPU occurs. When the PE, ORE, or FRE is active, data in the SIDR register becomes invalid.

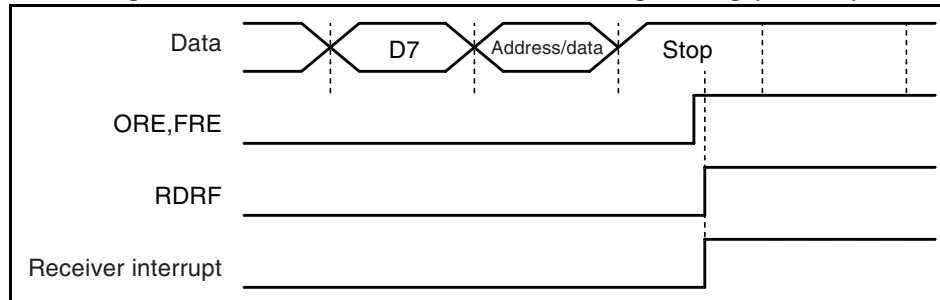
Figure 20.4-5 ORE, FRE, and RDRF Setting Timing (Mode 0)



○ During receiver operation in mode 1

The ORE, FRE, or RDRF is set when receive transfer is finished and the last stop bit is detected, and an interrupt request to the CPU occurs. Because of receivable data length of 8 bits, the last 9th bit data indicating an address or data becomes invalid. When the ORE or FRE is active, data in the SDR register becomes invalid.

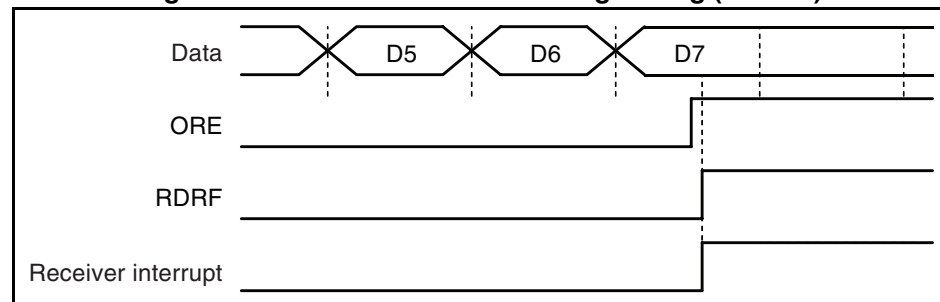
Figure 20.4-6 ORE, FRE, and RDRF Setting Timing (Mode 1)



○ During receiver operation in mode 2

The ORE or RDRF is set when receive transfer is finished and the last data (D7) is detected, and an interrupt request to the CPU occurs. When the ORE is active, data in the SDR register becomes invalid.

Figure 20.4-7 ORE and RDRF Setting Timing (Mode 2)



○ During transmitter operation in modes 0, 1, and 2

The TDRE is cleared when data is written into the SODR register, and set when the data is transferred to the internal shift register and the UART gets ready to write the next data. And an interrupt request to the CPU occurs. When 0 is written into the TXE of the SCR register during the transmitter operation (including RXE in mode 2), the TDRE of the SSR register is set to 1, the transmitter shifter stops and the UART transmitter operation is then disabled. After 0 is written into the TXE of the SCR register during the transmitter operation (including RXE in mode 2), the data written into the SODR register is transmitted before the transmitter stops.

Figure 20.4-8 TDRE Setting Timing (Modes 0 and 1)

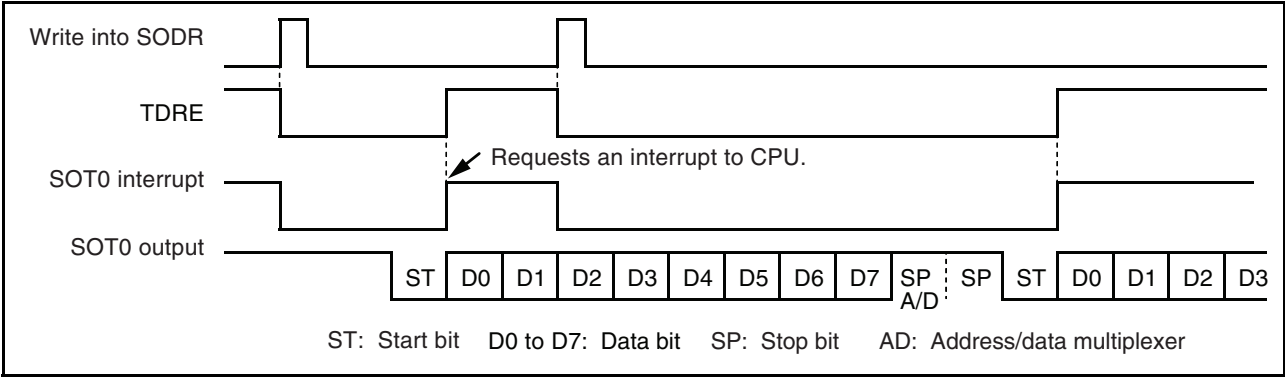
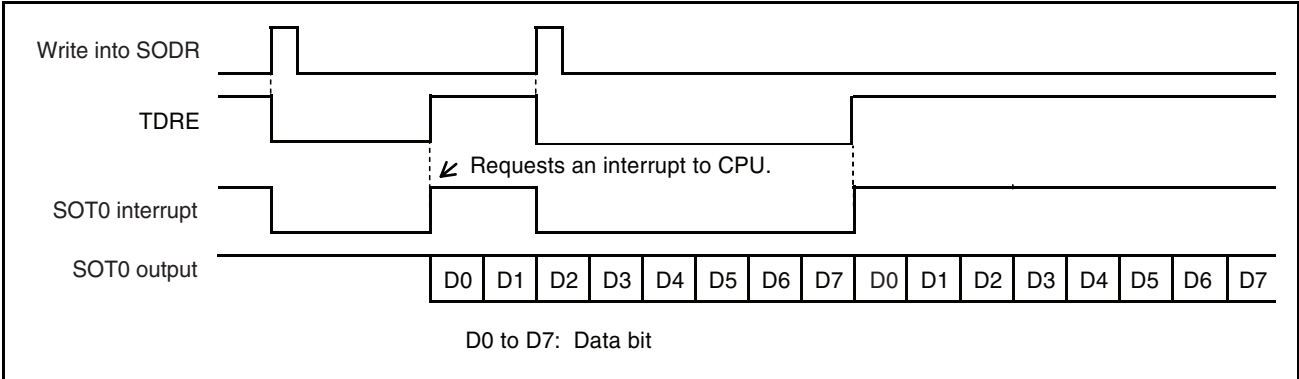


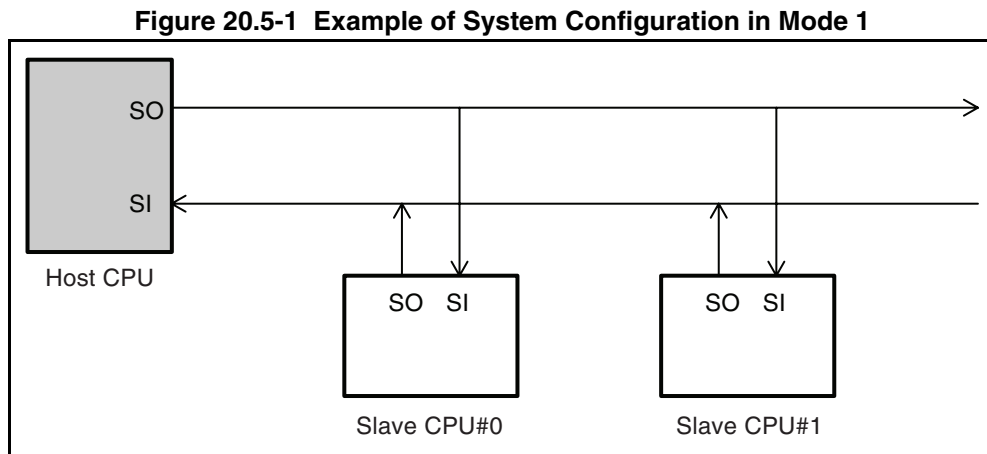
Figure 20.4-9 TDRE Setting Timing (mode 2)



20.5 Application of UART (During Operation in Mode 1)

Mode 1 is used if multiple slave CPUs are connected to one host CPU. This UART only supports the host communication interface (see Figure 20.5-1 "Example of System Configuration in Mode 1").

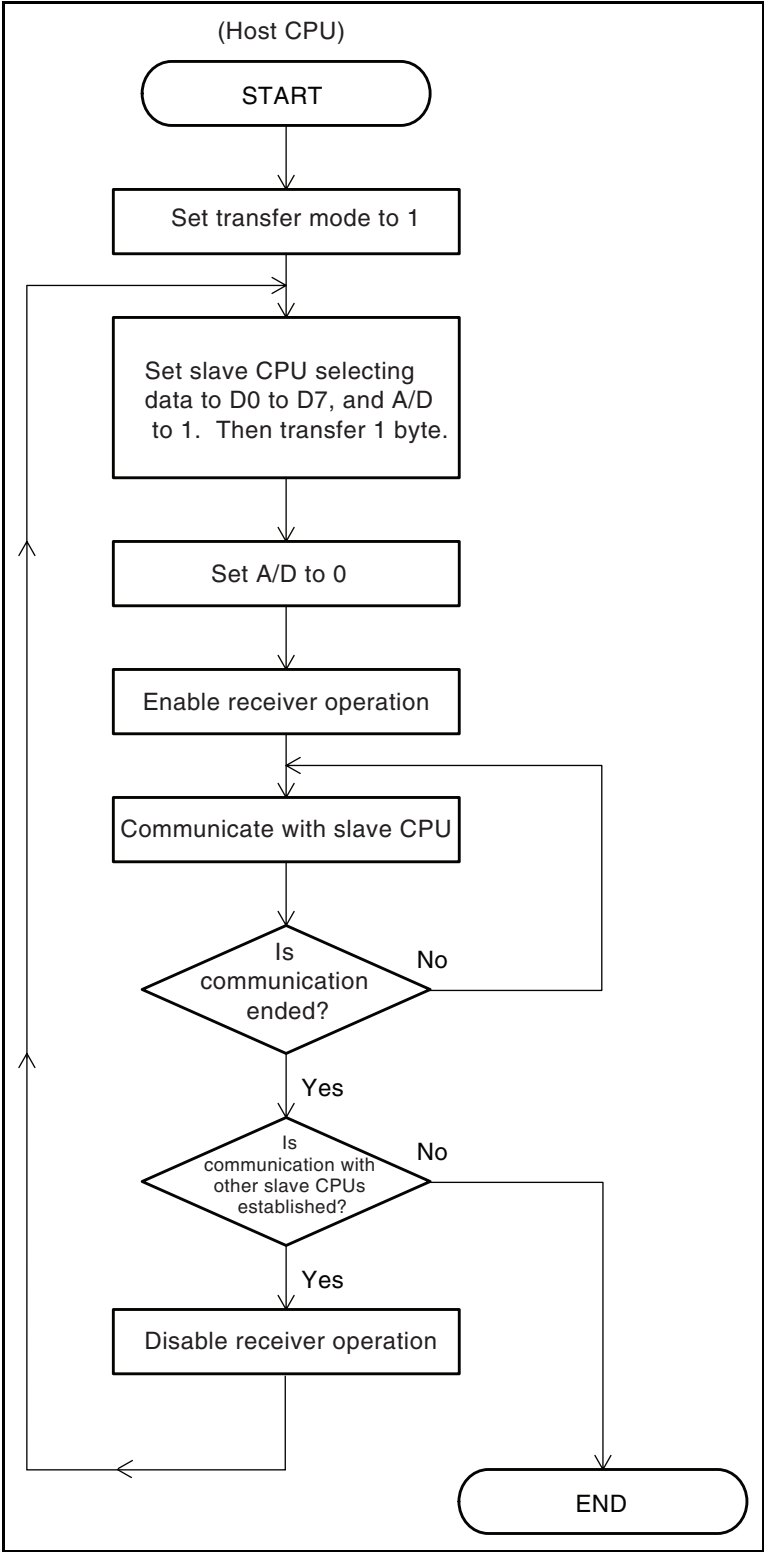
■ Application of UART (During Operation in Mode 1)



As shown in Figure 20.5-2 "Communication Flowchart in Mode 1" communication begins once the host CPU transfers address data. The address data means the data when the A/D of the SCR register is 1. This allows one of the slave CPUs to be selected as the receiver for communication with the host CPU. Normally, the data when the A/D of the SCR register is 0 is used.

In this mode, the parity check function is not available. Therefore, set the PEN bit of the SCR register to 0.

Figure 20.5-2 Communication Flowchart in Mode 1



CHAPTER 21 IEBus™ CONTROLLER

This chapter describes the functions and operation of the IEBus™ controller. Note that the MB90587C/CA model does not have the IEBus™ controller.

- 21.1 "Overview of IEBus™ Controller"
- 21.2 "Block Diagram for IEBus™ Controller"
- 21.3 "Registers of IEBus™ Controller"
- 21.4 "IEBus™ Transmission Control"
- 21.5 "IEBus™ Reception Control"
- 21.6 "Communication Control Status"
- 21.7 "Examples of the Flows of Main and Interrupt Processing Routines for IEBus™ Controller"
- 21.8 "IEBus™ Controller Operation at Transmission"
- 21.9 "IEBus™ Protocol Operation"
- 21.10 "Transmission Protocol"
- 21.11 "Transmission Data"
- 21.12 "Bit Format"

21.1 Overview of IEBus™ Controller

The Inter-Equipment Bus (IEBus™) is a small-scale two-wire serial bus interface designed for data transmission between different equipment.

IEBus™ is used, for example, as the bus interface for controlling car-mounted equipment.

■ Features of IEBus™ Controller

○ Multi-master mode

All the units connected to IEBus™ can transmit data to each other.

○ Broadcast function (communication between one unit and multiple units)

Group broadcast: Broadcast to a group of units

General broadcast: Broadcast to all other units

○ Selection from three modes using different transmission speeds

	Internal frequency of IEBus™	
	6 MHz	6.29 MHz
Mode 0	About 3.9 kbps	About 4.1 kbps
Mode 1	About 17 kbps	About 18 kbps
Mode 2	About 26 kbps	About 27 kbps

○ Transmission data buffer

8-byte FIFO

○ Reception data buffer

8-byte FIFO

○ Internal operating frequency of CPU (12 MHz or 12.58 MHz)

○ Frequency accuracy

In mode 0 or 1: plus or minus 1.5%

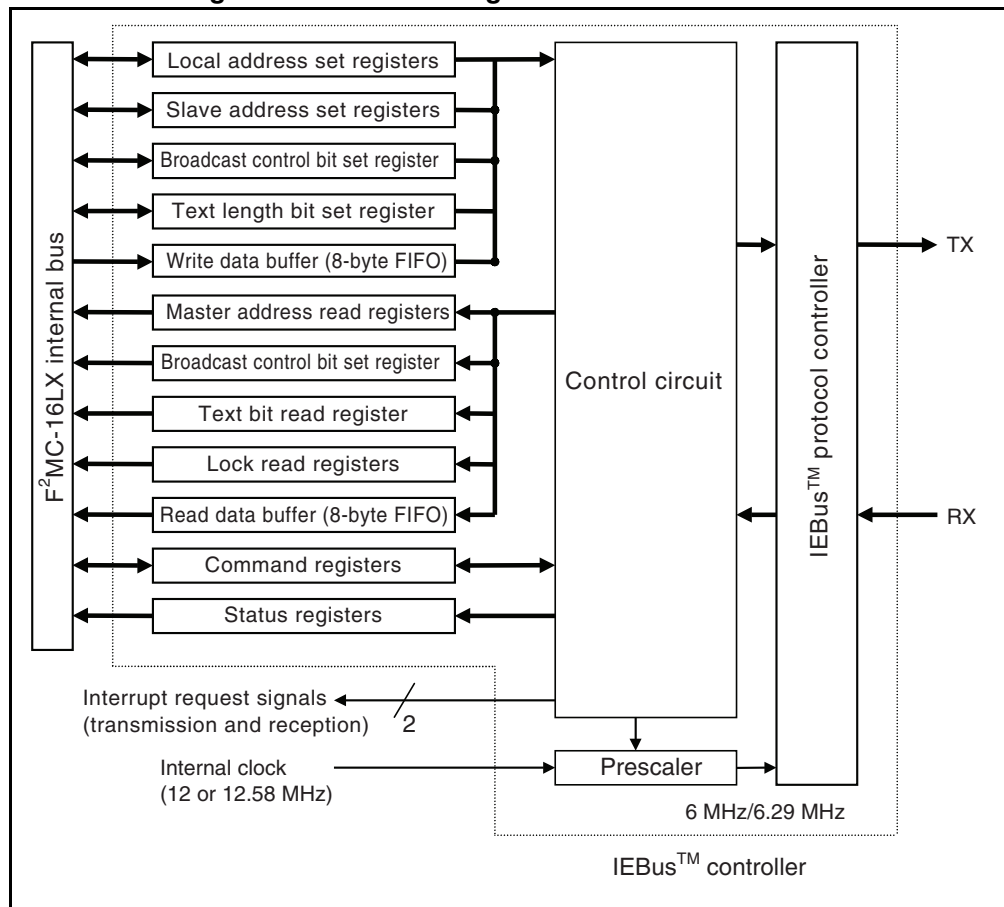
In mode 2: plus or minus 0.5%

21.2 Block Diagram for IEBus™ Controller

Figure 21.2-1 "Block Diagram for IEBus™ Controller" is a block diagram for the IEBus™ controller.

■ Block Diagram for IEBus™ Controller

Figure 21.2-1 Block Diagram for IEBus™ Controller



The control circuit of the IEBus™ controller performs the following operations:

- Controlling the number of bytes of transmission and reception data
- Controlling the maximum number of transmission bytes
- Detecting arbitration results
- Determining whether to return the acknowledgment of each field
- Generating interrupt signals

21.3 Registers of IEBus™ Controller

The IEBus™ controller has the following 18 registers:

- Local address set registers H and L
- Slave address set registers H and L
- Broadcast control bit set register
- Broadcast control bit read register
- Text length bit set register
- Text bit read register
- Command registers H and L
- Status registers H and L
- Lock read registers H and L
- Master address read registers H and L
- Read data buffer
- Write data buffer

■ Registers of IEBus™ Controller

Bit	15	14	13	12	11	10	9	8	
Address:000071 _H	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	Local address set register H
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	MAWH
Initial value	X	X	X	X	X	X	X	X	
Bit	7	6	5	4	3	2	1	0	
Address:000070 _H	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	Local address set register L
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	MAWL
Initial value	X	X	X	X	X	X	X	X	
Bit	15	14	13	12	11	10	9	8	
Address:000073 _H	Reserved	Reserved	Reserved	Reserved	SA11	SA10	SA09	SA08	Slave address set register H
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	SAWH
Initial value	X	X	X	X	X	X	X	X	
Bit	7	6	5	4	3	2	1	0	
Address:000072 _H	SA07	SA06	SA05	SA04	SA03	SA02	SA01	SA00	Slave address set register L
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	SAWL
Initial value	X	X	X	X	X	X	X	X	
Bit	15	14	13	12	11	10	9	8	
Address:000075 _H	DO3	DO2	DO1	DO0	C3	C2	C1	C0	Broadcast control bit set register
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	DCWR
Initial value	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	
Address:00007F _H	DO3	DO1	DO2	DO0	C3	C2	C1	C0	Broadcast control bit read register
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	DCRR
Initial value	0	0	0	X	X	X	X	X	

Bit	7	6	5	4	3	2	1	0	
Address:000074 _H	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	Text length bit set register
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	DEWR
Initial value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
Address:00007E _H	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	Text bit read register
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	DERR
Initial value	X	X	X	X	X	X	X	X	
Bit	15	14	13	12	11	10	9	8	
Address:000077 _H	MD1	MD0	PCOM	RIE	TIE	GOTM	GOTS	Reserved	Command register H
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	CMRH
Initial value	0	0	0	0	0	0	0	X	
Bit	7	6	5	4	3	2	1	0	
Address:000076 _H	RXS	TXS	TIT1	TIT0	CS1	CS0	RDBC	WDBC	Command register L
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	CMRL
Initial value	1	1	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	
Address:000079 _H	COM	TE	PEF	ACK	RIF	TIF	TSL	EOD	Status register H
Read/write	(R)	(R/W)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	STRH
Initial value	0	0	X	X	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
Address:000078 _H	WDBF	RDBF	WDBE	RDBE	ST3	ST2	ST1	ST0	Status register L
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	STRL
Initial value	0	0	1	1	X	X	X	X	
Bit	15	14	13	12	11	10	9	8	
Address:00007B _H	Reserved	Reserved	Reserved	LOC	LD11	LD10	LD09	LD08	Lock read register H
Read/write	(R)	(R)	(R)	(R/W)	(R)	(R)	(R)	(R)	LRRH
Initial value	1	1	1	0	X	X	X	X	
Bit	7	6	5	4	3	2	1	0	
Address:00007A _H	LD07	LD06	LD05	LD04	LD03	LD02	LD01	LD00	Lock read register L
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	LRRL
Initial value	X	X	X	X	X	X	X	X	
Bit	15	14	13	12	11	10	9	8	
Address:00007D _H	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	Master address read register H
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	MARH
Initial value	1	1	1	1	X	X	X	X	

Bit	7	6	5	4	3	2	1	0	
Address:00007C _H	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	Master address read register L
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	MARL
Initial value	X	X	X	X	X	X	X	X	
Bit	15	14	13	12	11	10	9	8	
Address:000081 _H	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	Read data buffer
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	RDB
Initial value	X	X	X	X	X	X	X	X	
Bit	7	6	5	4	3	2	1	0	
Address:000080 _H	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0	Write data buffer
Read/write	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	WDB
Initial value	X	X	X	X	X	X	X	X	

21.3.1 Local Address Set Registers (MAWH and HAWL)

The local address set registers are used to set the address (12 bits) of the local unit. Set the address of the local unit in these registers before the unit is released from the communication-disabled state.

■ Local Address Set Registers (MAWH and MAWL)

Figure 21.3-1 "Local Address Set Registers (MAWH and HAWL)" shows the structures of the local address set registers (MAWH and MAWL).

The address set in the MAWH and MAWL is used as the master address when the local unit is the master unit or for comparison with the slave address received when the local unit is in a slave unit.

Bits 15 to 12 are reserved and must always be 1. The values read from these bits are unpredictable.

Figure 21.3-1 Local Address Set Registers (MAWH and HAWL)

Bit	15	14	13	12	11	10	9	8	
Address:000071 _H	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	Local address set register H
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	MAWH
Initial value	X	X	X	X	X	X	X	X	
Bit	7	6	5	4	3	2	1	0	
Address:000070 _H	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	Local address set register L
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	MAWL
Initial value	X	X	X	X	X	X	X	X	

21.3.2 Slave Address Set Registers (SAWH and SAWL)

The slave address set registers (SAWH and SAWL) are used to set the slave address (12 bits) of the local unit when it is a slave unit. Set the slave address of the local unit in these registers before the unit is released from the communication-disabled state.

■ Slave Address Set Registers (SAWH and SAWL)

Figure 21.3-2 "Slave Address Set Registers (SAWH and SAWL)" shows the structures of the slave address set registers (SAWH and SAWL).

Bits 15 to 12 are reserved and must always be 1. The values read from these bits are unpredictable.

Figure 21.3-2 Slave Address Set Registers (SAWH and SAWL)

Bit	15	14	13	12	11	10	9	8	
Address:000073 _H	Reserved	Reserved	Reserved	Reserved	SA11	SA10	SA09	SA08	Slave address set register H
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	SAWH
Bit	7	6	5	4	3	2	1	0	
Address:000072 _H	SA07	SA06	SA05	SA04	SA03	SA02	SA01	SA00	Slave address set register L
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	SAWL

21.3.3 Broadcast Control Bit Set Register (DCWR)

The broadcast control bit set register (DCWR) is used to select a communication mode from broadcast and normal communication and control reading and data locking.

■ Broadcast Control Bit Set Register (DCWR)

Figure 21.3-3 Broadcast Control Bit Set Register (DCWR)

Bit	15	14	13	12	11	10	9	8	
Address:000075 _H	DO3	DO2	DO1	DO0	C3	C2	C1	C0	Broadcast control bit set register
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	DCWR

[Bits 15 to 12] D03 to D00

The D03, D02, D01, and D00 bits are used to select a communication mode from broadcast and normal communication.

Broadcast: 0_H (0000_B)

Normal communication: 8_H (1000_B)

Bits 14 to 12 must always be 0. When these bits are read, the read values are always 0.

[Bits 11 to 8] C3 to C0

The C3, C2, C1, and C0 bits are used to set four control bits.

Table 21.3-1 Control bit settings

	Bit 3 ^{*1}	Bit 2	Bit 1	Bit 0	Function ^{*2}
0 _H	0	0	0	0	Slave status reading
1 _H	0	0	0	1	Undefined
2 _H	0	0	1	0	Undefined
3 _H	0	0	1	1	Data reading and locking
4 _H	0	1	0	0	Lock address reading (lower 8 bits)
5 _H	0	1	0	1	Lock address reading (higher 4 bits)
6 _H	0	1	1	0	Slave status reading and unlocking
7 _H	0	1	1	1	Data reading
8 _H	1	0	0	0	Undefined
9 _H	1	0	0	1	Undefined
A _H	1	0	1	0	Command writing and locking
B _H	1	0	1	1	Data writing and locking
C _H	1	1	0	0	Undefined
D _H	1	1	0	1	Undefined
E _H	1	1	1	0	Command writing
F _H	1	1	1	1	Data writing

*1:

The value of bit 3 (MSB) determines the direction of transmission of the text length bits in the text length field and the data in the data field.

- When bit 3 is 1, data is transmitted from the master unit to the slave unit.
- When bit 3 is 0, data is transmitted from the slave unit to the master unit.

*2:

3_H, 6_H, A_H, and B_H are the control bit settings for locking and unlocking. If an undefined value 1_H, 2_H, 8_H, 9_H, C_H, or D_H is transmitted, no acknowledgment is returned.

21.3.4 Text Length Bit Set Register (DEWR)

The text length bit set register (DEWR) is used to specify the number of bytes of transmission data (an 8-bit value). The setting in this register is valid only for transmission.

■ Text Length Bit Set Register (DEWR)

Figure 21.3-4 Text Length Bit Set Register (DEWR)

Bit	7	6	5	4	3	2	1	0	
Address:000074 _H	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	Text length bit set register
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	DEWR
Initial value	0	0	0	0	0	0	0	0	

Table 21.3-2 "Settings of the Number of Bytes of Transmission Data" lists the settings of the number of bytes of transmission data.

Table 21.3-2 Settings of the Number of Bytes of Transmission Data

Number of bytes of transmission data	Command parameter
1 byte	1 _H
2 bytes	2 _H
:	:
255 bytes	FF _H
256 bytes	00 _H

○ Slave reception of command parameter

If one of the following command parameters is received in the control field when the local unit is a slave unit, the number of bytes of transmission data must always be set to 1 byte:

- 0_H (slave status reading)
- 4_H (reading of the lower 8 bits of lock address)
- 5_H (reading of the higher 4 bits of lock address)
- 6_H (slave status reading and unlocking)

○ Setting of a value over the maximum number of bytes

If a value over the maximum number of bytes of transmission data is set, data is transmitted using multiple frames. If it occurs, set the remaining number of bytes of the data to be transmitted in this register for the second and subsequent transmission operations.

For the remaining number of bytes, reference the value of the text length bit read register (DERR).

21.3.5 Command Register (Higher 8 Bits) (CMRH)

The command register (higher 8 bits) (CMRH) is used to specify the communication mode and control transmission, reception, and interrupts.

■ Command Register (Higher 8 Bits) (CMRH)

Figure 21.3-5 Command Register (Higher 8 Bits) (CMRH)

Bit	15	14	13	12	11	10	9	8	
Address:000077H	MD1	MD0	PCOM	RIE	TIE	GOTM	GOTS	Reserved	Command register H
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	X	CMRH

[Bits 15 and 14] MD1 and MD0

The MD1 and MD0 bits are used to specify the communication mode.

Table 21.3-3 Communication mode settings

MD1	MD0	Communication mode
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Setting inhibited

[Bit 13] PCOM

The PCOM bit enables or disables communication. Writing 1 in this bit sets the COM flag of the status register to 1 and enables communication. Writing 0 in this bit disables communication. Write 1 in this bit when the COM flag of the status register is 0.

[Bit 12] RIE

The RIE bit enables reception interrupts.

- 0: Disables reception interrupts.
- 1: Enables reception interrupts.

A reception interrupt occurs when:

- the 8-byte RDB is full of received data,
- data reception has ended normally,
- a frame has ended before the data of the number of data bytes specified by the text length bits has been received, or
- the local unit has failed in arbitration and has not been selected as a slave unit by the master unit.

[Bit 11] TIE

The TIE bit enables or disables transmission interrupts.

- 0: Disables transmission interrupts.
- 1: Enables transmission interrupts.

A transmission interrupt occurs when:

- the local unit has succeeded in arbitration and set a master address in the master address field during master transmission and remained to be the master unit,
- the local unit has received, from the master unit, the control bit value requesting data transmission during slave transmission,
- the writable area in WDB has reached the number of bytes set in the TIT1 and TIT0 bits of the command register,
- transmission of the number of bytes specified by the text length bits has ended within one frame, or
- the data of the number of bytes specified by the text length bits has not been transmitted by one frame and transmission has ended.

[Bit 10] GOTM

The GOTM bit starts master transmission. After communication is enabled, transmission starts when 1 is written in this bit.

Only writing of 1 is valid. Writing of 0 in this bit is invalid. When this bit is read, the read value is always 0.

[Bit 9] GOTS

The GOTS bit starts slave transmission. To transmit data when the local unit is a slave unit, write 1 in this bit after communication is enabled.

Only writing of 1 is valid. Writing of 0 in this bit is invalid. When this bit is read, the read value is always 0.

Relationships between the settings of GOTM and GOTS bits are as follows:

Table 21.3-4 Settings of GOTM and GOTS bits

GOTM	GOTS	Arbitration	Slave transmission	Operation
0	0	None	Disabled	Only reception is possible slave reception.
0	1	None	Enabled	Slave transmission is possible. The local unit cannot be the master.
1	0	To be done	Disabled	The local unit can perform only reception if it has failed in arbitration and become a slave.
1	1	To be done	Enabled	The local unit can perform transmission even if it has failed in arbitration and become a slave.

[Bit 8] Reserved

Bit 8 is reserved. This bit must always be 1. When this bit is read, the read value is unpredictable.

21.3.6 Command Register (Lower 8 Bits) (CMRL)

The command register (lower 8 bits) (CMRL) is used to set positive or negative logic, interrupt, and internal frequency.

■ Command Register (Lower 8 Bits) (CMRL)

Figure 21.3-6 Command Register (Lower 8 Bits) (CMRL)

Bit	7	6	5	4	3	2	1	0	
Address:000076H	RXS	TXS	TIT1	TIT0	CS1	CS0	RDBC	WDBC	Command register L
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value	1	1	0	0	0	0	0	0	CMRL

[Bit 7] RXS

The RXS bit is used to specify the input polarity of the RX1 pin. Select a polarity according to the specifications of the external driver or receiver to be connected.

Table 21.3-5 Specification of RX1 pin polarity

RXS	RX1 input
0	Positive logic input to RX1 <ul style="list-style-type: none">Logic "1": High levelLogic "0": Low level
1	Negative logic input to RX1 <ul style="list-style-type: none">Logic "1": Low levelLogic "0": High level

[Bit 6] TXS

The TXS bit is used to specify the output polarity of TX1 pin. Select a polarity according to the specifications of the external driver or receiver to be connected.

Table 21.3-6 Specification of TX1 pin polarity

TXS	TX1 output
0	Positive logic output from TX1 <ul style="list-style-type: none">Logic "1": High levelLogic "0": Low level
1	Negative logic output from TX1 <ul style="list-style-type: none">Logic "1": Low levelLogic "0": High level

Note:

At reset, the MB90580C series outputs an L level signal from the TX1 pin. If the connected driver or receiver uses the positive logic (enabling operation with an L level signal), an error occurs in the communication between other communication units. The error occurs because the L level signal is kept output after reset until the setting of the TXS bit.

To avoid such an error, external circuits must be designed so that an H level signal is input to the connected driver or receiver throughout the period from reset to the setting of the TXS bit.

[Bits 5 and 4] TIT1 and TIT0

The TIT1 and TIT0 bits are used to specify the timing of the interrupt that can occur when transmission data is written in the write data buffer.

Table 21.3-7 Settings of data transmission interrupt timing

TIT1	TIT0	Interrupt timing
0	0	When the writable area is 1 byte or more
0	1	When the writable area is 2 bytes or more
1	0	When the writable area is 4 bytes or more
1	1	When the writable area is 8 bytes

[Bits 3 and 2] CS1 and CS0

The CS1 and CS0 bits are used to specify the internal operating frequency of CPU and the internal frequency of IEBus™ as shown in Table 21.3-8 "Settings of CS1 and CS0 bits". Write 0 in both the CS1 and CS0 bits.

Table 21.3-8 Settings of CS1 and CS0 bits

CS1	CS0	Internal operating frequency of CPU ϕ	Internal frequency of IEBus™ ϕ	Calculation formula
0	0	12 MNz (12.58 MHz)	6 MNz (6.29 MHz)	$\phi_{IE} = \phi/2$
0	1	Setting inhibited	-	-
1	0	Setting inhibited	-	-
1	1	Setting inhibited	-	-

Note:

Use the formula shown above to calculate the internal operating frequency of CPU and the internal frequency of IEBus™. The internal frequency of CPU must not exceed the operation assurance range.

Frequency accuracy must be plus or minus 1.5% for mode 0 or 1 or plus or minus 0.5% for mode 2.

[Bit 1] RDBC

Writing 1 in the RDBC bit clears the 8-byte RDB to make the RDB blank (RDBE = 1). When this bit is read, the read value is always 0.

[Bit 0] WDBC

Writing 1 in the RDBC bit clears the 8-byte WDB to make the WDB blank (WDBE = 1). When this bit is read, the read value is always 0.

If this bit is set to 1 for the transmission of the data exceeding the maximum number of transmission bytes, the data written in the WDB in the preceding frame is invalidated and transmission starts with the data written in the current frame. If this bit is set to 0 in the same case and some of the transmission data written in the preceding frame remains, transmission starts with the remaining data.

If transmission has ended in the middle because of a timing error or for other reasons, the data being transmitted when the error occurred is not transmitted but the data following that data is transmitted even if this bit is 0.

21.3.7 Status Register (Higher 8 Bits) (STRH)

The status register (higher 8 bits) (STRH) indicates the communication status, whether an error has occurred, and whether an interrupt request has been generated.

■ Status Register (Higher 8 Bits) (STRH)

Figure 21.3-7 Status Register (Higher 8 Bits) (STRH)

Bit	15	14	13	12	11	10	9	8	
Address:000079 _H	COM	TE	PEF	ACK	RIF	TIF	TSL	EOD	Status register H
Read/write	(R)	(R/W)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	
Initial value	0	0	X	X	0	0	0	0	STRH

[Bit 5] COM

The COM bit is a flag that indicates the communication status.

- 0: Communication is disabled.
- 1: Communication is enabled.

When this flag is 0, it can be set by writing 1 in the PCOM bit that enables communication. This flag is cleared when communication ends.

[Bit 14] TE

The TE bit is set when a timing error occurs during communication. This bit is cleared by writing 0. Only writing 0 is valid for this bit.

[Bit 13] PEF

The PEF bit is set when a parity error is detected. If this bit is set on the receiving unit, the receiving unit does not return an acknowledgment. This bit is cleared when communication is enabled.

- 0: No parity error is detected.
- 1: A parity error is detected.

[Bit 12] ACK

During one-to-one communication, the data transmitted or received with the acknowledge bit is set in the ACK bit. The ACK bit is cleared when communication is enabled.

- 0: Acknowledgment bit is 0.
- 1: Acknowledgment bit is 1.

In the case of broadcast, the ACK bit is invalid and the value read from the ACK bit is unpredictable.

[Bit 11] RIF

The RIF bit is set to 1 when a reception interrupt occurs.

- 0: No reception interrupt occurs.
- 1: A reception interrupt occurs.

This bit is cleared by writing 0 or intelligent I/O service. Only writing 0 is valid for this bit.

[Bit 10] TIF

The TIF bit is set to 1 when a transmission interrupt occurs.

- 0: No transmission interrupt occurs.
- 1: A transmission interrupt occurs.

This bit is cleared by writing 0 or intelligent I/O service. Only writing 0 is valid for this bit.

[Bit 9] TSL

The TSL bit is set when a communication frame ends because the maximum number of transmission bytes has been reached in the data field during data transmission or reception. This bit is cleared when transmission of the next communication frame starts.

[Bit 8] EOD

The EOD bit is set when a communication frame ends because the maximum number of transmission bytes specified by the text length bits has been reached in the data field during data transmission or reception. When the EOD bit is 1, the relevant communication frame has ended normally.

This bit is cleared when transmission of the next communication frame starts.

21.3.8 Status Register (Lower 8 Bits) (STRL)

The status register (lower 8 bits) (STRL) indicates the status of the write data buffer (WDB) and the read data buffer (RDB).

It generates the following four communication statuses:

- Master or slave transmission
- Master reception
- Slave reception mode
- Broadcast reception

■ Status Register (Lower 8 Bits) (STRL)

Figure 21.3-8 Status Register (Lower 8 Bits) (STRL)

Bit	7	6	5	4	3	2	1	0	
Address:000078H	WDBF	RDBF	WDBE	RDBE	ST3	ST2	ST1	ST0	Status register L
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value	0	0	1	1	X	X	X	X	STRL

[Bit 7] WDBF

The WDBF bit is a flag that indicates the status of the write data buffer (WDB).

This bit is set when WDB is full and cleared when data is transmitted and at least one byte of the WDB area becomes writable.

- 0: WDB is not full.
- 1: WDB is full.

[Bit 6] RDBF

The WDBF bit is a flag that indicates the status of the read data buffer (RDB).

This bit is set when RDB is full of received data and is cleared when data is read and at least one byte of the RDB area becomes writable.

- 0: RDB is not full.
- 1: RDB is full.

[Bit 5] WDBE

The WDBE bit is a flag that indicates the status of the write data buffer (WDB).

This bit is set when WDB is empty and is cleared when at least one byte is written in WDB. This bit is also set when 1 is written in the WDBC bit of the command register.

- 0: WDB is not empty.
- 1: WDB is empty.

[Bit 4] RDBE

The RDBE bit is a flag that indicates the status of the read data buffer (RDB).

This bit is set when data has been read from RDB and RDB is empty and is cleared when at least one byte of received data is written in RDB. This bit is also set when 1 is written in the RDBC bit of the command register.

- 0: RDB is not empty.
- 1: RDB is empty.

[Bits 3 to 0] ST3 to ST0

The ST3, ST2, ST1, and ST0 bits store the status of the communication in progress and cause an interrupt. The current communication status can be known by reading these four bits.

Table 21.3-9 Status flag

ST 3	ST 2	ST 1	ST 0	Mode	Status
0	0	0	0	Master or slave transmission	Transmission started.
0	0	0	1		Transmission is in progress.
0	0	1	0		Transmission ended normally.
0	0	1	1		Transmission stopped in the middle.
0	1	0	0	Master reception	Master reception is started.
0	1	0	1		RDB is full of the data received by master reception.
0	1	1	0		Master reception ended in the middle.
0	1	1	1		Master reception ended in the middle.
1	0	0	0	Slave reception mode	Slave reception is started.
1	0	0	1		RDB is full of the data received by slave reception.
1	0	1	0		Slave reception ended normally.
1	0	1	1		Slave reception ended in the middle.
1	1	0	0	Broadcast reception	Broadcast reception started.
1	1	0	1		RDB is full of the data received by broadcast.
1	1	1	0		Broadcast reception ended normally.
1	1	1	1		Broadcast reception ended in the middle.

For the timing of the operation of these bits, see Section 21.6 "Communication Control Status".

21.3.9 Lock Read Registers (LRRH and LRRL)

The lock read registers (LRRH and LRRL) are used to read the lock status of the local unit.

■ Lock Read Registers (LRRH and LRRL)

Figure 21.3-9 Lock Read Registers (LRRH and LRRL)

Bit	15	14	13	12	11	10	9	8	
Address:00007BH	Reserved	Reserved	Reserved	LOC	LD11	LD10	LD09	LD08	Lock read register H
Read/write	(R)	(R)	(R)	(R/W)	(R)	(R)	(R)	(R)	LRRH
Initial value	1	1	1	0	X	X	X	X	
Bit	7	6	5	4	3	2	1	0	
Address:00007AH	LD07	LD06	LD05	LD04	LD03	LD02	LD01	LD00	Lock read register L
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	LRRL
Initial value	X	X	X	X	X	X	X	X	

[Bits 15 to 13] Reserved

Bits 15 to 13 are reserved. The values read from these bits are always 1.

[Bit 12] LOC

The LOC bit indicates the lock status, that is, whether the local unit is locked by another unit.

- 0: The local unit is not locked.
- 1: The local unit is locked.

Writing 0 in this bit unlocks the local unit. Writing 1 in this bit is invalid.

[Bits 11 to 0] LD11 to LD00

The LD11 to LD00 bits indicate the address of the unit that locks the local unit. If the local unit is not locked, the contents of these bits are invalid.

Note:

IEBus™ provides a lock function to enable connected units to communicate over multiple frames. On the other hand, if a unit that has locked another unit fails without unlocking it, the locked unit will not be able to receive data. To prevent this, the system using the lock function must periodically read the lock read register to monitor the lock status.

To unlock the local unit, write 0 in the LOC bit.

21.3.10 Master Address Read Registers (MARH and MARL)

The master address read registers (MARH and MARL) indicate the address of the current master unit.

■ Master Address Read Registers (MARH and MARL)

Figure 21.3-10 Master Address Read Registers (MARH and MARL)

Bit	15	14	13	12	11	10	9	8	
Address:00007D _H	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	Master address read register H
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value	1	1	1	1	X	X	X	X	MARH
Bit	7	6	5	4	3	2	1	0	
Address:00007C _H	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	Master address read register L
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value	X	X	X	X	X	X	X	X	MARL

[Bits 15 to 12] Reserved

Bits 15 to 12 are reserved. The values read from these bits are always 1.

[Bits 11 to 0] MA11 to MA00

The MA11 to MA00 bits indicate the address of the current master unit. If the local unit is the master unit, the local address set in the local address set registers is set in these bits.

Address data is set in the master address read registers when arbitration in the master address field ends.

21.3.11 Broadcast Control Bit Read Register (DCRR)

The broadcast control bit read register (DCRR) indicates the broadcast and control bits received from the master unit.

■ Broadcast Control Bit Read Register (DCRR)

Figure 21.3-11 Broadcast Control Bit Read Register (DCRR)

Bit	15	14	13	12	11	10	9	8	
Address:00007FH	DO3	DO1	DO2	DO0	C3	C2	C1	C0	Broadcast control bit read register
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	DCRR
Initial value	0	0	0	X	X	X	X	X	

[Bits 15 to 12] DO3 to DO0

The DO3 to DO0 bits indicate the broadcast bits that are received from the master unit when the local unit is a slave unit. If the local unit is the master unit, the DO3 to DO0 bits indicate the data set in the broadcast bits of the broadcast control bit set register. Broadcast bit data is set in DCRR automatically when it is transmitted or received.

- For one-to-one communication: 1_H (0001_B)
- For broadcast: 0_H (0000_B)

The values read from D03 to D01 are always "0".

[Bits 11 to 8] C3 to C0

The C3 to C0 bits indicate the control bits received from the master unit. If the local unit is the master unit, the C3 to C0 bits indicate the data set in the control bits of the broadcast control bit set register. Control bit data is set when the control field ends and an acknowledgment is detected.

See Table 21.3-1 "Control bit settings".

21.3.12 Text Length Bit Read Register (Lower 8 Bits) (DERR)

The text length bit read register (lower 8 bits) (DERR) indicates the number of bytes of the data to be transmitted or received. When the local unit transmits data, DERR indicates the value set in the text length bit set register. When the local unit receives data, DERR indicates the value received in the text length field.

■ Text Length Bit Read Register (Lower 8 Bits) (DERR)

Figure 21.3-12 Text Length Bit Read Register (Lower 8 Bits) (DERR)

Bit	7	6	5	4	3	2	1	0	
Address:00007EH	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	Text bit read register
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value	X	X	X	X	X	X	X	X	DERR

Data is set in the text length bit read register (lower 8 bits) (DERR) in the following timing:

- In master mode
 - When the number of bytes of transmission data is written in DEWR
 - When text length bits are received by master reception
 - When communication ends
- In slave mode
 - When the number of bytes of transmission data is written in DEWRI
 - When text length bits are received by slave reception
 - When communication ends

For the values of DERR, see Table 21.3-2 "Settings of the Number of Bytes of Transmission Data".

21.3.13 Read Data Buffer (RDB)

The read data buffer (RDB) is an 8-byte FIFO buffer to store the data received in the data field.

■ Read Data Buffer (RDB)

Figure 21.3-13 "Read Data Buffer (RDB)" shows the bit configuration of the read data buffer (RDB).

Figure 21.3-13 Read Data Buffer (RDB)

Bit	15	14	13	12	11	10	9	8	
Address:000081H	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	Read data buffer
Read/write	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value	X	X	X	X	X	X	X	X	RDB

If the 8-byte read data buffer (RDB) becomes full of received data, a reception interrupt occurs. If it occurs, data must be read from the read data buffer. If data is not read before the next data is received, an error occurs. If the error occurs, broadcast communication is stopped or one-to-one communication continues reception until received data exceeds the maximum number of bytes of transmission data without returning an acknowledgment. Table 21.3-10 "Time required until next data reception after reception interrupt" lists the time required until the next data is received after RDB has become full.

Even if RDB is not full of received data, a reception interrupt occurs when the data of the number of bytes specified by the text length bits or the maximum number of bytes of transmission data has been received. Data must be read from RDB when the reception interrupt occurs.

Writing 1 in the RDBC bit of the CMRL register clears the read data buffer. Confirm that the read data buffer is not empty before reading it.

Table 21.3-10 Time required until next data reception after reception interrupt

	Maximum time [μ s]	Number of cycles
Mode 0	1580	19000
Mode 1	400	4800
Mode 2	290	3400

21.3.14 Write Data Buffer (WDB)

The write data buffer (WDB) is an 8-byte FIFO buffer to store the data to be transmitted in the data field. The timing of transmission interrupt can be set in the TIT1 and TIT0 bits of the command register.

■ Write Data Buffer (WDB)

Figure 21.3-14 "Write Data Buffer (WDB)" shows the bit configuration of the write data buffer (WDB).

Figure 21.3-14 Write Data Buffer (WDB)								
Bit	7	6	5	4	3	2	1	0
Address:000080H	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0
Read/write	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value	X	X	X	X	X	X	X	X
Write data buffer								
WDB								

If a transmission interrupt is caused by the write data buffer (WDB), the data to be transmitted next must be written in the write data buffer. The written data is transmitted, the write data buffer becomes empty, and there is no more data to be transmitted. In this status, if the data to be transmitted next is not written within the time shown in Table 21.3-11 "Time allowed until next data writing after transmission interrupt" an error occurs and communication is stopped.

Writing 1 in the WDBC bit of the CMRL register clears the write data buffer. Data must not be written in the write data buffer when it is full. Confirm that the write data buffer is not full before writing data in it.

Table 21.3-11 Time allowed until next data writing after transmission interrupt

	Maximum time [μs]	Number of cycles
Mode 0	1580	19000
Mode 1	400	4800
Mode 2	290	3400

21.4 IEBus™ Transmission Control

There are the following three types of transmission on IEBus™:

- Master transmission
 - Slave transmission
 - Transmission of slave status and lock address
-

■ Master Transmission

Master transmission is to transmit data commands from a master unit to a slave unit.

When the local unit as the master unit transmits data to a slave unit, A_H, B_H, E_H, or F_H must be set in the broadcast control bit set register.

1. The master address is set in the local address set register, the slave address is set in the slave address set register, and broadcast and control bits are set in the broadcast control bit set register. The PCOM bit of the command register is then set to enable communication.
2. When the local unit succeeded in arbitration as the master unit (at the end of master address field), the state code (0_H) indicating the start of transmission is set in the ST3 to ST0 bits of the status register and a transmission interrupt is generated. The number of bytes of transmission data is then set in the text bit length set register and transmission data is set in WDB while WDB is being checked to confirm that it is not full.
3. Each time one byte of data is transmitted, the data set in WDB is eliminated by one byte. When the writable area in WDB reaches the number of bytes specified by the TIT1 and TIT0 bits of the command register (that is, a timing of transmission interrupt), a transmission interrupt occurs. At that time, transmission data must be set in WDB if the state code is 1_H (data transmission) and WDB is not full.
4. When the data or commands of the number of bytes of transmission data have been transmitted normally, state code "2_H" (end of transmission) is set in the ST3 to ST0 bits of the status register and the EOD bit is set, then a transmission interrupt occurs.
5. Transmission of data over the maximum number of bytes or a transmission error may occur and transmission may stop in the middle without the number of bytes set in the text length bit set register transmitted. If it occurs, state code "3_H" (transmission stopped in the middle) is set in the ST3 to ST0 bits of the status register and a transmission interrupt occurs. In this case, the TSL, REF, and TE bits of the status register indicate the content of the communication error.

If a timing error occurs in transmission and all data cannot be transmitted, the data remaining in WDB can be transmitted by clearing the TE bit and retrying transmission. The retry of transmission starts with the transmission of the data that follows the data transmitted at the time the error occurred.

If transmission is to be resumed, 1 must be written in the WDBC bit of the command register to clear WDB.

If the master unit fails in arbitration, it enters the slave reception state. If it receives a slave address field and it is not slave reception, slave results or broadcast reception may stop in the middle (status code B_H or F_H) and a reception interrupt occurs.

Because the hardware does not support transmission retry, a program must specify the number of retries and execute those retries.

■ Slave Transmission

Slave transmission is to transmit data from a slave unit to a master unit when the slave unit receives control bit data of 3_H or 7_H from the master unit.

1. When the slave unit receives control bit data of 3_H or 7_H from the master unit, state code "0_H" (start of transmission) is set in the ST3 to ST0 bits of the status register and a transmission interrupt occurs. The number of bytes of transmission data is then set in the text bit length set register and transmission data is set in WDB while WDB is being checked to confirm that it is not full. (See the notes below.)
2. When transmission of text data is started, state code "1_H" (transmission in progress) is set in the ST3 to ST0 bits of the status register and a transmission interrupt occurs. Transmission data is then set in WDB while WDB is being checked to confirm that it is not full.
3. Each time one byte of data is transmitted, the data set in WDB is eliminated by one byte. When the writable area in WDB reaches the number of bytes specified by the TIT1 and TIT0 bits of the command register (that is, a timing of transmission interrupt), a transmission interrupt occurs. At that time, transmission data must be set in WDB if the state code is 1_H (data transmission) and WDB is not full.
4. When the data or commands of the number of bytes of transmission data have transmitted normally, state code "2_H" (end of transmission) is set in the ST3 to ST0 bits of the status register and the EOD bit is set, then a transmission interrupt occurs.
5. Transmit data over the maximum number of bytes or a transmission error may occur and transmission may stop in the middle without the number of bytes set in the text length bit set register transmitted. If it occurs, state code "3_H" (transmission stopped in the middle) is set in the ST3 to ST0 bits of the status register and a transmission interrupt occurs. In this case, the TSL, REF, and TE bits of the status register indicate the content of the communication error.

■ Notes on the Reception of Control Bits from Master Unit

The number of bytes of transmission data and transmission data can also be set by the interrupt after reception of control bits.

If WDB is empty and transmission data is set first by that interrupt, the time allowed until the next interrupt for text length bit transmission is short. Note the following:

- Confirm that WDB is not full before writing data.
- When setting the number of bytes of transmission data by that interrupt, set it within the limit time (shown in Table 21.4-1 "Time allowed until setting of the number of bytes of transmission data after transmission interrupt") from that interrupt. If it cannot be set within the limit time, no error occurs but the data of the number of bytes of transmission data set previously is transmitted. If the number of bytes is the initial value, 256 bytes of data are transmitted.
- When no transmission data has been set in WDB before that interrupt, set at least one byte of data in WDB within the limit time (shown in Table 21.4-1 "Time allowed until setting of the number of bytes of transmission data after transmission interrupt") from that interrupt. If no data is set within the limit time, WDB is determined to be empty, an error occurs after the text length bits are transmitted, and communication ends.

Table 21.4-1 Time allowed until setting of the number of bytes of transmission data after transmission interrupt

	Maximum time [μ s]	Number of cycles
Mode 0	About 158	About 1900
Mode 1	About 40	About 480
Mode 2	About 29	About 350

■ Transmission of Slave Status and Lock Address

When a slave unit receives control bit data of 0_H, 4_H, 5_H, or 6_H, the slave unit automatically generates slave status and a lock address and transmits them to the master unit. The slave status and lock address do not have to be set as transmission data in WDB. However, one byte must be set as the number of bytes of transmission data in the text length bit set register for the slave status and lock address.

21.5 IEBus™ Reception Control

There are three types of reception on IEBus™ as follows:

- Master reception
 - Slave reception
 - Broadcast reception
-

■ Master Reception

Master reception is to receive data, slave status, or lock address from a slave unit. The local unit is designated as the master unit by setting 0_H, 3_H, 4_H, 5_H, 6_H, or 7_H in the control bits.

1. After receiving the control bits, the slave unit transmits the text length bits to the master unit. When the master unit returns an acknowledgment after receiving the text length bits, the number of bytes of reception data is set in the text length bit read register.
2. When the slave unit receives acknowledgment of the master unit transmitted, the slave unit transmits data in the data field. Whenever one byte of data is received, the data is set in RDB.
3. Each time 8 bytes (RDB size) of data are received, state code "5_H" (RDB buffer is full) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. When the reception interrupt occurs, RDB is read while being checked to confirm that it is not full.
4. When all data of one frame is received and set in RDB, state code "5_H" (normal end of reception) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. This reception interrupt occurs even if RDB is not full.
5. Reception of data over the maximum number of bytes or a transmission error may occur and reception may stop in the middle without the number of bytes set in the text length bit set register received from the slave unit. If this occurs, state code "7_H" (reception stopped in the middle) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs.

■ Slave Reception

The slave unit receives A_H, B_H, E_H, or F_H as the control bits from the master unit. Slave reception is to receive data or commands from the master unit.

1. When the slave unit returns an acknowledgment in the text length field, the number of bytes of reception data is set in the text length bit read register.
2. When the slave unit returns the acknowledgment after reception of the text length bits, the master unit transmits data in the data field. Each time one byte of data is received normally, the data is set in RDB.
3. Each time 8 bytes (RDB size) of data are received, state code "9_H" (RDB buffer is full) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. When the reception interrupt occurs, RDB is read while being checked to confirm that it is not full.
4. After the last data of one frame is received and set in RDB, state code "A_H" (normal end of reception) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. This reception interrupt occurs even if RDB is not full.
5. Reception of data over the maximum number of bytes or a transmission error may occur and reception may stop in the middle without the number of bytes set in the text length bit set

register received from the slave unit. If this occurs, state code "B_H" (reception stopped in the middle) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs.

■ Broadcast Reception

1. When the slave unit returns an acknowledgment in the text length field, the number of bytes of reception data is set in the text length bit read register.
2. When the slave unit returns the acknowledgment after reception of the text length bits, the master unit transmits data in the data field. Each time one byte of data is received normally, the data is set in RDB.
3. Each time 8 bytes (RDB size) of data are received, state code "E_H" (RDB buffer is full) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. When the reception interrupt occurs, RDB is read while being checked to confirm that it is not full.
4. After the last data of one frame is received and set in RDB, state code "D_H" (normal end of reception) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. This reception interrupt occurs even if RDB is not full.
5. Reception of data over the maximum number of bytes or a transmission error may occur and reception may stop in the middle without the number of bytes set in the text length bit set register received from the slave unit. If this occurs, state code "F_H" (reception stopped in the middle) is set in the ST3 to ST0 bits of the status register and a reception interrupt occurs. For the settings of the ST3 to ST0 bits of the status register, see Section 21.6 "Communication Control Status".

21.6 Communication Control Status

The ST3 to ST0 bits of the status register are used to indicate a status code. When a status code is set in these bits, an interrupt request occurs. During the interrupt, these bits can be read to know the communication status.

■ Master or Slave Transmission (Occurrence of Transmission Interrupt)

Table 21.6-1 "Codes in ST3 to ST0 Bits for Master or Slave Transmission" lists the codes that can be set in the ST3 to ST0 bits when the local unit has succeeded in arbitration in the master address field and has become the master unit and transmits data or commands to a slave unit (also in case of broadcast transmission) or when a slave unit transmits data to the master unit.

Table 21.6-1 Codes in ST3 to ST0 Bits for Master or Slave Transmission

Code name	Code	Description
Start of transmission	0000 _B	This code indicates that master or slave transmission is started. The master and slave modes are different in the timing of the code setting. <ul style="list-style-type: none"> Master transmission The code is set when the local unit has ended the master address field and become the master unit. Slave transmission The code is set when the control bits (0_H, 3_H, 4_H, 5_H, 6_H, or 7_H) requesting data transmission are received from the master unit.
Data transmission	0001 _B	This code indicates that the master or slave unit is transmitting data using the data field. This code is set when the transmission of text length bits is started.
Normal end of transmission	0010 _B	This code indicates that all the data of the number of bytes specified by the text length bits has been transmitted in one frame.
Transmission ended in the middle	0011 _B	This code indicates that communication ended without transmitting all data of the number of bytes specified by the text length bits in one frame.

■ Master Reception (Occurrence of Reception Interrupt)

Table 21.6-2 "Codes in ST3 to ST0 Bits for Master Reception" lists the codes that can be set in the ST3 to ST0 bits when the local unit has succeeded in arbitration in the master address field and become the master unit and receives data, status, or lock address from a slave unit (also in case of broadcast reception).

Table 21.6-2 Codes in ST3 to ST0 Bits for Master Reception

Code name	Code	Description
Start of master reception	0100 _B	This code indicates that master reception has started. The code is set when the master unit has received a text length code normally from the slave unit.
Data full in master reception	0101 _B	This code indicates that 8 bytes of data has been received and RDB is full. When this code is set, the master unit requests the host controller to read received data from RDB.
Normal end of master reception	0110 _B	This code indicates that all the data of the number of bytes specified by the text length bits has been received in one frame.
Master reception ended in the middle	0111 _B	This code indicates that communication ended without receiving all the data of the number of bytes specified by the text length bits in one frame.

■ **Slave Reception (Occurrence of Reception Interrupt)**

Table 21.6-3 "Codes in ST3 to ST0 Bits for Slave Reception" lists the codes that can be set in the ST3 to ST0 bits when the local unit as a slave unit receives data or commands from the master unit.

Table 21.6-3 Codes in ST3 to ST0 Bits for Slave Reception

Code name	Code	Description
Start of slave reception	1000 _B	This code indicates that slave reception is started. The code is set when the slave unit has received a text length code normally from the master unit.
Data full in slave reception	1001 _B	This code indicates that 8 bytes of data has been received and RDB is full. When this code is set, the slave unit requests the host controller to read received data from RDB.
Normal end of slave reception	1010 _B	This code indicates that all the data of the number of bytes specified by the text length bits has been received in one frame.
Slave reception ended in the middle	1011 _B	This code indicates that communication ended without receiving all the data of the number of bytes specified by the text length bits in one frame.

■ Broadcast Reception (Occurrence of Reception Interrupt)

Table 21.6-4 "Codes in ST3 to ST0 Bits for Broadcast Reception" lists the codes that can be set in the ST3 to ST0 bits when the local unit as a slave unit receives data or commands by broadcast from the master unit.

Table 21.6-4 Codes in ST3 to ST0 Bits for Broadcast Reception

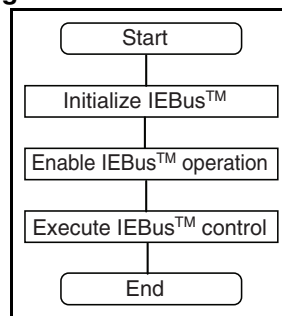
Code name	Code	Description
Start of broadcast reception	1100 _B	This code indicates that broadcast reception has started. The code is set when the slave unit has received a text length code normally from the master unit.
Data full in broadcast reception	1101 _B	This code indicates that 8 bytes of data have been received and RDB is full. When this code is set, the slave unit requests the host controller to read received data from RDB.
Normal end of broadcast reception	1110 _B	This code indicates that all the data of the number of bytes specified by the text length bits has been received in one frame.
Broadcast reception ended in the middle	1111 _B	This code indicates that communication ended without receiving all the data of the number of bytes specified by the text length bits in one frame.

21.7 Examples of the Flows of Main and Interrupt Processing Routines for IEBus™ Controller

This section explains the flows of IEBus™ processing on the MB90580C series with examples.

■ Main Routine

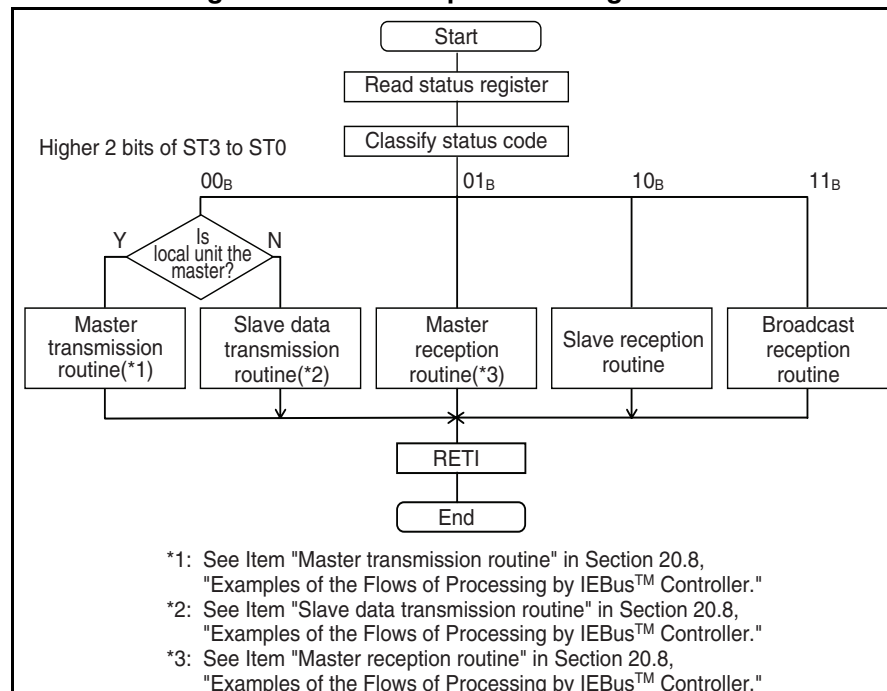
Figure 21.7-1 Main Routine



■ Interrupt Processing Routine

Figure 21.7-2 "Interrupt Processing Routine" shows an example of the routine that is executed by the IEBus™ controller when transmission is started or a reception completion interrupt occurs. In this routine, the status code set in the ST3 to ST0 bits of the status register is read, transmission data is written, and received data is read.

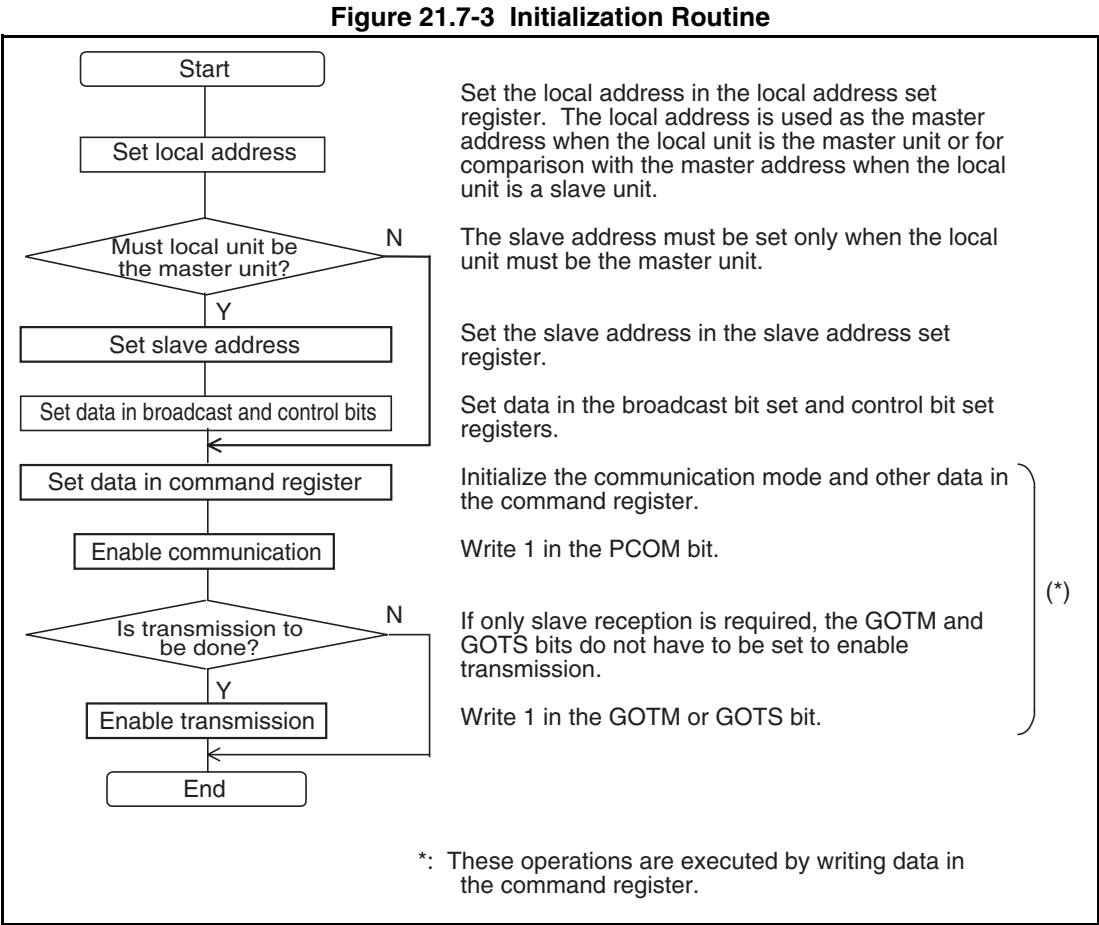
Figure 21.7-2 Interrupt Processing Routine



21.7.1 Initialization Routine

Figure 21.7-3 "Initialization Routine" shows the flow of IEBus™ initialization. In this routine, the local address and a slave address are set, data is set in the command register, and communication is enabled. The slave address, broadcast bits, and control bits must be set only when the local unit must be the master unit.

■ Initialization Routine

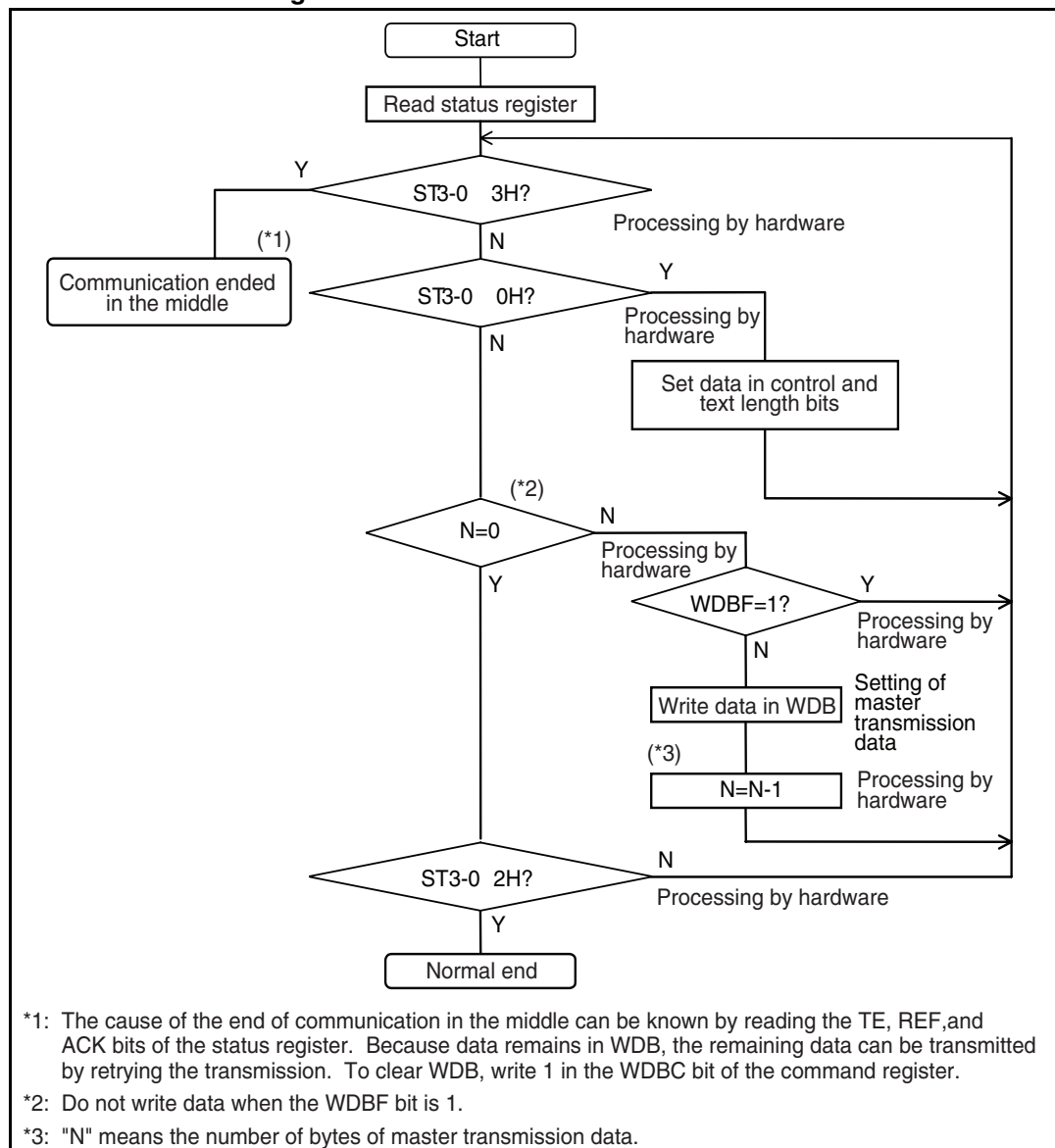


21.7.2 Master Transmission Routine

Figure 21.7-4 "Master Transmission Routine" shows the flow of the master transmission routine. The master transmission routine is used after communication is enabled when the local unit has succeeded in arbitration and become the master unit and transmits data to a slave unit. This routine is executed when the master transmission state code (higher 2 bits are 00B) is set in the ST3 to ST0 bits of the status register in the interrupt processing routine.

■ Master Transmission Routine

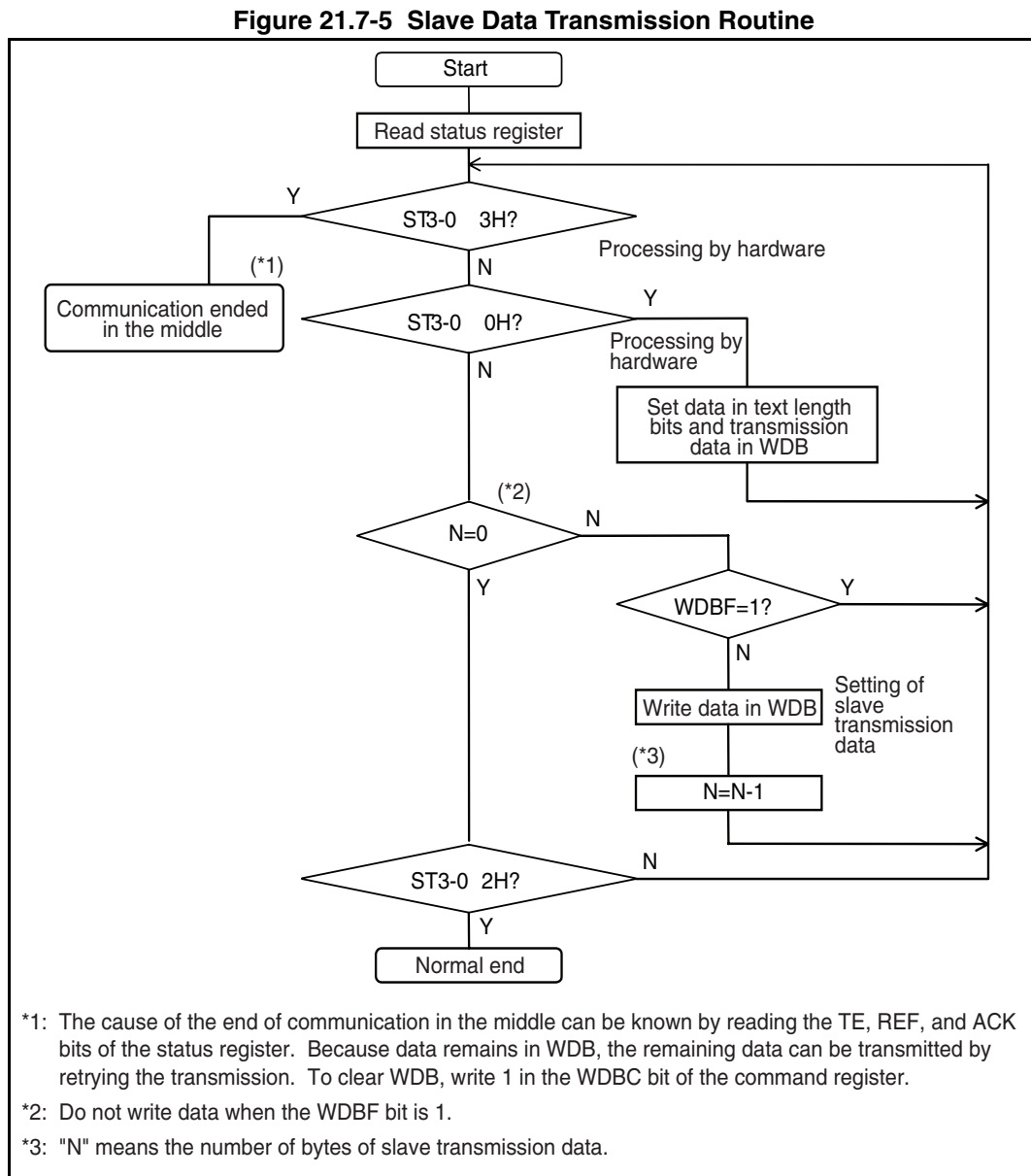
Figure 21.7-4 Master Transmission Routine



21.7.3 Slave Data Transmission Routine

Figure 21.7-5 "Slave Data Transmission Routine" shows the flow of the slave data transmission routine. The slave data transmission routine is used when the slave unit transmits data to the master unit after the slave unit receives control bits from the master unit. This routine is executed when the slave data transmission state code (higher 2 bits are 00_B) is set in the ST3 to ST0 bits of the status register in the interrupt processing routine.

■ Slave Data Transmission Routine



21.7.4 Master Reception Routine

The master reception routine is used when the master unit receives data, slave status, or lock address from the slave unit after the master unit has started data transmission.

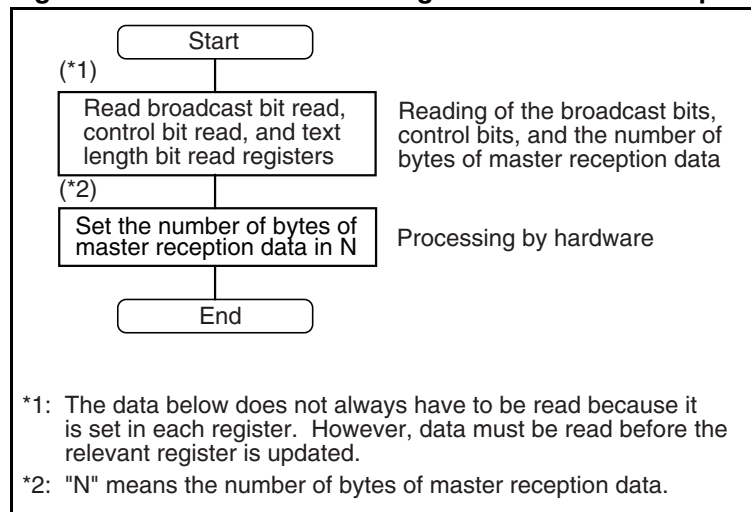
■ Master Reception Routine

The master reception routine consists of the following four routines executed according to the contents of the ST3 to ST0 bits of the status register:

○ Processing to be performed when status code is "4_H" (start of master reception)

Figure 21.7-6 "Flow of Processing to Start Master Reception" shows the flow of the processing to be executed when status code is "4_H" (start of master reception). This status code is set when the master unit has received the text length bits normally from the slave unit and indicates the start of master reception. (No interrupt occurs.)

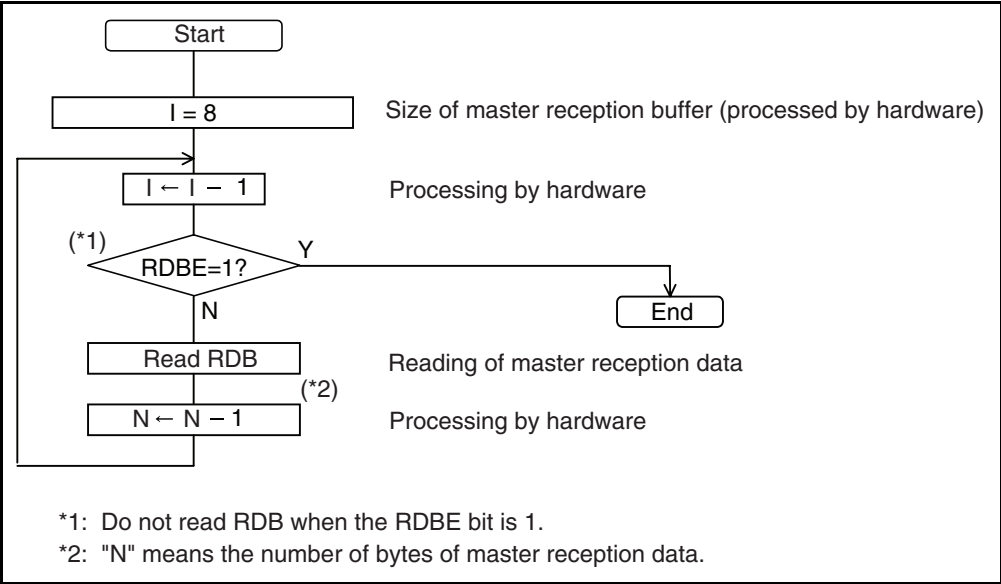
Figure 21.7-6 Flow of Processing to Start Master Reception



○ Processing to be performed when status code is "5_H" (request for reading master reception data)

Figure 21.7-7 "Flow of Processing of Master Reception Data Reading" shows the flow of the processing to be executed when status code is "5_H" (request for reading master reception data).

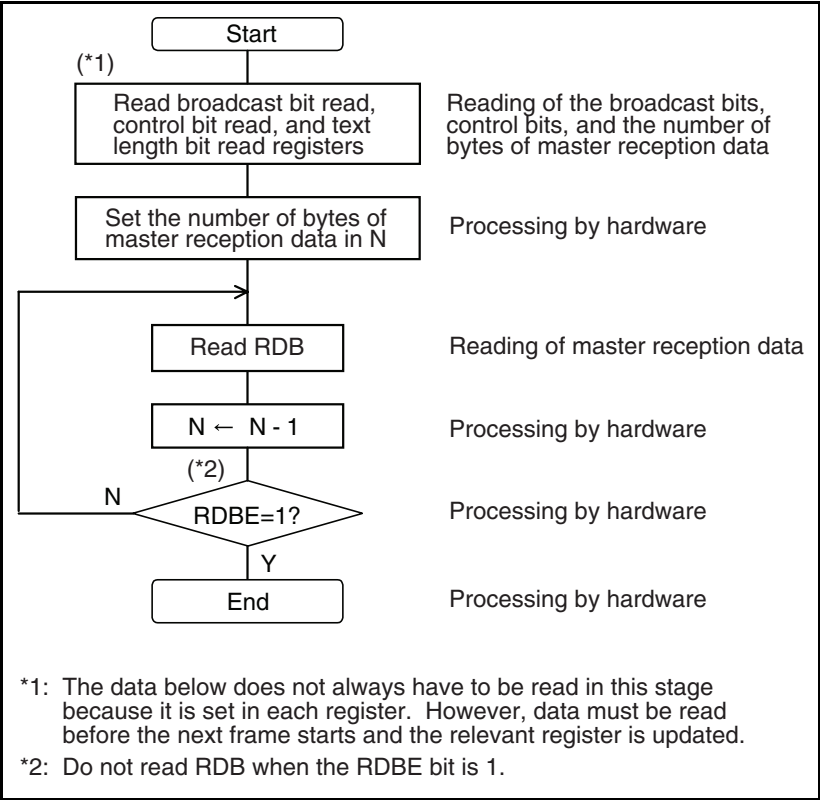
Figure 21.7-7 Flow of Processing of Master Reception Data Reading



○ Processing to be performed when status code is "6_H" (normal end of master reception)

Figure 21.7-8 "Flow of Processing of the Normal End of Master Reception" shows the flow of the processing to be executed when status code is "6_H" (normal end of master reception).

Figure 21.7-8 Flow of Processing of the Normal End of Master Reception



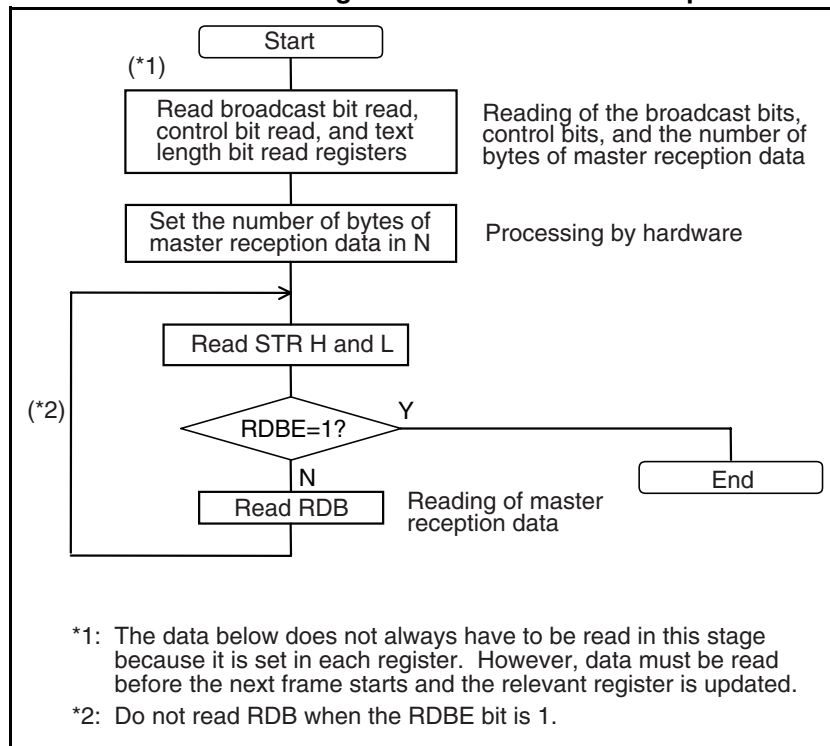
21.7 Examples of the Flows of Main and Interrupt Processing Routines for IEBus™ Controller

○ Processing to be performed when status code is "7_H" (end of master reception in the middle)

Figure 21.7-9 "Flow of Processing of the End of Master Reception in the Middle" shows the flow of the processing to be executed when status code is "7_H" (end of master reception in the middle).

Note that the routines for slave reception and broadcast reception are the same as those for master reception, but the status codes to be used are different. For the status codes, see Section 21.6 "Communication Control Status".

Figure 21.7-9 Flow of Processing of the End of Master Reception in the Middle



21.8 IEBus™ Controller Operation at Transmission

Figure 21.8-1 "Operation when WDBC is set to 1 (on the master for master transmission)" and Figure 21.8-2 "Operation when WDBC is set to 0 (on the master for master transmission)" show the transmission operations over multiple frames. Figure 21.8-3 "Example of Operation at Occurrence of Error on Slave Unit" and Figure 21.8-4 "Example of Operation at Occurrence of Error on Master Unit" show the transmission operations when an error occurs.

■ Transmission Operation over Multiple Frames

Figure 21.8-1 Operation when WDBC is set to 1 (on the master for master transmission)

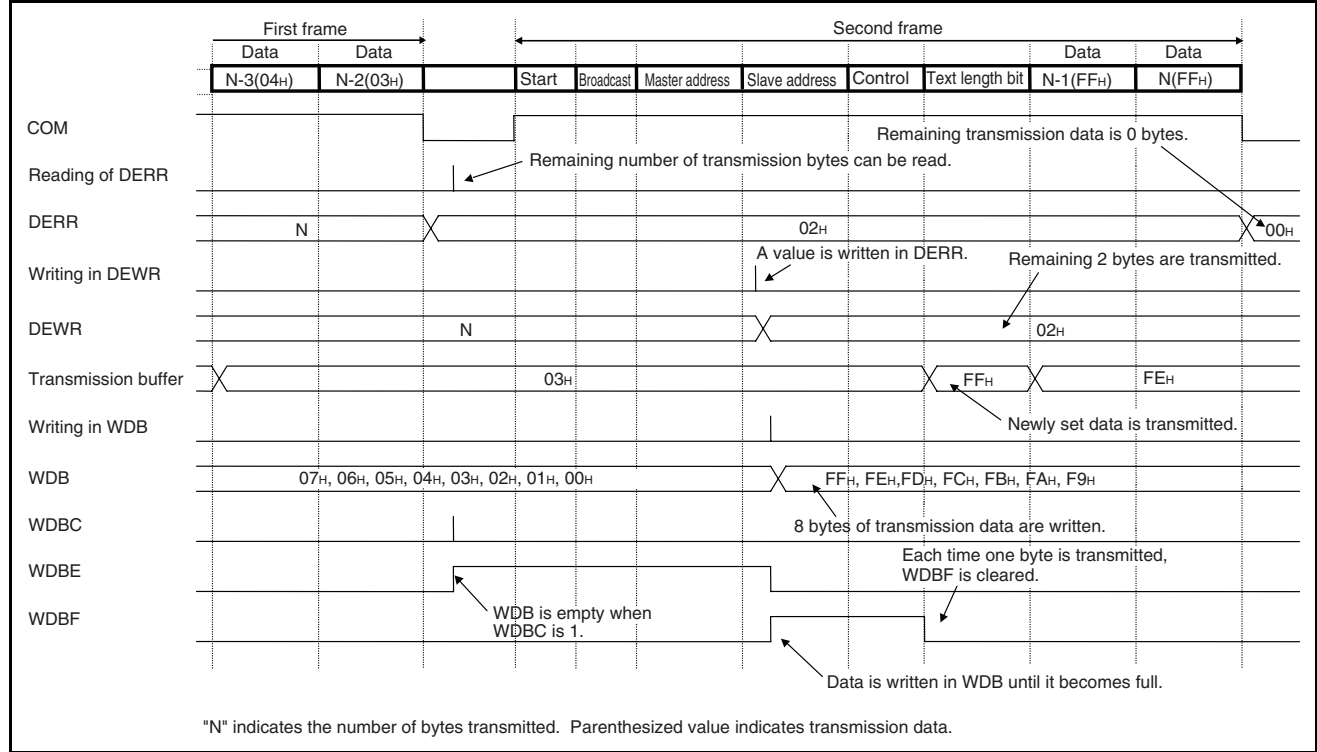
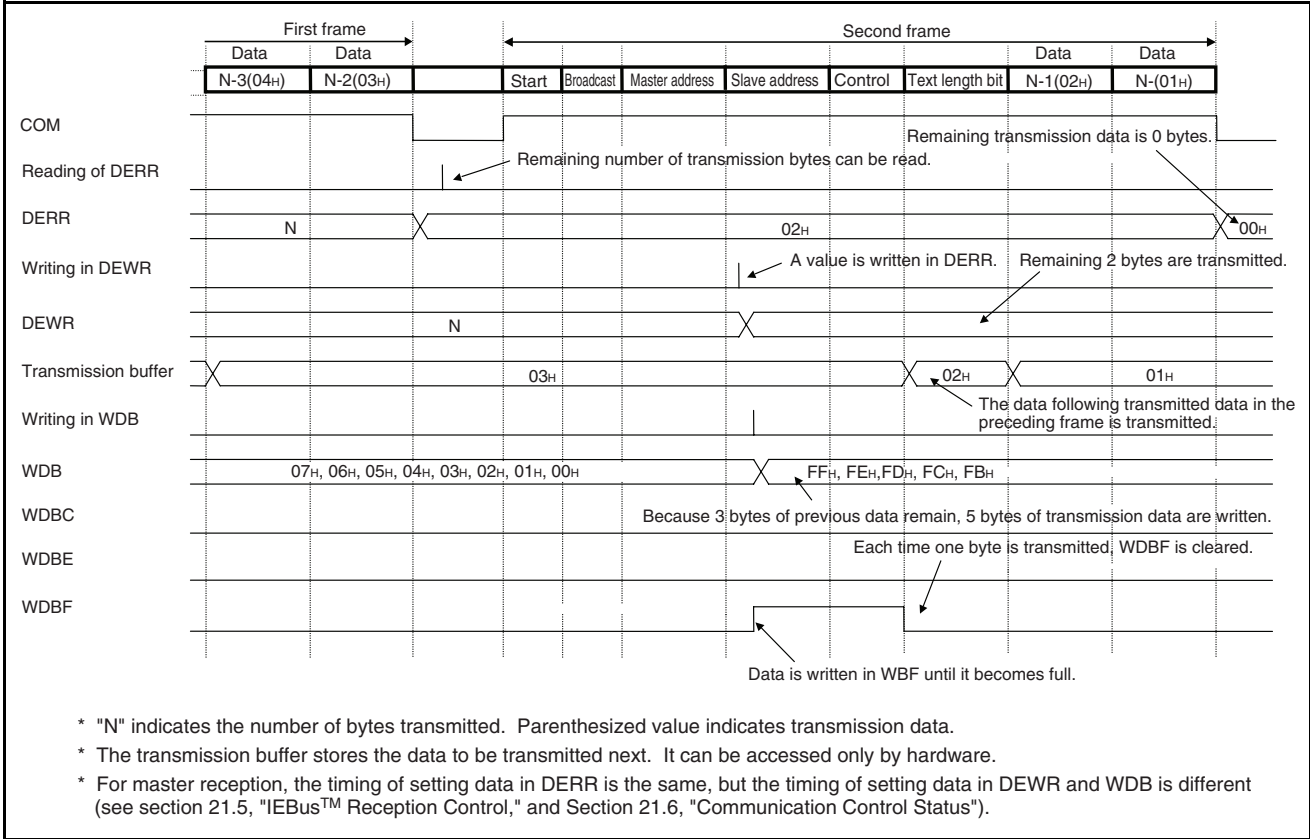


Figure 21.8-2 Operation when WDBC is set to 0 (on the master for master transmission)



■ Transmission Operation at Occurrence of Error

Figure 21.8-3 "Example of Operation at Occurrence of Error on Slave Unit" shows an example of the master transmission operation in which an error occurs in data in the second byte on the slave unit, the master unit receives NAK, and the master unit transmits subsequent data in the second frame.

Figure 21.8-3 Example of Operation at Occurrence of Error on Slave Unit

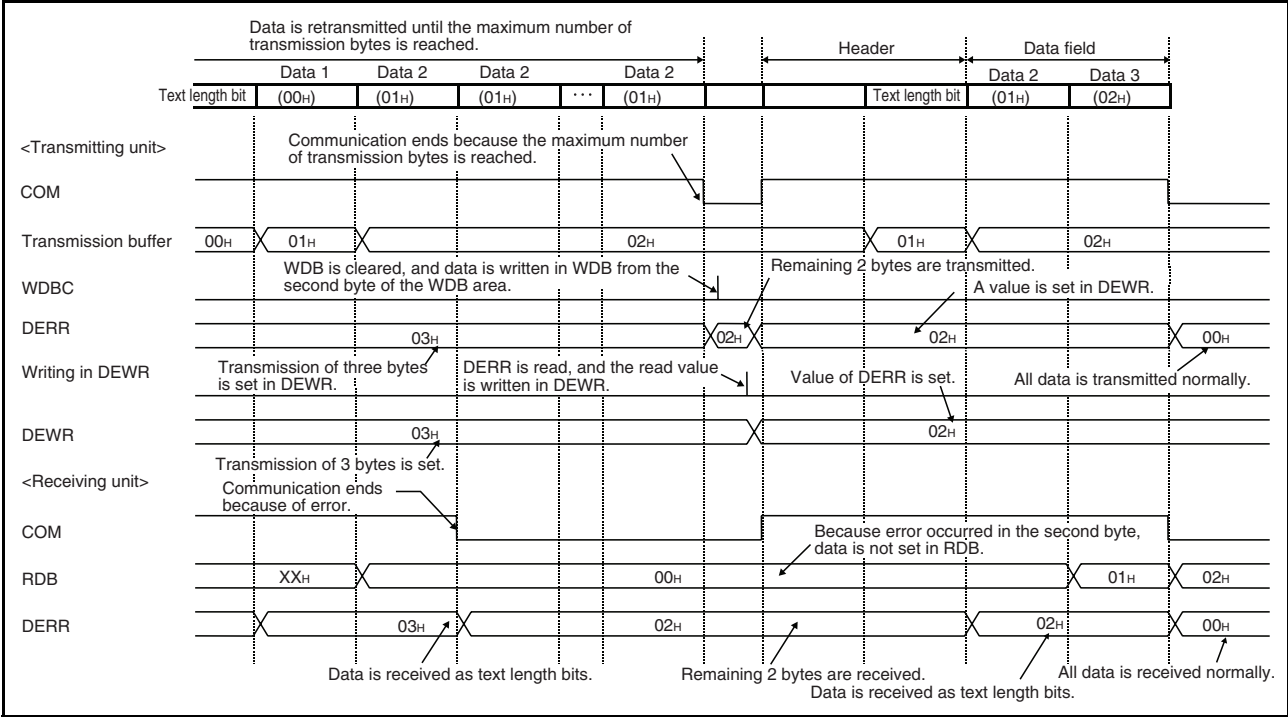
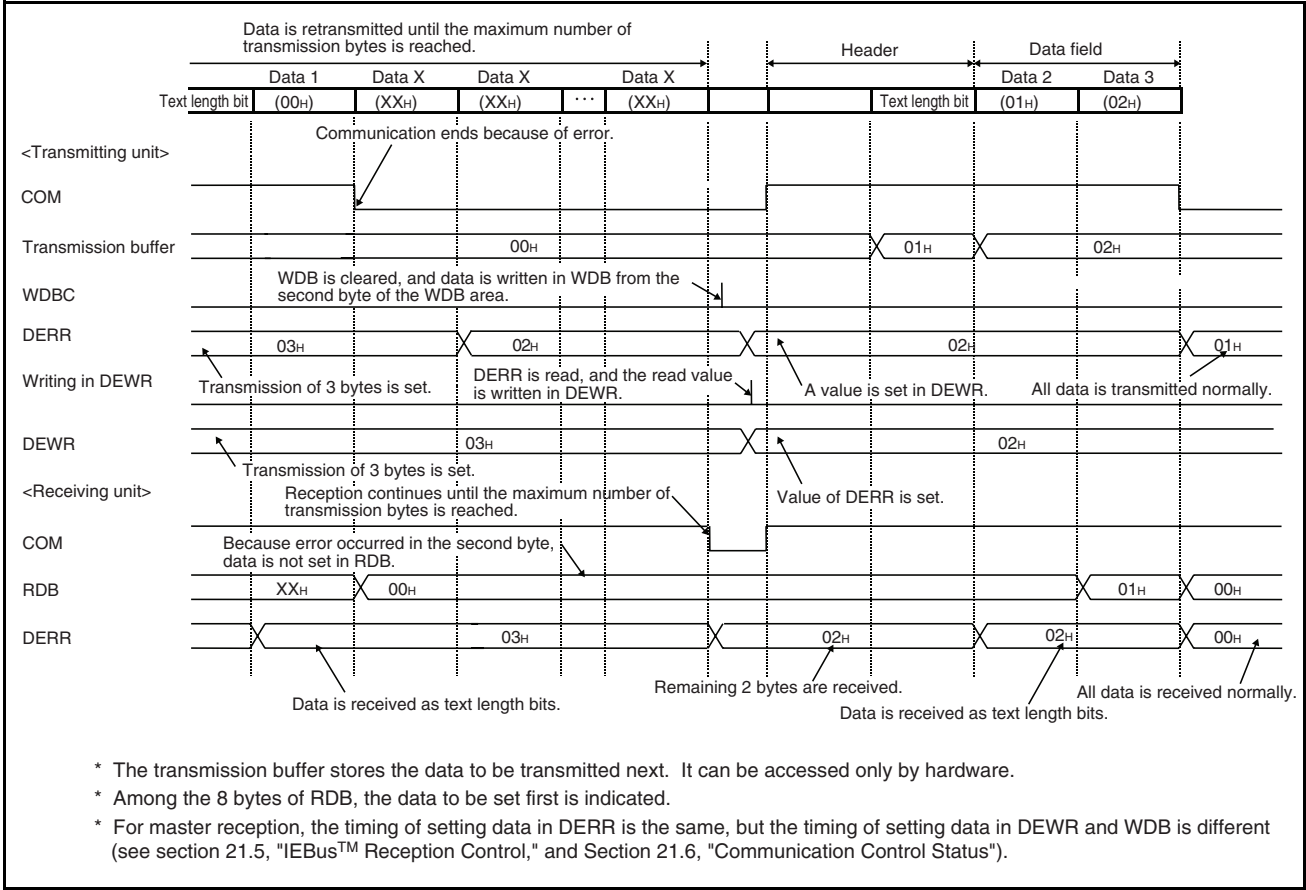


Figure 21.8-4 "Example of Operation at Occurrence of Error on Master Unit" shows an example of the master transmission operation in which an error occurs in data in the second byte on the master unit and the master unit transmits subsequent data in the second frame.

Figure 21.8-4 Example of Operation at Occurrence of Error on Master Unit



21.9 IEBus™ Protocol Operation

The Inter-Equipment Bus (IEBus™) is a small-scale digital data transmission system bus designed for data transmission between different equipment.

■ Overview of IEBus™ Protocol Operation

The following gives an overview of the IEBus™ protocol operation:

○ Communication mode

Half-duplex asynchronous communication

○ Multi-master mode

All the units connected to IEBus™ can transmit data to each other.

○ Broadcast function (communication between one unit and multiple units)

- Group broadcast: Broadcast to a group of units
- General broadcast: Broadcast to all other units

○ Selection from three modes using different transmission speeds

Table 21.9-1 Transmission speeds by mode of IEBus™ protocol operation

	Internal frequency of IEBus™: 6 MHz	Internal frequency of IEBus™: 6.29 MHz	Maximum number of transmission bytes (number of bytes per frame)
Mode 0	About 3.9 K(bps)	About 4.1 K(bps)	16
Mode 1	About 17 K(bps)	About 18 K(bps)	32
Mode 2	About 26 K(bps)	About 27 K(bps)	128

○ Access control

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

○ Priority for the use of bus

1. Broadcast has priority over normal communication (i.e., communication between a unit and another unit).
2. A unit with a master address has priority over all the other units with higher master addresses.

○ Scale of communication

- Maximum number of units: 50
- Maximum cable length: 150 m (using a twisted-pair cable with resistance of 0.1 Ω /m or less)

- Maximum load capacity:
 - 8,000 pF (between BUS- and BUS+) at internal frequency of IEBus™ of 6 MHz
 - 7,100 pF (between BUS- and BUS+) at internal frequency of IEBus™ of 6.29 MHz
- Terminating resistance: 120 Ω

The scale of communication is the one on the system that includes the IEBus™ driver and receiver.

■ Determining the Priority for Using the Bus (Arbitration)

A unit connected to IEBus™ executes an operation to use the bus exclusively for the control of other units. This operation is called "arbitration".

When multiple units start transmission at the same time, arbitration is executed to give one of those units the right to use the bus exclusively.

By arbitration, the unit that is given the right to use the bus exclusively is determined under the following priority conditions for the exclusive use of the bus:

○ Priority due to communication type

Broadcast (communication between a unit and multiple units) has priority over normal communication (communication between two units).

○ Priority due to master address

When the communication type is the same, a unit with a master address has priority over all the other units with higher master addresses.

The master address consists of 12 bits. The unit with master address of 000_H has the highest level of priority; the unit with master address of FFF_H has the lowest level of priority.

■ Communication Modes

IEBus™ supports three communication modes that are different in transmission speed. The table below lists the communication modes, transmission speeds, and the maximum numbers of transmission bytes per frame.

Table 21.9-2 Transmission speed and maximum number of transmission bytes in each communication mode

Communication mode	Maximum number of transmission bytes (number of bytes per frame)	Effective transmission speed *1	
		Internal frequency of IEBus™ *2	Internal frequency of IEBus™ *2
0	16	About 3.9 K(bps)	About 4.1 K(bps)
1	32	About 17 K(bps)	About 18 K(bps)
2	128	About 26 K(bps)	About 27 K(bps)

*1:

This indicates the effective transmission speed available for the transmission of the maximum number of transmission bytes.

*2:

For the relation between the internal frequency of IEBus™ and the internal frequency of CPU, see the calculation formula shown in Table 21.3-4 "Settings of GOTM and GOTS bits".

Note:

- A communication mode must be selected on each unit connected to IEBus™ before starting communication. If a master unit and a slave unit use different communication modes, normal communication between the units is not possible.
- If the internal frequency of IEBus™ set on a unit differs from the one set on another unit, normal communication between these units is not possible even if the same communication type is used. The same internal frequency must be set on every unit.

■ Communication address

On IEBus™, a unique 12-bit communication address is given to each unit. The communication address consists of the following numbers:

- Higher 4 bits: Group number (identifying the group where each unit belongs)
- Lower 8 bits: Unit number (identifying each unit in the relevant group)

■ Broadcast

In normal communication, a master unit transmits or receives data to/from a slave unit. In broadcast, a master unit transmits data to multiple slave units, and the slave units do not return acknowledgment signals to the master unit.

The communication type, broadcast or normal communication, can be selected by the setting of the broadcast control bits.

Broadcast is available in two kinds as follows:

○ Group broadcast

Group broadcast transmits data to the slave units having the same group number (higher 4 bits of communication address).

○ General broadcast

General broadcast transmits data to all slave units regardless of their group numbers. Group broadcast and general broadcast can be distinguished by the value of the slave address.

21.10 Transmission Protocol

Table 21.10-1 "Transmission protocol signal format" shows the format of transmission protocol signals of IEBus™.

Communication data is transmitted as a series of signals called the "frame". The number of bytes of data that can be transmitted by a frame and transmission speed vary depending on the communication mode.

■ Transmission Protocol

Table 21.10-1 Transmission protocol signal format

Field name	Header		Master address field		Slave address field			Control field			Text length field			Data field							
Number of bits	1	1	12	1	12	1	1	4	1	1	8	1	1	8	1	1	8	1	1		
Transmission time	Start bit	Broad-cast bit	Master address	P	Slave address	P	A	Control bits	P	A	Text length bits	P	A	Data (1 byte)	P	A	...	Data (1 byte)	P	A	
Mode 0	About 7330 μs													About 1590 x Nμs							
Mode 1	About 2090 μs													About 410 x Nμs							
Mode 2	About 1590 μs													About 300 x Nμs							

P: Parity bit (1 bit)

N: Number of data bytes

A: Acknowledgment bit (1 bit)

0: ACK

1: NAK

* For broadcast, the value of each acknowledgment bit is ignored.

21.10.1 Header in Transmission Protocol Signal Format

The header in the transmission signal format consists of a start bit and a broadcast bit.

■ Start Bit

The start bit is the signal to indicate the start of data transmission to other units. A unit that is to start data transmission outputs a low-level signal (start bit) for a specified time, then outputs the next signal (broadcast bit).

If a unit has already output the start bit signal when another unit attempts to output its start bit signal, the latter unit stops start bit output and waits until the former unit ends start bit output. In synchronization with the end of start bit output, the latter unit outputs the broadcast bit signal. A unit other than the unit that has started transmission detects the start bit output and enters the receiving state.

■ Broadcast Bit

The broadcast bit is used to identify the communication type as either broadcast or normal communication. When the broadcast bit is 0, the communication type is broadcast. When it is 1, the communication type is normal. Broadcast is available in two types, group broadcast and general broadcast, which are distinguishable by the value of the slave address.

Because multiple slave units are communication partners in broadcast, the acknowledgment bit in each field is not returned from each slave unit.

If two or more units start transmitting frames at the same time, broadcast has priority over normal communication and succeeds in arbitration.

21.10.2 Master Address Field in Transmission Protocol Signal Format

The master address field in the transmission signal format determines the master unit.

■ Master Address Field

The master address field consists of a 12-bit master address and a parity bit.

If two or more units start transmission with the same broadcast bit value at the same time, determination by arbitration is undertaken in the master address field. In the master address field, the local unit compares each bit it outputs with the data on the bus. If the master address the local unit outputs is different from the data on the bus, the local unit determines that it has failed in arbitration, stops transmission, and enters the receiving state.

IEBusTM has a wired AND logic. The unit having the lowest master address among the units participated in arbitration (arbitrating master units) succeeds in arbitration.

After the units have output 12-bit master addresses, only one unit remains the master unit. The master unit outputs a parity bit (*) to determine the master address for other units, then starts the output of the slave address field.

A master address consists of 12 bits, and its output begins with the MSB.

*: IEBusTM uses an even parity. The parity bit is set when the master address field contains an odd number of set bits.

21.10.3 Slave Address Field in Transmission Protocol Signal Format

The slave address field in the transmission signal format determines the slave unit.

■ Slave Address Field

The slave address field consists of a 12-bit slave address, a parity bit, and an acknowledgment bit.

A slave address consists of 12 bits, and its output begins with the MSB. After the output of a 12-bit slave address, the master address outputs a parity bit to avoid incorrect reception of the slave address. The master unit then attempts to detect the acknowledgment signal from the slave unit to confirm that the slave unit exists on the bus. When the master unit detects the acknowledgment signal, it starts the output of the control field. For broadcast, the master unit starts the output of the control field without detecting an acknowledgment signal.

The slave unit outputs an acknowledgment signal when it has detected that the slave address transmitted matches its own slave address and that parities for the master address and the slave address are even. If a parity is odd, the slave unit determines that the master or slave address has not been received correctly and does not output an acknowledgment signal. If it occurs, the master unit enters the standby (monitoring) state and communication ends.

For broadcast, the slave address is used to identify the broadcast type, group or general broadcast, as follows:

- Slave address is FFF_H : General broadcast
- Slave address is an address other than FFF_H : Group broadcast

The group number applied to a group broadcast is the value set in the higher 4 bits of the slave address.

21.10.4 Control Field in Transmission Protocol Signal Format

The control field in the transmission signal format determines the type and direction of the data field.

■ Control Field

The control field consists of four control bits, a parity bit, and an acknowledgment bit.

The four control bits are output sequentially from the MSB. After the control bits, the parity bit is output. When the slave unit determines that the parity is even and it can execute the function requested by the master unit, it outputs an acknowledgment signal and starts receiving the text length bit field. If the slave unit cannot execute the function requested by the master unit when the parity is even or if the parity is odd, the slave unit does not output an acknowledgment signal and enters the standby state.

After confirming the acknowledgment signal, the master unit starts the output of the text length bit field. If the master unit cannot confirm the acknowledgment signal, the master unit enters the standby (monitoring) state and communication ends. For broadcast, the master unit starts the output of the text length bit field without confirming an acknowledgment signal.

21.10.5 Text Length Field

The text length field specifies the number of bytes of transmission data.

■ Text Length Field

The text length field consists of eight text length bits, a parity bit, and an acknowledgment bit. The eight text length bits are output sequentially from the MSB. Table 21.10-2 "Settings of text length bits" lists the numbers of bytes of transmission data that can be specified by the text length bits.

Table 21.10-2 Settings of text length bits

Text length bits (hexadecimal)	Number of bytes of transmission data
01 _H	1 byte
02 _H	2 bytes
:	:
FF _H	255 bytes
00 _H	256 bytes

Remarks:
If a value over the maximum number of transmission bytes per frame set by the communication mode is specified, multiple frames are used for communication. If it occurs, the value of the text length bits for the second and subsequent frames will be the remaining number of bytes of transmission data.

The operation of the text length bit field is different between master transmission (control bit 3 is 1) and master reception (control bit 3 is 0).

○ Master transmission

The master unit outputs the text length and parity bits. The slave unit outputs an acknowledgment signal when it detects an even parity, then starts receiving the data field.

For broadcast, if the parity is odd, the slave unit determines that the text length bits have not been received correctly and enters the standby state without returning an acknowledgment signal. If this occurs, the master unit also enters the standby (monitoring) state, and communication ends.

○ Master reception

The slave unit outputs the text length and parity bits. The master unit outputs an acknowledgment signal when it detects an even parity.

If the parity is odd, the master unit determines that the text length bits have not been received correctly and enters the standby state without returning an acknowledgment signal. If this occurs, the slave unit also enters the standby state, and communication ends.

21.10.6 Data Field

The data field is used to transmit and receive data according to control bits.

■ Data Field

The data field consists of 10-bit sets of eight data bits, a parity bit, and an acknowledgment bit.

The eight data bits are output sequentially from the MSB. After the data bits, the parity bit is output by the master unit and the acknowledgment bit is output by the slave unit.

For broadcast, only transmission operation of the master unit is performed and acknowledgment signals are ignored.

The operation of the data field is different between master transmission and master reception as follows:

○ Master transmission

When the master unit transmits data to the slave unit, the master unit transmits the data and parity bits to the slave unit. When the slave unit receives the data and parity bits and detects that the parity is even and the reception buffer has a space, the slave unit outputs an acknowledgment signal. If the parity is odd or the reception buffer is full, the slave unit rejects the reception of the relevant data and does not output an acknowledgment signal.

If the slave unit does not return an acknowledgment signal, the master unit transmits the same data again. The master unit repeats this operation until it detects the acknowledgment signal from the slave unit or the size of data exceeds the maximum number of bytes of transmission data. When the slave unit has detected an even parity and output an acknowledgment signal, the master unit transmits the next data if the next data exists and does not exceeds the maximum number of bytes of transmission data.

For broadcast, the slave units do not output an acknowledgment signal, and the master unit transmits data in units of bytes.

○ Master reception

When the master unit receives data from the slave unit, the master unit outputs the synchronization signals corresponding to all bits to be read.

The slave unit outputs the data and parity bits to the bus according to the synchronization signals from the master unit.

The master unit reads the data and parity bits output by the slave unit and checks the parity.

If the parity is odd or the reception buffer is full, the master unit rejects the reception of the relevant data and does not output an acknowledgment signal. The master unit repeats the reading of the same data if the data does not reach the maximum number of bytes of data that can be transmitted in one frame.

If the parity is even and the reception buffer has a space, the master unit receives data and returns an acknowledgment signal. The master unit then reads the next data if the data does not reach the maximum number of bytes of data that can be transmitted in one frame.

21.10.7 Parity Bit

The parity bit is used to check whether transmission data is correct.

The parity bit is added to the master address, slave address, control, text length, and data bits.

■ Parity Bit

IEBus™ uses an even parity. The parity bit is set when the relevant data contains an odd number of set bits. The parity bit is cleared when the relevant data contains an even number of set bits.

21.10.8 Acknowledgment Bit

For normal communication (between two units), the acknowledgment bit is added in each of the following positions to check that data has been received normally:

- End of slave address field
 - End of control field
 - End of text bit field
 - End of data field
-

■ Acknowledgment Bit

The acknowledgment bit is defined as follows:

0 (ACK): Indicates that transmission data was acknowledged.

1 (NAK): Indicates that transmission data was not acknowledged.

○ Acknowledgment bit at the end of the slave address field

When one of the following conditions is satisfied, the acknowledgment bit at the end of the slave address field is set (NAK) and communication is canceled:

- The parity for master address or slave address bits is invalid.
- A timing error (bit format error) has occurred.
- The specified slave unit is not found.

○ Acknowledgment bit at the end of control field

When one of the following conditions is satisfied, the acknowledgment bit at the end of the control field is set (NAK) and communication is canceled:

- The parity for control bits is invalid.
- Control bit 3 is 1 (writing operation) although the slave reception buffer (see Note) is not empty.
- Control bit data is 3_H or 7_H although the slave reception buffer (see Note) is empty.
- Although the local unit is locked, the control bit data of 3_H, 6_H, 7_H, A_H, B_H, E_H, or F_H has been received from a unit other than the locking unit.
- Although the local unit is not locked, the control bit data read from the lock address is 4_H.
- A timing error has occurred.
- An undefined value is set in the control bits.

Note:

For the slave reception buffer, see Item "Reading slave status (SSR) (control bits: 0_H or 6_H)" of Section 21.11 "Transmission Data".

○ Acknowledgment bit at the end of the text length field

When one of the following conditions is satisfied, the acknowledgment bit at the end of the text length field is set (NAK):

- The parity for text length bits is invalid.
- A timing error has occurred.

○ Acknowledgment bit at the end of the data field

When one of the following conditions is satisfied, the acknowledgment bit at the end of the data field is set (NAK):

- The parity for data bits is invalid (see Note).
- A timing error occurred after the previous output of an acknowledgment bit.
- The reception buffer is full and additional data cannot be received [JH1] (see Note).

Note:

Even in this case, the transmitting unit retries transmission of the relevant data field until the data reaches the maximum number of bytes of data that can be transmitted in one frame.

21.11 Transmission Data

The data field contains the transmission data specified by the control bits.

■ Transmission Data

Table 21.11-1 Settings of Control Bits

	Bit 3 ^{*1}	Bit 2	Bit 1	Bit 0	Function ^{*2}
0 _H	0	0	0	0	Slave status reading
1 _H	0	0	0	1	Undefined
2 _H	0	0	1	0	Undefined
3 _H	0	0	1	1	Data reading and locking
4 _H	0	1	0	0	Lock address reading (lower 8 bits)
5 _H	0	1	0	1	Lock address reading (higher 4 bits)
6 _H	0	1	1	0	Slave status reading and unlocking
7 _H	0	1	1	1	Data reading
8 _H	1	0	0	0	Undefined
9 _H	1	0	0	1	Undefined
A _H	1	0	1	0	Command writing and locking
B _H	1	0	1	1	Data writing and locking
C _H	1	1	0	0	Undefined
D _H	1	1	0	1	Undefined
E _H	1	1	1	0	Command writing
F _H	1	1	1	1	Data writing

*1: The value of bit 3 (MSB) determines the direction of transmission of the text length bits in the text length field and the data in the data field.

- When bit 3 is 1, data is transmitted from the master unit to the slave unit.
- When bit 3 is 0, data is transmitted from the slave unit to the master unit.

*2: 3_H, 6_H, A_H, and B_H are control bit settings for locking and unlocking. If an undefined value 1_H, 2_H, 8_H, 9_H, C_H, or D_H is transmitted, no acknowledgment is returned.

When a slave unit has been locked by the master unit, the unit rejects reception of data if the control bit value received from a unit other than the master unit is a value other than those values listed in Table 21.11-2 "Control Fields that can be Received by Locked Slave Units". Then the unit does not output an acknowledgment signal.

Table 21.11-2 Control Fields that can be Received by Locked Slave Units

	Bit 3	Bit 2	Bit 1	Bit 0	Function
0 _H	0	0	0	0	Slave status reading
4 _H	0	1	0	0	Lock address reading (lower 8 bits)
5 _H	0	1	0	1	Lock address reading (higher 4 bits)

■ Reading Slave Status (SSR) (Control Bits: 0H or 6H)

The master unit can know why the slave unit did not return the acknowledgment signal (ACK) by reading the slave status (control bit value: 0_H or 6_H).

The slave status of a slave unit is determined in relation to the results of the last communication the slave unit executed. All slave units can provide slave status information. Figure 21.11-1 "Bit Configuration of Slave Status" shows the bit configuration of slave status information.

Figure 21.11-1 Bit Configuration of Slave Status

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Table 21.11-3 Meanings of Slave Status

Bit	Value	Meaning	
Bit 0 ^{*1}	0	The slave transmission buffer is empty.	
	1	The slave transmission buffer is not empty.	
Bit 1 ^{*2}	0	The slave reception buffer is not full.	
	1	The slave reception buffer is full.	
Bit 2	0	The unit is not locked.	
	1	The unit is locked.	
Bit 3	0	Always 0	
Bit 4 ^{*3}	0	Slave transmission is stopped.	
	1	Slave transmission is enabled.	
Bit 5	0	Always 0	
Bits 6 and 7	00	Mode 0	These bits indicate the highest level of mode supported by the unit. ^{*4}
	01	Mode 1	
	10	Mode 2	
	11	Setting inhibited	

^{*1}: The slave transmission buffer is the buffer that is accessed when data is read (control bit value: 3_H or 7_H). It corresponds to the write data buffer (WDB).

^{*2}: The slave reception buffer is the buffer that is accessed when data is written (control bit value: 8_H, A_H, B_H, E_H, or F_H). It corresponds to the read data buffer (WDB).

^{*3}: The setting of this bit can be selected by the operation of the PCOM bit of the command register.

^{*4}: On the MB90580C series, the setting of these bits is always 10.

■ Transferring Data or Commands (Control Bits: 3_H or 7_H [Reading] or A_H, B_H, E_H, or F_H [Writing])

When the value of control bits is 3_H or 7_H (data reading), the data stored in the data buffer of the slave unit is read by the master unit.

When the value of control bits is B_H or F_H (data writing) or AE or E_H (command writing), the data received by a slave unit is processed according to the operation specifications of the slave unit.

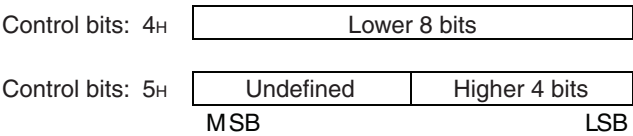
Remarks:

1. The user can freely specify the data or command to be transferred depending on the user system.
2. Under specific communication conditions or status, locking is set for the operations indicated by control bit values such as 3_H, A_H, and B_H.

■ Reading the Lock Address (Control Bits: 4_H or 5_H)

When the value of control bits is 4_H or 5_H (lock address reading), the 12-bit address of the master unit that issued the relevant lock instruction is read. The address to be read has a one-byte structure as shown in Figure 21.11-2 "Structure of lock address".

Figure 21.11-2 Structure of lock address



■ Locking and Unlocking

The lock function is used when a unit transmits a message to another unit over multiple frames. A locked unit does not receive data from units other than the unit that locked it.

Locking and unlocking are performed as follows:

○ Locking

A slave unit is locked by the master unit when a frame ends without all the data of the number of bytes specified by the text length bits transmitted or received after transmission and reception of the acknowledgment (ACK) for the text length field with a control bit value (3_H, A_H, or B_H) specifying locking. If this occurs, bit 2 (indicating locking) among the slave status bits is set to 1.

○ Unlocking

A slave unit is unlocked by the master unit when all the data of the number of bytes specified by the text length bits was transmitted or received within one frame with a control bit value (3_H, A_H, or B_H) specifying locking or (6_H) specifying unlocking. If this occurs, bit 2 (indicating locking) among the slave status bits is cleared to 0.

Locking and unlocking are not used for broadcast.

21.12 Bit Format

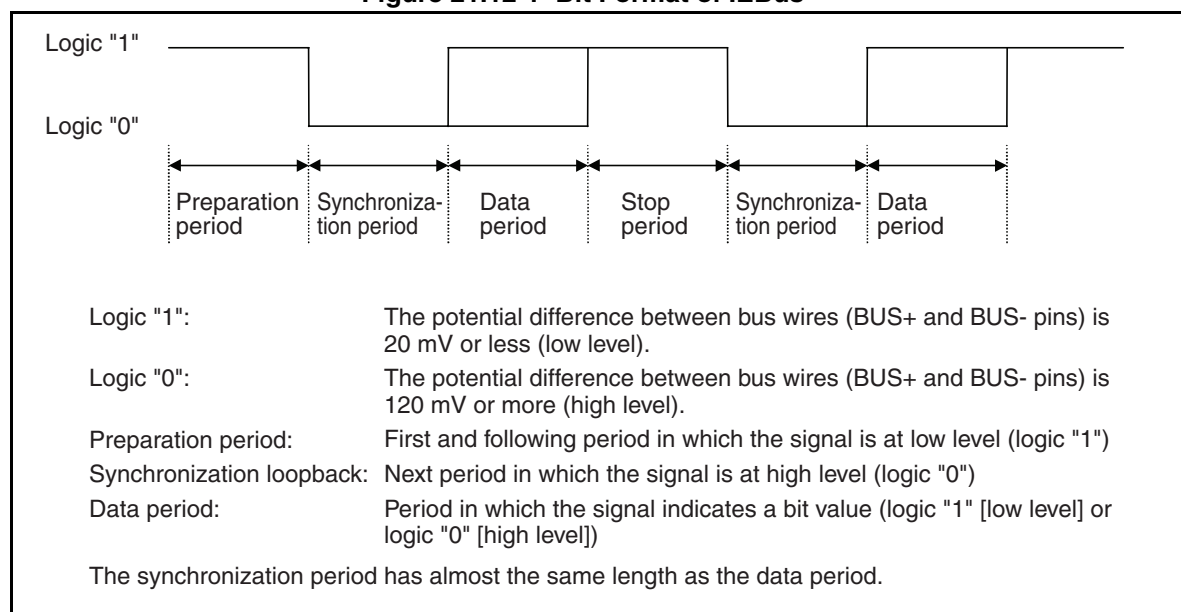
IEBus™ sets synchronization in units of bits. The specifications of the time assigned to the whole bit and the time assigned to each period within each bit is dependent on the type of transmission bit and the type of unit, i.e., master or slave unit.

During communication, each unit, master or slave, checks all periods (e.g., preparation, synchronization, and data periods) to determine whether each signal is output for the specified time. If a signal is not output for a specified time, the master or slave unit detects a timing error, stops communication immediately, and enters the standby state.

■ Bit Format

Figure 21.12-1 "Bit Format of IEBus™" shows the bit format of the frame of IEBus™.

Figure 21.12-1 Bit Format of IEBus™



CHAPTER 22 CLOCK MONITOR FUNCTION

This chapter describes the functions and operation of the clock monitor.

22.1 "Overview of the Clock Monitor Functions"

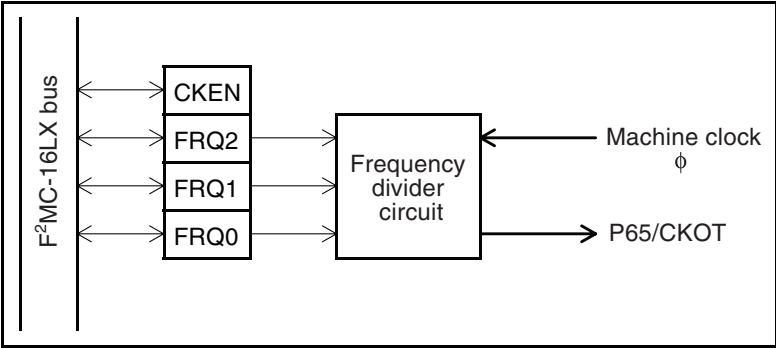
22.2 "Clock Output Permission Register (CLKR)"

22.1 Overview of the Clock Monitor Functions

The clock monitor function is to output from the CKOT pin a divided clock of the machine clock(clock for monitoring) that is running in main clock , PLL clock, or subclock mode.

■ Block Diagram of the Clock Monitor Functions

Figure 22.1-1 Block Diagram of the Clock Monitor Functions



22.2 Clock Output Permission Register (CLKR)

The bits of the clock output permission register (CLKR) are used for selection of the CKOT output permission and clock output frequency.

■ Clock Output Permission Register (CLKR)

Figure 22.2-1 Clock Output Permission Register (CLKR)

Clock output permission 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 3] CKEN

The CKEN bit is the CKOT output permission bit.

Table 22.2-1 Functions of CKEN Bit

CKEN	Function
0	Normal port
1	CKOT output

[Bit 2,1,and 0] FRQ2, FRQ1, and FRQ0

The FRQ2, FRQ1, and FRQ0 bits are used to select the clock output frequency.

Table 22.2-2 Functions of the FRQ2, FRQ1, and FRQ0 Bits

FRQ2	FRQ1	FRQ0	Output clock	$\phi = 16 \text{ MHz}$	$\phi = 8 \text{ MHz}$	$\phi = 4 \text{ MHz}$	$\phi = 8 \text{ kHz}$
0	0	0	$\phi/2^1$	125 ns	250 ns	500 ns	250 μs
0	0	1	$\phi/2^2$	250 ns	500 ns	1 μs	500 μs
0	1	0	$\phi/2^3$	500 ns	1 μs	2 μs	1 ms
0	1	1	$\phi/2^4$	1 μs	2 μs	4 μs	2 ms
1	0	0	$\phi/2^5$	2 μs	4 μs	8 μs	4 ms
1	0	1	$\phi/2^6$	4 μs	8 μs	16 μs	8 ms
1	1	0	$\phi/2^7$	8 μs	16 μs	32 μs	16 ms
1	1	1	$\phi/2^8$	16 μs	32 μs	64 μs	32 ms

CHAPTER 23 ADDRESS MATCH DETECTION FUNCTION

This chapter describes the address match detection function and operation.

23.1 "Overview of the Address Match Detection Function"

23.2 "Registers of the Address Match Detection Function"

23.3 "Operation of the Address Match Detection Function"

23.4 "Example of the Address Match Detection Function"

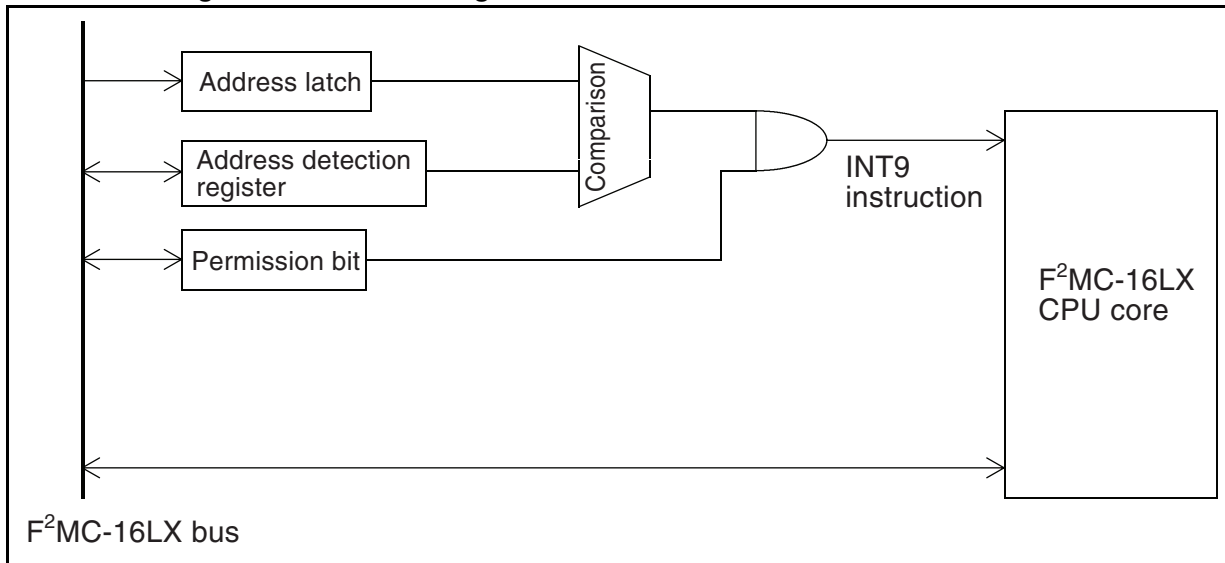
23.1 Overview of the Address Match Detection Function

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 (01H). Consequently, the CPU executes the INT9 instruction when executing a specified instruction. The address match detection function can be achieved using the INT9 interrupt routine for processing.

There are two address detection registers, each with an interrupt permission bit. When an address matches the value set in the address detection register and the interrupt permission bit is 1, the instruction code to be read by the CPU is replaced with the INT9 instruction code.

■ Block Diagram of the Address Match Detection Function

Figure 23.1-1 Block Diagram of the Address Match Detection Function



23.2 Registers of the Address Match Detection Function

The two types of registers for the address match detection function are as follows:

- Program address detection registers (PADR0 and PADR1)
- Program address detection control status register (PACSR)

■ Program Address Detection Registers (PADR0 and PADR1)

The program address detection registers 0 and 1 (PADR0 and PADR1) compare the address with the value written in each register. If they match when the interrupt permission bit corresponding to ADCSR is 1, the CPU is requested to issue the INT9 instruction.

When the corresponding interrupt bit is 0, nothing occurs.

Figure 23.2-1 Program Address Detection Registers (PADR0 and PADR1)

Program address detection registers	byte	byte	byte	Access	Initial value
PADR0 1FF2H/1FF1H/1FF0H				R/W	Not defined
PADR1 1FF5H/1FF4H/1FF3H				R/W	Not defined

Table 23.2-1 "Correspondence between PADR0 and PADR1 Registers and PACSR" lists the correspondence between the program address detection registers (PADR0 and PADR1) and PACSR.

Table 23.2-1 Correspondence between PADR0 and PADR1 Registers and PACSR

Address detection register	Interrupt permission bit
PADR0	AD0E
PADR1	AD1E

■ Program Address Detection Control Status Register (PACSR)

The program address detection control register (PACSR) controls the operation of the address detection function.

Figure 23.2-2 Program Address Detection Control 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)	(-)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

[Bits 7 to 4] Reserved bits

Bits 7 to 4 are reserved. Set these bits to 0 before setting PACSR.

[Bit 3] AD1E (Address Detect register 1 Enable)

The AD1E bit is the operation permission bit of ADR1.

When this bit is 1, the address is compared with the PADR1 register. If they match, the INT9 instruction is issued.

[Bit 2] Reserved bit

Bit 2 is a reserved bit. When PACSR is set, be sure to set bit 2 to 0.

[Bit 1] AD0E (Address Detect register 0 Enable)

The AD0E bit is the operation permission bit of ADR0.

When this bit is 1, the address is compared with the PADR0 register. If they match, the INT9 instruction is issued.

[Bit 0] Reserved bit

Bit 0 is a reserved bit. When PACSR is set, be sure to set bit 0 to 0.

23.3 Operation of the Address Match Detection Function

If the program counter specifies the same address as the address match detection register, the INT9 instruction is executed. The address match detection function can be achieved by processing the INT9 instruction routine.

■ Operation of the Address Match Detection Function

There are two address detection registers with a compare enable bit. When the value set in the address detection register and the value of the program counter match and the compare enable bit is set to 1, the CPU executes the INT9 instruction.

Note:

If the value of the address detection register and the value of the program counter match, the contents of internal data bus is changed to 01_H. Consequently, the INT9 instruction is executed. Before changing the contents of the address detection register, always set the compare enable bit to 0. While the compare enable bit is set to 1, changing the contents of the address detection register may result in a malfunction.

23.4 Example of the Address Match Detection Function

Figure 23.4-1 "System Configuration Example of the Address Match Detection Function" shows a system configuration example of the address match detection function. Table 23.4-1 "EEPROM Memory Map" lists the EEPROM memory map.

■ System Configuration Example of the Address Match Detection Function

Figure 23.4-1 System Configuration Example of the Address Match Detection Function

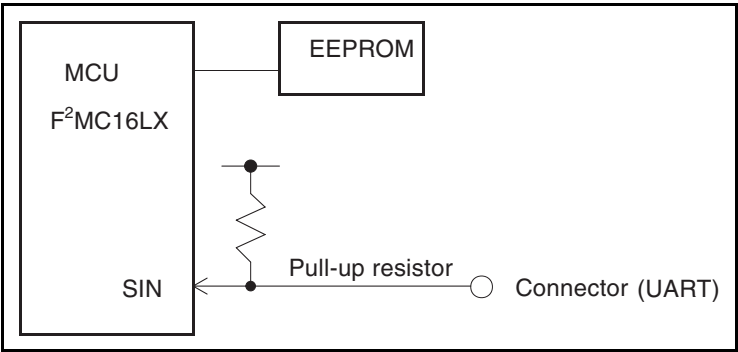


Table 23.4-1 EEPROM Memory Map

Address	Description
0000 _H	Number of bytes of patch program No.0 (If 0, no program error exists.)
0001 _H	Program address No.0 bits 7 to 0
0002 _H	Program address No.0 bits 15 to 8
0003 _H	Program address No.0 bits 24 to 16
0004 _H	Number of bytes of patch program No.1 (If 0, no program error exists.)
0005 _H	Program address No.1 bits 7 to 0
0006 _H	Program address No.1 bits 15 to 8
0007 _H	Program address No.1 bits 24 to 16
0010 _H or higher	Main body of patch program No. 0

○ Initial status

EEPROM is set to all 0s.

○ 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 EEPROM.

○ Reset sequence

The MCU reads the value of EEPROM after reset. If the number of bytes of the patch program is not 0, the main body of the patch program is read from EEPROM and written to RAM. The MCU then uses either PADR0 or PADR1 to set the patch address and sets the compare enable bit. If the relocatable patch program is required, the first address of the patched program can be written to the RAM area. In this case, the INT9 routine accesses this user-defined RAM area and jumps to the patched program.

○ INT9 interrupt

The interrupt routine can know the address where the interrupt occurs by checking the value of the stack program counter. The information that has been placed on the stack during the interrupt is discarded.

■ Example of program patch processing

Figure 23.4-2 Example of program patch processing

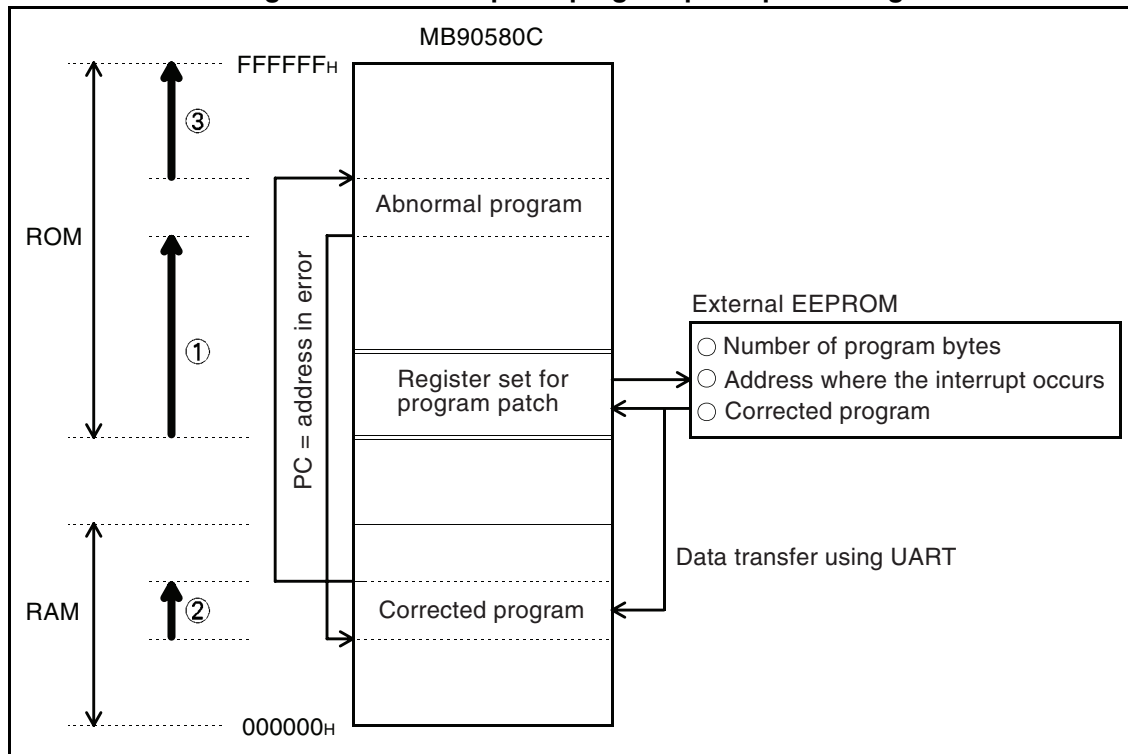
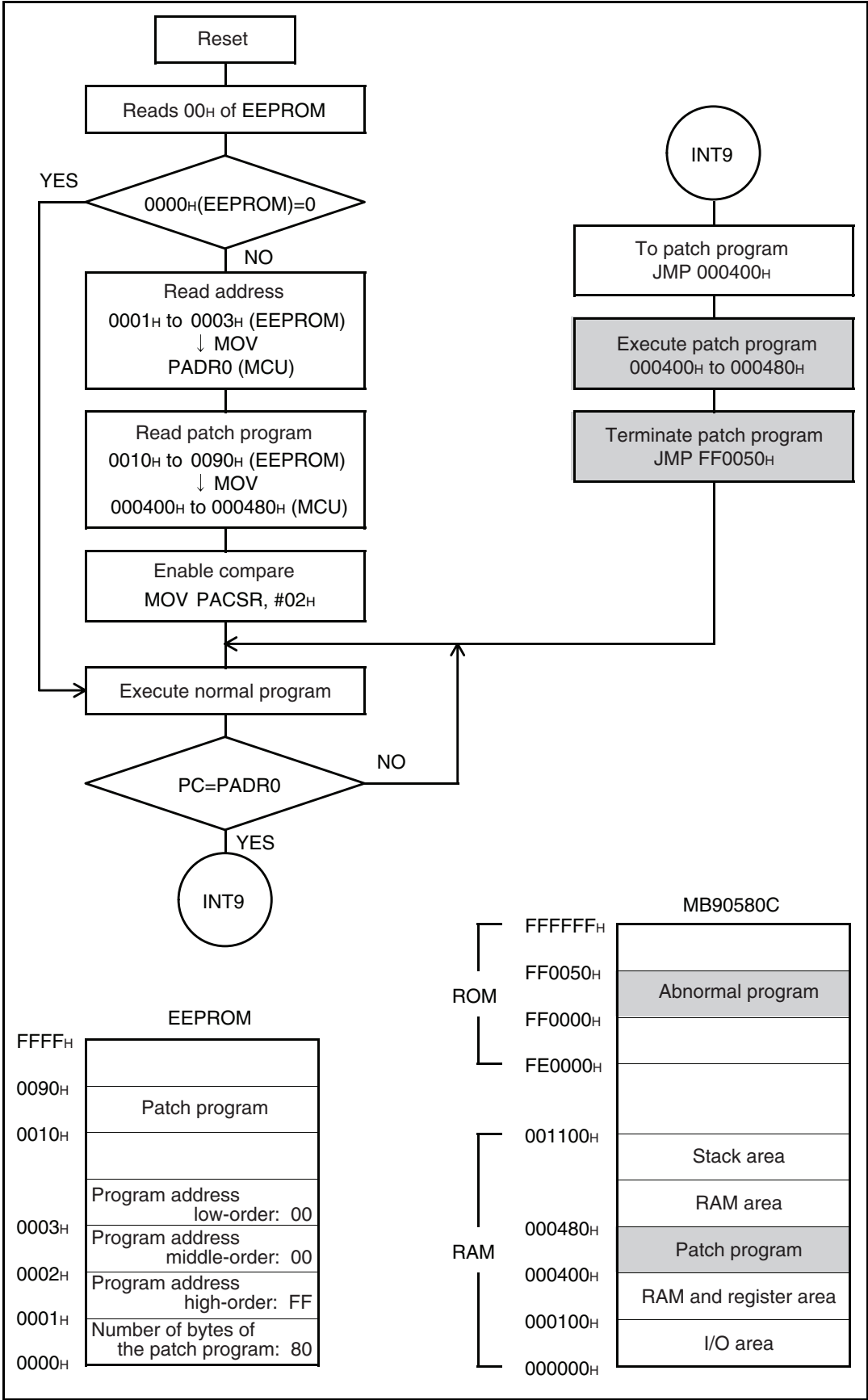


Figure 23.4-3 Flow of program patch processing



CHAPTER 24 ROM MIRROR FUNCTION SELECTION MODULE

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

24.1 "Overview of the ROM Mirror Function Selection Module"

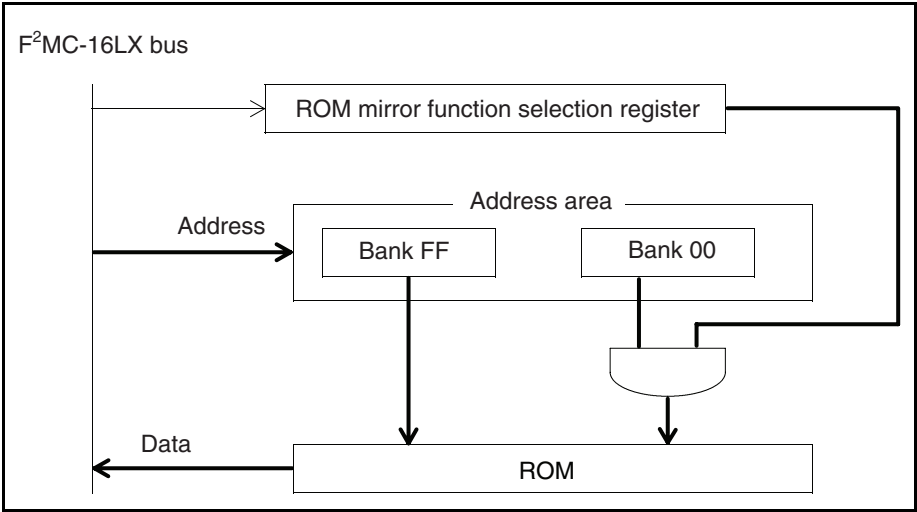
24.2 "ROM Mirror Function Selection Register (ROMM)"

24.1 Overview of the ROM Mirror Function Selection Module

By setting a register, the ROM mirror function selection module can determine that bank FF containing the ROM is read in bank 00.

■ Block Diagram of the ROM Mirror Function Selection Module

Figure 24.1-1 Block Diagram of the ROM Mirror Function Selection Module



24.2 ROM Mirror Function Selection Register (ROMM)

Figure 24.2-1 "ROM Mirror Function Selection Register (ROMM)" shows the ROM mirror function selection register (ROMM).

■ ROM Mirror Function Selection Register (ROMM)

Figure 24.2-1 ROM Mirror Function Selection Register (ROMM)

ROM mirror register	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 00006F _H	—	—	—	—	—	—	—	MI	ROMM
Read/write ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(W)	
Initial value ⇐	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(1)	

Note:

Do not access this register while the operation in addresses 004000_H to 00FFFF_H are ongoing.

[Bit 8] MI

When 1 is written to the MI bit, ROM data in bank FF can be read in bank 00. When 0 is written to this bit, processing such as memory mapping is not executed. This bit is written only.

In single-chip mode and internal ROM external bus mode, the memory space is as follows.

Note:

Addresses FF4000_H to FFFFFFF_H are mirrored to addresses 004000_H to 00FFFF_H only when the ROM mirror function is activated. Therefore, addresses FF0000_H to FF3FFF_H are not mirrored to bank 00.

Table 24.2-1 Memory Space Address

	MB90583C/CA	MB90F583C/CA	MB90V580B	MB90587C/CA
Address 1	FF0000 _H	FE0000 _H	-	FF0000 _H
Address 2	001900 _H	001900 _H	001900 _H	001100 _H

Figure 24.2-2 Memory Space in Single-chip Mode

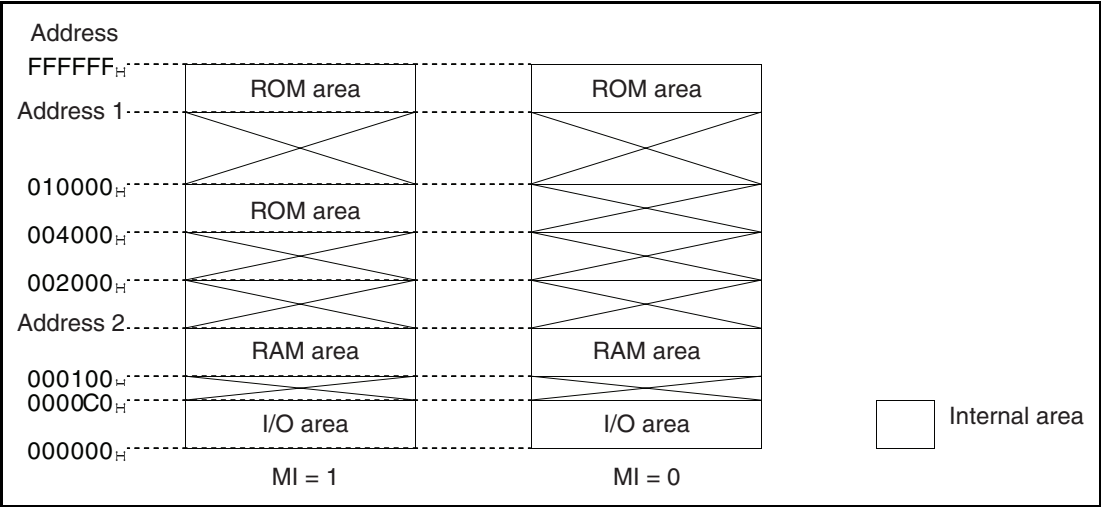
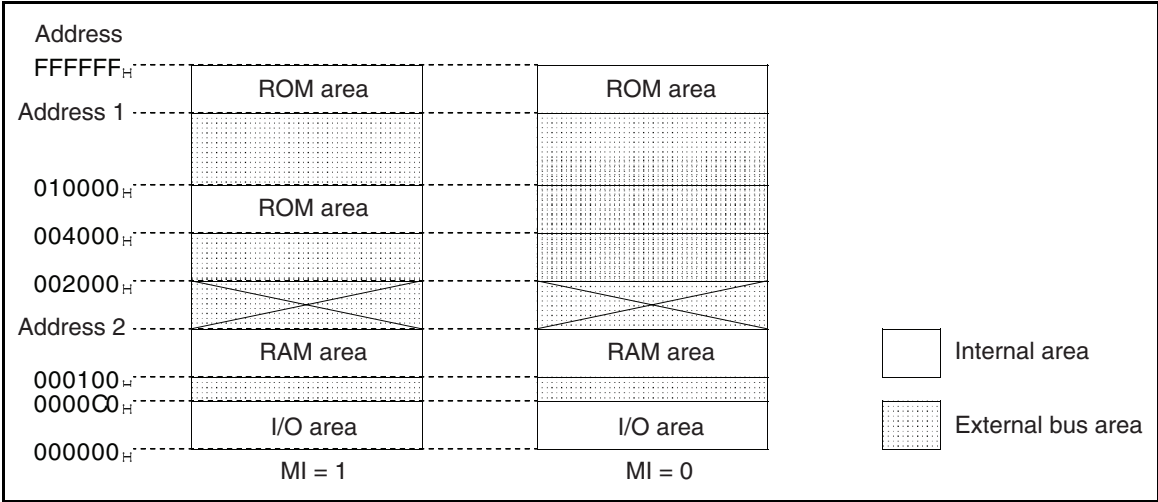


Figure 24.2-3 Memory Space in the Internal ROM External Bus Mode



CHAPTER 25 1M-BIT FLASH MEMORY

This chapter describes the functions and operations of the 1M-bit flash memory. The following three methods are supported to write or delete data for the flash memory:

- **Parallel programmer**
- **Serial programmer**
- **Executing programs to write/erase data**

This chapter describes "Executing programs to write/erase data".

- 25.1 "Overview of the 1M-Bit Flash Memory"
- 25.2 "Sector Configuration of the Flash Memory"
- 25.3 "Flash Memory Control Status Register (FMCS)"
- 25.4 "Activating the Automatic Algorithm of the Flash Memory"
- 25.5 "Confirming the Automatic Algorithm Execution Status"
- 25.6 "Detailed Explanations of Flash Memory Writing and Deletion"
- 25.7 "Example of the 1M-Bit Flash Memory Program"

25.1 Overview of the 1M-Bit Flash Memory

The 1M-bit flash memory is allocated in banks FE_H to FF_H on the CPU memory map. As in mask ROM, it can be subjected to a read access and program access from the CPU using the flash memory interface circuit function. Because data can be written or deleted for the flash memory by instructions from the CPU through the flash memory interface circuit, data can be rewritten in the installation status by control of the internal CPU. This enables programs and data to be improved. Sector operations such as enable and sector protect cannot be used.

■ Features of the 1M-bit Flash Memory

- Configuration of the 128 K words x 8 K or 64 K words x 16 bits (16 K + 8 K x 2 + 32 K + 64 K) sector
- Automatic program algorithm (Embedded Algorithm: Same as for MBM29LV200)
- Function for temporarily stopping deletion and function for restarting it
- Detecting the completion of writing and deletion using the data polling and toggle bits
- Deleting the completion of writing and deletion using CPU interrupts
- Compatibility with the JEDEC standard commands
- Enabling data to be deleted for each sector (combining sectors freely)
- Number of times of writing or number of times of deletions (minimum): 10,000
- Flash read cycle time (Min.): 2 machine cycles

Embedded Algorithm is a trademark of the Advanced Micro Device, Inc.

■ Writing and Deleting Data for the Flash Memory

Writing or deletion and reading for the flash memory cannot occur at the same time. In other words, when data is written or deleted for the flash memory, writing only is possible by the following operation without a program access from the flash memory: the program on the flash memory is copied onto the RAM and the RAM is executed.

■ Flash Memory Register

○ Flash memory control status register (FMCS)

Bit No.	7	6	5	4	3	2	1	0
Address: 0000AE _H	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0
Read/write	(R/W)	(R/W)	(R/W)	(R)	(W)	(R/W)	(W)	(R/W)
Initial value	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)

25.2 Sector Configuration of the Flash Memory

Figure 25.2-1 "Sector configuration of the 1M-bit flash memory" shows the sector configuration of the flash memory.

■ Sector configuration of the 1M-bit flash memory

Figure 25.2-1 "Sector configuration of the 1M-bit flash memory" shows the sector configuration of the 1M-bit flash memory and the high-order and low-order addresses of each sector.

For access from the CPU, SA0 is stored in the FE bank register and SA1 to SA4 are stored in the FF bank register.

Figure 25.2-1 Sector configuration of the 1M-bit flash memory

Flash memory	CPU address	Programmer address(*)
SA4 (16 Kbytes)	FFFFFF _H	7FFFF _H
	FFC000 _H	7C000 _H
SA3 (8 Kbytes)	FFBFFF _H	7BFFF _H
	FFA000 _H	7A000 _H
SA2 (8 Kbytes)	FF9FFF _H	79FFF _H
	FF8000 _H	78000 _H
SA1 (32 Kbytes)	FF7FFF _H	77FFF _H
	FF0000 _H	70000 _H
SA0 (64 Kbytes)	FEFFFF _H	6FFFF _H
	FE0000 _H	60000 _H

*: A programmer address is associated with a CPU address when data is written in the flash memory by the parallel programmer. This address is used to perform writing or deletion using the general-purpose programmer.

25.3 Flash Memory Control Status Register (FMCS)

The control status register (FMCS) exists in the flash memory interface circuit and is used for flash memory writing and deletion.

■ Control Status Register (FMCS)

Bit No.	7	6	5	4	3	2	1	0
Address: 0000AEH	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0
Read/write	(R/W)	(R/W)	(R/W)	(R)	(W)	(R/W)	(W)	(R/W)
Initial value	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)

○ Bits

[Bit 7] INTE (INTerrupt Enable)

Used to generate an interrupt to the CPU at the end of flash memory writing or deletion.

An interrupt to the CPU occurs when the INTE bit is 1 and RDYINT bit is 1. No interrupt occurs if the INTE bit is 0.

0: Disables an interrupt at the end of writing or deletion.

1: Enables an interrupt at the end of writing or deletion.

[Bit 6] RDYINT (ReaDY INTerrupt)

Used to indicate the flash memory operation status.

This bit is set to 1 after the end of flash memory writing or deletion. Flash memory writing or deletion is impossible while this bit is 0 after flash memory writing or deletion. If this bit is set to 1 after writing or deletion ends, flash memory writing or deletion is possible.

This bit is cleared to 0 by writing 0. Writing 1 is ignored. This bit is set to 1 when the automatic algorithm of the flash memory (see Section 25.4 "Activating the Automatic Algorithm of the Flash Memory") ends. Value 1 can be read when the read modify write (RMW) instruction is used.

0: Indicates that writing or deletion is ongoing.

1: Indicates that writing or deletion ends. (Interrupt request generation)

[Bit 5] WE (Write Enable)

Used to enable writing to the flash memory area.

When this bit is 1, writing to the flash memory area is set after the command sequence to banks FC to FF (see Section 25.4 "Activating the Automatic Algorithm of the Flash Memory") is issued. When this bit is 0, no signals for writing and deletion are generated. This bit is used to activate the commands for flash memory writing and deletion.

When neither writing nor deletion is executed, it is recommended that this bit always be set to 0 so that no data is written in the flash memory by mistake.

0: Disables flash memory writing and deletion.

1: Enables flash memory writing and deletion.

[Bit 4] RDY (ReaDY)

Used to enable flash memory writing or deletion.

While this bit is 0, writing and deletion are impossible for the flash memory. In this status, a read command, reset command, and suspend commands such as sector deletion temporary stop can be accepted.

0: Indicates that writing or deletion is ongoing (writing or deletion of the next data is impossible).

1: Indicates that the writing or deletion ends (writing or deletion of the next data is possible).

[Bit 3] Reserved bit

Reserved for testing. Set this bit to 0 for normal use.

[Bits 1] Free bits

Set these bits to 0 for normal use.

[Bit 2 and 0] LPM1, LPM0 (Low Power Mode)

LPM1 and LPM0 bits can be used to control power consumption of the flash memory. However, because the time for access from the CPU to the flash memory is very dependent on the setting, select the setting value according to the operation frequency of the CPU.

01: Low-power consumption mode
(operation at internal operation frequency of 4 MHz or less)

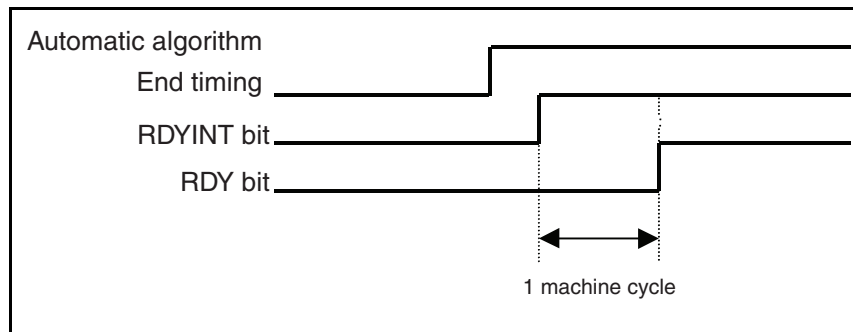
10: Low-power consumption mode
(operation at internal operation frequency of 8 MHz or less)

11: Low-power consumption mode
(operation at internal operation frequency of 12.58 MHz or less)

00: Ordinary power consumption mode
(operation at internal operation frequency of 16 MHz or less)

Note:

The RDYINT and RDY bits do not change at the same time. Create a program so that one or the other is used.



25.4 Activating the Automatic Algorithm of the Flash Memory

To activate the automatic algorithm of the flash memory, the following four types of commands are supported: read, reset, writing, and chip deletion. For the sector deletion, temporary stop and restart can be controlled.

■ Command Sequence Table

Table 25.4-1 "Command Sequence Table" lists the commands used for flash memory writing and deletion. All data items to be written in the command register are in byte units. However, write data by word access. In this case, the data for high-order bytes is ignored.

Table 25.4-1 Command Sequence Table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read or Reset (*)	1	FxXXXX	XXF0	-	-	-	-	-	-	-	-	-	-
Read or Reset (*)	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	-	-	-	-
Write program	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip deletion	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector deletion	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector deletion temporary stop		Sector deletion temporarily stops by input of address FxXXXX data (xxB0 _H).											
Sector deletion restart		After a temporary stop of the sector deletion, deletion starts by the input of address FxXXXX data (xx30 _H).											
Auto-select	3	FxAAA	XXAA	Fx5554	XX55	FxAAAA	XX90	-	-	-	-	-	-

Note:

- Address Fx in the table indicates FF or FE. For each operation, use a value of the bank to be accessed.
- An address in the table is a value on the CPU memory map. All addresses and data are represented in hexadecimal. However, X is any value.
- RA: Read address
- PA: Only a write address and even address can be specified.
- SA: Sector address. See Section 25.2 "Sector Configuration of the Flash Memory".
- RD: Read data
- PD: Only write data and word data can be specified.

*: Both read and reset commands can reset the flash memory to the read mode.

25.4 Activating the Automatic Algorithm of the Flash Memory

The Auto-select command shown in Table 25.4-1 "Command Sequence Table" is used to know the state of sector protection. When using the Auto-select command, set the address as follows.

Table 25.4-2 Address Setting at Auto-select

	AQ13 to AQ16	AQ7	AQ2	AQ1	AQ0	DQ7 to DQ0
Sector protection	Sector Address	L	H	L	L	CODE [*]

^{*}: When the sector address is protected, the output is "01H".

When the sector address is not protected, the output is "00H".

25.5 Confirming the Automatic Algorithm Execution Status

To provide the writing or deletion flow with an automatic algorithm, the flash memory contains hardware that notifies an operator of the operating status or of operation completion within the flash memory. This automatic algorithm enables the operating status of the built-in flash memory to be confirmed using the following hardware sequence:

■ Hardware Sequence Flag

The hardware sequence flag consists of the 5-bit outputs DQ7, DQ6, DQ5, DQ3, and DQ2 which have functions of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit excess flag (DQ5), sector deletion timer flag (DQ3), and toggle bit 2 flag (DQ2), thereby enabling an operator to confirm whether the end of writing, the end of chip or sector deletion, and deletion code writing are valid.

The hardware sequence flag can be referenced by a read access to the address of the target sector within the flash memory after the command sequence (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory") is set. 25.5-1 "Bit Allocation for the Hardware Sequence Flags" shows the bit allocation for the hardware sequence flags.

Table 25.5-1 Bit Allocation for the Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	DQ2	-	-

To determine whether automatic writing or chip or sector deletion is ongoing, check the hardware sequence flag or the RDY bit of the flash memory control register (FMCS), thereby enabling the operator to determine whether the writing ends. After the writing or deletion is completed, the read or reset status is returned. To create a program, perform the next processing such as data reading after confirming the end of the automatic writing or deletion using either of the flags. The hardware sequence flag can be used to confirm whether the second and subsequent sector deletion code writings are valid. The subsequent sections explain the hardware sequence flags. Table 25.5-2 "Hardware Sequence Flag Functions" lists the hardware sequence flag functions.

Table 25.5-2 Hardware Sequence Flag Functions

Status		DQ7	DQ6	DQ5	DQ3	DQ2
Status change at normal operation	Writing --> writing completion (At write address specification)	$\overline{DQ7}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3	1 --> DATA:2
	Chip or sector deletion --> deletion completion	0 --> 1	Toggle --> Stop	0 --> 1	1	Toggle --> Stop
	Sector deletion wait --> deletion start	0	Toggle	0	0 --> 1	Toggle
	Deletion --> temporary stop of sector deletion (Sector being deleted)	0 --> 1	Toggle --> 1	0	1 --> 0	Toggle
	Temporary stop of sector deletion --> deletion restart (Sector being deleted)	1 --> 0	1 --> Toggle	0	0 --> 1	Toggle
	Temporary stop of sector deletion ongoing (Sector not being deleted)	DATA:7	DATA:6	DATA:5	DATA:3	DATA:2
Abnormal operation	Writing	$\overline{DQ7}$	Toggle	1	0	1
	Chip or sector deletion operation	0	Toggle	1	1	*1

Note: If the DQ5 outputs "1" (exceed the timing limit), successive reads from a writing or erasing sector cause DQ2 to toggle. DQ2 does not toggle when the successive reads are executed from other sectors.

25.5.1 Data Polling Flag (DQ7)

Using the data polling function, the data polling flag (DQ7) notifies an operator that the automatic algorithm execution is ongoing or ends.

■ Data Polling Flag (DQ7)

Table 25.5-3 "Status Transition of the Data Polling Flag (Status Changes At Normal Operation)" and Table 25.5-4 "Status Transition of the Data Polling Flag (Status Changes at Abnormal Operation)" list the transitions of data polling flag statuses.

Table 25.5-3 Status Transition of the Data Polling Flag (Status Changes At Normal Operation)

Operation status	Writing --> completion	Chip or sector deletion --> completion	Sector deletion wait --> start	Sector deletion --> temporary stop of deletion Sector being deleted	Temporary stop of sector deletion --> restart Sector being deleted	Temporary stop of sector deletion ongoing Sector not being deleted
DQ7	$\overline{\text{DQ7}}$ --> DATA:7	0 --> 1	0	0 --> 1	1 --> 0	DATA:7

Table 25.5-4 Status Transition of the Data Polling Flag (Status Changes at Abnormal Operation)

Operation status	Writing	Chip or sector deletion
DQ7	$\overline{\text{DQ7}}$	0

○ At writing

If a read access is made while the automatic write algorithm is executed, the flash memory outputs the reverse data of bit 7 of the data last written, regardless of the indicated address. If a read access is made when the automatic write algorithm ends, the flash memory outputs bit 7 of the read value of the indicated address.

○ At chip or sector deletion

The flash memory outputs 0 while the chip deletion or sector deletion algorithm is executed if a read access is made from the deleted sector at sector deletion or a read access is made regardless of the indicated address at chip deletion. Similarly, the flash memory outputs 1 at the end of the operation.

○ At temporary stop of sector deletion

The flash memory outputs 1 if a read access is made at temporary stop of sector deletion and the indicated address specifies the sector being deleted. If the indicated address does not specify the sector being deleted, the flash memory outputs bit 7 (DATA:7) of the read value of the indicated address. Referencing this bit together with the toggle bit flag (DQ6) enables the operator to determine whether the sector stops temporarily or which sector is being deleted.

Note:

At activation of the automatic algorithm, a read access to the specified address is ignored. At data reading, other bits can be output when the data polling flag (DQ7) ends. Therefore, after the automatic algorithm ends, data must be read following the read access for which the end of data polling is confirmed.

25.5.2 Toggle Bit Flag (DQ6)

Using the toggle bit function, the toggle bit flag (DQ6) informs an operator that the automatic algorithm execution is ongoing or ends, as with the data polling flag (DQ7).

■ Toggle Bit Flag (DQ6)

Table 25.5-5 "Status Transition of The Toggle Bit Flag (Status Changes at Normal Operation)" and Table 25.5-6 "Status Transition of the Toggle Bit Flag (Status Changes at Abnormal Operation)" show status transitions of the toggle bit flag.

Table 25.5-5 Status Transition of The Toggle Bit Flag (Status Changes at Normal Operation)

Operation status	Writing --> completion	Chip or sector deletion --> completion	Sector deletion wait --> start	Sector deletion --> temporary stop of deletion Sector being deleted	Temporary stop of sector deletion --> restart Sector being deleted	Temporary stop of sector deletion ongoing Sector not being deleted
DQ6	Toggle --> DATA:6	Toggle --> Stop	Toggle	Toggle --> 1	1 --> Toggle	DATA:6

Table 25.5-6 Status Transition of the Toggle Bit Flag (Status Changes at Abnormal Operation)

Operation status	Writing	Chip or sector deletion
DQ6	Toggle	Toggle

○ At writing or at chip or sector deletion

Assume that read access is made continuously while the automatic write algorithm or chip or sector deletion algorithm is executed. In this case, the flash memory outputs the toggle status in which 1 and 0 are alternately output for each read access, regardless of the indicated address. If read access is made continuously at the end of the automatic write algorithm or chip or sector deletion algorithm, the flash memory stops the toggle operation of bit 6 and outputs bit 6 (DATA:6) of the read value of the indicated address.

○ At temporary stop of sector deletion

When a read access is made at temporary stop of sector deletion, the flash memory outputs 1 if the indicated address belongs to the sector being deleted. If the address does not belong to the sector being deleted, the flash memory outputs bit 6 (DATA:6) of the read value of the indicated address.

Reference:

At writing, if the sector in which an attempt is made to write data is protected against rewriting, the toggle operation ends without rewriting data after the toggle operation of about 2 μ s is performed.

At deletion, if all selected sectors are protected against rewriting, the toggle bits involve the toggle operation of about 100 μ s, and then the read or reset status is returned without rewriting data.

25.5.3 Timing Limit Excess Flag (DQ5)

The timing limit excess flag (DQ5) informs the operator that the automatic algorithm execution exceeded the time specified within the flash memory (number of internal pulses).

■ Timing Limit Excess Flag (DQ5)

Table 25.5-7 "Status Transition of The Timing Limit Excess Flag (Status Changes at Normal Operation)" and Table 25.5-8 "Status Transition of the Timing Limit Excess Flag (Status Changes at Abnormal Operation)" show status transitions of the timing limit excess flag.

Table 25.5-7 Status Transition of The Timing Limit Excess Flag (Status Changes at Normal Operation)

Operation status	Writing --> completion	Chip or sector deletion --> completion	Sector deletion wait --> start	Sector deletion --> temporary stop of deletion Sector being deleted	Temporary stop of sector deletion --> restart Sector being deleted	Temporary stop of sector deletion ongoing Sector not being deleted
DQ5	0 --> DATA:5	0 --> 1	0	0	0	DATA:5

Table 25.5-8 Status Transition of the Timing Limit Excess Flag (Status Changes at Abnormal Operation)

Operation status	Writing	Chip or sector deletion
DQ5	1	1

○ At writing or chip or sector deletion

Assume that a read access is made after the automatic algorithm for the writing or chip or sector deletion is activated. The flag outputs 0 when the automatic algorithm execution is within the specified time (required for writing or deletion). It outputs 1 if the automatic algorithm execution exceeds the specified time. This output does not depend on whether the automatic algorithm is being executed or ends; therefore, the operator can determine whether the writing or deletion is successful. In other words, if this flag is 1, and the automatic algorithm is being executed by the data polling function or toggle bit function, the operator is able to recognize that the writing has failed.

For example, if an attempt is made to write 1 in the flash memory address in which 0 is already written, a fail occurs. In this case, the flash memory is locked and the automatic algorithm does not end. Therefore, no valid data is output from the data polling flag (DQ7). The toggle bit flag (DQ6) does not stop the toggle operation and the time limit is exceeded. The timing limit excess flag (DQ5) outputs 1. This status indicates that the flash memory is not faulty and that it was not used correctly. If this status occurs, execute the reset command.

25.5.4 Sector Deletion Timer Flag (DQ3)

The sector deletion timer flag (DQ3) informs an operator whether the sector deletion wait period is ongoing after the sector deletion command is activated.

■ Sector Deletion Timer Flag (DQ3)

Table 25.5-9 "Status Transition of the Sector Deletion Timer Flag (Status Changes at Normal Operation)" and Table 25.5-10 "Status Transition of the Sector Deletion Timer Flag (Status Changes at Abnormal Operation)" show status transitions of the sector deletion timer flag.

Table 25.5-9 Status Transition of the Sector Deletion Timer Flag (Status Changes at Normal Operation)

Operation status	Writing --> completion	Chip or sector deletion --> completion	Sector deletion wait --> start	Sector deletion --> temporary stop of deletion Sector being deleted	Temporary stop of sector deletion --> restart Sector being deleted	Temporary stop of sector deletion ongoing Sector not being deleted
DQ3	0 --> DATA:3	1	0 --> 1	1 --> 0	0 --> 1	DATA:3

Table 25.5-10 Status Transition of the Sector Deletion Timer Flag (Status Changes at Abnormal Operation)

Operation status	Writing	Chip or sector deletion
DQ3	0	1

○ At sector deletion

Assume that a read access is made after the sector deletion command is activated. The flash memory outputs 0 if the sector deletion wait period is ongoing or it outputs 1 if this period is exceeded, regardless of the address indicated by the address signal of the sector for which the command was issued.

When the data polling or toggle bit function indicates that the deletion algorithm execution is ongoing, if this flag is 1, the deletion to be internally controlled is starting. Any command other than the subsequent sector deletion code writing or temporary stop of deletion is ignored until the deletion ends.

If this flag is 0, the flash memory accepts the writing of the additional sector deletion code. To confirm this, it is recommended to check the status of this flag before the subsequent sector deletion code writing. If the flag status is 1 at the second status check, the deletion code of the added sector may not be accepted.

○ At sector deletion

When a read access is made during temporary stop of sector deletion, the flash memory outputs 1 if the indicated address belongs to the sector being deleted. If it does not belong to the sector being deleted, the flash memory outputs bit 3 (DATA:3) of the read value of the indicated address.

25.5.5 Toggle Bit 2 Flag (DQ2)

The toggle bit 2 flag (DQ2) is a flag that uses the toggle bit function to indicate that the sector is in the erase-suspended state.

■ Toggle Bit 2 Flag (DQ2)

Table 25.5-11 "Toggle Bit 2 Flag State Transitions (State Change for Normal Operation)" and Table 25.5-12 "Toggle Bit 2 Flag State Transitions (State Change for Abnormal Operation)" list the state transitions of the toggle bit flag.

Table 25.5-11 Toggle Bit 2 Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ2	1 --> DATA:2	Toggle --> Stop	Toggle	Toggle	Toggle	DATA:2

Table 25.5-12 Toggle Bit 2 Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ2	1	*1

*1: If the DQ5 outputs "1" (exceed the timing limit), successive reads from a writing or erasing sector cause DQ2 to toggle. DQ2 does not toggle when the successive reads are executed from other sectors.

○ During a sector erase operation

If successive reads are executed during the execution of the chip sector erase algorithm, a flash memory toggles to output "1" and "0" to addresses alternately at every read access regardless of the location indicated by the addresses. If successive reads are executed after the chip sector erase algorithm is completed, the flash memory stops the toggle operation of the bit 2 and outputs the read value of the bit 2 (DATA: 2) to the location indicated by the address.

○ While a sector erase operation is suspended

If successive reads are executed while a sector erase operation is suspended, and if the address indicates the sector to be erased, the flash memory toggles to alternately output "1" and "0". If the address indicates the sector is not to be erased, the flash memory outputs the read value of the bit 2 (DATA: 2) to the location indicated by the address.

In the erase-suspend-program mode, successive reads from the non-erase suspended sector causes the flash memory to output "1".

Both DQ2 and DQ6 are used for detecting an erase-suspended sector (DQ2 toggles, but DQ6 does not).

DQ2 is also used for detecting an erasing sector. While erasing a sector, if a read access is executed from the erasing sector, DQ2 toggles.

Reference:

If all sectors selected for erasing are write-protected, the toggle bit 2 toggles for about 100 μ s, and then returns to the read/reset mode without writing the data.

25.6 Detailed Explanations of Flash Memory Writing and Deletion

This section describes procedures that are provided by issuing commands that activate the automatic algorithm. The procedures are reading and reset, writing, chip deletion, sector deletion, temporary stop of sector deletion, and restart of sector deletion with regard to the flash memory.

■ Detailed Explanation of Flash Memory Writing and Deletion

The flash memory can execute the automatic algorithm when the reading, reset, writing, chip deletion, sector deletion, temporary stop of deletion, or restart of deletion performs write cycles for the bus of the command sequence (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"). Write cycles for each bus must be performed continuously. The end of the automatic algorithm can be determined by the data polling function. After normal operation, the read or reset status is returned.

The subsequent sections explain the operations in the following order:

- Setting reading and reset status
- Writing data
- Deleting all data items (deleting all chips)
- Deleting any data item (deleting a sector)
- Stopping sector deletion temporarily
- Restarting sector deletion

25.6.1 Setting the Flash Memory in the Read or Reset Status

This section describes the procedure for setting the flash memory in the read or reset status by issuing the read or reset command.

■ Setting the Flash Memory to the Read or Reset Status

The flash memory can be set to the read or reset status by continuously sending read or reset commands, listed in the command sequence table (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"), to the target sector in the flash memory.

The read and reset commands have the following two command sequence types: one-bus operation and three-bus operation, though there is no essential difference between the two.

The read and reset statuses are initial statuses of the flash memory. At power-on or normal operation of the command, the flash memory is always set to a read and reset status. In these statuses, the system waits for other commands to be input.

In the read and reset statuses, data can be read by an ordinary read access. As with the mask ROM, a program access from the CPU is possible. The read and reset commands are not needed to read data at ordinary reading. If a command does not end normally, these commands are used primarily to initialize the automatic algorithm.

25.6.2 Writing Data in the Flash Memory

This section describes the procedure for writing data in the flash memory issuing the write command.

■ Writing Data in the Flash Memory

The data writing automatic algorithm of the flash memory can be activated by continuously sending the write command, listed in the command sequence table (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"), to the target sector in the flash memory. When the data writing to the target address ends in the fourth cycle, the automatic algorithm is activated and the automatic writing starts.

○ Addressing

Only an even address is possible as the write address to be specified in the write data cycle. If an odd address is specified, data cannot be written correctly. In other words, writing to the even address in word data units is required.

Writing is possible in any address order and outside the sector boundary. However, only one-word data can be written by one execution of the write command.

○ Notes on writing data

Data 0 cannot be returned to data 1 by writing. If data 1 is written in data 0, the data polling algorithm (DQ7) or toggle operation (DQ6) does not end and the flash memory element is determined to be faulty. The timing limit excess flag (DQ6) assumes an error because the specified writing time is exceeded, or it seems as if data 1 is written. However, if data is read in the read or reset status, data remains 0. Only a deletion operation can set data 0 to 1.

During automatic writing, all commands are ignored. Note that if the hardware reset is activated during writing, the data written in the address is not guaranteed.

■ Procedure for Writing Data in the Flash Memory

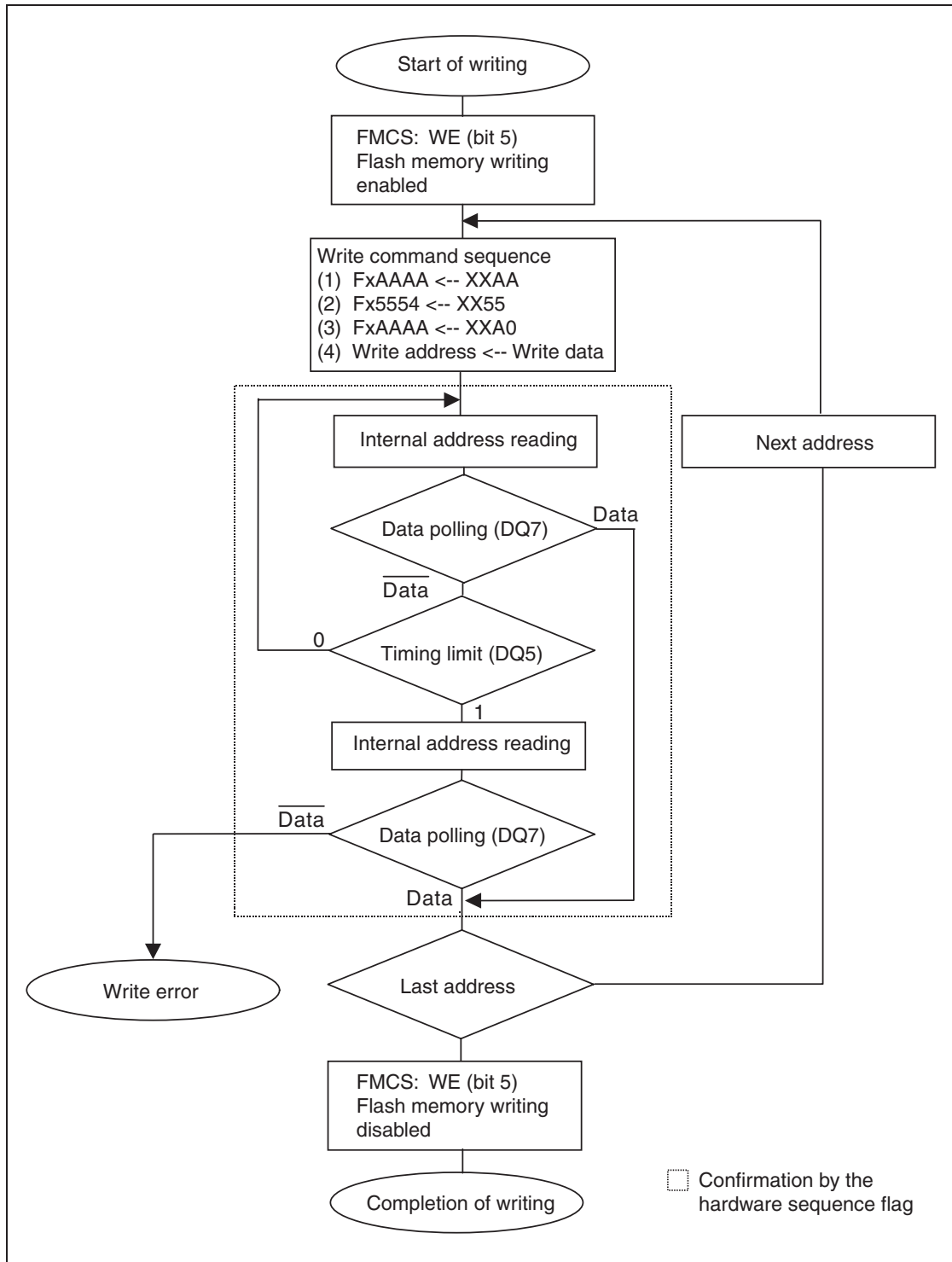
Figure 25.6-1 "Example of the Procedure for Flash Memory Writing" shows an example of the flash memory writing. The status of the automatic algorithm in the flash memory can be determined using the hardware sequence flag (see Section 25.5 "Confirming the Automatic Algorithm Execution Status"). The data polling flag (DQ7) is used to confirm the end of writing.

The data to be read for flag checking is read from the address in which the last writing was made.

The data polling flag (DQ7) changes when the timing limit excess flag (DQ5) changes. Therefore, even if the timing limit excess flag (DQ5) is 1, the data polling bit flag bit (DQ7) must be rechecked.

Similarly, the toggle bit flag (DQ6) stops the toggle operation when the timing limit excess flag bit (DQ5) changes to 1. Therefore, the toggle bit flag (DQ6) must be rechecked.

Figure 25.6-1 Example of the Procedure for Flash Memory Writing



25.6.3 Deleting all Data Items from the Flash Memory (Chip Deletion)

This section describes the procedure for deleting all data items from the flash memory issuing the chip deletion command.

■ Deleting the Data From the Flash Memory (Chip Deletion)

All data items can be deleted from the flash memory by continuously sending chip deletion commands, listed in the command sequence table (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"), to the target sector within the flash memory.

The chip deletion command is executed by six bus operations. The chip deletion starts when the writing in the 6th cycle is completed. Before the chip deletion, the user need not perform writing in the flash memory. During execution of the automatic deletion algorithm, the flash memory performs verification by writing 0 before automatically deleting all cells.

25.6.4 Deleting any Data Item from the Flash Memory (Sector Deletion)

This section describes the procedure for deleting a data item from the flash memory (sector deletion) by issuing the sector deletion command. Data can be deleted for each sector and two or more sectors can be specified at the same time.

■ Flash Memory from which any Data Item is Deleted (Sector Deletion)

Any sector can be deleted from the flash memory by continuously sending sector deletion commands, listed in the command sequence table (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"), to the target sector within the flash memory.

○ Specifying a sector

The sector deletion command is executed by six bus operations. The sector deletion wait of 50 μ s starts, in the 6th cycle, by writing the sector deletion code (30H) into any even address that can be accessed within the target sector. To delete two or more sectors, write the deletion code (30H) in the address within the target sector to be deleted in accordance with the above processing.

○ Notes on specifying two or more sectors

The deletion starts when the sector deletion wait period of 50 μ s from the writing of the last sector deletion code is completed. In other words, to delete two or more sectors at the same time, the address of the next deletion sector and the deletion code (6th cycle of the command sequence) each must be input within 50 μ s. If this limit is exceeded, the address and code may not be accepted. The sector deletion timer (hardware sequence flag DQ3) can be used to check whether the writing of the subsequent sector deletion code is valid. In this case, set the address for reading the sector deletion timer so that it indicates the sector to be deleted.

■ Procedure for Deleting a Sector from the Flash Memory

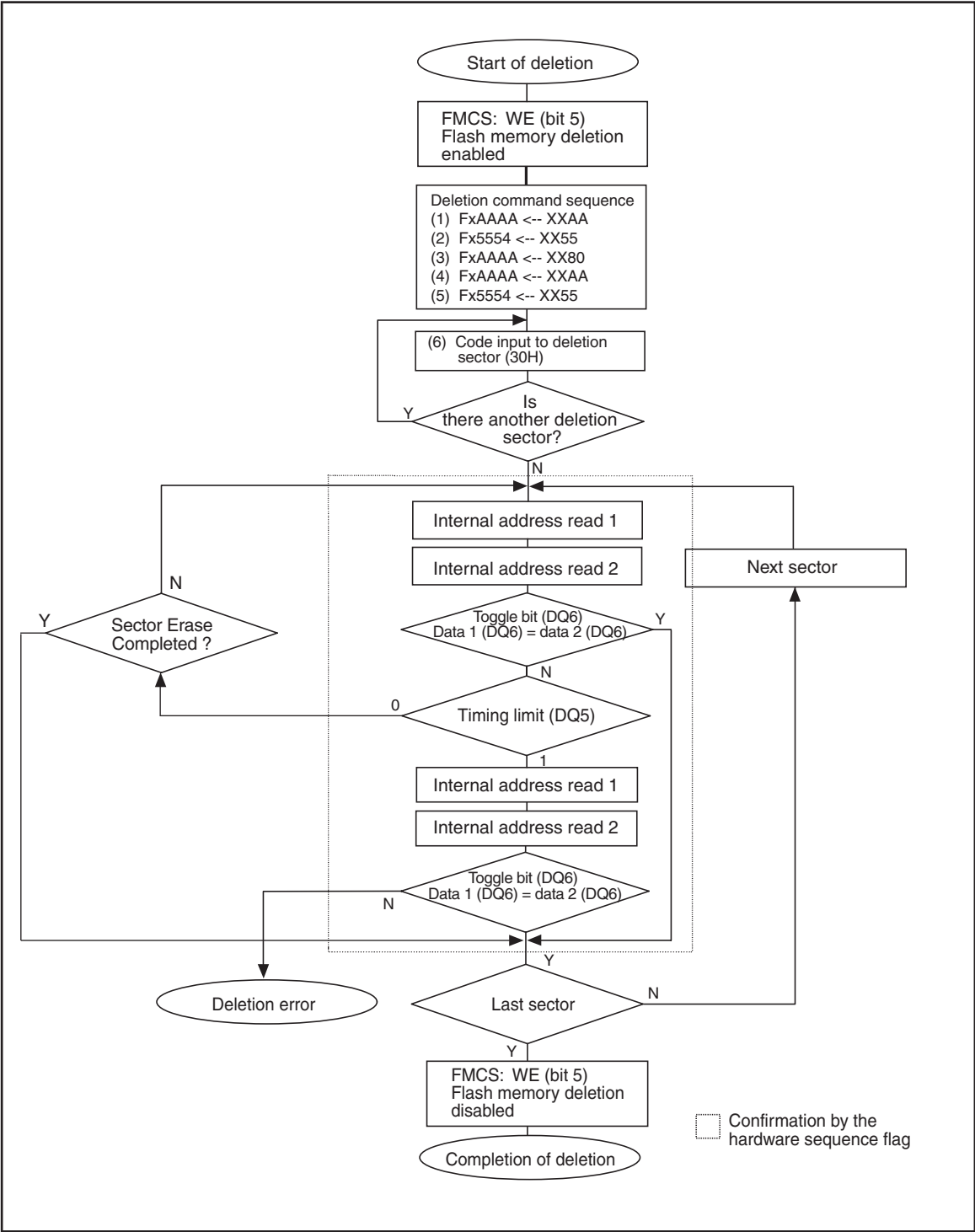
The status of the automatic algorithm within the flash memory can be determined using the hardware sequence flag (see Section 25.5 "Confirming the Automatic Algorithm Execution Status"). Figure 25.6-2 "Example of the Procedure for Deleting a Sector from the Flash Memory" shows an example of the procedure for deleting a flash memory sector. The toggle bit flag (DQ6) is used to confirm the end of deletion.

Note that the data to be read for flag checking is read from the sector to be deleted.

The toggle bit flag (DQ6) stops the toggle operation when the timing limit excess flag (DQ5) changes to 1. Therefore, even if the timing limit excess flag (DQ5) is 1, the toggle bit flag (DQ6) must be rechecked.

Similarly, because the data polling flag (DQ7) changes when the timing limit excess flag (DQ5) changes, it must be rechecked.

Figure 25.6-2 Example of the Procedure for Deleting a Sector from the Flash Memory



25.6.5 Temporarily Stopping the Sector Deletion from the Flash Memory

This section describes the procedure for temporarily stopping the sector deletion from the flash memory by issuing the sector deletion temporary stop command. Data can be read from the sector not being deleted.

■ Temporarily Stopping the Sector Deletion from the Flash Memory

The sector deletion from the flash memory can be temporarily stopped by continuously sending sector deletion temporary stop commands, listed in the command sequence table (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"), to the flash memory.

The sector deletion temporary stop command temporarily stops the sector deletion to enable data to be read from the sector not being deleted. In this status, only reading is possible (writing is not possible). This command is valid only during sector deletion, including deletion wait time, and is ignored during chip deletion or writing.

This command is executed by writing the deletion temporary stop code (B0H). However, any address in the flash memory must be indicated. The reexecution of the deletion temporary stop command is ignored for the temporary stop of the deletion.

If the sector deletion temporary command is input during the sector deletion wait period, the sector deletion wait ends immediately and deletion is suspended. The deletion stop status is entered. If the deletion temporary stop command is input during sector deletion after the sector deletion wait period, the deletion temporary stop status is entered after the maximum time of 15 μ s.

25.6.6 Restarting the Flash Memory Sector Deletion

This section describes the procedure for restarting the flash memory sector deletion temporarily stopped by issuing the sector deletion restart command.

■ Restarting the Flash Memory Sector Deletion

The sector deletion temporarily stopped can be restarted by continuously sending sector deletion restart commands, listed in the command sequence table (see Table 25.4-1 "Command Sequence" in Section 25.4 "Activating the Automatic Algorithm of the Flash Memory"), to the flash memory.

The sector deletion restart command is used to restart the sector deletion from the sector deletion temporary stop status set by the sector deletion temporary stop command. This command is executed by writing the deletion restart command (30H). However, any address in the flash memory area must be indicated.

The issuance of the sector deletion restart command is ignored during sector deletion.

25.7 Example of the 1M-Bit Flash Memory Program

This section provides an example of the 1M-bit flash memory program.

■ Example of the 1M-bit Flash Memory Program

```

NAME      FLASHWE
TITLE     FLASHWE

;-----
;1M-bit FLASH sample program
;
;1: Transferring the program (address FFBC00H, sector SA4)
;   in FLASH to the RAM (address 000700H)
;2: Executing the program on the RAM
;3: Writing a PDR1 value to FLASH (address FE0000H, sector SA0)
;4: Reading the written value (address FE0000H, sector SA0) and
;   outputting it to PDR2
;5: Deleting the written sector (SA0)
;6: Outputting the deletion data confirmation
; Conditions
;   -Number of RAM transfer bytes: 100H (256B)
;   -Judgment for the end of writing and deletion
;       Judgment with DQ5 (timing limit excess flag)
;       Judgment with DQ6 (toggle bit flag)
;       Judgment with RDY (FMCS)
;   -Error handling
;       Outputting Hi to P00 to P07
;       Issuing the reset command
;-----
;
RESOUS    IOSEG    ABS=00          ;Definition of the RESOUS I/O
          ORG      0000H          segment
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

```

CHAPTER 25 1M-BIT FLASH MEMORY

```

SSTA      ENDS
;
DATA      DSEG      ABS=0FFH          ;FLASH command address
          ORG        5554H
COMADR2   RW        1
          ORG        0AAAAH
COMADR1   RW        1
DATA      ENDS
;////////////////////////////////////
;Main program (SA1)
;////////////////////////////////////
CODE      CSEG
START:
          ;////////////////////////////////////
          ;Initialization
          ;////////////////////////////////////
          MOV        CKSCR,#0BAH      ;Setting to threefold
          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 confirmation
          MOV        DDR0,#0FFH
          MOV        PDR1,#00H        ;Data input port
          MOV        DDR1,#00H
          MOV        PDR2,#00H        ;Data output port
          MOV        DDR2,#0FFH
          ;////////////////////////////////////
          ;The FLASH write deletion program (FFBC00H) is transferred
          ;to the RAM (address 700H).
          ;////////////////////////////////////
          MOVW       A,#0700H        ;Transfer destination RAM area
          MOVW       A,#0BC00H        ;Transfer source address (program
;                                     location)
          MOVW       RW0,#100H        ;Number of bytes to be transferred
          MOVS       ADB,PCB          ;100H transfer from FFBC00H to
;                                     000700H
          CALLP      000700H          ;Jump to the address in which the
;                                     transferred program exists
          ;////////////////////////////////////
          ;Data output
          ;////////////////////////////////////
OUT        MOV        A,#0FEH
          MOV        ADB,A
          MOVW       RW2,#0000H
          MOVW       A,@RW2+00
          MOV        PDR2,A
END        JMP        *
CODE      ENDS
;////////////////////////////////////
;FLASH write deletion program (SA4)
;////////////////////////////////////
RAMPRG    CSEG      ABS=0FFH
          ORG        0BC00H
;////////////////////////////////////

```

25.7 Example of the 1M-Bit Flash Memory Program

```

;      Initialization
;      ///////////////////////////////////////////////////
MOVW   RW0,#0500H      ;RW0:  RAM space for input data
;                        acquisition          00:0500 to
MOVW   RW2,#0000H      ;RW2:  Flash memory writing address
;                        FD:0000 to
;
MOV     A,#00H          ;DTB change
MOV     DTB,A           ;@RW0 bank specification
MOV     A,#0FEH         ;ADB change 1
MOV     ADB,A           ;Specification of the bank for the
;                        write mode specification address
MOV     PDR3,#00H       ;Switch initialization
MOV     DDR3,#00H
;
WAIT1   BBC     PDR3:0,WAIT1    ;PDR3:0  Start of writing with Hi
;
;      ///////////////////////////////////////////////////
;      Writing(SA0)
;      ///////////////////////////////////////////////////
MOV     A,PDR1
MOVW    @RW0+00,A          ;Allocation of PDR1 data in
;                        the RAM
MOV     FMCS,#20H          ;Write mode setting
MOVW    ADB:COMADR1,#00AAH ;Flash write command 1
MOVW    ADB:COMADR2,#0055H ;Flash write command 2
MOVW    ADB:COMADR1,#00A0H ;Flash write command 3
;
MOVW    A,@RW0+00          ;Writing of input data (RW0)
;                        into the flash memory (RW2)
MOVW    @RW2+00,A
WRITE   ;Wait time check
;      ///////////////////////////////////////////////////
;      Error when the time limit excess check flag is set and the
;      toggle operation is ongoing
;      ///////////////////////////////////////////////////
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 if the
;                        value is different)
AND     A,#40H             ;Is the DQ6 toggle bit
;                        different?
BNZ     ERROR              ;If it is different, go to
;                        ERROR.
;      ///////////////////////////////////////////////////
;      Write end check (FMCS-RDY)
;      ///////////////////////////////////////////////////
NTOW    MOVW    A,FMCS
AND     A,#10H             ;Extraction of the FMCS RDY
;                        bit (4 bits)
BZ      WRITE              ;Is writing ended?
MOV     FMCS,#00H          ;Release of the write mode
;      ///////////////////////////////////////////////////
;      Write data output
;      ///////////////////////////////////////////////////

```


CHAPTER 25 1M-BIT FLASH MEMORY

```

        MOVW    RW2,#0000H                ;Write data output
        MOVW    A,@RW2+00
        MOV     PDR2,A
;
WAIT2   BBC     PDR3:1,WAIT2              ;PDR3:1 Start of the sector
                                           deletion with Hi
;
;//////////////////////////////////////////
;Sector deletion (SA0)
;//////////////////////////////////////////
        MOV     @RW2+00,#0000H           ;Address initialization
        MOV     FMCS,#20H                ;Deletion mode setting
        MOVW    ADB:COMADR1,#00AAH       ;Flash deletion command 1
        MOVW    ADB:COMADR2,#0055H       ;Flash deletion command 2
        MOVW    ADB:COMADR1,#0080H       ;Flash deletion command 3
        MOVW    ADB:COMADR1,#00AAH       ;Flash deletion command 4
        MOVW    ADB:COMADR2,#0055H       ;Flash deletion command 5
        MOV     @RW2+00,#0030H           ;Issuance of the deletion
                                           command to the sector to be
                                           deleted 6
;
;
        ELS     ; Wait time check
;
;//////////////////////////////////////////
;Error when the time limit excess check flag is set and the
;toggle operation is ongoing
;//////////////////////////////////////////
        MOVW    A,@RW2+00
        AND     A,#20H                   ;DQ5 time limit check
        BZ      NTOE                     ;Time limit over
        MOVW    A,@RW2+00                ;AH      Hi and low are
                                           alternately output,
;                                           for each reading,
        MOVW    A,@RW2+00                ;AL      from DQ6 during
                                           writing.
;                                           XOR of AH and AL (1 writing
        XORW    A                        ;(writing ongoing)
                                           if the DQ6 value is
;                                           different)
        AND     A,#40H                   ;Is the DQ6 toggle bit Hi?
        BNZ     ERROR                    ;If it is Hi, go to ERROR.
;//////////////////////////////////////////
;Deletion end check (FMCS-RDY)
;//////////////////////////////////////////
NTOE    MOVW    A,FMCS                    ;
        AND     A,#10H                   ;Extraction of the FMCS RDY
                                           bit (4 bits)
;                                           Is sector deletion ended?
        BZ      ELS                      ;Release of the FLASH
        MOV     FMCS,#00H                ;deletion mode
;                                           Return to the main program
        RETP                               ;
;//////////////////////////////////////////
;Error
;//////////////////////////////////////////
ERROR    MOV     ADB:COMADR1,#0F0H        ;Reset command (reading
                                           possible)
;                                           Release of the FLASH mode
        MOV     FMCS,#00H                ;Confirmation of the error
        MOV     PDR0,#0FFH               ;handling
;

```

25.7 Example of the 1M-Bit Flash Memory Program

```

                                RETP                                ;Return to the main program
RAMPRG  ENDS
;////////////////////////////////////
VECT    CSEG    ABS=0FFH
        ORG     0FFDCH
        DSL     START
        DB      00H
VECT    ENDS
;
        END     START
```


CHAPTER 26 EXAMPLE OF MB90F583C/CA SERIAL PROGRAMMING CONNECTION

This chapter provides examples of serial programming connection using the flash microcomputer programmer manufactured by YDC corporation.

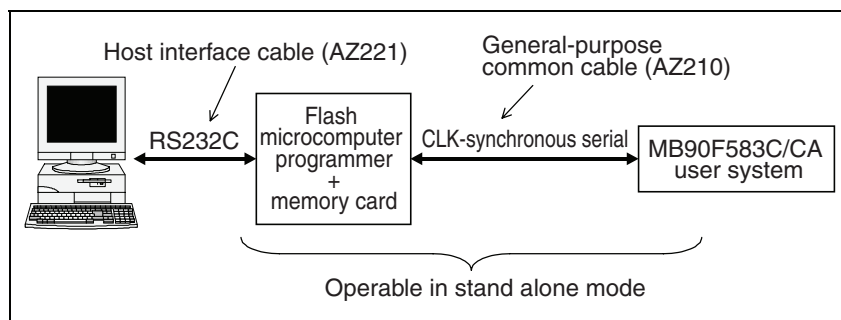
- 26.1 "Basic Configuration of MB90F583C/CA Serial Programming Connection"
- 26.2 "Example of Serial Programming Connection (When User Power Supply Is Used)"
- 26.3 "Example of Serial Programming Connection (When Power Is Supplied from a Programmer)"
- 26.4 "Example of Minimal Connection with the Flash Microcomputer Programmer (When User Power Supply Is Used)"
- 26.5 "Example of Minimal Connection with the Flash Microcomputer Programmer (When Power Is Supplied from a Programmer)"

26.1 Basic Configuration of MB90F583C/CA Serial Programming Connection

The MB90F583C/CA supports flash ROM serial onboard programming (Fujitsu standard). This section describes the specifications.

■ Basic Configuration of MB90F583C/CA Serial Programming Connection

The AF200 flash microcomputer programmer manufactured by YDC corporation. is used for Fujitsu standard serial onboard programming.



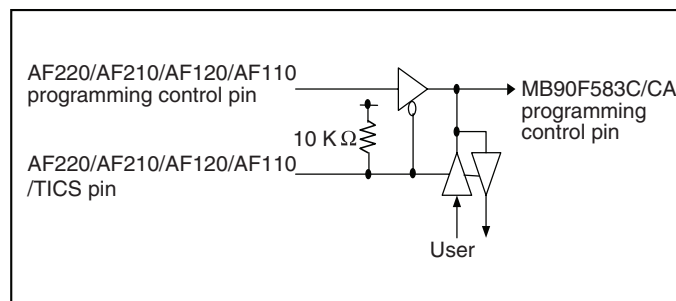
Note:

For the functions and operation of the flash microcomputer programmer (AF220/AF210/AF120/AF110), the general-purpose common cable (AZ210), and the connector, contact YDC corporation.

Table 26.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming

Pin	Function	Description
MD2, MD1 MD0	Mode pins	Programming mode is controlled from the flash microcomputer programmer.
X0, X1	Oscillation pin	In programming mode, the CPU internal operating clock pulse is generated by multiplying the PLL clock pulse by 1. Therefore, the oscillation clock is available as the internal operating clock. An oscillator used for serial reprogramming oscillates at 3 to 16 MHz.
P00, P01	Programming program activation pin	-
$\overline{\text{RST}}$	Reset pin	-
SIN0	Serial data input pin	UART is used as CLK synchronization mode.
SOT0	Serial data output pin	
SCK0	Serial clock pulse input pin	
C	C pin	Capacitor pin for power supply stabilization. Connect a ceramic capacitor of about 0.1 μF externally.
V_{CC}	Power supply voltage supply pin	Connection with the flash microcomputer programmer is not required if the programming voltage ($5\text{ V} \pm 10\%$) is supplied from the user system. Be sure not to connect the pin with the user power supply circuit when wiring.
V_{SS}	GND pin	Common to the GND of the flash microcomputer programmer.
$\overline{\text{HST}}$	Hardware standby pin	Input the H level in serial programming mode.

If the P00, SIN, SOT0, and SCK0 pins are used also by the user system, the following control circuit is required. The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.



Sections 26.1 "Basic Configuration of MB90F583C/CA Serial Programming Connection" to 26.5 "Example of Minimal Connection with the Flash Microcomputer Programmer (When Power Is Supplied from a Programmer)" show the following examples of serial programming connections for reference:

- Example of serial programming connection (when user power supply is used)
- Example of serial programming connection (when power is supplied from a programmer)
- Example of minimal connection with the flash microcomputer programmer (when user power supply is used)
- Example of minimal connection with the flash microcomputer programmer (when power is supplied from a programmer)

■ Oscillation Clock Frequency and Serial Clock Input Frequency

The serial clock frequency that can be entered into the MB90F583C/CA is obtained from the following equation. Change the serial clock input frequency in accordance with the user's oscillation clock frequency by setting the flash microcomputer programmer.

Input-enabling serial clock frequency = 0.125 x oscillation clock frequency

Table 26.1-2

Oscillation clock frequency	Maximum serial clock frequency that can be entered by the microcomputer	Maximum serial clock frequency that can be set by AF220/AF210/AF120/AF110	Maximum serial clock frequency that can be set by 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

■ System Configuration of Flash Microcomputer Programmer (Manufactured by YDC Corporation)

Table 26.1-3 System Configuration of Flash Microcomputer Programmer (Manufactured by YDC Corporation)

Type		Function
Main unit	AF220/AC4P	Built-in Ethernet interface model with 100 VAC to 220 VAC power adapter
	AF210/AC4P	Standard model with 100 VAC to 220 VAC power adapter
	AF120/AC4P	Single-key built-in Ethernet interface model with 100 VAC to 220 VAC power adapter
	AF110/AC4P	Single-key model with 100 VAC to 220 VAC power adapter
AZ221		Programmer-dedicated RS232C cable for PC/AT
AZ210		Standard target probe (a) length: 1 m
FF201		Control module for Fujitsu F ² MC-16LX flash microcomputer
/P2		2-MB PC card (option) flash memory capacity - for 128 KB
/P4		4-MB PC card (option) flash memory capacity - for 512 KB

Note:

Although production of the AF200 flash microcomputer programmer has ended, it can still be used if the control module FF201 is used. The serial programming connection is used in the connection example given in the next section.

26.2 Example of Serial Programming Connection (When User Power Supply Is Used)

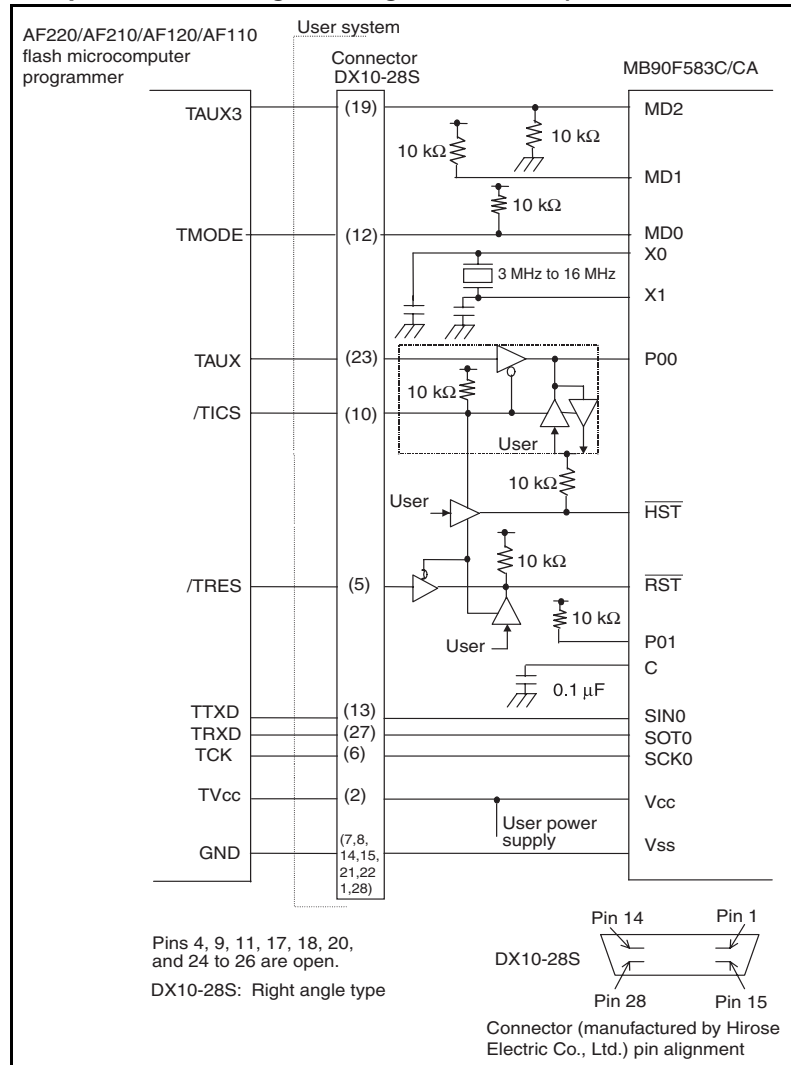
Figure 26.2-1 "Example of Serial Programming Connection (when User Power Supply Is Used)" shows an example of serial programming connection when the power supply voltage of the microcomputer is supplied from the user power supply.

The value 1 and 0 are input to mode pins MD2 and MD0 from TAUX3 and TMODE of the AF220/AF210/AF120/AF110 programmer.

Serial reprogramming mode: MD2, MD1, MD0 = 110.

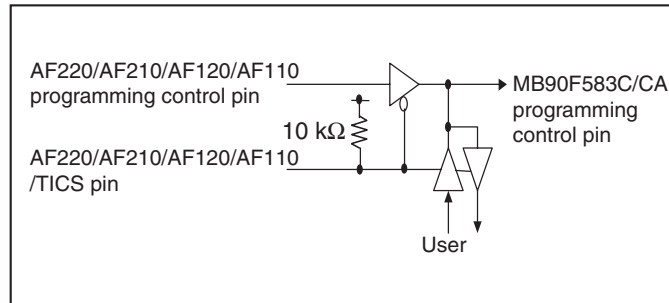
■ Example of Serial Programming Connection (when User Power Supply is Used)

Figure 26.2-1 Example of Serial Programming Connection (when User Power Supply Is Used)



26.2 Example of Serial Programming Connection (When User Power Supply Is Used)

- As with the case in which the P00 pin is also used, the following control circuit is required when the SIN0, SOT0, and SCK0 pins are also used by the user system. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect the AF220/AF210/AF120/AF110, turning the user power supply off.

26.3 Example of Serial Programming Connection (When Power is Supplied from a Programmer)

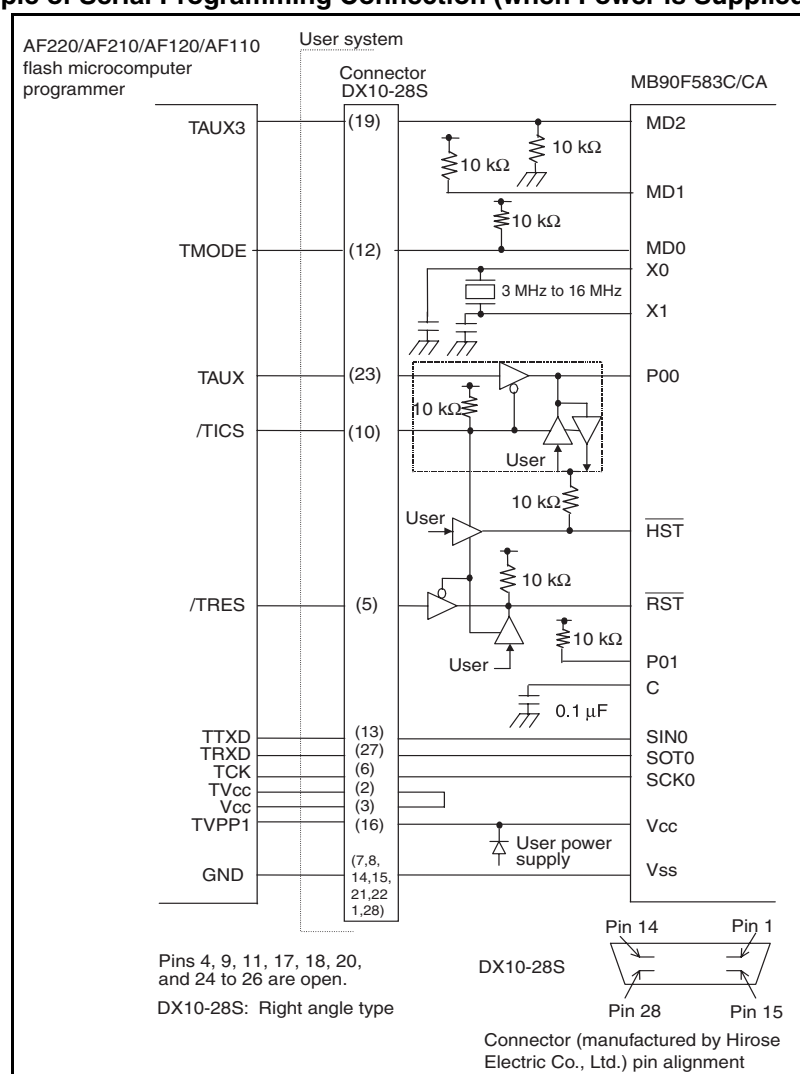
Figure 26.3-1 "Example of Serial Programming Connection (when Power is Supplied from a Programmer)" shows an example of serial programming connection when the power supply voltage of the microcomputer is supplied from programmer power supply.

The value 1 and 0 are input to mode pins MD2 and MD0 from TAUX3 and TMODE of the AF220/AF210/AF120/AF110 programmer.

Serial reprogramming mode: MD2, MD1, MD0 = 110.

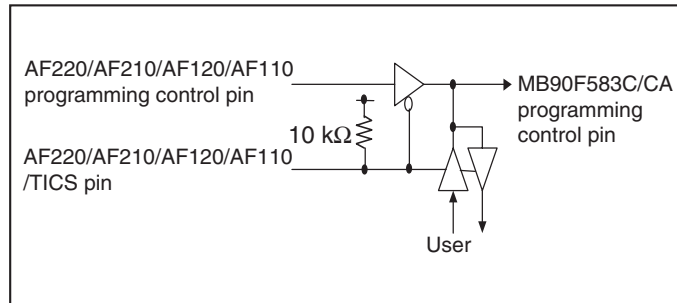
■ Example of Serial Programming Connection (when Power is Supplied from a Programmer)

Figure 26.3-1 Example of Serial Programming Connection (when Power is Supplied from a Programmer)



26.3 Example of Serial Programming Connection (When Power is Supplied from a Programmer)

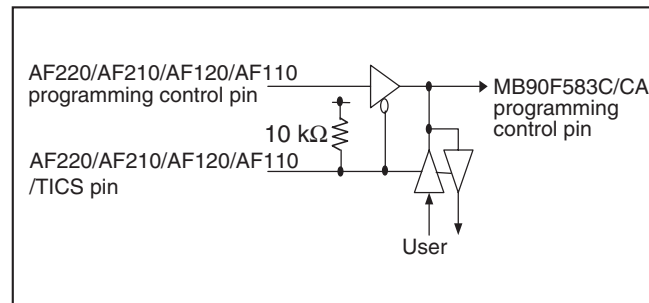
- As with the case in which the P00 pin is also used, the following control circuit is required when the SIN0, SOT0, and SCK0 pins are also used by the user system. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect the AF220/AF210/AF120/AF110, turning the user power supply off.
- When programming power supply is supplied from the AF220/AF210/AF120/AF110, be sure not to connect the power supply pin with the user power supply circuit.

26.4 Example of Minimum Connection with the Flash Microcomputer Programmer (When User Power Supply is Used)

- When the SIN0, SOT0, and SCK0 pins are also used by the user system, the following control circuit is required. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect the AF220/AF210/AF120/AF110, turning the user power supply off.

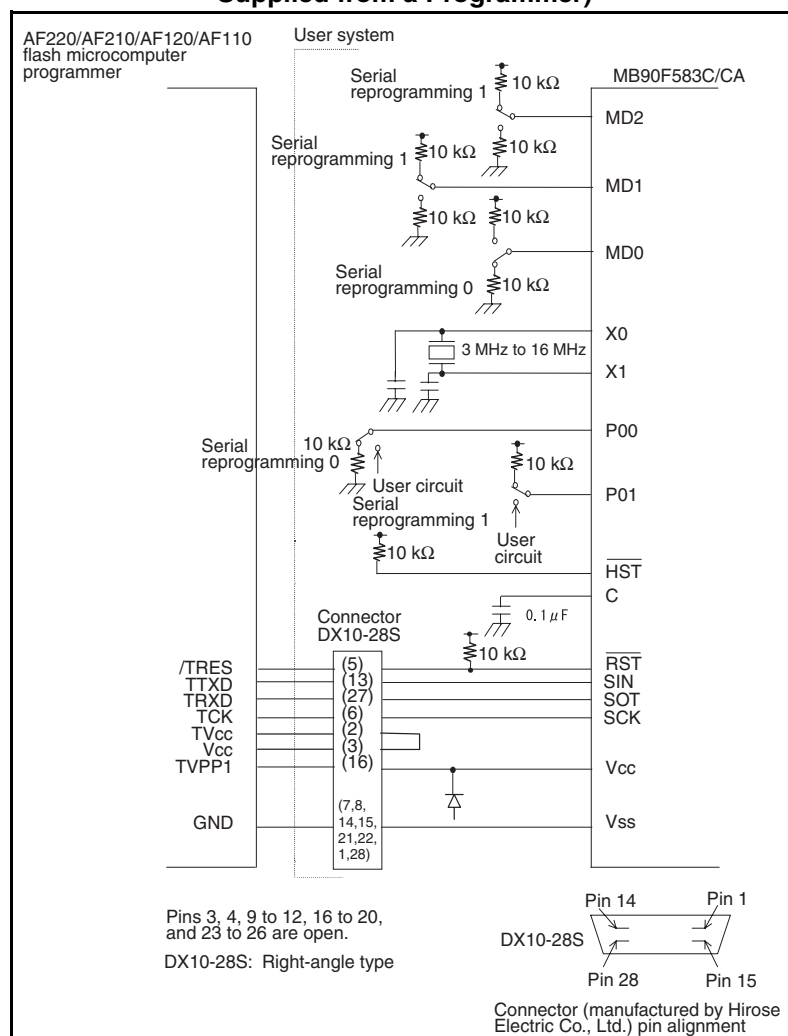
26.5 Example of Minimum Connection with the Flash Microcomputer Programmer (When Power is Supplied from a Programmer)

Figure 26.5-1 "Example of Minimum Connection with Flash Microcomputer Programmer (when Power is Supplied from a Programmer)" shows an example of minimum connection with the flash microcomputer programmer when the power supply voltage of the microcomputer is supplied from a programmer. Serial reprogramming mode: MD2, MD1, MD0 = 110.

■ Example of Minimum Connection with the Flash Microcomputer Programmer (when Power is Supplied from a Programmer)

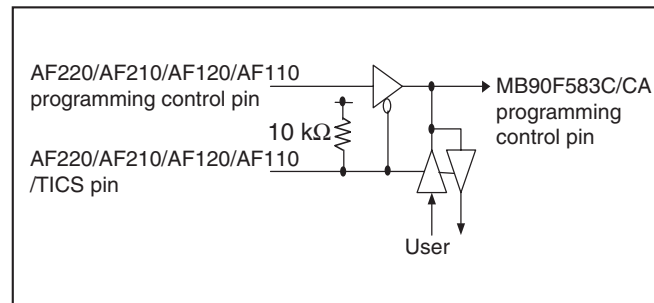
The MD2, MD1, MD0, and P00 pins need not be connected to the flash microcomputer programmer if each pin is connected, as shown below, for flash memory programming.

Figure 26.5-1 Example of Minimum Connection with Flash Microcomputer Programmer (when Power is Supplied from a Programmer)



26.5 Example of Minimum Connection with the Flash Microcomputer Programmer (When Power is Supplied from a Programmer)

- When the SIN0, SOT0, and SCK0 pins are also used by the user system, the following control circuit is required. (The user circuit can be disconnected by the /TICS signal of the flash microcomputer programmer during serial programming.)



- Connect the AF220/AF210/AF120/AF110, turning the user power supply off.
- When programming power supply is supplied from the AF220/AF210/AF120/AF110, be sure not to connect the power supply pin with the user power supply circuit.

APPENDIX

The appendix describes the I/O map and instructions.

APPENDIX A "I/O Map"

APPENDIX B "Instructions"

APPENDIX A I/O Map

The addresses are assigned to the registers for each resource of this micro controller as follows.

■ I/O Map

Table A-1 I/O Map

Address	Register	Abbreviation	Access	Resource	Initial value
00 _H	Port 0 data register	PDR0	R/W	Port 0	XXXXXXXX _B
01 _H	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX _B
02 _H	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX _B
03 _H	Port 3 data register	PDR3	R/W	Port 3	XXXXXXXX _B
04 _H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX _B
05 _H	Port 5 data register	PDR5	R/W	Port 5	11111111 _B
06 _H	Port 6 data register	PDR6	R/W	Port 6	--XXXX _B
07 _H	Port 7 data register	PDR7	R/W	Port 7	---XXXX _B
08 _H	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX _B
09 _H	Port 9 data register	PDR9	R/W	Port 9	XXXXXXXX _B
0A _H	Port A data register	PDRA	R/W	Port A	-----XXX _B
0B _H to 0F _H	Disabled				
10 _H	Port 0 data direction register	DDR0	R/W	Port 0	00000000 _B
11 _H	Port 1 data direction register	DDR1	R/W	Port 1	00000000 _B
12 _H	Port 2 data direction register	DDR2	R/W	Port 2	00000000 _B
13 _H	Port 3 data direction register	DDR3	R/W	Port 3	00000000 _B
14 _H	Port 4 data direction register	DDR4	R/W	Port 4	00000000 _B
15 _H	Port 5 data direction register	DDR5	R/W	Port 5	00000000 _B
16 _H	Port 6 data direction register	DDR6	R/W	Port 6	--000000 _B
17 _H	Port 7 data direction register	DDR7	R/W	Port 7	---0000- _B
18 _H	Port 8 data direction register	DDR8	R/W	Port 8	00000000 _B
19 _H	Port 9 data direction register	DDR9	R/W	Port 9	00000000 _B
1A _H	Port A data direction register	DDRA	R/W	Port A	-----000 _B

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
1B _H	Port 4 output terminal register	ODR4	R/W	Port 4	00000000 _B
1C _H	Port 5 analog input enable register	ADER	R/W	Port 5 and A/D	11111111 _B
1D _H to 1F _H	Disabled				
20 _H	Serial mode register 0	SMR0	R/W	UART0	00000000 _B
21 _H	Serial control register 0	SCR0	R/W		00000100 _B
22 _H	Serial input /serial output register 0	SIDR0/ SODR0	R/W		XXXXXXXX _B
23 _H	Serial status register 0	SSR0	R/W		00001-00 _B
24 _H	Serial mode register 1	SMR1	R/W	UART1	00000000 _B
25 _H	Serial control register 1	SCR1	R/W		00000100 _B
26 _H	Serial input /serial output register 1	SIDR1/ SODR1	R/W		XXXXXXXX _B
27 _H	Serial status register 1	SSR1	R/W		000001-00 _B
28 _H	Serial mode register 2	SMR2	R/W	UART2	00000000 _B
29 _H	Serial control register 2	SCR2	R/W		00000100 _B
2A _H	Serial input /serial output register 2	SIDR2/ SODR2	R/W		XXXXXXXX _B
2B _H	Serial status register 2	SSR2	R/W		00001-00 _B
2C _H	Clock division control register 0	CDCR0	R/W	Communica- tion prescaler 0	0---1111 _B
2D _H	Disabled				
2E _H	Clock division control register 1	IDCR1	R/W	Communica- tion prescaler 1	0---1111 _B
2F _H	Disabled				
30 _H	Interrupt/DTP enable register	ENIR	R/W	DTP/external interrupt	00000000 _B
31 _H	Interrupt/DTP cause register	EIRR	R/W		XXXXXXXX _B
32 _H	Request level setting register (low- order)	ELVR	R/W		00000000 _B
33 _H	Request level setting register (high- order)				00000000 _B
34 _H	Clock division control register 2	CDCR2	R/W	Communica- tion prescaler 2	0---1111 _B

APPENDIX

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
35 _H	Disabled				
36 _H	Control status register (low-order)	ADCS1	R/W	A/D converter	00000000 _B
37 _H	Control status register (high-order)	ADCS2	R/W		00000000 _B
38 _H	Data register (low-order)	ADCR1	R		XXXXXXXX _B
39 _H	Data register (high-order)	ADCR2			00001-XX _B
3A _H	D/A converter data register 0	DAT0	R/W	D/A converter	00000000 _B
3B _H	D/A converter data register 1	DAT1	R/W		00000000 _B
3C _H	D/A control register 0	DACR0	R/W		-----0 _B
3D _H	D/A Control register 1	DACR1	R/W		-----0 _B
3E _H	Clock output enable register	CLKR	R/W	Clock monitor function	---0000 _B
3F _H	Disabled				
40 _H	Reload register L (ch.0)	PRLLO	R/W	8/16 bit PPG 0/1	XXXXXXXX _B
41 _H	Reload register H (ch. 0)	PRLH0	R/W		XXXXXXXX _B
42 _H	Reload register L (ch. 1)	PRLLO	R/W		XXXXXXXX _B
43 _H	Reload register H (ch. 1)	PRLH1	R/W		XXXXXXXX _B
44 _H	PPG0 operating mode control register	PPGC0	R/W		0X000XX1 _B
45 _H	PPG1 operating mode control register	PPGO1	R/W		0X000001 _B
46 _H	PPG0 and PPG1 operation output control register	PPGOE	R/W		00000000 _B
47 _H	Disabled				
48 _H	Timer control status register 0 (low-order)	TMCSR0	R/W	16-bit reload timer 0	00000000 _B
49 _H	Timer control status register 0 (high-order)				---0000 _B
4A _H	16-bit timer register 0/16-bit reload register 0	TMR0/ TMRLR0	R/W		XXXXXXXX _B
4B _H					XXXXXXXX _B
4C _H	Timer control status register 1 (low-order)	TMCSR1	R/W	16-bit reload timer 1	00000000 _B
4D _H	Timer control status register 1 (high-order)				---0000 _B
4E _H	16-bit timer register 1/16-bit reload register 1	TMR1/ TMRLR1	R/W		XXXXXXXX _B
4F _H					XXXXXXXX _B

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value	
50 _H	Lower timer control status register 2	TMCSR2	R/W	16-bit reload timer 2	00000000 _B	
51 _H	Upper timer control status register 2				----0000 _B	
52 _H	16-bit timer register 2/16-bit reload register 2	TMR2/ TMRLR2	R/W		XXXXXXXX _B	
53 _H					XXXXXXXX _B	
54 _H	PWC control status register (low-order)	PWCSR	R/W	16-bit PWC timer	00000000 _B	
55 _H	PWC control status register (high-order)				00000000 _B	
56 _H	PWC data buffer register (low-order)	PWCR	R/W		XXXXXXXX _B	
57 _H	PWC data buffer register (high-order)				XXXXXXXX _B	
58 _H	Dividing ratio control register	DIVR	R/W		-----00 _B	
59 _H	Disabled					
5A _H	Compare register ch.0	OCCP0	R/W	Output compare (ch.0 to ch.1)	XXXXXXXX _B	
5B _H					XXXXXXXX _B	
5C _H	Compare register ch.1	OCCP1	R/W		XXXXXXXX _B	
5D _H					XXXXXXXX _B	
5E _H	Compare control status register ch.0	OCS0	R/W		0000--00 _B	
5F _H	Compare control status register ch.1	OCS1	R/W		----0000 _B	
60 _H	Input capture register 0	IPCP0	R	Input capture (ch.0 to ch.3)	XXXXXXXX _B	
61 _H					XXXXXXXX _B	
62 _H	Input capture register 1	IPCP1	R		XXXXXXXX _B	
63 _H					XXXXXXXX _B	
64 _H	Input capture register 2	IPCP2	R		XXXXXXXX _B	
65 _H					XXXXXXXX _B	
66 _H	Input capture register 3	IPCP3	R		XXXXXXXX _B	
67 _H					XXXXXXXX _B	
68 _H	Input capture control status register ch.0 and ch.1	ICS01	R/W		00000000 _B	
69 _H	Disabled					
6A _H	Input capture control status register ch.2 and ch.3	ICS23	R/W			00000000 _B

APPENDIX

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
6B _H	Disabled				
6C _H	Timer data register (low-order)	TCDTL	R/W	Free-run timer	00000000 _B
6D _H	Timer data register (high-order)	TCDTH	R/W		00000000 _B
6E _H	Timer control status register	TCCS	R/W		00000000 _B
6F _H	ROM mirror function selection register	ROMM	W	ROM mirror function	-----1 _B
70 _H	Local address setting register (low-order)	MAWL	R/W	IEBus contoroler	XXXXXXXX _B
71 _H	Local address setting register (high-order)	MAWH	R/W		XXXXXXXX _B
72 _H	Slave address setting register (low-order)	SAWL	R/W		XXXXXXXX _B
73 _H	Slave address setting register (high-order)	SAWH	R/W		XXXXXXXX _B
74 _H	Text length bit setting register	DEWR	R/W		00000000 _B
75 _H	Broadcast control bit setting register	DCWR	R/W		00000000 _B
76 _H	Command register (low-order)	CMRL	R/W		11000000 _B
77 _H	Command register (high-order)	CMRH	R/W		0000000X _B
78 _H	Status register (low-order)	STRL	R		0011XXXX _B
79 _H	Status register (high-order)	STRH	R/W		00XX0000 _B
7A _H	Lock read register (low-order)	LRRL	R		XXXXXXXX _B
7B _H	Lock read register (high-order)	LRRH	R/W		1110XXXX _B
7C _H	Master address read register (low-order)	MARL	R		XXXXXXXX _B
7D _H	Master address read register (high-order)	MARH	R		1111XXXX _B
7E _H	Text length bit read register	DERR	R		XXXXXXXX _B
7F _H	Broadcast control bit read register	DCRR	R		000XXXXX _B
80 _H	Write data buffer	WDB	W	UART3	XXXXXXXX _B
81 _H	Read data buffer	RDB	R		XXXXXXXX _B
82 _H	Serial mode register 3	SMR3	R/W		00000000 _B
83 _H	Serial control register 3	SCR3	R/W		00000100 _B
84 _H	Serial input/serial output register 3	SIDR3/ SODR3	R/W		XXXXXXXX _B
85 _H	Serial status register 3	SSR3	R/W		00001-00 _B

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
86 _H	PWC noise filter register	RNCR	R/W	PWC noise filter	-----000 _B
87 _H	Clock division control register 3	CDCR3	R/W	Communication prescaler 3	0---1111 _B
88 _H	Serial mode register 4	SMR4	R/W	UART4	00000000 _B
89 _H	Serial control register 4	SCR4	R/W		00000100 _B
8A _H	Serial input/serial output register 4	SIDR4/ SODR4	R/W		XXXXXXXX _B
8B _H	Serial status register 4	SSR4	R/W		00001-00 _B
8C _H	Port 0 input pull-up resistor setting register	RDR0	R/W	Port 0	00000000 _B
8D _H	Port 1 input pull-up resistor setting register	RDR1	R/W	Port 1	00000000 _B
8E _H	Port 6 input pull-up resistor setting register	RDR6	R/W	Port 6	--000000 _B
8F _H	Clock division control register 4	CDCR4	R/W	Communication prescaler 4	0---1111 _B
90 _H to 9D _H	Disabled				
9E _H	Program address detection control/status register	PACSR	R/W	Address match detection function	00000000 _B
9F _H	Delayed interrupt source generation/cancel register	DIRR	R/W	Delayed interrupt generating module	-----0 _B
A0 _H	Low-power consumption mode control register	LPMCR	R/W	Low-power consumption control circuit	0001100- _B
A1 _H	Clock selection register	CKSCR	R/W		11111100 _B
A2 _H to A4 _H	Disabled				
A5 _H	Automatic ready function selection register	ARSR	W	External bus pin control circuit	0011-00 _B
A6 _H	External address output control register	HACR	W		00000000 _B
A7 _H	Bus control signal selection register	ECSR	W		0000000- _B

APPENDIX

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
A8 _H	Watchdog control register	WDTC	R/W	Watchdog timer	XXXXXX111 _B
A9 _H	Time-based timer control register	TBTC	R/W	Time-base timer	1--00100 _B
AA _H	Clock timer control register	WTC	R/W	Clock timer	1X000000 _B
AB _H to AD _H	Disabled				
AE _H	Flash memory control status register	FMCS	R/W	Flash interface	000X000 _B
AF _H	Disabled				
B0 _H	Interrupt control register 00	ICR00	R/W	Interrupt controller	00000111 _B
B1 _H	Interrupt control register 01	ICR01	R/W		00000111 _B
B2 _H	Interrupt control register 02	ICR02	R/W		00000111 _B
B3 _H	Interrupt control register 03	ICR03	R/W		00000111 _B
B4 _H	Interrupt control register 04	ICR04	R/W		00000111 _B
B5 _H	Interrupt control register 05	ICR05	R/W		00000111 _B
B6 _H	Interrupt control register 06	ICR06	R/W		00000111 _B
B7 _H	Interrupt control register 07	ICR07	R/W		00000111 _B
B8 _H	Interrupt control register 08	ICR08	R/W		00000111 _B
B9 _H	Interrupt control register 09	ICR09	R/W		00000111 _B
BA _H	Interrupt control register 10	ICR10	R/W		00000111 _B
BB _H	Interrupt control register 11	ICR11	R/W		00000111 _B
BC _H	Interrupt control register 12	ICR12	R/W		00000111 _B
BD _H	Interrupt control register 13	ICR13	R/W		00000111 _B
BE _H	Interrupt control register 14	ICR14	R/W		00000111 _B
BF _H	Interrupt control register 15	ICR15	R/W		00000111 _B
C0 _H to FF _H	External area				
100 _H to # _H	RAM area				
# _H to 1FEF _H	Reserved area				

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
1FF0 _H	Program address detection register 0 (low-order)	PADR0	R/W	Address match detection function	XXXXXXXX _B
1FF1 _H	Program address detection register 0 (middle-order)		R/W		XXXXXXXX _B
1FF2 _H	Program address detection register 0 (high-order)		R/W		XXXXXXXX _B
1FF3 _H	Program address detection register 1 (low-order)	PADR1	R/W		XXXXXXXX _B
1FF4 _H	Program address detection register 1 (middle-order)		R/W		XXXXXXXX _B
1FF5 _H	Program address detection register 1 (high-order)		R/W		XXXXXXXX _B
1FF6 _H to 1FFF _H	Reserved area				

- Initial values --> 0: 0, 1: 1, X: Not defined, -: Not defined (none)
- An area of address 00FFH and smaller addresses is a reserved area.
- The border address #_H between RAM area and reserved area is dependent on the part number.

Note:

Values initialized by a reset are described as initial values of writable bits. Note that the values are not read values.

Initialization of the LPMCR, CKSCR, and WDTC registers is dependent on the reset type. The initial values, set when initialized, are described.

APPENDIX B Instructions

APPENDIX B describes the instructions used by the F²MC-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 F²MC-16LX Instruction List
- B.9 Instruction Map

B.1 Instruction Types

The F²MC-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 F²MC-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 F²MC-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 F²MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)

■ Effective Address Field

Table B.2-1 lists the address formats specified by the effective address field.

Table B.2-1 Effective Address Field

Code	Representation			Address format	Default bank
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	DTB
09	@RW1				DTB
0A	@RW2				ADB
0B	@RW3				SPB
0C	@RW0+			Register indirect with post increment	DTB
0D	@RW1+				DTB
0E	@RW2+				ADB
0F	@RW3+				SPB
10	@RW0+disp8			Register indirect with 8-bit displacement	DTB
11	@RW1+disp8				DTB
12	@RW2+disp8				ADB
13	@RW3+disp8				SPB
14	@RW4+disp8			Register indirect with 8-bit displacement	DTB
15	@RW5+disp8				DTB
16	@RW6+disp8				ADB
17	@RW7+disp8				SPB
18	@RW0+disp16			Register indirect with 16-bit displacement	DTB
19	@RW1+disp16				DTB
1A	@RW2+disp16				ADB
1B	@RW3+disp16				SPB
1C	@RW0+RW7			Register indirect with index	DTB
1D	@RW1+RW7			Register indirect with index	DTB
1E	@PC+disp16			PC indirect with 16-bit displacement	PCB
1F	addr16			Direct address	DTB

B.3 Direct Addressing

An operand value, register, or address is specified explicitly in direct addressing mode.

■ Direct Addressing

- Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

Figure B.3-1 Example of Immediate Addressing (#imm)

MOVW A, #01212H (This instruction stores the operand value in A.)											
Before execution	A	<table><tr><td>2</td><td>2</td><td>3</td><td>3</td><td>:</td><td>4</td><td>4</td><td>5</td><td>5</td></tr></table>	2	2	3	3	:	4	4	5	5
2	2	3	3	:	4	4	5	5			
After execution	A	<table><tr><td>4</td><td>4</td><td>5</td><td>5</td><td>:</td><td>1</td><td>2</td><td>1</td><td>2</td></tr></table> (Some instructions transfer AL to AH.)	4	4	5	5	:	1	2	1	2
4	4	5	5	:	1	2	1	2			

- Register direct addressing

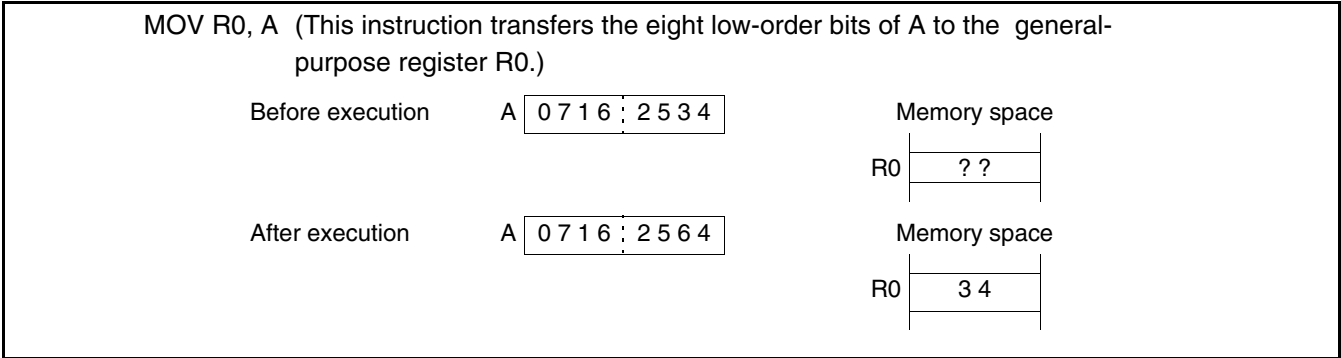
Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

Table B.3-1 Direct Addressing Registers

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

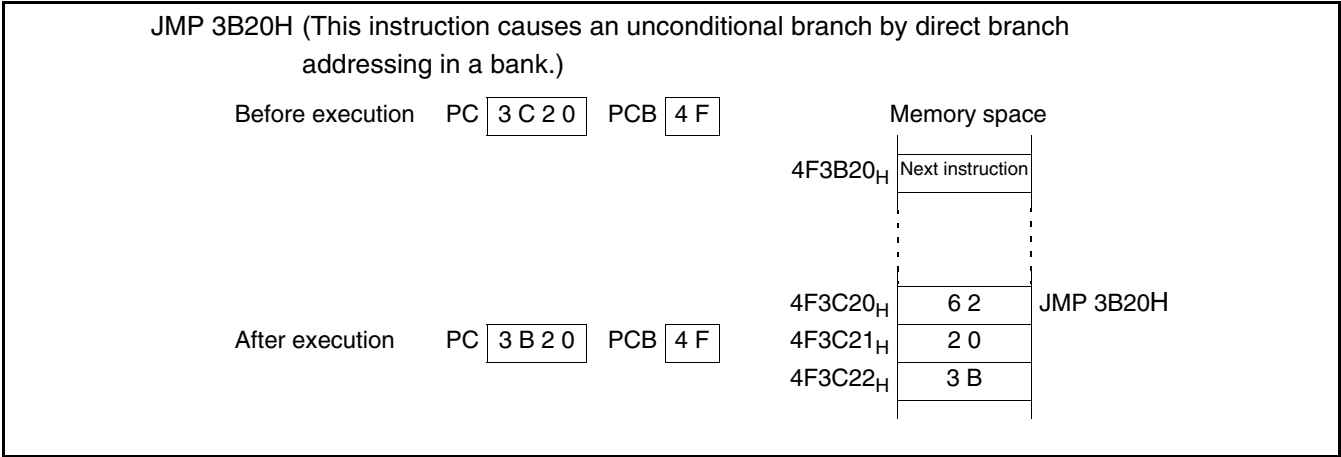
Figure B.3-2 Example of Register Direct Addressing



● Direct branch addressing (addr16)

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bit23 to bit16 of the address are specified by the program counter bank register (PCB).

Figure B.3-3 Example of Direct Branch Addressing (addr16)

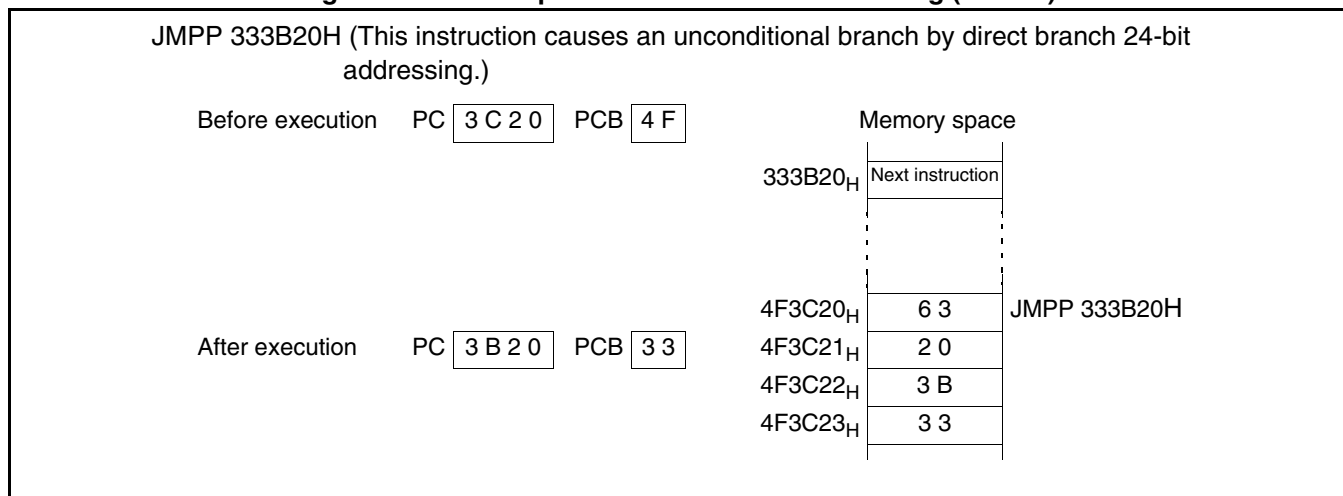


APPENDIX B Instructions

- 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.

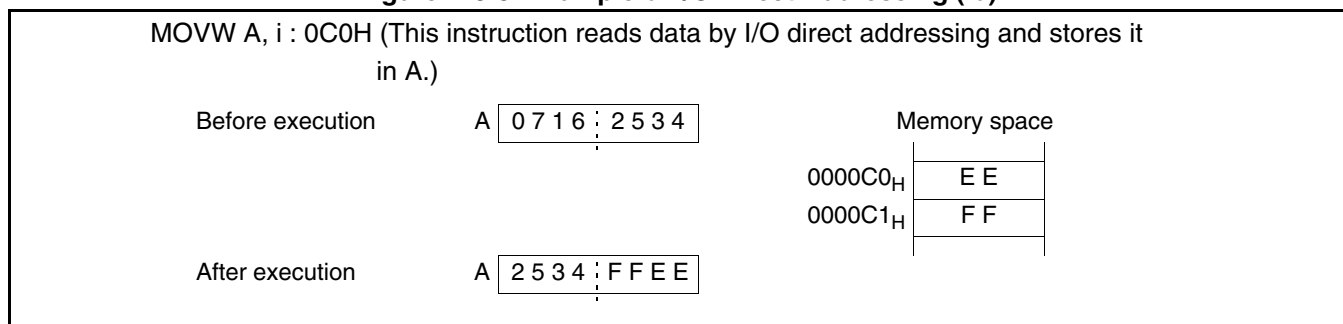
Figure B.3-4 Example of Direct Branch Addressing (addr24)



- 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_H to 0000FF_H is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

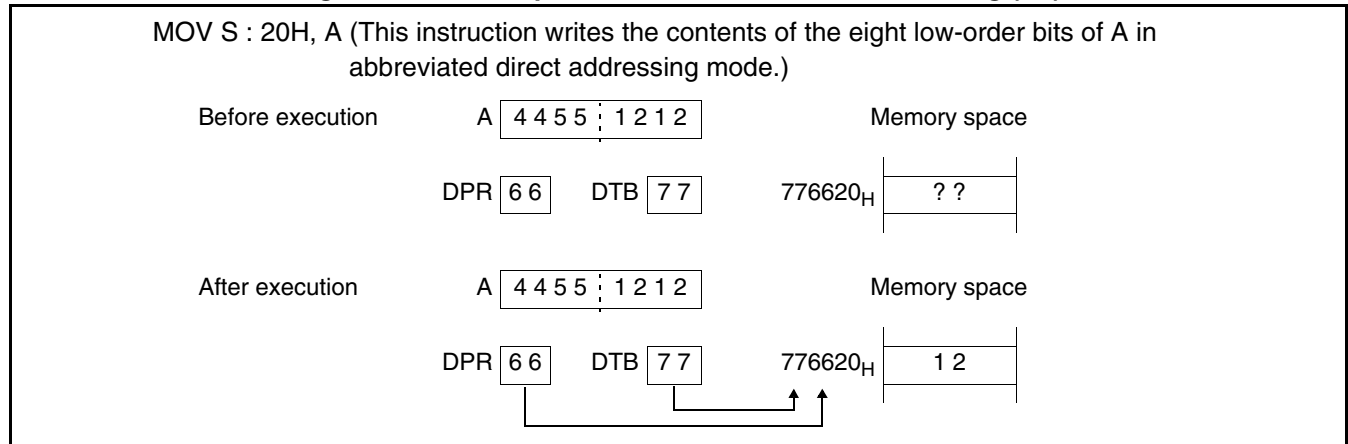
Figure B.3-5 Example of I/O Direct Addressing (io)



- Abbreviated direct addressing (dir)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB).

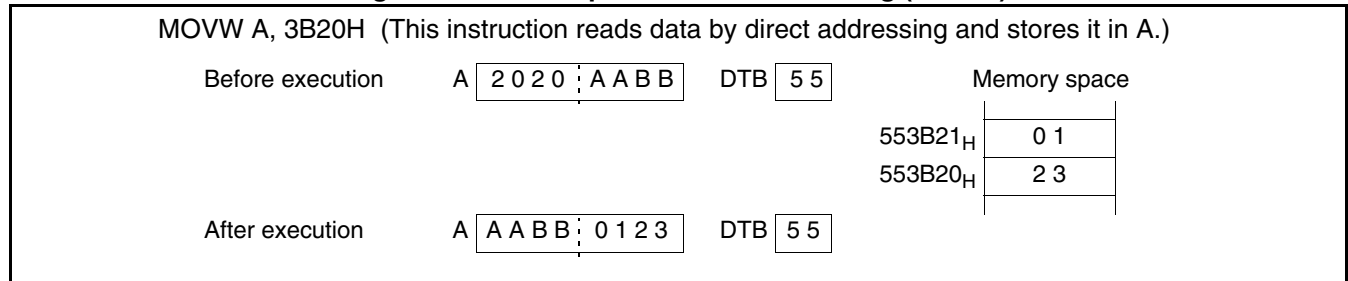
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)



- Direct addressing (addr16)

Specify the 16 low-order bits of a memory address explicitly in an operand. Address bits 16 to 23 are specified by the data bank register (DTB). A prefix instruction for access space addressing is invalid for this mode of addressing.

Figure B.3-7 Example of Direct Addressing (addr16)

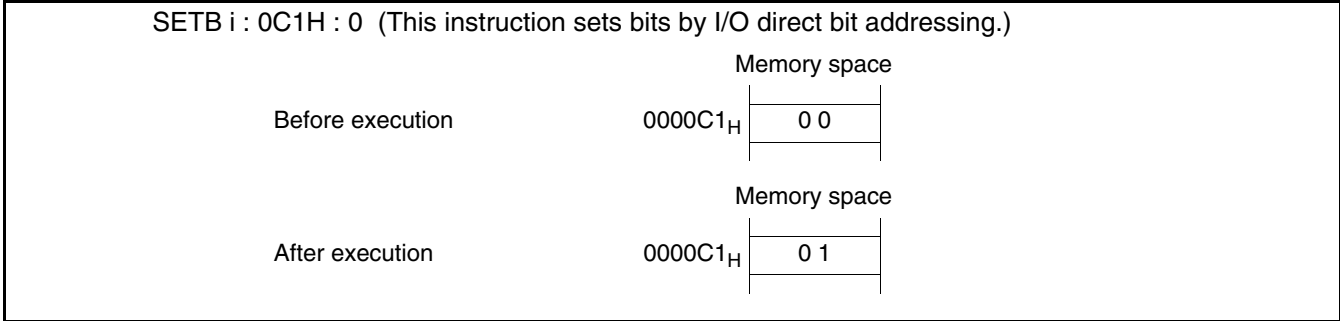


APPENDIX B Instructions

● I/O direct bit addressing (io:bp)

Specify bits in physical addresses 000000_H to 0000FF_H explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

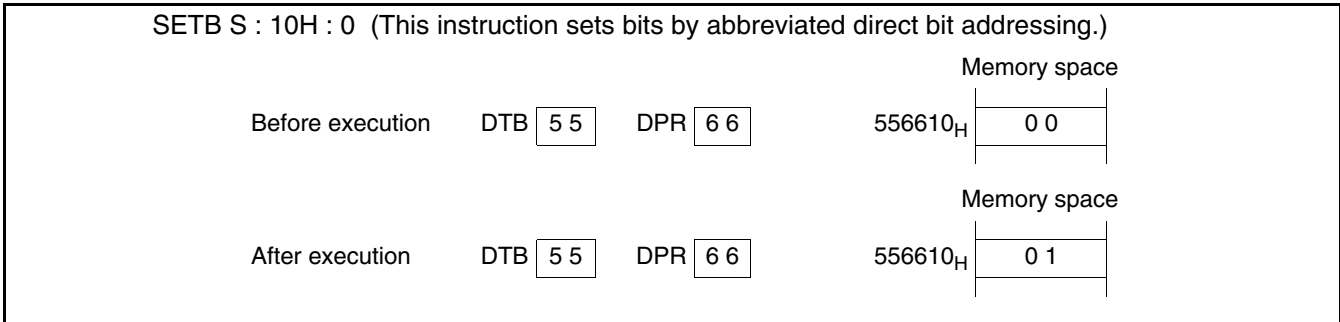
Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)



● Abbreviated direct bit addressing (dir:bp)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

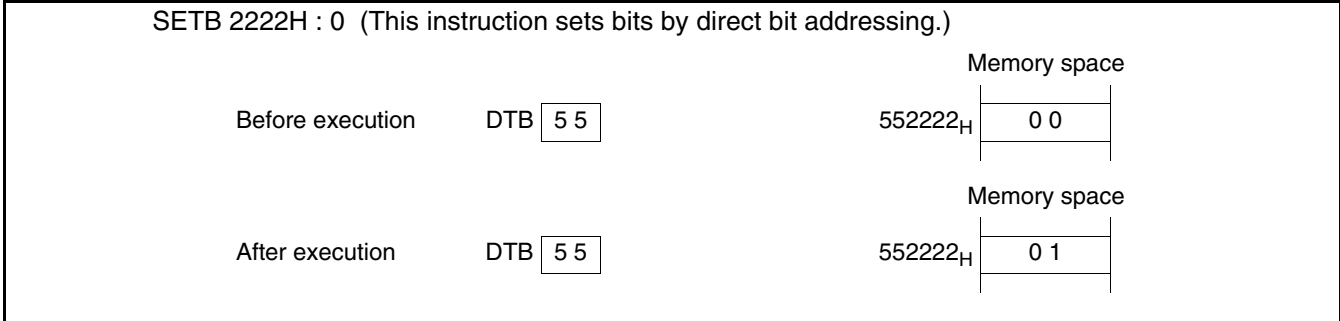
Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)



● Direct bit addressing (addr16:bp)

Specify arbitrary bits in 64 kilobytes explicitly. Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

Figure B.3-10 Example of Direct Bit Addressing (addr16:bp)



● Vector Addressing (#vct)

Specify vector data in an operand to indicate the branch destination address. There are two sizes for vector numbers: 4 bits and 8 bits. Vector addressing is used for a subroutine call or software interrupt instruction.

Figure B.3-11 Example of Vector Addressing (#vct)

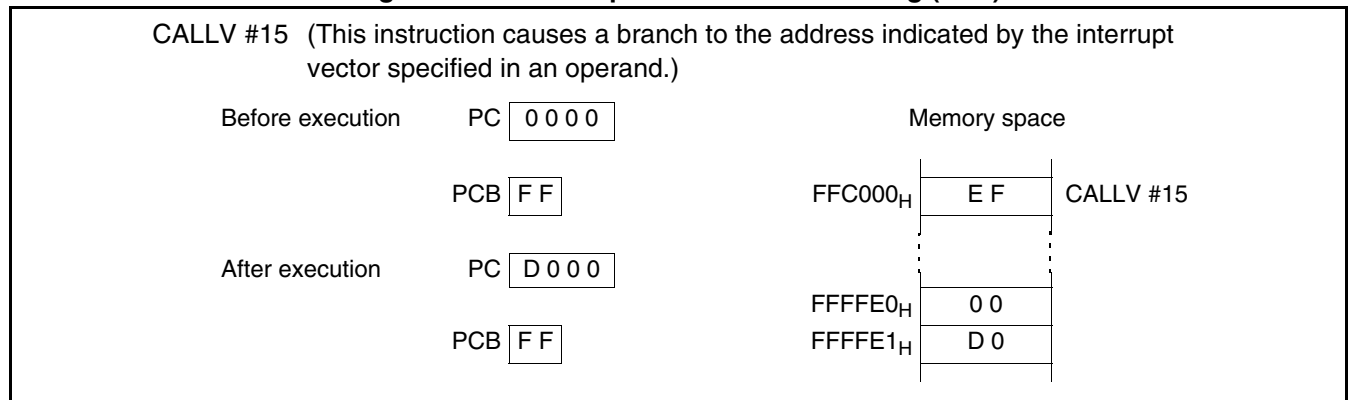


Table B.3-2 CALLV Vector List

Instruction	Vector address L	Vector address H
CALLV #0	XXFFFE _H	XXFFFF _H
CALLV #1	XXFFFC _H	XXFFFD _H
CALLV #2	XXFFFA _H	XXFFFB _H
CALLV #3	XXFFF8 _H	XXFFF9 _H
CALLV #4	XXFFF6 _H	XXFFF7 _H
CALLV #5	XXFFF4 _H	XXFFF5 _H
CALLV #6	XXFFF2 _H	XXFFF3 _H
CALLV #7	XXFFF0 _H	XXFFF1 _H
CALLV #8	XXFFEE _H	XXFFE _F _H
CALLV #9	XXFFEC _H	XXFFED _H
CALLV #10	XXFFEA _H	XXFFEB _H
CALLV #11	XXFFE8 _H	XXFFE9 _H
CALLV #12	XXFFE6 _H	XXFFE7 _H
CALLV #13	XXFFE4 _H	XXFFE5 _H
CALLV #14	XXFFE2 _H	XXFFE3 _H
CALLV #15	XXFFE0 _H	XXFFE1 _H

Note: A PCB register value is set in XX.

Note:

When the program counter bank register (PCB) is FF_H, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2).

B.4 Indirect Addressing

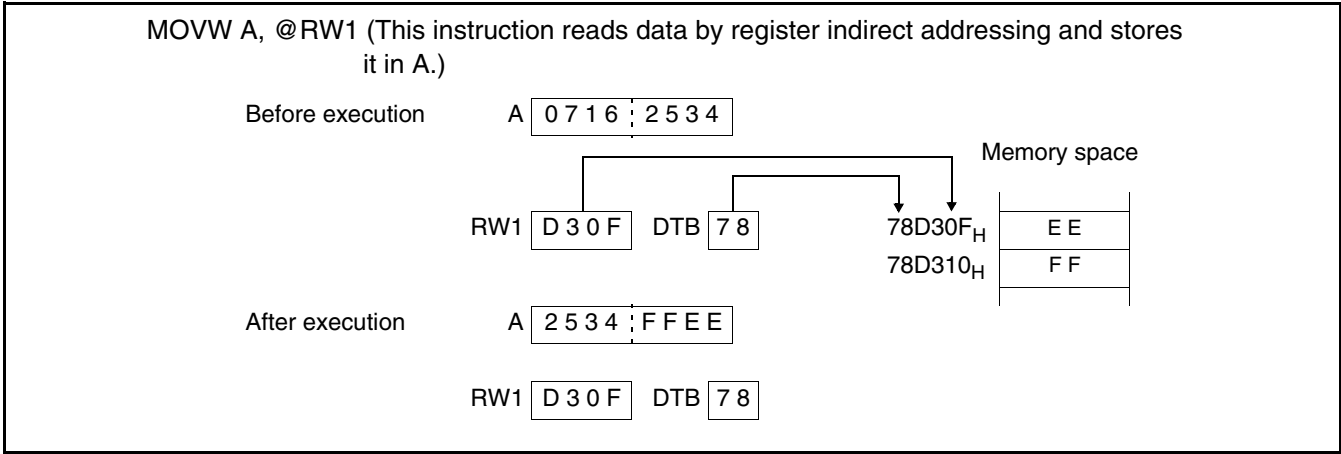
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ Indirect Addressing

- Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

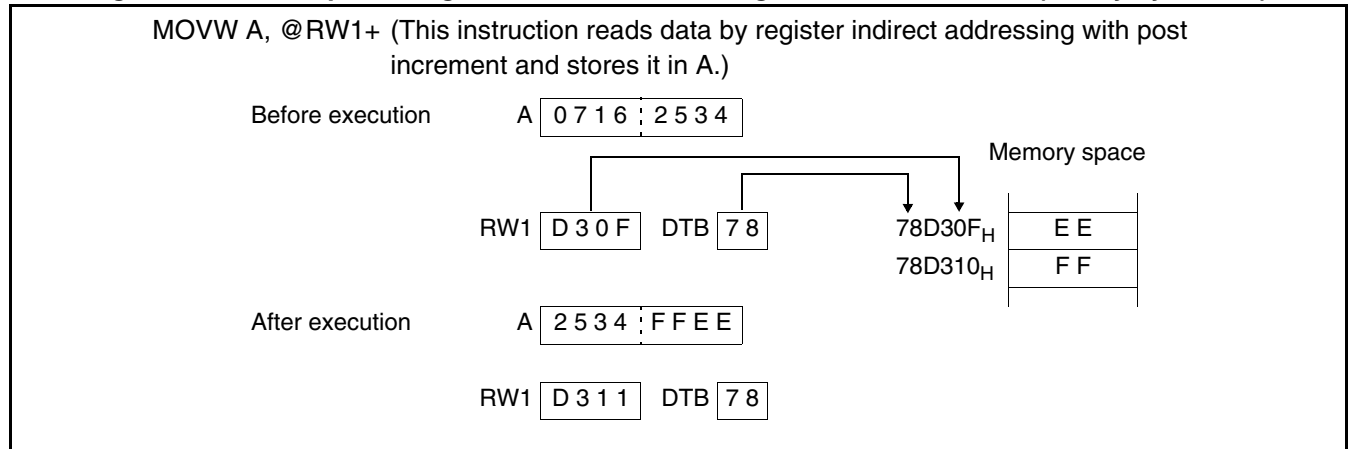
Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)



- Register indirect addressing with post increment (@RWj+ j = 0 to 3)

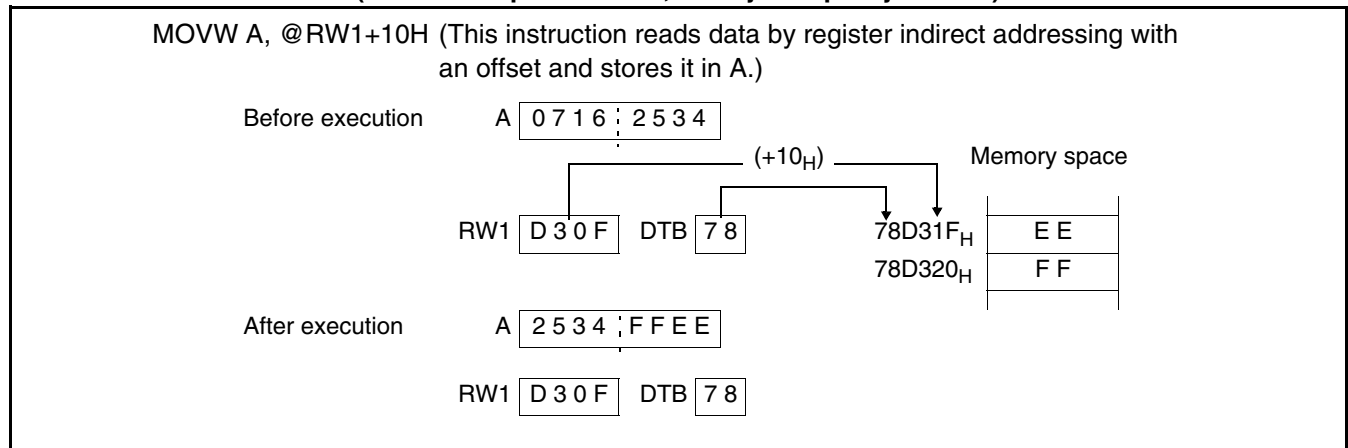
Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.

Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj+ j = 0 to 3)

● Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

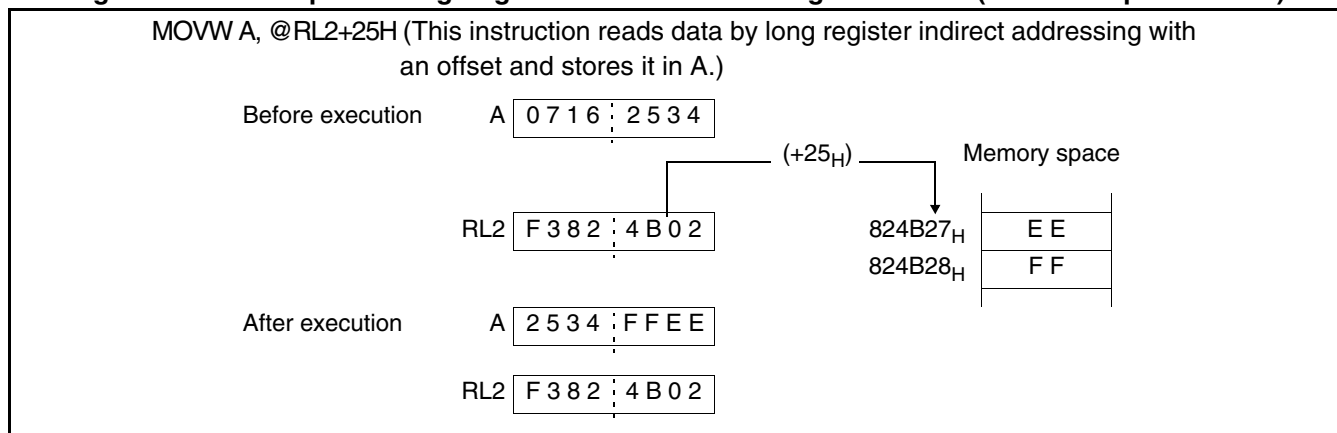
Figure B.4-3 Example of Register Indirect Addressing with Offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)

APPENDIX B Instructions

- Long register indirect addressing with offset ($@R_{Li} + \text{disp8}$ $i = 0$ 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 + disp8 i = 0 to 3)

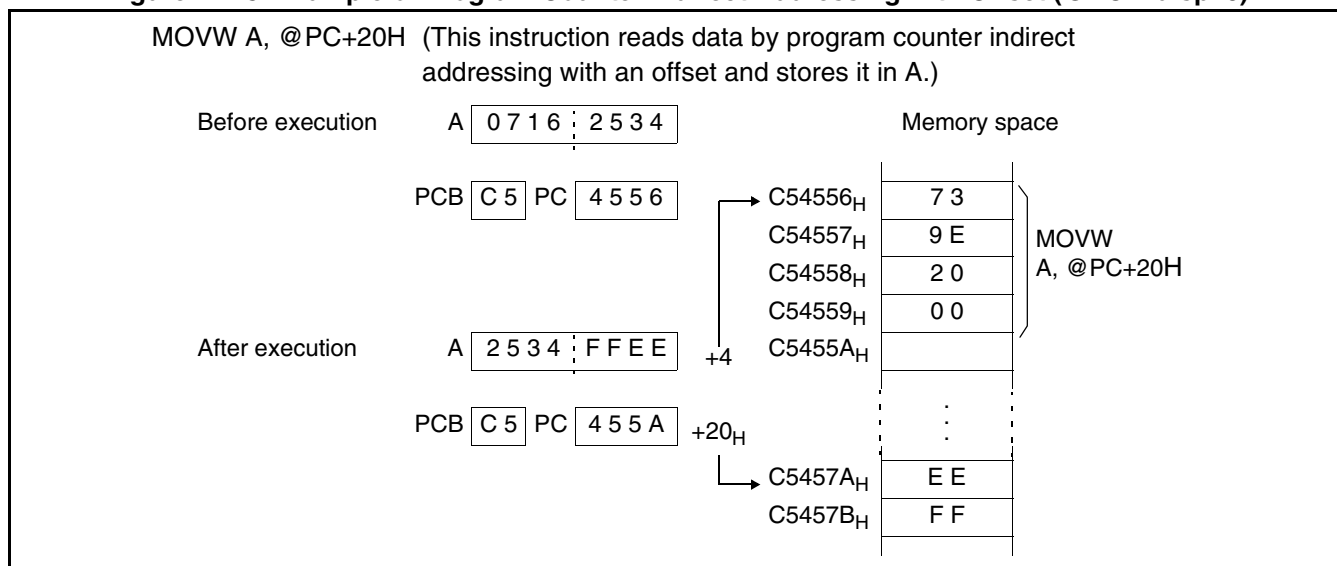


- Program counter indirect addressing with offset ($@PC + \text{disp16}$)

Memory is accessed using the address indicated by (instruction address + 4 + disp16). The offset is one word long. Address bits 16 to 23 are specified by the program counter bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address + disp16):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8
- MOVW eam, #imm16

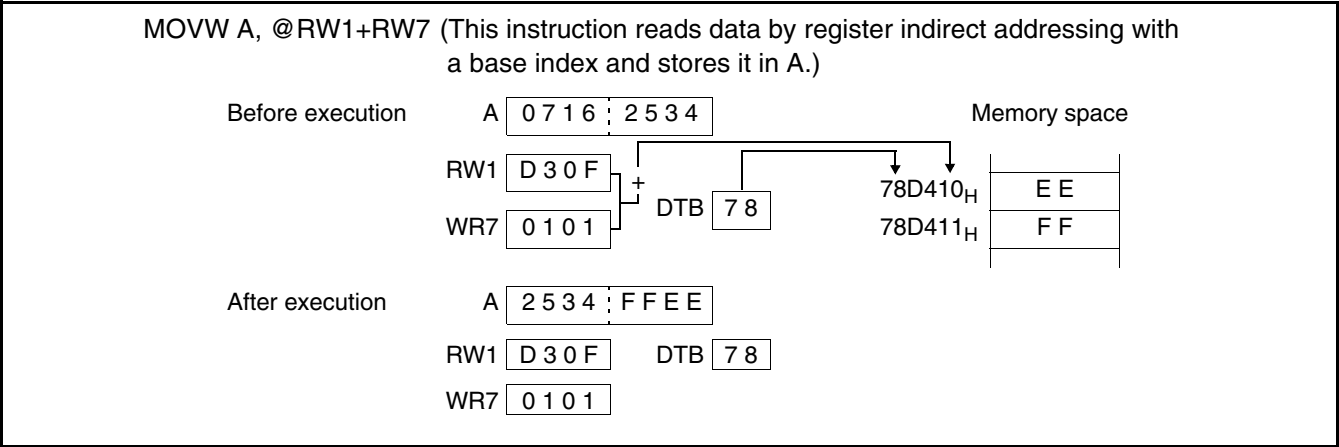
Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16)



● Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)

Memory is accessed using the address determined by adding RW0 or RW1 to the contents of general-purpose register RW7. Address bits 16 to 23 are indicated by the data bank register (DTB).

Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)

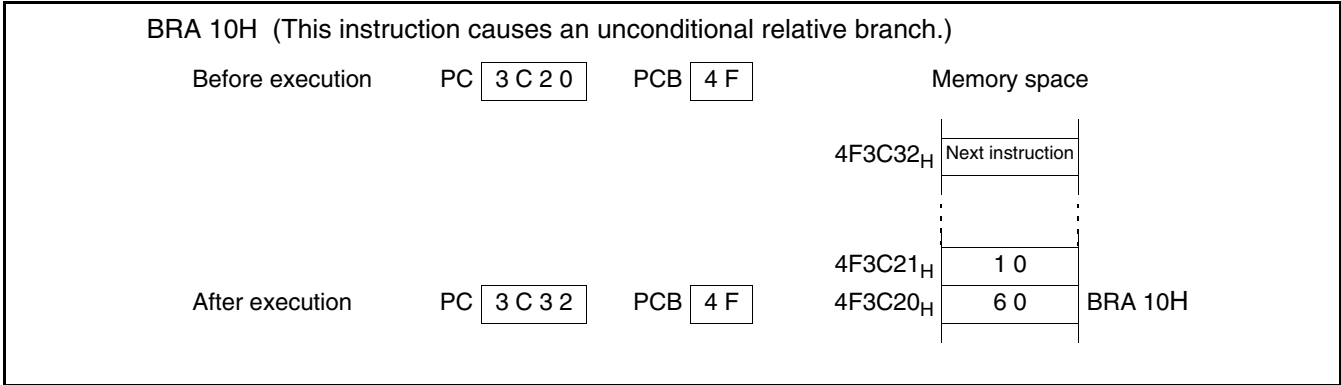


APPENDIX B Instructions

● 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

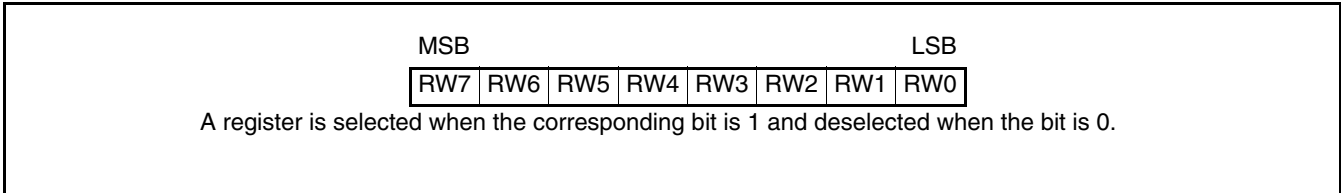
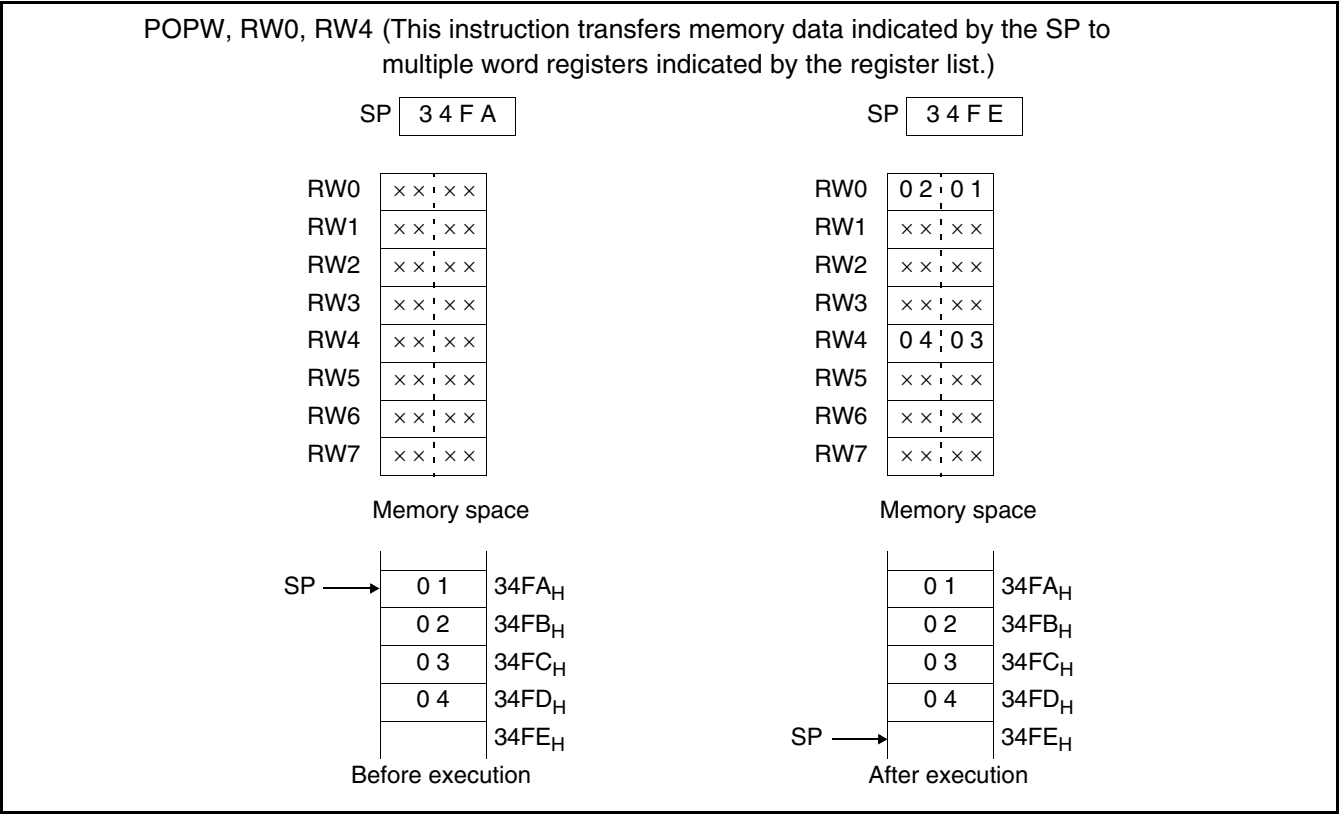


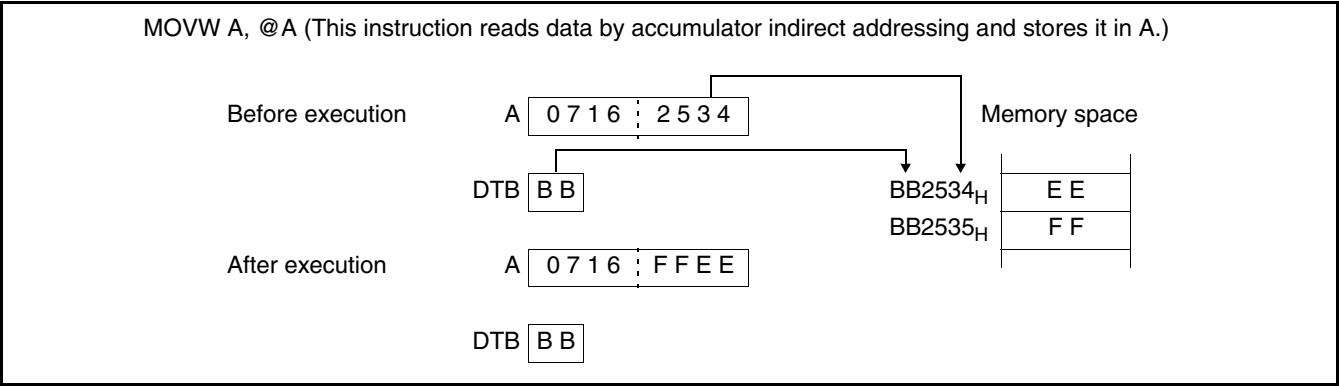
Figure B.4-9 Example of Register List (rlist)



● 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)

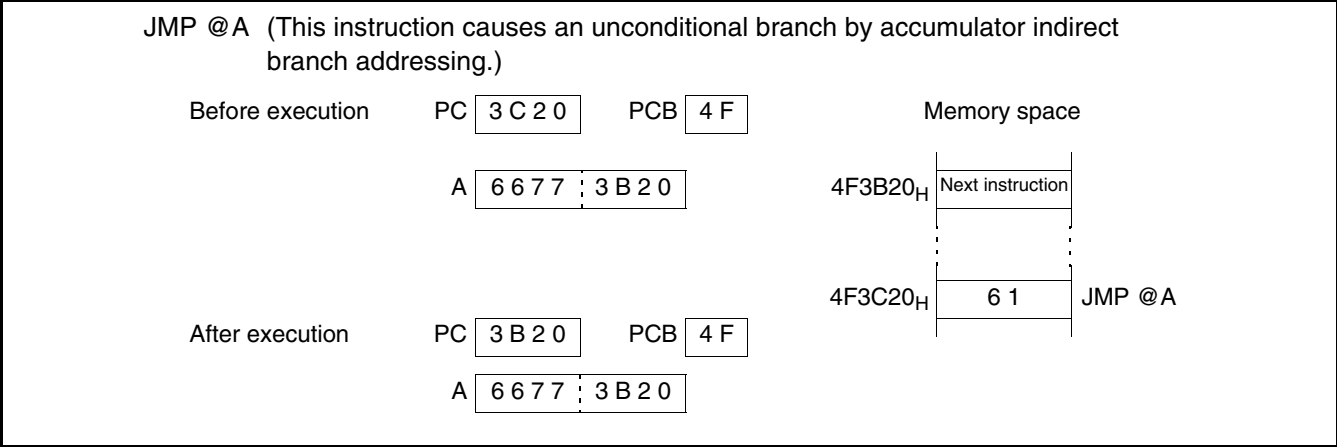


APPENDIX B Instructions

● Accumulator indirect branch addressing (@A)

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program counter bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

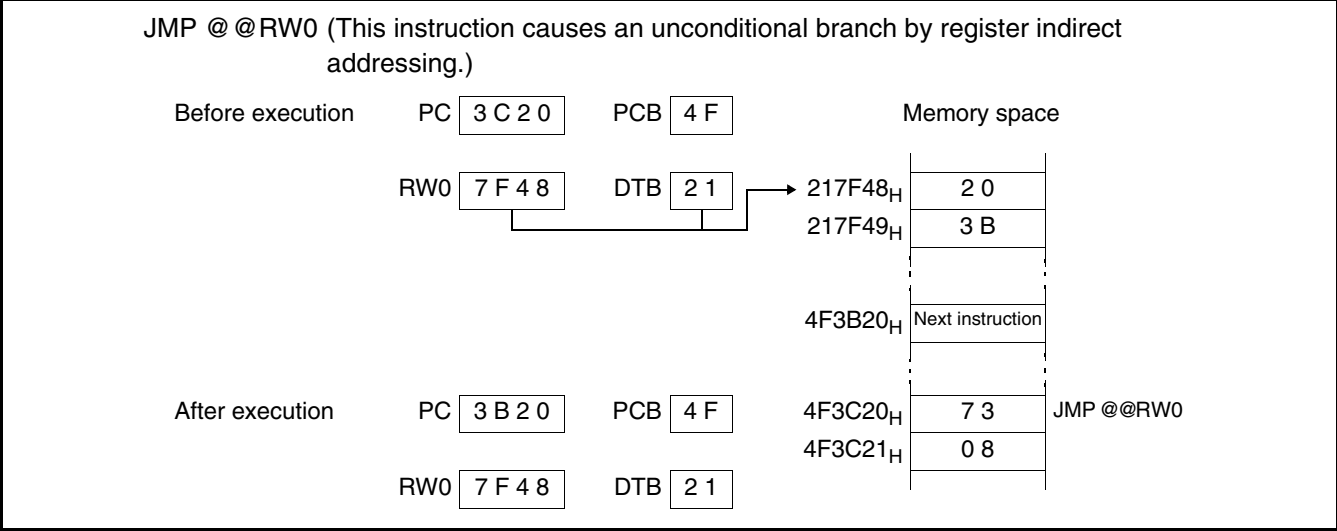
Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)



● Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

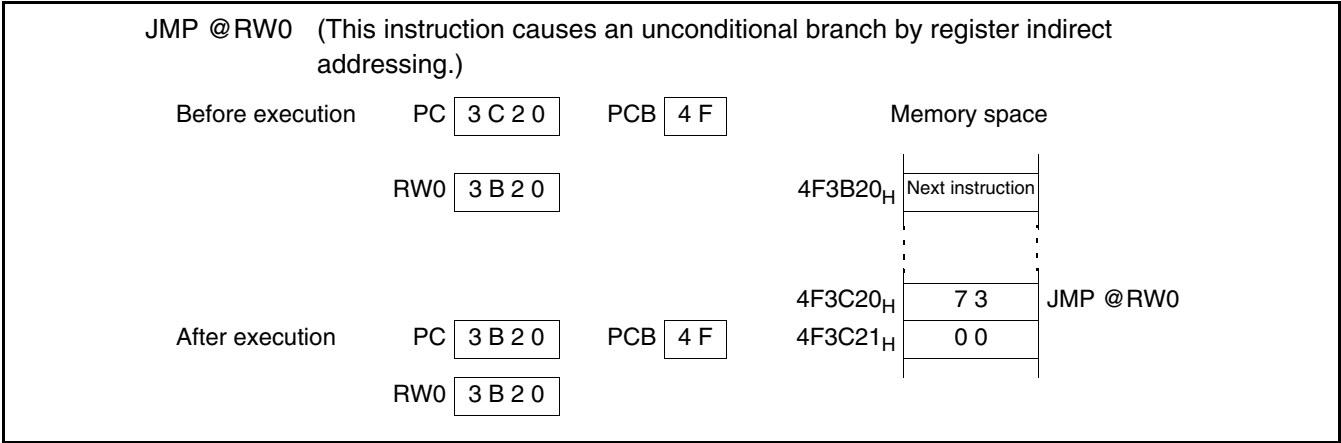
Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)



● Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



B.5 Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.

■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

■ Calculating the Execution Cycle Count

Table B.5-1 lists execution cycle counts and Table B.5-2 and Table B.5-3 summarize correction value data.

Table B.5-1 Execution Cycle Counts in Each Addressing Mode

Code	Operand	(a) *	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00 07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08 0B	@RWj	2	1
0C 0F	@RWj+	4	2
10 17	@RWi+disp8	2	1
18 1B	@RWi+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

*: (a) is used for ~ (cycle count) and B (correction value) in "B.8 F²MC-16LX Instruction List".

Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles

Operand	(b) byte *		(c) word *		(d) long *	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus 16-bit even address	+1	1	+1	1	+2	2
External data bus 16-bit odd address	+1	1	+4	2	+8	4
External data bus 8-bits	+1	1	+4	2	+8	4

*: (b), (c), and (d) are used for ~ (cycle count) and B (correction value) in "B.8 F²MC-16LX Instruction List".

Note:

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16-bits	-	+3
External data bus 8-bits	+3	-

Notes:

- When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.
- Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.

B.6 Effective address field

Table B.6-1 shows the effective address field.

■ Effective Address Field

Table B.6-1 Effective Address Field

Code	Representation			Address format	Byte count of extended address part *
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	0
09	@RW1				
0A	@RW2				
0B	@RW3				
0C	@RW0+			Register indirect with post increment	0
0D	@RW1+				
0E	@RW2+				
0F	@RW3+				
10	@RW0+disp8			Register indirect with 8-bit displacement	1
11	@RW1+disp8				
12	@RW2+disp8				
13	@RW3+disp8				
14	@RW4+disp8				
15	@RW5+disp8				
16	@RW6+disp8				
17	@RW7+disp8				
18	@RW0+disp16			Register indirect with 16-bit displacement	2
19	@RW1+disp16				
1A	@RW2+disp16				
1B	@RW3+disp16				
1C	@RW0+RW7			Register indirect with index	0
1D	@RW1+RW7			Register indirect with index	0
1E	@PC+disp16			PC indirect with 16-bit displacement	2
1F	addr16			Direct address	2

*1: Each byte count of the extended address part applies to + in the # (byte count) column in "B.8 F²MC-16LX Instruction List".

B.7 How to Read the Instruction List

Table B.7-1 describes the items used in "B.8 F²MC-16LX Instruction List", and Table B.7-2 describes the symbols used in the same list.

■ Description of Instruction Presentation Items and Symbols

Table B.7-1 Description of Items in the Instruction List (1/2)

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number of following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table B.2-1 for the alphabetical letters in items.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bit15 to bit08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00 _H or FF _H 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 -: Not Read Modify Write instruction</p> <p>Note: 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	The bit length used varies depending on the 32-bit accumulator instruction. Byte: Low-order 8 bits of byte AL Word: 16 bits of word AL Long word: 32 bits of AL and AH
AH	16 high-order bits of A
AL	16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	program counter bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
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 _H to 0000FF _H)
#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 F²MC-16LX Instruction List

Table B.8-1 to Table B.8-18 list the instructions used by the F²MC-16LX.

■ F²MC-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.

APPENDIX B Instructions

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.

Table B.8-3 42 Addition/Subtraction Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD A,#imm8	2	2	0	0	byte (A) \leftarrow (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	5	0	(b)	byte (A) \leftarrow (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	3	1	0	byte (A) \leftarrow (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	4 + (a)	0	(b)	byte (A) \leftarrow (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	3	2	0	byte (ear) \leftarrow (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD eam,A	2+	5 + (a)	0	2 \times (b)	byte (eam) \leftarrow (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	0	byte (A) \leftarrow (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	3	1	0	byte (A) \leftarrow (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	4 + (a)	0	(b)	byte (A) \leftarrow (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A	1	3	0	0	byte (A) \leftarrow (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	0	byte (A) \leftarrow (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	5	0	(b)	byte (A) \leftarrow (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	3	1	0	byte (A) \leftarrow (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	4 + (a)	0	(b)	byte (A) \leftarrow (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	3	2	0	byte (ear) \leftarrow (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB eam,A	2+	5 + (a)	0	2 \times (b)	byte (eam) \leftarrow (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	0	byte (A) \leftarrow (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	3	1	0	byte (A) \leftarrow (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	4 + (a)	0	(b)	byte (A) \leftarrow (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A	1	3	0	0	byte (A) \leftarrow (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	0	word (A) \leftarrow (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	3	1	0	word (A) \leftarrow (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) \leftarrow (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	0	word (A) \leftarrow (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	3	2	0	word (ear) \leftarrow (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW eam,A	2+	5+(a)	0	2 \times (c)	word (eam) \leftarrow (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	3	1	0	word (A) \leftarrow (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) \leftarrow (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	0	word (A) \leftarrow (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	3	1	0	word (A) \leftarrow (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) \leftarrow (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	0	word (A) \leftarrow (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	3	2	0	word (ear) \leftarrow (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW eam,A	2+	5+(a)	0	2 \times (c)	word (eam) \leftarrow (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	3	1	0	word (A) \leftarrow (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) \leftarrow (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL A,ear	2	6	2	0	long (A) \leftarrow (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) \leftarrow (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	0	long (A) \leftarrow (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	6	2	0	long (A) \leftarrow (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) \leftarrow (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	0	long (A) \leftarrow (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

APPENDIX B Instructions

Table B.8-4 12 Increment/decrement Instructions (Byte, Word, Long Word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2 \times (b)	byte (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2 \times (b)	byte (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2 \times (c)	word (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2 \times (c)	word (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2 \times (d)	long (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2 \times (d)	long (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-5 11 Compare Instructions (Byte, Word, Long Word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-6 11 Unsigned Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU A	1	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW A	1	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0 7: Overflow 15: Normal

*2: 4: Division by 0 8: Overflow 16: Normal

*3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal

*4: 4: Division by 0 7: Overflow 22: Normal

*5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal

*6: (b): Division by 0 or overflow 2 × (b): Normal

*7: (c): Division by 0 or overflow 2 × (c): Normal

*8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.

*9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.

*10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.

*11: 3: Word (AH) is 0. 11: Word (AH) is not 0.

*12: 4: Word (ear) is 0. 12: Word (ear) is not 0.

*13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

APPENDIX B Instructions

Table B.8-7 11 Signed Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV A	2	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL A	2	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW A	2	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal

*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal

*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal

*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal
When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal

*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal
When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal

*6: (b): Division by 0 or overflow, 2 × (b): Normal

*7: (c): Division by 0 or overflow, 2 × (c): Normal

*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative

*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative

*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative

*11: 3: Word (AH) is 0, 16: result is positive, 19: result is negative

*12: 4: Word (ear) is 0, 17: result is positive, 20: result is negative

*13: 5+(a): Word (eam) is 0, 18+(a): result is positive, 21+(a): result is negative

Notes:

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-8 39 Logic 1 Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A,#imm8	2	2	0	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	3	1	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	2	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	3	1	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	2	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	3	1	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	2	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	3	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOT eam	2+	5+(a)	0	2 × (b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	3	1	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	2	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	3	1	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	2	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	3	1	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	2	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	3	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW eam	2+	5+(a)	0	2 × (c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

APPENDIX B Instructions

Table B.8-9 6 Logic 2 Instructions (Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	6	2	0	long (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	long (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	long (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (d) in the table.

Table B.8-10 6 Sign Inversion Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	0	byte (A) ← 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	byte (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 × (b)	byte (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	word (A) ← 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 × (c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-11 1 Normalization Instruction (Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A,R0	2	*1	1	0	long (A) ← Shift left to the position where '1' is set for the first time. byte (R0) ← Shift count at that time	-	-	-	-	-	-	*	-	-	-

*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

Table B.8-12 18 Shift Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	3	2	0	byte (ear) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	*
ASR A,R0	2	*1	1	0	byte (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSR A,R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSL A,R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW A/SHRW A	1	2	0	0	word (A) ← Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW A/SHLW A	1	2	0	0	word (A) ← Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW A,R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW A,R0	2	*1	1	0	word (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL A,R0	2	*2	1	0	long (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL A,R0	2	*2	1	0	long (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL A,R0	2	*2	1	0	long (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

*1: 6 when R0 is 0; otherwise, 5 + (R0)

*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

APPENDIX B Instructions

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: $3 \times (c) + (b)$

*3: Read (word) of branch destination address

*4: W: Save to stack (word) R: Read (word) of branch destination address

*5: Save to stack (word)

*6: W: Save to stack (long word), R: Read (long word) of branch destination address

*7: Save to stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-14 19 Branch 2 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE eam,#imm16,rel *9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ ear,rel	3	*5	2	0	byte (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ eam,rel	3+	*6	2	2 × (b)	byte (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*5	2	0	word (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ eam,rel	3+	*6	2	2 × (c)	word (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	16	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	17	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI	1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK #imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK	1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET *10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP *11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

*1: 5 when a branch is made; otherwise, 4

*2: 13 when a branch is made; otherwise, 12

*3: 7+(a) when a branch is made; otherwise, 6+(a)

*4: 8 when a branch is made; otherwise, 7

*5: 7 when a branch is made; otherwise, 6

*6: 8+(a) when a branch is made; otherwise, 7+(a)

*7: 3 × (b) + 2 × (c) when jumping to the next interruption request; 6 × (c) when returning from the current interruption

*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption

*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.

*10: Return from stack (word)

*11: Return from stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

APPENDIX B Instructions

Table B.8-15 28 Other Control Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*5	*4	(SP) \leftarrow (SP) - 2n, ((SP)) \leftarrow (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	0	(c)	word (A) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	0	(c)	word (AH) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	4	0	(c)	word (PS) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*5	*4	(rlst) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2n	-	-	-	-	-	-	-	-	-	-
JCTX @A	1	14	0	6 \times (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	0	byte (CCR) \leftarrow (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	0	byte (CCR) \leftarrow (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	0	byte (RP) \leftarrow imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	0	byte (ILM) \leftarrow imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	1	0	word (RWi) \leftarrow ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	1	0	word (RWi) \leftarrow eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	1	0	0	word (A) \leftarrow ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	0	word (A) \leftarrow eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	0	word (SP) \leftarrow (SP) + ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	0	word (SP) \leftarrow (SP) + imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	0	byte (A) \leftarrow (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	0	byte (brg2) \leftarrow (A)	-	-	-	-	-	*	*	-	-	-
NOP	1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB	1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB	1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB	1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB	1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC	1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR	1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2

*2: $7 + 3 \times (\text{POP count}) + 2 \times (\text{POP last register number})$, 7 when RLST = 0 (no transfer register)

*3: $29 + 3 \times (\text{PUSH count}) - 3 \times (\text{PUSH last register number})$, 8 when RLST = 0 (no transfer register)

*4: $(\text{POP count}) \times (c)$ or $(\text{PUSH count}) \times (c)$

*5: (POP count) or (PUSH count)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (c) in the table.

Table B.8-16 21 Bit Operand Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A,dir:bp	3	5	0	(b)	byte (A) \leftarrow (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,addr16:bp	4	5	0	(b)	byte (A) \leftarrow (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,io:bp	3	4	0	(b)	byte (A) \leftarrow (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB dir:bp,A	3	7	0	2 \times (b)	bit (dir:bp)b \leftarrow (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	7	0	2 \times (b)	bit (addr16:bp)b \leftarrow (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	6	0	2 \times (b)	bit (io:bp)b \leftarrow (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	7	0	2 \times (b)	bit (dir:bp)b \leftarrow 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	7	0	2 \times (b)	bit (addr16:bp)b \leftarrow 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	7	0	2 \times (b)	bit (io:bp)b \leftarrow 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	7	0	2 \times (b)	bit (dir:bp)b \leftarrow 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	7	0	2 \times (b)	bit (addr16:bp)b \leftarrow 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	7	0	2 \times (b)	bit (io:bp)b \leftarrow 0	-	-	-	-	-	-	-	-	-	*
BBC dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*3	0	2 \times (b)	Branch on (addr16:bp) b = 1, bit (addr16:bp) b \leftarrow 1	-	-	-	-	-	-	*	-	-	*
WBTS io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1: 8 when a branch is made; otherwise, 7

*2: 7 when a branch is made; otherwise, 6

*3: 10 when the condition is met; otherwise, 9

*4: Undefined count

*5: Until the condition is met

Note:

See Table B.5-1 and Table B.5-2 for information on (b) in the table.

Table B.8-17 6 Accumulator Operation Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 \leftrightarrow (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW	1	2	0	0	word (AH) \leftrightarrow (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extension	-	Z	-	-	-	R	*	-	-	-

APPENDIX B Instructions

Table B.8-18 10 String Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*5	*3	byte transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*5	*3	byte transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*8	*4	byte search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*8	*4	byte search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*8	*3	byte fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*5	*6	word transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*5	*6	word transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*8	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*8	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*8	*6	word fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, $4 + 7 \times (RW0)$ when the counter expires, or $7n + 5$ when a match occurs

*2: 5 when RW0 is 0; otherwise, $4 + 8 \times (RW0)$

*3: $(b) \times (RW0) + (b) \times (RW0)$ When the source and destination access different areas, calculate the (b) item individually.

*4: $(b) \times n$

*5: $2 \times (b) \times (RW0)$

*6: $(c) \times (RW0) + (c) \times (RW0)$ When the source and destination access different areas, calculate the (c) item individually.

*7: $(c) \times n$

*8: $(b) \times (RW0)$

Note:

m: RW0 value (counter value), n: Loop count

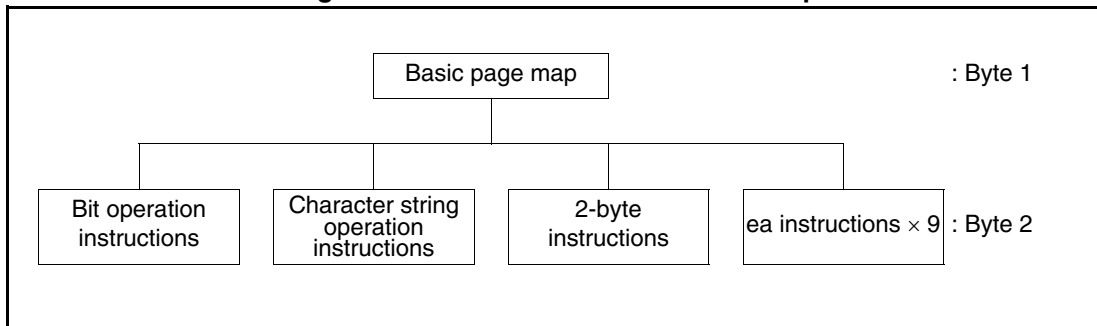
See Table B.5-1 and Table B.5-2 for information on (b) and (c) in the table.

B.9 Instruction Map

Each F²MC-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 F²MC-16LX instruction map.

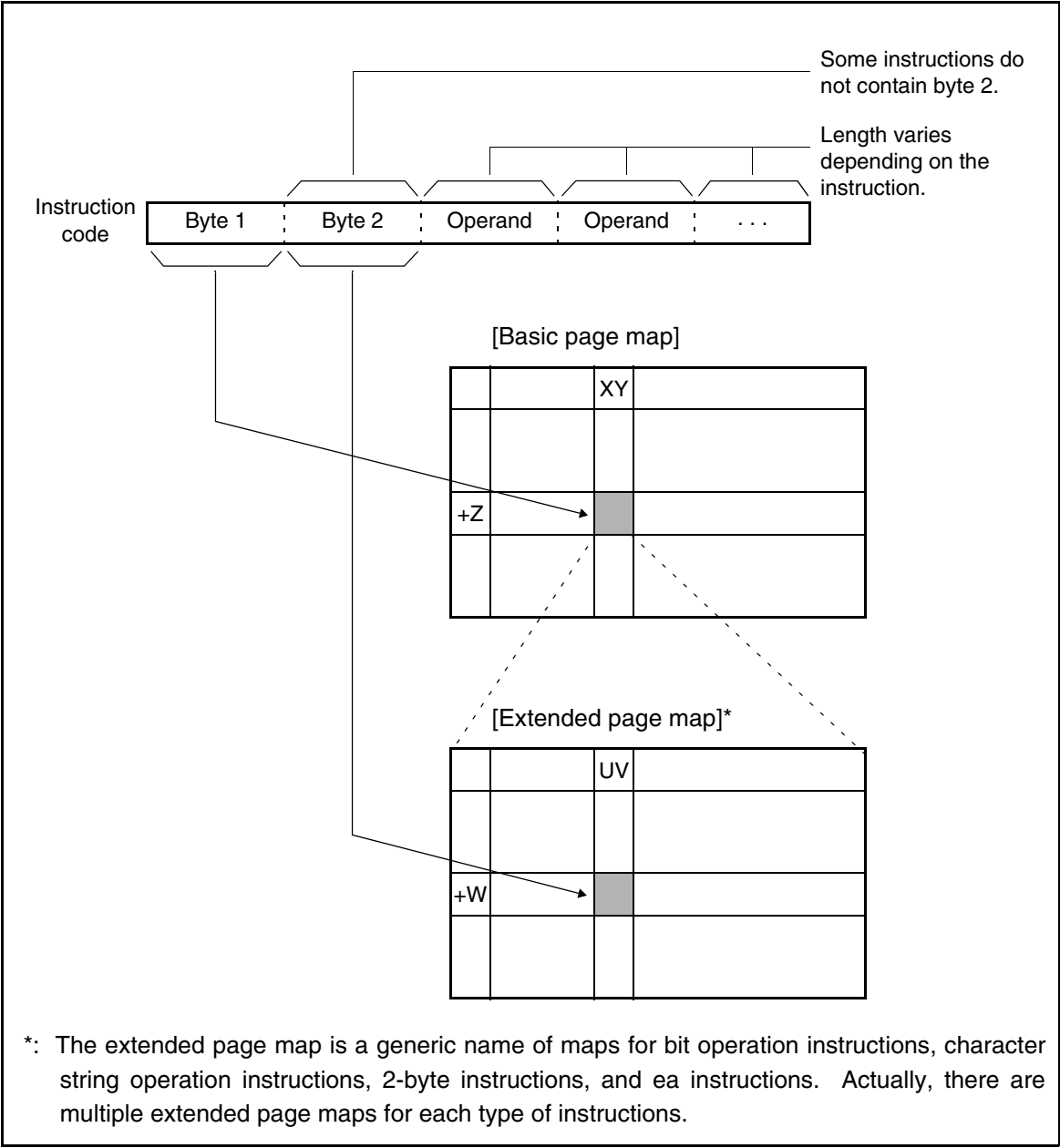
■ Structure of Instruction Map

Figure B.9-1 Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 shows the correspondence between an actual instruction code and instruction map.

Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map



An example of an instruction code is shown in Table B.9-1.

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-
MOV A, ADB	60 +F=6F	00 +0=00
@RW2+d8, #8, rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	NOP	CMR	ADD A, dir	ADD A, #8	MOV A, dir	MOV A, io	BRA rel instruction 1	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	MOV A, @RWi+d8	MOVW @RWi+d8, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT	MOVEA RWi, ea								BNT rel
+A	MOV RP, #8	MOV ILM, #8	CBNE A, #8, rel	CWBNE A, #16, rel	MOVW A, #16	MOVW A, addr16	INTP 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 A	ANDW A	ANDW A, #16	PUSHW A	POPW A	Bit operation instruction	MOV ea, Ri								BLE rel
+D		ZEXTW	ORW A	ORW A, #16	PUSHW AH	POPW AH		MOVW ea, RWi								BGT rel
+E	ASRW A	SWAPW A	XORW A	XORW A, #16	PUSHW PS	POPW PS	Character string operation instruction	XCH Ri, ea								BLS rel
+F	LSRW A	ADDSP #16	MULW A	NOTW A	PUSHW r1st	POPW r1st	2-byte instruction	XCHW RWi, ea								BHI rel

Table B.9-3 Bit Operation Instruction Map (First Byte = 6C_H)

	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		WBTS, io:bp		WBTC, io:bp	
+1																
+2																
+3																
+4																
+5																
+6																
+7																
+8	MOV B, A, dir:bp	MOV B, A, dir:bp	MOV B, dir:bp, A	MOV B, dir:bp, A	CLRB, dir:bp	CLRB, dir:bp	SETB, dir:bp	SETB, dir:bp	BBC, dir:bp, rel	BBC, dir:bp, rel	BBS, dir:bp, rel	BBS, dir:bp, rel				SBBS, dir:bp
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table B.9-4 Character String Operation Instruction Map (First Byte = 6EH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI PCB, PCB	MOVSD PCB, PCB	MOVSWI PCB, PCB	MOVSWD PCB, PCB					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB			
+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_H)

	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_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBNE ↓, CWBNE ↓										CBNE ↓	CBNE ↓
+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	SUBL A, @RW0	SUBL A, @RW0	SUBL A, @RW0	CMPL A, @RW0	ANDL A, @RW0	ANDL A, @RW0	ANDL A, @RW0	ANDL A, @RW0	ANDL A, @RW0	ORL A, @RW0	ORL A, @RW0	XORL A, @RW0	XORL A, @RW0	@RW0, #8, rel	@RW0, #8, rel
+9	ADDL A, @RW1	SUBL A, @RW1	SUBL A, @RW1	SUBL A, @RW1	CMPL A, @RW1	ANDL A, @RW1	ANDL A, @RW1	ANDL A, @RW1	ANDL A, @RW1	ANDL A, @RW1	ORL A, @RW1	ORL A, @RW1	XORL A, @RW1	XORL A, @RW1	@RW1, #8, rel	@RW1, #8, rel
+A	ADDL A, @RW2	SUBL A, @RW2	SUBL A, @RW2	SUBL A, @RW2	CMPL A, @RW2	ANDL A, @RW2	ANDL A, @RW2	ANDL A, @RW2	ANDL A, @RW2	ANDL A, @RW2	ORL A, @RW2	ORL A, @RW2	XORL A, @RW2	XORL A, @RW2	@RW2, #8, rel	@RW2, #8, rel
+B	ADDL A, @RW3	SUBL A, @RW3	SUBL A, @RW3	SUBL A, @RW3	CMPL A, @RW3	ANDL A, @RW3	ANDL A, @RW3	ANDL A, @RW3	ANDL A, @RW3	ANDL A, @RW3	ORL A, @RW3	ORL A, @RW3	XORL A, @RW3	XORL A, @RW3	@RW3, #8, rel	@RW3, #8, rel
+C	ADDL A, @RW0+RW7	SUBL A, @RW0+RW7	SUBL A, @RW0+RW7	SUBL A, @RW0+RW7	CMPL A, @RW0+RW7	ANDL A, @RW0+RW7	ANDL A, @RW0+RW7	ANDL A, @RW0+RW7	ANDL A, @RW0+RW7	ANDL A, @RW0+RW7	ORL A, @RW0+RW7	ORL A, @RW0+RW7	XORL A, @RW0+RW7	XORL A, @RW0+RW7	Use prohibited	Use prohibited
+D	ADDL A, @RW1+RW7	SUBL A, @RW1+RW7	SUBL A, @RW1+RW7	SUBL A, @RW1+RW7	CMPL A, @RW1+RW7	ANDL A, @RW1+RW7	ANDL A, @RW1+RW7	ANDL A, @RW1+RW7	ANDL A, @RW1+RW7	ANDL A, @RW1+RW7	ORL A, @RW1+RW7	ORL A, @RW1+RW7	XORL A, @RW1+RW7	XORL A, @RW1+RW7	Use prohibited	Use prohibited
+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	Use prohibited
+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	Use prohibited

Table B.9-7 ea Instruction 2 (First Byte = 71_H)

	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_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ROLc	ROLc	RORc	RORc	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	A, R0' @RW0+d8	A, R0' @RW0+d8	R0, A' @RW0+d8, A	MOV	MOVX	MOVX	A, R0' @RW0+d8	A, R0' @RW0+d8
+1	ROLc	ROLc	RORc	RORc	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	A, R1' @RW1+d8	A, R1' @RW1+d8	R1, A' @RW1+d8, A	MOV	MOVX	MOVX	A, R1' @RW1+d8	A, R1' @RW1+d8
+2	ROLc	ROLc	RORc	RORc	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	A, R2' @RW2+d8	A, R2' @RW2+d8	R2, A' @RW2+d8, A	MOV	MOVX	MOVX	A, R2' @RW2+d8	A, R2' @RW2+d8
+3	ROLc	ROLc	RORc	RORc	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	A, R3' @RW3+d8	A, R3' @RW3+d8	R3, A' @RW3+d8, A	MOV	MOVX	MOVX	A, R3' @RW3+d8	A, R3' @RW3+d8
+4	ROLc	ROLc	RORc	RORc	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	A, R4' @RW4+d8	A, R4' @RW4+d8	R4, A' @RW4+d8, A	MOV	MOVX	MOVX	A, R4' @RW4+d8	A, R4' @RW4+d8
+5	ROLc	ROLc	RORc	RORc	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	A, R5' @RW5+d8	A, R5' @RW5+d8	R5, A' @RW5+d8, A	MOV	MOVX	MOVX	A, R5' @RW5+d8	A, R5' @RW5+d8
+6	ROLc	ROLc	RORc	RORc	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	A, R6' @RW6+d8	A, R6' @RW6+d8	R6, A' @RW6+d8, A	MOV	MOVX	MOVX	A, R6' @RW6+d8	A, R6' @RW6+d8
+7	ROLc	ROLc	RORc	RORc	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	A, R7' @RW7+d8	A, R7' @RW7+d8	R7, A' @RW7+d8, A	MOV	MOVX	MOVX	A, R7' @RW7+d8	A, R7' @RW7+d8
+8	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW0+d16	R0' @RW0+d16	R0' @RW0+d16	R0' @RW0+d16	R0' @RW0+d16	R0' @RW0+d16	R0' @RW0+d16	A, R0' @RW0+d16	A, R0' @RW0+d16	R0' @RW0, A' @RW0+d16, A	MOV	MOVX	MOVX	A, R0' @RW0+d16	A, R0' @RW0+d16
+9	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW1' @RW1+d16	R0' @RW1' @RW1+d16	R0' @RW1' @RW1+d16	R0' @RW1' @RW1+d16	R0' @RW1' @RW1+d16	R0' @RW1' @RW1+d16	R0' @RW1' @RW1+d16	A, R0' @RW1' @RW1+d16	A, R0' @RW1' @RW1+d16	R0' @RW1, A' @RW1+d16, A	MOV	MOVX	MOVX	A, R0' @RW1' @RW1+d16	A, R0' @RW1' @RW1+d16
+A	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW2' @RW2+d16	R0' @RW2' @RW2+d16	R0' @RW2' @RW2+d16	R0' @RW2' @RW2+d16	R0' @RW2' @RW2+d16	R0' @RW2' @RW2+d16	R0' @RW2' @RW2+d16	A, R0' @RW2' @RW2+d16	A, R0' @RW2' @RW2+d16	R0' @RW2, A' @RW2+d16, A	MOV	MOVX	MOVX	A, R0' @RW2' @RW2+d16	A, R0' @RW2' @RW2+d16
+B	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW3' @RW3+d16	R0' @RW3' @RW3+d16	R0' @RW3' @RW3+d16	R0' @RW3' @RW3+d16	R0' @RW3' @RW3+d16	R0' @RW3' @RW3+d16	R0' @RW3' @RW3+d16	A, R0' @RW3' @RW3+d16	A, R0' @RW3' @RW3+d16	R0' @RW3, A' @RW3+d16, A	MOV	MOVX	MOVX	A, R0' @RW3' @RW3+d16	A, R0' @RW3' @RW3+d16
+C	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW0+ @RW0+RW7	R0' @RW0+ @RW0+RW7	R0' @RW0+ @RW0+RW7	R0' @RW0+ @RW0	R0' @RW0+ @RW0	R0' @RW0+ @RW0	R0' @RW0+ @RW0	A, R0' @RW0+ @RW0	A, R0' @RW0+ @RW0	R0' @RW0, A' @RW0+RW7, A	MOV	MOVX	MOVX	A, R0' @RW0+ @RW0+RW7	A, R0' @RW0+ @RW0+RW7
+D	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW1+ @RW1+RW7	R0' @RW1+ @RW1+RW7	R0' @RW1+ @RW1+RW7	R0' @RW1+ @RW1+RW7	R0' @RW1+ @RW1+RW7	R0' @RW1+ @RW1+RW7	R0' @RW1+ @RW1+RW7	A, R0' @RW1+ @RW1+RW7	A, R0' @RW1+ @RW1+RW7	R0' @RW1, A' @RW1+RW7, A	MOV	MOVX	MOVX	A, R0' @RW1+ @RW1+RW7	A, R0' @RW1+ @RW1+RW7
+E	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW2+ @PC+d16	R0' @RW2+ @PC+d16	R0' @RW2+ @PC+d16	R0' @RW2+ @PC+d16	R0' @RW2+ @PC+d16	R0' @RW2+ @PC+d16	R0' @RW2+ @PC+d16	A, R0' @RW2+ @PC+d16	A, R0' @RW2+ @PC+d16	R0' @RW2, A' @PC+d16, A	MOV	MOVX	MOVX	A, R0' @RW2+ @PC+d16	A, R0' @RW2+ @PC+d16
+F	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
		R0' @RW3+ addr16	R0' @RW3+ addr16	R0' @RW3+ addr16	R0' @RW3+ addr16	R0' @RW3+ addr16	R0' @RW3+ addr16	R0' @RW3+ addr16	A, R0' @RW3+ addr16	A, R0' @RW3+ addr16	R0' @RW3, A' addr16, A	MOV	MOVX	MOVX	A, R0' @RW3+ addr16	A, R0' @RW3+ addr16

Table B.9-9 ea Instruction 4 (First Byte = 73_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMP @RW0, @RW0+d8	JMP @RW0, @RW0+d8	CALL RW0, @RW0+d8	CALL RW0, @RW0+d8	INCW RW0, @RW0+d8	INCW RW0, @RW0+d8	DECW RW0, @RW0+d8	DECW RW0, @RW0+d8	MOVW A, RW0, @RW0+d8	MOVW A, RW0, @RW0+d8	MOVW RW0, #16, @RW0+d8	MOVW RW0, #16, @RW0+d8	MOVW RW0, #16, @RW0+d8	MOVW RW0, #16, @RW0+d8	XCHW A, RW0, @RW0+d8	XCHW A, RW0, @RW0+d8
+1	JMP @RW1, @RW1+d8	JMP @RW1, @RW1+d8	CALL RW1, @RW1+d8	CALL RW1, @RW1+d8	INCW RW1, @RW1+d8	INCW RW1, @RW1+d8	DECW RW1, @RW1+d8	DECW RW1, @RW1+d8	MOVW A, RW1, @RW1+d8	MOVW A, RW1, @RW1+d8	MOVW RW1, #16, @RW1+d8	MOVW RW1, #16, @RW1+d8	MOVW RW1, #16, @RW1+d8	MOVW RW1, #16, @RW1+d8	XCHW A, RW1, @RW1+d8	XCHW A, RW1, @RW1+d8
+2	JMP @RW2, @RW2+d8	JMP @RW2, @RW2+d8	CALL RW2, @RW2+d8	CALL RW2, @RW2+d8	INCW RW2, @RW2+d8	INCW RW2, @RW2+d8	DECW RW2, @RW2+d8	DECW RW2, @RW2+d8	MOVW A, RW2, @RW2+d8	MOVW A, RW2, @RW2+d8	MOVW RW2, #16, @RW2+d8	MOVW RW2, #16, @RW2+d8	MOVW RW2, #16, @RW2+d8	MOVW RW2, #16, @RW2+d8	XCHW A, RW2, @RW2+d8	XCHW A, RW2, @RW2+d8
+3	JMP @RW3, @RW3+d8	JMP @RW3, @RW3+d8	CALL RW3, @RW3+d8	CALL RW3, @RW3+d8	INCW RW3, @RW3+d8	INCW RW3, @RW3+d8	DECW RW3, @RW3+d8	DECW RW3, @RW3+d8	MOVW A, RW3, @RW3+d8	MOVW A, RW3, @RW3+d8	MOVW RW3, #16, @RW3+d8	MOVW RW3, #16, @RW3+d8	MOVW RW3, #16, @RW3+d8	MOVW RW3, #16, @RW3+d8	XCHW A, RW3, @RW3+d8	XCHW A, RW3, @RW3+d8
+4	JMP @RW4, @RW4+d8	JMP @RW4, @RW4+d8	CALL RW4, @RW4+d8	CALL RW4, @RW4+d8	INCW RW4, @RW4+d8	INCW RW4, @RW4+d8	DECW RW4, @RW4+d8	DECW RW4, @RW4+d8	MOVW A, RW4, @RW4+d8	MOVW A, RW4, @RW4+d8	MOVW RW4, #16, @RW4+d8	MOVW RW4, #16, @RW4+d8	MOVW RW4, #16, @RW4+d8	MOVW RW4, #16, @RW4+d8	XCHW A, RW4, @RW4+d8	XCHW A, RW4, @RW4+d8
+5	JMP @RW5, @RW5+d8	JMP @RW5, @RW5+d8	CALL RW5, @RW5+d8	CALL RW5, @RW5+d8	INCW RW5, @RW5+d8	INCW RW5, @RW5+d8	DECW RW5, @RW5+d8	DECW RW5, @RW5+d8	MOVW A, RW5, @RW5+d8	MOVW A, RW5, @RW5+d8	MOVW RW5, #16, @RW5+d8	MOVW RW5, #16, @RW5+d8	MOVW RW5, #16, @RW5+d8	MOVW RW5, #16, @RW5+d8	XCHW A, RW5, @RW5+d8	XCHW A, RW5, @RW5+d8
+6	JMP @RW6, @RW6+d8	JMP @RW6, @RW6+d8	CALL RW6, @RW6+d8	CALL RW6, @RW6+d8	INCW RW6, @RW6+d8	INCW RW6, @RW6+d8	DECW RW6, @RW6+d8	DECW RW6, @RW6+d8	MOVW A, RW6, @RW6+d8	MOVW A, RW6, @RW6+d8	MOVW RW6, #16, @RW6+d8	MOVW RW6, #16, @RW6+d8	MOVW RW6, #16, @RW6+d8	MOVW RW6, #16, @RW6+d8	XCHW A, RW6, @RW6+d8	XCHW A, RW6, @RW6+d8
+7	JMP @RW7, @RW7+d8	JMP @RW7, @RW7+d8	CALL RW7, @RW7+d8	CALL RW7, @RW7+d8	INCW RW7, @RW7+d8	INCW RW7, @RW7+d8	DECW RW7, @RW7+d8	DECW RW7, @RW7+d8	MOVW A, RW7, @RW7+d8	MOVW A, RW7, @RW7+d8	MOVW RW7, #16, @RW7+d8	MOVW RW7, #16, @RW7+d8	MOVW RW7, #16, @RW7+d8	MOVW RW7, #16, @RW7+d8	XCHW A, RW7, @RW7+d8	XCHW A, RW7, @RW7+d8
+8	JMP @RW0, @RW0+d16	JMP @RW0, @RW0+d16	CALL @RW0, @RW0+d16	CALL @RW0, @RW0+d16	INCW @RW0, @RW0+d16	INCW @RW0, @RW0+d16	DECW @RW0, @RW0+d16	DECW @RW0, @RW0+d16	MOVW A, RW0, @RW0+d16	MOVW A, RW0, @RW0+d16	MOVW @RW0, A, @RW0+d16	MOVW @RW0, A, @RW0+d16	MOVW @RW0, A, @RW0+d16	MOVW @RW0, A, @RW0+d16	XCHW A, RW0, @RW0+d16	XCHW A, RW0, @RW0+d16
+9	JMP @RW1, @RW1+d16	JMP @RW1, @RW1+d16	CALL @RW1, @RW1+d16	CALL @RW1, @RW1+d16	INCW @RW1, @RW1+d16	INCW @RW1, @RW1+d16	DECW @RW1, @RW1+d16	DECW @RW1, @RW1+d16	MOVW A, RW1, @RW1+d16	MOVW A, RW1, @RW1+d16	MOVW @RW1, A, @RW1+d16	MOVW @RW1, A, @RW1+d16	MOVW @RW1, A, @RW1+d16	MOVW @RW1, A, @RW1+d16	XCHW A, RW1, @RW1+d16	XCHW A, RW1, @RW1+d16
+A	JMP @RW2, @RW2+d16	JMP @RW2, @RW2+d16	CALL @RW2, @RW2+d16	CALL @RW2, @RW2+d16	INCW @RW2, @RW2+d16	INCW @RW2, @RW2+d16	DECW @RW2, @RW2+d16	DECW @RW2, @RW2+d16	MOVW A, RW2, @RW2+d16	MOVW A, RW2, @RW2+d16	MOVW @RW2, A, @RW2+d16	MOVW @RW2, A, @RW2+d16	MOVW @RW2, A, @RW2+d16	MOVW @RW2, A, @RW2+d16	XCHW A, RW2, @RW2+d16	XCHW A, RW2, @RW2+d16
+B	JMP @RW3, @RW3+d16	JMP @RW3, @RW3+d16	CALL @RW3, @RW3+d16	CALL @RW3, @RW3+d16	INCW @RW3, @RW3+d16	INCW @RW3, @RW3+d16	DECW @RW3, @RW3+d16	DECW @RW3, @RW3+d16	MOVW A, RW3, @RW3+d16	MOVW A, RW3, @RW3+d16	MOVW @RW3, A, @RW3+d16	MOVW @RW3, A, @RW3+d16	MOVW @RW3, A, @RW3+d16	MOVW @RW3, A, @RW3+d16	XCHW A, RW3, @RW3+d16	XCHW A, RW3, @RW3+d16
+C	JMP @RW0+, @RW0+RW7	JMP @RW0+, @RW0+RW7	CALL @RW0+, @RW0+RW7	CALL @RW0+, @RW0+RW7	INCW @RW0+, @RW0+RW7	INCW @RW0+, @RW0+RW7	DECW @RW0+, @RW0+RW7	DECW @RW0+, @RW0+RW7	MOVW A, RW0+, @RW0+RW7	MOVW A, RW0+, @RW0+RW7	MOVW @RW0+, A, @RW0+RW7	MOVW @RW0+, A, @RW0+RW7	MOVW @RW0+, A, @RW0+RW7	MOVW @RW0+, A, @RW0+RW7	XCHW A, RW0+, @RW0+RW7	XCHW A, RW0+, @RW0+RW7
+D	JMP @RW1+, @RW1+RW7	JMP @RW1+, @RW1+RW7	CALL @RW1+, @RW1+RW7	CALL @RW1+, @RW1+RW7	INCW @RW1+, @RW1+RW7	INCW @RW1+, @RW1+RW7	DECW @RW1+, @RW1+RW7	DECW @RW1+, @RW1+RW7	MOVW A, RW1+, @RW1+RW7	MOVW A, RW1+, @RW1+RW7	MOVW @RW1+, A, @RW1+RW7	MOVW @RW1+, A, @RW1+RW7	MOVW @RW1+, A, @RW1+RW7	MOVW @RW1+, A, @RW1+RW7	XCHW A, RW1+, @RW1+RW7	XCHW A, RW1+, @RW1+RW7
+E	JMP @RW2+, @PC+d16	JMP @RW2+, @PC+d16	CALL @RW2+, @PC+d16	CALL @RW2+, @PC+d16	INCW @RW2+, @PC+d16	INCW @RW2+, @PC+d16	DECW @RW2+, @PC+d16	DECW @RW2+, @PC+d16	MOVW A, RW2+, @PC+d16	MOVW A, RW2+, @PC+d16	MOVW @RW2+, A, @PC+d16	MOVW @RW2+, A, @PC+d16	MOVW @RW2+, A, @PC+d16	MOVW @RW2+, A, @PC+d16	XCHW A, RW2+, @PC+d16	XCHW A, RW2+, @PC+d16
+F	JMP @RW3+, @addr16	JMP @RW3+, @addr16	CALL @RW3+, @addr16	CALL @RW3+, @addr16	INCW @RW3+, @addr16	INCW @RW3+, @addr16	DECW @RW3+, @addr16	DECW @RW3+, @addr16	MOVW A, RW3+, @addr16	MOVW A, RW3+, @addr16	MOVW @RW3+, A, @addr16	MOVW @RW3+, A, @addr16	MOVW @RW3+, A, @addr16	MOVW @RW3+, A, @addr16	XCHW A, RW3+, @addr16	XCHW A, RW3+, @addr16

Table B.9-10 ea Instruction 5 (First Byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0, @RW0+d8	SUB A, R0, @RW0+d8	SUB A, R0, @RW0+d8	SUB A, R0, @RW0+d8	ADDC A, R0, @RW0+d8	ADDC A, R0, @RW0+d8	CMP A, R0, @RW0+d8	CMP A, R0, @RW0+d8	AND A, R0, @RW0+d8	AND A, R0, @RW0+d8	OR A, R0, @RW0+d8	OR A, R0, @RW0+d8	XOR A, R0, @RW0+d8	XOR A, R0, @RW0+d8	DBNZ @R0, r, 'RW0+d8, r	DBNZ @R0, r, 'RW0+d8, r
+1	ADD A, R1, @RW1+d8	SUB A, R1, @RW1+d8	SUB A, R1, @RW1+d8	SUB A, R1, @RW1+d8	ADDC A, R1, @RW1+d8	ADDC A, R1, @RW1+d8	CMP A, R1, @RW1+d8	CMP A, R1, @RW1+d8	AND A, R1, @RW1+d8	AND A, R1, @RW1+d8	OR A, R1, @RW1+d8	OR A, R1, @RW1+d8	XOR A, R1, @RW1+d8	XOR A, R1, @RW1+d8	DBNZ @R1, r, 'RW1+d8, r	DBNZ @R1, r, 'RW1+d8, r
+2	ADD A, R2, @RW2+d8	SUB A, R2, @RW2+d8	SUB A, R2, @RW2+d8	SUB A, R2, @RW2+d8	ADDC A, R2, @RW2+d8	ADDC A, R2, @RW2+d8	CMP A, R2, @RW2+d8	CMP A, R2, @RW2+d8	AND A, R2, @RW2+d8	AND A, R2, @RW2+d8	OR A, R2, @RW2+d8	OR A, R2, @RW2+d8	XOR A, R2, @RW2+d8	XOR A, R2, @RW2+d8	DBNZ @R2, r, 'RW2+d8, r	DBNZ @R2, r, 'RW2+d8, r
+3	ADD A, R3, @RW3+d8	SUB A, R3, @RW3+d8	SUB A, R3, @RW3+d8	SUB A, R3, @RW3+d8	ADDC A, R3, @RW3+d8	ADDC A, R3, @RW3+d8	CMP A, R3, @RW3+d8	CMP A, R3, @RW3+d8	AND A, R3, @RW3+d8	AND A, R3, @RW3+d8	OR A, R3, @RW3+d8	OR A, R3, @RW3+d8	XOR A, R3, @RW3+d8	XOR A, R3, @RW3+d8	DBNZ @R3, r, 'RW3+d8, r	DBNZ @R3, r, 'RW3+d8, r
+4	ADD A, R4, @RW4+d8	SUB A, R4, @RW4+d8	SUB A, R4, @RW4+d8	SUB A, R4, @RW4+d8	ADDC A, R4, @RW4+d8	ADDC A, R4, @RW4+d8	CMP A, R4, @RW4+d8	CMP A, R4, @RW4+d8	AND A, R4, @RW4+d8	AND A, R4, @RW4+d8	OR A, R4, @RW4+d8	OR A, R4, @RW4+d8	XOR A, R4, @RW4+d8	XOR A, R4, @RW4+d8	DBNZ @R4, r, 'RW4+d8, r	DBNZ @R4, r, 'RW4+d8, r
+5	ADD A, R5, @RW5+d8	SUB A, R5, @RW5+d8	SUB A, R5, @RW5+d8	SUB A, R5, @RW5+d8	ADDC A, R5, @RW5+d8	ADDC A, R5, @RW5+d8	CMP A, R5, @RW5+d8	CMP A, R5, @RW5+d8	AND A, R5, @RW5+d8	AND A, R5, @RW5+d8	OR A, R5, @RW5+d8	OR A, R5, @RW5+d8	XOR A, R5, @RW5+d8	XOR A, R5, @RW5+d8	DBNZ @R5, r, 'RW5+d8, r	DBNZ @R5, r, 'RW5+d8, r
+6	ADD A, R6, @RW6+d8	SUB A, R6, @RW6+d8	SUB A, R6, @RW6+d8	SUB A, R6, @RW6+d8	ADDC A, R6, @RW6+d8	ADDC A, R6, @RW6+d8	CMP A, R6, @RW6+d8	CMP A, R6, @RW6+d8	AND A, R6, @RW6+d8	AND A, R6, @RW6+d8	OR A, R6, @RW6+d8	OR A, R6, @RW6+d8	XOR A, R6, @RW6+d8	XOR A, R6, @RW6+d8	DBNZ @R6, r, 'RW6+d8, r	DBNZ @R6, r, 'RW6+d8, r
+7	ADD A, R7, @RW7+d8	SUB A, R7, @RW7+d8	SUB A, R7, @RW7+d8	SUB A, R7, @RW7+d8	ADDC A, R7, @RW7+d8	ADDC A, R7, @RW7+d8	CMP A, R7, @RW7+d8	CMP A, R7, @RW7+d8	AND A, R7, @RW7+d8	AND A, R7, @RW7+d8	OR A, R7, @RW7+d8	OR A, R7, @RW7+d8	XOR A, R7, @RW7+d8	XOR A, R7, @RW7+d8	DBNZ @R7, r, 'RW7+d8, r	DBNZ @R7, r, 'RW7+d8, r
+8	ADD A, @RW0, @RW0+d16	SUB A, @RW0, @RW0+d16	SUB A, @RW0, @RW0+d16	SUB A, @RW0, @RW0+d16	ADDC A, @RW0, @RW0+d16	ADDC A, @RW0, @RW0+d16	CMP A, @RW0, @RW0+d16	CMP A, @RW0, @RW0+d16	AND A, @RW0, @RW0+d16	AND A, @RW0, @RW0+d16	OR A, @RW0, @RW0+d16	OR A, @RW0, @RW0+d16	XOR A, @RW0, @RW0+d16	XOR A, @RW0, @RW0+d16	DBNZ @RW0, r, 'W0+d16, r	DBNZ @RW0, r, 'W0+d16, r
+9	ADD A, @RW1, @RW1+d16	SUB A, @RW1, @RW1+d16	SUB A, @RW1, @RW1+d16	SUB A, @RW1, @RW1+d16	ADDC A, @RW1, @RW1+d16	ADDC A, @RW1, @RW1+d16	CMP A, @RW1, @RW1+d16	CMP A, @RW1, @RW1+d16	AND A, @RW1, @RW1+d16	AND A, @RW1, @RW1+d16	OR A, @RW1, @RW1+d16	OR A, @RW1, @RW1+d16	XOR A, @RW1, @RW1+d16	XOR A, @RW1, @RW1+d16	DBNZ @RW1, r, 'W1+d16, r	DBNZ @RW1, r, 'W1+d16, r
+A	ADD A, @RW2, @RW2+d16	SUB A, @RW2, @RW2+d16	SUB A, @RW2, @RW2+d16	SUB A, @RW2, @RW2+d16	ADDC A, @RW2, @RW2+d16	ADDC A, @RW2, @RW2+d16	CMP A, @RW2, @RW2+d16	CMP A, @RW2, @RW2+d16	AND A, @RW2, @RW2+d16	AND A, @RW2, @RW2+d16	OR A, @RW2, @RW2+d16	OR A, @RW2, @RW2+d16	XOR A, @RW2, @RW2+d16	XOR A, @RW2, @RW2+d16	DBNZ @RW2, r, 'W2+d16, r	DBNZ @RW2, r, 'W2+d16, r
+B	ADD A, @RW3, @RW3+d16	SUB A, @RW3, @RW3+d16	SUB A, @RW3, @RW3+d16	SUB A, @RW3, @RW3+d16	ADDC A, @RW3, @RW3+d16	ADDC A, @RW3, @RW3+d16	CMP A, @RW3, @RW3+d16	CMP A, @RW3, @RW3+d16	AND A, @RW3, @RW3+d16	AND A, @RW3, @RW3+d16	OR A, @RW3, @RW3+d16	OR A, @RW3, @RW3+d16	XOR A, @RW3, @RW3+d16	XOR A, @RW3, @RW3+d16	DBNZ @RW3, r, 'W3+d16, r	DBNZ @RW3, r, 'W3+d16, r
+C	ADD A, @RW0+, @RW0+R7	SUB A, @RW0+, @RW0+R7	SUB A, @RW0+, @RW0+R7	SUB A, @RW0+, @RW0+R7	ADDC A, @RW0+, @RW0+R7	ADDC A, @RW0+, @RW0+R7	CMP A, @RW0+, @RW0+R7	CMP A, @RW0+, @RW0+R7	AND A, @RW0+, @RW0+R7	AND A, @RW0+, @RW0+R7	OR A, @RW0+, @RW0+R7	OR A, @RW0+, @RW0+R7	XOR A, @RW0+, @RW0+R7	XOR A, @RW0+, @RW0+R7	DBNZ @RW0+, r, 'W0+R7, r	DBNZ @RW0+, r, 'W0+R7, r
+D	ADD A, @RW1+, @RW1+R7	SUB A, @RW1+, @RW1+R7	SUB A, @RW1+, @RW1+R7	SUB A, @RW1+, @RW1+R7	ADDC A, @RW1+, @RW1+R7	ADDC A, @RW1+, @RW1+R7	CMP A, @RW1+, @RW1+R7	CMP A, @RW1+, @RW1+R7	AND A, @RW1+, @RW1+R7	AND A, @RW1+, @RW1+R7	OR A, @RW1+, @RW1+R7	OR A, @RW1+, @RW1+R7	XOR A, @RW1+, @RW1+R7	XOR A, @RW1+, @RW1+R7	DBNZ @RW1+, r, 'W1+R7, r	DBNZ @RW1+, r, 'W1+R7, r
+E	ADD A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	ADDC A, @RW2+, @PC+d16	ADDC A, @RW2+, @PC+d16	CMP A, @RW2+, @PC+d16	CMP A, @RW2+, @PC+d16	AND A, @RW2+, @PC+d16	AND A, @RW2+, @PC+d16	OR A, @RW2+, @PC+d16	OR A, @RW2+, @PC+d16	XOR A, @RW2+, @PC+d16	XOR A, @RW2+, @PC+d16	DBNZ @RW2+, r, 'PC+d16, r	DBNZ @RW2+, r, 'PC+d16, r
+F	ADD A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	ADDC A, @RW3+, A, addr16	ADDC A, @RW3+, A, addr16	CMP A, @RW3+, A, addr16	CMP A, @RW3+, A, addr16	AND A, @RW3+, A, addr16	AND A, @RW3+, A, addr16	OR A, @RW3+, A, addr16	OR A, @RW3+, A, addr16	XOR A, @RW3+, A, addr16	XOR A, @RW3+, A, addr16	DBNZ @RW3+, r, 'addr16, r	DBNZ @RW3+, r, 'addr16, r

Table B.9-11 ea Instruction 6 (First Byte = 75_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB A, R0, @RW0+d8, A	SUBC A, R0, @RW0+d8, A	SUBC A, R0, @RW0+d8, A	NEG R0, @RW0+d8, A	NEG A, 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 A, R1, @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 A, R2, @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 A, R3, @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 A, R4, @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 A, R5, @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 A, R6, @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 A, R7, @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 A, @RW0, @RW0+d16, A	SUBC A, @RW0, @RW0+d16, A	SUBC A, @RW0, @RW0+d16, A	NEG @RW0, @RW0+d16, A	NEG A, @RW0, @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 A, @RW1, @RW1+d16, A	SUBC A, @RW1, @RW1+d16, A	SUBC A, @RW1, @RW1+d16, A	NEG @RW1, @RW1+d16, A	NEG A, @RW1, @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 A, @RW2, @RW2+d16, A	SUBC A, @RW2, @RW2+d16, A	SUBC A, @RW2, @RW2+d16, A	NEG @RW2, @RW2+d16, A	NEG A, @RW2, @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 A, @RW3, @RW3+d16, A	SUBC A, @RW3, @RW3+d16, A	SUBC A, @RW3, @RW3+d16, A	NEG @RW3, @RW3+d16, A	NEG A, @RW3, @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 A, @RW0+, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7, A	NEG @RW0+, @RW0+RW7, A	NEG A, @RW0+, @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 A, @RW1+, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7, A	NEG @RW1+, @RW1+RW7, A	NEG A, @RW1+, @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 A, @RW2+, @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 A, @RW3+, 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_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW A, RW0, @RW0+d8	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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	SUBW 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+R7	SUBW A, RW0+, @RW0+R7	SUBW A, RW0+, @RW0+R7	SUBW A, RW0+, @RW0+R7	ADDCW A, RW0+, @RW0+R7	ADDCW A, RW0+, @RW0+R7	CMPW A, RW0+, @RW0+R7	CMPW A, RW0+, @RW0+R7	ANDW A, RW0+, @RW0+R7	ANDW A, RW0+, @RW0+R7	ORW A, RW0+, @RW0+R7	ORW A, RW0+, @RW0+R7	XORW A, RW0+, @RW0+R7	XORW A, RW0+, @RW0+R7	DWBZ RW0+, r, @RW0+R7,r	DWBZ RW0+, r, @RW0+R7,r
+D	ADDW A, RW1+, @RW1+R7	SUBW A, RW1+, @RW1+R7	SUBW A, RW1+, @RW1+R7	SUBW A, RW1+, @RW1+R7	ADDCW A, RW1+, @RW1+R7	ADDCW A, RW1+, @RW1+R7	CMPW A, RW1+, @RW1+R7	CMPW A, RW1+, @RW1+R7	ANDW A, RW1+, @RW1+R7	ANDW A, RW1+, @RW1+R7	ORW A, RW1+, @RW1+R7	ORW A, RW1+, @RW1+R7	XORW A, RW1+, @RW1+R7	XORW A, RW1+, @RW1+R7	DWBZ RW1+, r, @RW1+R7,r	DWBZ RW1+, r, @RW1+R7,r
+E	ADDW A, RW2+, @PC+d16	SUBW 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	SUBW 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_H)

	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, A, @RW0+d8, A	NOTW RW0, A, @RW0+d8, A
+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, A, @RW1+d8, A	NOTW RW1, A, @RW1+d8, A
+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, A, @RW2+d8, A	NOTW RW2, A, @RW2+d8, A
+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, A, @RW3+d8, A	NOTW RW3, A, @RW3+d8, A
+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, A, @RW4+d8, A	NOTW RW4, A, @RW4+d8, A
+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, A, @RW5+d8, A	NOTW RW5, A, @RW5+d8, A
+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, A, @RW6+d8, A	NOTW RW6, A, @RW6+d8, A
+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, A, @RW7+d8, A	NOTW RW7, A, @RW7+d8, A
+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, A, @RW0+d16, A	NOTW @RW0, A, @RW0+d16, A
+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, A, @RW1+d16, A	NOTW @RW1, A, @RW1+d16, A
+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, A, @RW2+d16, A	NOTW @RW2, A, @RW2+d16, A
+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, A, @RW3+d16, A	NOTW @RW3, A, @RW3+d16, A
+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+, A, @RW0+RW7, A	NOTW @RW0+, A, @RW0+RW7, A
+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+, A, @RW1+RW7, A	NOTW @RW1+, A, @RW1+RW7, A
+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+, A, @PC+d16, A	NOTW @RW2+, A, @PC+d16, A
+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+, A, addr16, A	NOTW @RW3+, A, addr16, A

Table B.9-14 ea Instruction 9 (First Byte = 78_H)

	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+ @PC+d16	MULUW A, @RW2+ @PC+d16	MUL A, @RW2+ @PC+d16	MUL A, @RW2+ @PC+d16	MULW A, @RW2+ @PC+d16	MULW A, @RW2+ @PC+d16	DIVU A, @RW2+ @PC+d16	DIVU A, @RW2+ @PC+d16	DIVUW A, @RW2+ @PC+d16	DIVUW A, @RW2+ @PC+d16	DIV A, @RW2+ @PC+d16	DIV A, @RW2+ @PC+d16	DIVW A, @RW2+ @PC+d16	DIVW A, @RW2+ @PC+d16
+F	MULU A, @RW3+ addr16	MULU A, @RW3+ addr16	MULUW A, @RW3+ addr16	MULUW A, @RW3+ addr16	MUL A, @RW3+ addr16	MUL A, @RW3+ addr16	MULW A, @RW3+ addr16	MULW A, @RW3+ addr16	DIVU A, @RW3+ addr16	DIVU A, @RW3+ addr16	DIVUW A, @RW3+ addr16	DIVUW A, @RW3+ addr16	DIV A, @RW3+ addr16	DIV A, @RW3+ addr16	DIVW A, @RW3+ addr16	DIVW A, @RW3+ addr16

Table B.9-15 MOVEA RWi, ea Instruction (First Byte = 79_H)

[illegible]

Table B.9-16 MOV Ri, ea Instruction (First Byte = 7A_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0 @ RW0+d8	MOV R1, R1 @ RW0+d8	MOV R2, R2 @ RW0+d8	MOV R3, R3 @ RW0+d8	MOV R4, R4 @ RW0+d8	MOV R5, R5 @ RW0+d8	MOV R6, R6 @ RW0+d8	MOV R7, R7 @ RW0+d8	MOV R8, R8 @ RW0+d8	MOV R9, R9 @ RW0+d8	MOV R10, R10 @ RW0+d8	MOV R11, R11 @ RW0+d8	MOV R12, R12 @ RW0+d8	MOV R13, R13 @ RW0+d8	MOV R14, R14 @ RW0+d8	MOV R15, R15 @ RW0+d8
+1	MOV R0, R1 @ RW1+d8	MOV R1, R2 @ RW1+d8	MOV R2, R3 @ RW1+d8	MOV R3, R4 @ RW1+d8	MOV R4, R5 @ RW1+d8	MOV R5, R6 @ RW1+d8	MOV R6, R7 @ RW1+d8	MOV R7, R8 @ RW1+d8	MOV R8, R9 @ RW1+d8	MOV R9, R10 @ RW1+d8	MOV R10, R11 @ RW1+d8	MOV R11, R12 @ RW1+d8	MOV R12, R13 @ RW1+d8	MOV R13, R14 @ RW1+d8	MOV R14, R15 @ RW1+d8	MOV R15, R16 @ RW1+d8
+2	MOV R0, R2 @ RW2+d8	MOV R1, R3 @ RW2+d8	MOV R2, R4 @ RW2+d8	MOV R3, R5 @ RW2+d8	MOV R4, R6 @ RW2+d8	MOV R5, R7 @ RW2+d8	MOV R6, R8 @ RW2+d8	MOV R7, R9 @ RW2+d8	MOV R8, R10 @ RW2+d8	MOV R9, R11 @ RW2+d8	MOV R10, R12 @ RW2+d8	MOV R11, R13 @ RW2+d8	MOV R12, R14 @ RW2+d8	MOV R13, R15 @ RW2+d8	MOV R14, R16 @ RW2+d8	MOV R15, R17 @ RW2+d8
+3	MOV R0, R3 @ RW3+d8	MOV R1, R4 @ RW3+d8	MOV R2, R5 @ RW3+d8	MOV R3, R6 @ RW3+d8	MOV R4, R7 @ RW3+d8	MOV R5, R8 @ RW3+d8	MOV R6, R9 @ RW3+d8	MOV R7, R10 @ RW3+d8	MOV R8, R11 @ RW3+d8	MOV R9, R12 @ RW3+d8	MOV R10, R13 @ RW3+d8	MOV R11, R14 @ RW3+d8	MOV R12, R15 @ RW3+d8	MOV R13, R16 @ RW3+d8	MOV R14, R17 @ RW3+d8	MOV R15, R18 @ RW3+d8
+4	MOV R0, R4 @ RW4+d8	MOV R1, R5 @ RW4+d8	MOV R2, R6 @ RW4+d8	MOV R3, R7 @ RW4+d8	MOV R4, R8 @ RW4+d8	MOV R5, R9 @ RW4+d8	MOV R6, R10 @ RW4+d8	MOV R7, R11 @ RW4+d8	MOV R8, R12 @ RW4+d8	MOV R9, R13 @ RW4+d8	MOV R10, R14 @ RW4+d8	MOV R11, R15 @ RW4+d8	MOV R12, R16 @ RW4+d8	MOV R13, R17 @ RW4+d8	MOV R14, R18 @ RW4+d8	MOV R15, R19 @ RW4+d8
+5	MOV R0, R5 @ RW5+d8	MOV R1, R6 @ RW5+d8	MOV R2, R7 @ RW5+d8	MOV R3, R8 @ RW5+d8	MOV R4, R9 @ RW5+d8	MOV R5, R10 @ RW5+d8	MOV R6, R11 @ RW5+d8	MOV R7, R12 @ RW5+d8	MOV R8, R13 @ RW5+d8	MOV R9, R14 @ RW5+d8	MOV R10, R15 @ RW5+d8	MOV R11, R16 @ RW5+d8	MOV R12, R17 @ RW5+d8	MOV R13, R18 @ RW5+d8	MOV R14, R19 @ RW5+d8	MOV R15, R20 @ RW5+d8
+6	MOV R0, R6 @ RW6+d8	MOV R1, R7 @ RW6+d8	MOV R2, R8 @ RW6+d8	MOV R3, R9 @ RW6+d8	MOV R4, R10 @ RW6+d8	MOV R5, R11 @ RW6+d8	MOV R6, R12 @ RW6+d8	MOV R7, R13 @ RW6+d8	MOV R8, R14 @ RW6+d8	MOV R9, R15 @ RW6+d8	MOV R10, R16 @ RW6+d8	MOV R11, R17 @ RW6+d8	MOV R12, R18 @ RW6+d8	MOV R13, R19 @ RW6+d8	MOV R14, R20 @ RW6+d8	MOV R15, R21 @ RW6+d8
+7	MOV R0, R7 @ RW7+d8	MOV R1, R8 @ RW7+d8	MOV R2, R9 @ RW7+d8	MOV R3, R10 @ RW7+d8	MOV R4, R11 @ RW7+d8	MOV R5, R12 @ RW7+d8	MOV R6, R13 @ RW7+d8	MOV R7, R14 @ RW7+d8	MOV R8, R15 @ RW7+d8	MOV R9, R16 @ RW7+d8	MOV R10, R17 @ RW7+d8	MOV R11, R18 @ RW7+d8	MOV R12, R19 @ RW7+d8	MOV R13, R20 @ RW7+d8	MOV R14, R21 @ RW7+d8	MOV R15, R22 @ RW7+d8
+8	MOV R0, R8 @ RW8+d8	MOV R1, R9 @ RW8+d8	MOV R2, R10 @ RW8+d8	MOV R3, R11 @ RW8+d8	MOV R4, R12 @ RW8+d8	MOV R5, R13 @ RW8+d8	MOV R6, R14 @ RW8+d8	MOV R7, R15 @ RW8+d8	MOV R8, R16 @ RW8+d8	MOV R9, R17 @ RW8+d8	MOV R10, R18 @ RW8+d8	MOV R11, R19 @ RW8+d8	MOV R12, R20 @ RW8+d8	MOV R13, R21 @ RW8+d8	MOV R14, R22 @ RW8+d8	MOV R15, R23 @ RW8+d8
+9	MOV R0, R9 @ RW9+d8	MOV R1, R10 @ RW9+d8	MOV R2, R11 @ RW9+d8	MOV R3, R12 @ RW9+d8	MOV R4, R13 @ RW9+d8	MOV R5, R14 @ RW9+d8	MOV R6, R15 @ RW9+d8	MOV R7, R16 @ RW9+d8	MOV R8, R17 @ RW9+d8	MOV R9, R18 @ RW9+d8	MOV R10, R19 @ RW9+d8	MOV R11, R20 @ RW9+d8	MOV R12, R21 @ RW9+d8	MOV R13, R22 @ RW9+d8	MOV R14, R23 @ RW9+d8	MOV R15, R24 @ RW9+d8
+A	MOV R0, R10 @ RW10+d8	MOV R1, R11 @ RW10+d8	MOV R2, R12 @ RW10+d8	MOV R3, R13 @ RW10+d8	MOV R4, R14 @ RW10+d8	MOV R5, R15 @ RW10+d8	MOV R6, R16 @ RW10+d8	MOV R7, R17 @ RW10+d8	MOV R8, R18 @ RW10+d8	MOV R9, R19 @ RW10+d8	MOV R10, R20 @ RW10+d8	MOV R11, R21 @ RW10+d8	MOV R12, R22 @ RW10+d8	MOV R13, R23 @ RW10+d8	MOV R14, R24 @ RW10+d8	MOV R15, R25 @ RW10+d8
+B	MOV R0, R11 @ RW11+d8	MOV R1, R12 @ RW11+d8	MOV R2, R13 @ RW11+d8	MOV R3, R14 @ RW11+d8	MOV R4, R15 @ RW11+d8	MOV R5, R16 @ RW11+d8	MOV R6, R17 @ RW11+d8	MOV R7, R18 @ RW11+d8	MOV R8, R19 @ RW11+d8	MOV R9, R20 @ RW11+d8	MOV R10, R21 @ RW11+d8	MOV R11, R22 @ RW11+d8	MOV R12, R23 @ RW11+d8	MOV R13, R24 @ RW11+d8	MOV R14, R25 @ RW11+d8	MOV R15, R26 @ RW11+d8
+C	MOV R0, R12 @ RW12+d8	MOV R1, R13 @ RW12+d8	MOV R2, R14 @ RW12+d8	MOV R3, R15 @ RW12+d8	MOV R4, R16 @ RW12+d8	MOV R5, R17 @ RW12+d8	MOV R6, R18 @ RW12+d8	MOV R7, R19 @ RW12+d8	MOV R8, R20 @ RW12+d8	MOV R9, R21 @ RW12+d8	MOV R10, R22 @ RW12+d8	MOV R11, R23 @ RW12+d8	MOV R12, R24 @ RW12+d8	MOV R13, R25 @ RW12+d8	MOV R14, R26 @ RW12+d8	MOV R15, R27 @ RW12+d8
+D	MOV R0, R13 @ RW13+d8	MOV R1, R14 @ RW13+d8	MOV R2, R15 @ RW13+d8	MOV R3, R16 @ RW13+d8	MOV R4, R17 @ RW13+d8	MOV R5, R18 @ RW13+d8	MOV R6, R19 @ RW13+d8	MOV R7, R20 @ RW13+d8	MOV R8, R21 @ RW13+d8	MOV R9, R22 @ RW13+d8	MOV R10, R23 @ RW13+d8	MOV R11, R24 @ RW13+d8	MOV R12, R25 @ RW13+d8	MOV R13, R26 @ RW13+d8	MOV R14, R27 @ RW13+d8	MOV R15, R28 @ RW13+d8
+E	MOV R0, R14 @ RW14+d8	MOV R1, R15 @ RW14+d8	MOV R2, R16 @ RW14+d8	MOV R3, R17 @ RW14+d8	MOV R4, R18 @ RW14+d8	MOV R5, R19 @ RW14+d8	MOV R6, R20 @ RW14+d8	MOV R7, R21 @ RW14+d8	MOV R8, R22 @ RW14+d8	MOV R9, R23 @ RW14+d8	MOV R10, R24 @ RW14+d8	MOV R11, R25 @ RW14+d8	MOV R12, R26 @ RW14+d8	MOV R13, R27 @ RW14+d8	MOV R14, R28 @ RW14+d8	MOV R15, R29 @ RW14+d8
+F	MOV R0, R15 @ RW15+d8	MOV R1, R16 @ RW15+d8	MOV R2, R17 @ RW15+d8	MOV R3, R18 @ RW15+d8	MOV R4, R19 @ RW15+d8	MOV R5, R20 @ RW15+d8	MOV R6, R21 @ RW15+d8	MOV R7, R22 @ RW15+d8	MOV R8, R23 @ RW15+d8	MOV R9, R24 @ RW15+d8	MOV R10, R25 @ RW15+d8	MOV R11, R26 @ RW15+d8	MOV R12, R27 @ RW15+d8	MOV R13, R28 @ RW15+d8	MOV R14, R29 @ RW15+d8	MOV R15, R30 @ RW15+d8

536

Table B.9-17 MOVW RWi, ea Instruction (First Byte = 7B_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW RW0, RW0, @RW0+d8	MOVW RW0, RW1, @RW1+d8	MOVW RW1, RW0, @RW0+d8	MOVW RW1, RW0, @RW0+d8	MOVW RW2, RW0, @RW0+d8	MOVW RW2, RW1, @RW1+d8	MOVW RW3, RW0, @RW0+d8	MOVW RW3, RW0, @RW0+d8	MOVW RW4, RW0, @RW0+d8	MOVW RW4, RW1, @RW1+d8	MOVW RW5, RW0, @RW0+d8	MOVW RW5, RW0, @RW0+d8	MOVW RW6, RW0, @RW0+d8	MOVW RW6, RW1, @RW1+d8	MOVW RW7, RW0, @RW0+d8	MOVW RW7, RW1, @RW1+d8
+1	MOVW RW0, RW1, @RW1+d8	MOVW RW1, RW2, @RW2+d8	MOVW RW1, RW2, @RW2+d8	MOVW RW1, RW2, @RW2+d8	MOVW RW2, RW1, @RW1+d8	MOVW RW2, RW2, @RW2+d8	MOVW RW3, RW1, @RW1+d8	MOVW RW3, RW2, @RW2+d8	MOVW RW4, RW1, @RW1+d8	MOVW RW4, RW2, @RW2+d8	MOVW RW5, RW1, @RW1+d8	MOVW RW5, RW2, @RW2+d8	MOVW RW6, RW1, @RW1+d8	MOVW RW6, RW2, @RW2+d8	MOVW RW7, RW1, @RW1+d8	MOVW RW7, RW2, @RW2+d8
+2	MOVW RW0, RW2, @RW2+d8	MOVW RW1, RW3, @RW3+d8	MOVW RW1, RW3, @RW3+d8	MOVW RW1, RW3, @RW3+d8	MOVW RW2, RW3, @RW3+d8	MOVW RW2, RW4, @RW4+d8	MOVW RW3, RW3, @RW3+d8	MOVW RW3, RW4, @RW4+d8	MOVW RW4, RW3, @RW3+d8	MOVW RW4, RW4, @RW4+d8	MOVW RW5, RW3, @RW3+d8	MOVW RW5, RW4, @RW4+d8	MOVW RW6, RW3, @RW3+d8	MOVW RW6, RW4, @RW4+d8	MOVW RW7, RW3, @RW3+d8	MOVW RW7, RW4, @RW4+d8
+3	MOVW RW0, RW3, @RW3+d8	MOVW RW1, RW4, @RW4+d8	MOVW RW1, RW4, @RW4+d8	MOVW RW1, RW4, @RW4+d8	MOVW RW2, RW4, @RW4+d8	MOVW RW2, RW5, @RW5+d8	MOVW RW3, RW5, @RW5+d8	MOVW RW3, RW6, @RW6+d8	MOVW RW4, RW5, @RW5+d8	MOVW RW4, RW6, @RW6+d8	MOVW RW5, RW5, @RW5+d8	MOVW RW5, RW6, @RW6+d8	MOVW RW6, RW5, @RW5+d8	MOVW RW6, RW6, @RW6+d8	MOVW RW7, RW5, @RW5+d8	MOVW RW7, RW6, @RW6+d8
+4	MOVW RW0, RW4, @RW4+d8	MOVW RW1, RW5, @RW5+d8	MOVW RW1, RW5, @RW5+d8	MOVW RW1, RW5, @RW5+d8	MOVW RW2, RW5, @RW5+d8	MOVW RW2, RW6, @RW6+d8	MOVW RW3, RW6, @RW6+d8	MOVW RW3, RW7, @RW7+d8	MOVW RW4, RW6, @RW6+d8	MOVW RW4, RW7, @RW7+d8	MOVW RW5, RW6, @RW6+d8	MOVW RW5, RW7, @RW7+d8	MOVW RW6, RW6, @RW6+d8	MOVW RW6, RW7, @RW7+d8	MOVW RW7, RW6, @RW6+d8	MOVW RW7, RW7, @RW7+d8
+5	MOVW RW0, RW5, @RW5+d8	MOVW RW1, RW6, @RW6+d8	MOVW RW1, RW6, @RW6+d8	MOVW RW1, RW6, @RW6+d8	MOVW RW2, RW6, @RW6+d8	MOVW RW2, RW7, @RW7+d8	MOVW RW3, RW7, @RW7+d8	MOVW RW3, RW8, @RW8+d8	MOVW RW4, RW7, @RW7+d8	MOVW RW4, RW8, @RW8+d8	MOVW RW5, RW7, @RW7+d8	MOVW RW5, RW8, @RW8+d8	MOVW RW6, RW7, @RW7+d8	MOVW RW6, RW8, @RW8+d8	MOVW RW7, RW7, @RW7+d8	MOVW RW7, RW8, @RW8+d8
+6	MOVW RW0, RW6, @RW6+d8	MOVW RW1, RW7, @RW7+d8	MOVW RW1, RW7, @RW7+d8	MOVW RW1, RW7, @RW7+d8	MOVW RW2, RW7, @RW7+d8	MOVW RW2, RW8, @RW8+d8	MOVW RW3, RW8, @RW8+d8	MOVW RW3, RW9, @RW9+d8	MOVW RW4, RW8, @RW8+d8	MOVW RW4, RW9, @RW9+d8	MOVW RW5, RW8, @RW8+d8	MOVW RW5, RW9, @RW9+d8	MOVW RW6, RW8, @RW8+d8	MOVW RW6, RW9, @RW9+d8	MOVW RW7, RW8, @RW8+d8	MOVW RW7, RW9, @RW9+d8
+7	MOVW RW0, RW7, @RW7+d8	MOVW RW1, RW8, @RW8+d8	MOVW RW1, RW8, @RW8+d8	MOVW RW1, RW8, @RW8+d8	MOVW RW2, RW8, @RW8+d8	MOVW RW2, RW9, @RW9+d8	MOVW RW3, RW9, @RW9+d8	MOVW RW3, RW10, @RW10+d8	MOVW RW4, RW9, @RW9+d8	MOVW RW4, RW10, @RW10+d8	MOVW RW5, RW9, @RW9+d8	MOVW RW5, RW10, @RW10+d8	MOVW RW6, RW9, @RW9+d8	MOVW RW6, RW10, @RW10+d8	MOVW RW7, RW9, @RW9+d8	MOVW RW7, RW10, @RW10+d8
+8	MOVW RW0, @RW0, @RW0+d16	MOVW RW1, @RW1, @RW1+d16	MOVW RW1, @RW1, @RW1+d16	MOVW RW1, @RW1, @RW1+d16	MOVW RW2, @RW2, @RW2+d16	MOVW RW2, @RW3, @RW3+d16	MOVW RW3, @RW3, @RW3+d16	MOVW RW3, @RW4, @RW4+d16	MOVW RW4, @RW4, @RW4+d16	MOVW RW4, @RW5, @RW5+d16	MOVW RW5, @RW5, @RW5+d16	MOVW RW5, @RW6, @RW6+d16	MOVW RW6, @RW6, @RW6+d16	MOVW RW6, @RW7, @RW7+d16	MOVW RW7, @RW7, @RW7+d16	MOVW RW7, @RW8, @RW8+d16
+9	MOVW RW0, @RW1, @RW1+d16	MOVW RW1, @RW2, @RW2+d16	MOVW RW1, @RW2, @RW2+d16	MOVW RW1, @RW2, @RW2+d16	MOVW RW2, @RW2, @RW2+d16	MOVW RW2, @RW3, @RW3+d16	MOVW RW3, @RW3, @RW3+d16	MOVW RW3, @RW4, @RW4+d16	MOVW RW4, @RW4, @RW4+d16	MOVW RW4, @RW5, @RW5+d16	MOVW RW5, @RW5, @RW5+d16	MOVW RW5, @RW6, @RW6+d16	MOVW RW6, @RW6, @RW6+d16	MOVW RW6, @RW7, @RW7+d16	MOVW RW7, @RW7, @RW7+d16	MOVW RW7, @RW8, @RW8+d16
+A	MOVW RW0, @RW2, @RW2+d16	MOVW RW1, @RW3, @RW3+d16	MOVW RW1, @RW3, @RW3+d16	MOVW RW1, @RW3, @RW3+d16	MOVW RW2, @RW3, @RW3+d16	MOVW RW2, @RW4, @RW4+d16	MOVW RW3, @RW4, @RW4+d16	MOVW RW3, @RW5, @RW5+d16	MOVW RW4, @RW5, @RW5+d16	MOVW RW4, @RW6, @RW6+d16	MOVW RW5, @RW6, @RW6+d16	MOVW RW5, @RW7, @RW7+d16	MOVW RW6, @RW7, @RW7+d16	MOVW RW6, @RW8, @RW8+d16	MOVW RW7, @RW8, @RW8+d16	MOVW RW7, @RW9, @RW9+d16
+B	MOVW RW0, @RW3, @RW3+d16	MOVW RW1, @RW4, @RW4+d16	MOVW RW1, @RW4, @RW4+d16	MOVW RW1, @RW4, @RW4+d16	MOVW RW2, @RW4, @RW4+d16	MOVW RW2, @RW5, @RW5+d16	MOVW RW3, @RW5, @RW5+d16	MOVW RW3, @RW6, @RW6+d16	MOVW RW4, @RW6, @RW6+d16	MOVW RW4, @RW7, @RW7+d16	MOVW RW5, @RW7, @RW7+d16	MOVW RW5, @RW8, @RW8+d16	MOVW RW6, @RW8, @RW8+d16	MOVW RW6, @RW9, @RW9+d16	MOVW RW7, @RW9, @RW9+d16	MOVW RW7, @RW10, @RW10+d16
+C	MOVW RW0, @RW0+, @RW0+RW7	MOVW RW1, @RW1+, @RW1+RW7	MOVW RW1, @RW1+, @RW1+RW7	MOVW RW1, @RW1+, @RW1+RW7	MOVW RW2, @RW1+, @RW1+RW7	MOVW RW2, @RW2+, @RW2+RW7	MOVW RW3, @RW2+, @RW2+RW7	MOVW RW3, @RW3+, @RW3+RW7	MOVW RW4, @RW3+, @RW3+RW7	MOVW RW4, @RW4+, @RW4+RW7	MOVW RW5, @RW4+, @RW4+RW7	MOVW RW5, @RW5+, @RW5+RW7	MOVW RW6, @RW5+, @RW5+RW7	MOVW RW6, @RW6+, @RW6+RW7	MOVW RW7, @RW6+, @RW6+RW7	MOVW RW7, @RW7+, @RW7+RW7
+D	MOVW RW0, @RW1+, @RW1+RW7	MOVW RW1, @RW2+, @RW2+RW7	MOVW RW1, @RW2+, @RW2+RW7	MOVW RW1, @RW2+, @RW2+RW7	MOVW RW2, @RW2+, @RW2+RW7	MOVW RW2, @RW3+, @RW3+RW7	MOVW RW3, @RW3+, @RW3+RW7	MOVW RW3, @RW4+, @RW4+RW7	MOVW RW4, @RW4+, @RW4+RW7	MOVW RW4, @RW5+, @RW5+RW7	MOVW RW5, @RW5+, @RW5+RW7	MOVW RW5, @RW6+, @RW6+RW7	MOVW RW6, @RW6+, @RW6+RW7	MOVW RW6, @RW7+, @RW7+RW7	MOVW RW7, @RW7+, @RW7+RW7	MOVW RW7, @RW8+, @RW8+RW7
+E	MOVW RW0, @RW2+, @PC+d16	MOVW RW1, @RW3+, @PC+d16	MOVW RW1, @RW3+, @PC+d16	MOVW RW1, @RW3+, @PC+d16	MOVW RW2, @RW3+, @RW3+d16	MOVW RW2, @RW4+, @RW4+d16	MOVW RW3, @RW4+, @RW4+d16	MOVW RW3, @RW5+, @RW5+d16	MOVW RW4, @RW5+, @RW5+d16	MOVW RW4, @RW6+, @RW6+d16	MOVW RW5, @RW6+, @RW6+d16	MOVW RW5, @RW7+, @RW7+d16	MOVW RW6, @RW7+, @RW7+d16	MOVW RW6, @RW8+, @RW8+d16	MOVW RW7, @RW8+, @RW8+d16	MOVW RW7, @RW9+, @RW9+d16
+F	MOVW RW0, @RW3+, @RW0, addr16	MOVW RW1, @RW4+, @RW1, addr16	MOVW RW1, @RW4+, @RW1, addr16	MOVW RW1, @RW4+, @RW1, addr16	MOVW RW2, @RW4+, @RW2, addr16	MOVW RW2, @RW5+, @RW2, addr16	MOVW RW3, @RW5+, @RW3, addr16	MOVW RW3, @RW6+, @RW3, addr16	MOVW RW4, @RW6+, @RW4, addr16	MOVW RW4, @RW7+, @RW4, addr16	MOVW RW5, @RW7+, @RW5, addr16	MOVW RW5, @RW8+, @RW5, addr16	MOVW RW6, @RW8+, @RW6, addr16	MOVW RW6, @RW9+, @RW6, addr16	MOVW RW7, @RW9+, @RW7, addr16	MOVW RW7, @RW10+, @RW7, addr16

Table B.9-18 MOV ea, Ri Instruction (First Byte = 7C_H)

	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, 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
+1	MOV R1, R0, @RW1+d8, R0	MOV R1, R1, @RW1+d8, R1	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
+2	MOV R2, R0, @RW2+d8, R0	MOV R2, R1, @RW2+d8, R1	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
+3	MOV R3, R0, @RW3+d8, R0	MOV R3, R1, @RW3+d8, R1	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
+4	MOV R4, R0, @RW4+d8, R0	MOV R4, R1, @RW4+d8, R1	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
+5	MOV R5, R0, @RW5+d8, R0	MOV R5, R1, @RW5+d8, R1	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
+6	MOV R6, R0, @RW6+d8, R0	MOV R6, R1, @RW6+d8, R1	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
+7	MOV R7, R0, @RW7+d8, R0	MOV R7, R1, @RW7+d8, R1	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
+8	MOV @RW0, R0, @RW0+d16, R0	MOV @RW0, R1, @RW0+d16, R1	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
+9	MOV @RW1, R0, @RW1+d16, R0	MOV @RW1, R1, @RW1+d16, R1	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
+A	MOV @RW2, R0, @RW2+d16, R0	MOV @RW2, R1, @RW2+d16, R1	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
+B	MOV @RW3, R0, @RW3+d16, R0	MOV @RW3, R1, @RW3+d16, R1	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
+C	MOV @RW0+, R0, @RW0+RW7, R0	MOV @RW0+, R1, @RW0+RW7, R1	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
+D	MOV @RW1+, R0, @RW1+RW7, R0	MOV @RW1+, R1, @RW1+RW7, R1	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
+E	MOV @RW2+, R0, @PC+d16, R0	MOV @RW2+, R1, @PC+d16, R1	MOV @RW2+, R1, @PC+d16, R1	MOV @RW2+, R2, @PC+d16, R2	MOV @RW2+, R3, @PC+d16, R3	MOV @RW2+, R4, @PC+d16, R4	MOV @RW2+, R5, @PC+d16, R5	MOV @RW2+, R6, @PC+d16, R6	MOV @RW2+, R7, @PC+d16, R7	MOV @RW2+, R8, @PC+d16, R8	MOV @RW2+, R9, @PC+d16, R9	MOV @RW2+, R10, @PC+d16, R10	MOV @RW2+, R11, @PC+d16, R11	MOV @RW2+, R12, @PC+d16, R12	MOV @RW2+, R13, @PC+d16, R13	MOV @RW2+, R14, @PC+d16, R14
+F	MOV @RW3+, R0, addr16, R0	MOV @RW3+, R1, addr16, R1	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

Table B.9-19 MOVW ea, Rwi Instruction (First Byte = 7D_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW R0, R0, @R0+0, R0	MOVW R0, R0, @R0+0, R0	MOVW R0, R1, @R0+0, R1	MOVW R0, R1, @R0+0, R1	MOVW R0, R2, @R0+0, R2	MOVW R0, R2, @R0+0, R2	MOVW R0, R3, @R0+0, R3	MOVW R0, R3, @R0+0, R3	MOVW R0, R4, @R0+0, R4	MOVW R0, R4, @R0+0, R4	MOVW R0, R5, @R0+0, R5	MOVW R0, R5, @R0+0, R5	MOVW R0, R6, @R0+0, R6	MOVW R0, R6, @R0+0, R6	MOVW R0, R7, @R0+0, R7	MOVW R0, R7, @R0+0, R7
+1	MOVW R1, R0, @R1+0, R0	MOVW R1, R0, @R1+0, R0	MOVW R1, R1, @R1+0, R1	MOVW R1, R1, @R1+0, R1	MOVW R1, R2, @R1+0, R2	MOVW R1, R2, @R1+0, R2	MOVW R1, R3, @R1+0, R3	MOVW R1, R3, @R1+0, R3	MOVW R1, R4, @R1+0, R4	MOVW R1, R4, @R1+0, R4	MOVW R1, R5, @R1+0, R5	MOVW R1, R5, @R1+0, R5	MOVW R1, R6, @R1+0, R6	MOVW R1, R6, @R1+0, R6	MOVW R1, R7, @R1+0, R7	MOVW R1, R7, @R1+0, R7
+2	MOVW R2, R0, @R2+0, R0	MOVW R2, R0, @R2+0, R0	MOVW R2, R1, @R2+0, R1	MOVW R2, R1, @R2+0, R1	MOVW R2, R2, @R2+0, R2	MOVW R2, R2, @R2+0, R2	MOVW R2, R3, @R2+0, R3	MOVW R2, R3, @R2+0, R3	MOVW R2, R4, @R2+0, R4	MOVW R2, R4, @R2+0, R4	MOVW R2, R5, @R2+0, R5	MOVW R2, R5, @R2+0, R5	MOVW R2, R6, @R2+0, R6	MOVW R2, R6, @R2+0, R6	MOVW R2, R7, @R2+0, R7	MOVW R2, R7, @R2+0, R7
+3	MOVW R3, R0, @R3+0, R0	MOVW R3, R0, @R3+0, R0	MOVW R3, R1, @R3+0, R1	MOVW R3, R1, @R3+0, R1	MOVW R3, R2, @R3+0, R2	MOVW R3, R2, @R3+0, R2	MOVW R3, R3, @R3+0, R3	MOVW R3, R3, @R3+0, R3	MOVW R3, R4, @R3+0, R4	MOVW R3, R4, @R3+0, R4	MOVW R3, R5, @R3+0, R5	MOVW R3, R5, @R3+0, R5	MOVW R3, R6, @R3+0, R6	MOVW R3, R6, @R3+0, R6	MOVW R3, R7, @R3+0, R7	MOVW R3, R7, @R3+0, R7
+4	MOVW R4, R0, @R4+0, R0	MOVW R4, R0, @R4+0, R0	MOVW R4, R1, @R4+0, R1	MOVW R4, R1, @R4+0, R1	MOVW R4, R2, @R4+0, R2	MOVW R4, R2, @R4+0, R2	MOVW R4, R3, @R4+0, R3	MOVW R4, R3, @R4+0, R3	MOVW R4, R4, @R4+0, R4	MOVW R4, R4, @R4+0, R4	MOVW R4, R5, @R4+0, R5	MOVW R4, R5, @R4+0, R5	MOVW R4, R6, @R4+0, R6	MOVW R4, R6, @R4+0, R6	MOVW R4, R7, @R4+0, R7	MOVW R4, R7, @R4+0, R7
+5	MOVW R5, R0, @R5+0, R0	MOVW R5, R0, @R5+0, R0	MOVW R5, R1, @R5+0, R1	MOVW R5, R1, @R5+0, R1	MOVW R5, R2, @R5+0, R2	MOVW R5, R2, @R5+0, R2	MOVW R5, R3, @R5+0, R3	MOVW R5, R3, @R5+0, R3	MOVW R5, R4, @R5+0, R4	MOVW R5, R4, @R5+0, R4	MOVW R5, R5, @R5+0, R5	MOVW R5, R5, @R5+0, R5	MOVW R5, R6, @R5+0, R6	MOVW R5, R6, @R5+0, R6	MOVW R5, R7, @R5+0, R7	MOVW R5, R7, @R5+0, R7
+6	MOVW R6, R0, @R6+0, R0	MOVW R6, R0, @R6+0, R0	MOVW R6, R1, @R6+0, R1	MOVW R6, R1, @R6+0, R1	MOVW R6, R2, @R6+0, R2	MOVW R6, R2, @R6+0, R2	MOVW R6, R3, @R6+0, R3	MOVW R6, R3, @R6+0, R3	MOVW R6, R4, @R6+0, R4	MOVW R6, R4, @R6+0, R4	MOVW R6, R5, @R6+0, R5	MOVW R6, R5, @R6+0, R5	MOVW R6, R6, @R6+0, R6	MOVW R6, R6, @R6+0, R6	MOVW R6, R7, @R6+0, R7	MOVW R6, R7, @R6+0, R7
+7	MOVW R7, R0, @R7+0, R0	MOVW R7, R0, @R7+0, R0	MOVW R7, R1, @R7+0, R1	MOVW R7, R1, @R7+0, R1	MOVW R7, R2, @R7+0, R2	MOVW R7, R2, @R7+0, R2	MOVW R7, R3, @R7+0, R3	MOVW R7, R3, @R7+0, R3	MOVW R7, R4, @R7+0, R4	MOVW R7, R4, @R7+0, R4	MOVW R7, R5, @R7+0, R5	MOVW R7, R5, @R7+0, R5	MOVW R7, R6, @R7+0, R6	MOVW R7, R6, @R7+0, R6	MOVW R7, R7, @R7+0, R7	MOVW R7, R7, @R7+0, R7
+8	MOVW @R0, R0, +0, R0	MOVW @R0, R0, +0, R0	MOVW @R0, R1, +0, R1	MOVW @R0, R1, +0, R1	MOVW @R0, R2, +0, R2	MOVW @R0, R2, +0, R2	MOVW @R0, R3, +0, R3	MOVW @R0, R3, +0, R3	MOVW @R0, R4, +0, R4	MOVW @R0, R4, +0, R4	MOVW @R0, R5, +0, R5	MOVW @R0, R5, +0, R5	MOVW @R0, R6, +0, R6	MOVW @R0, R6, +0, R6	MOVW @R0, R7, +0, R7	MOVW @R0, R7, +0, R7
+9	MOVW @R1, R0, +0, R0	MOVW @R1, R0, +0, R0	MOVW @R1, R1, +0, R1	MOVW @R1, R1, +0, R1	MOVW @R1, R2, +0, R2	MOVW @R1, R2, +0, R2	MOVW @R1, R3, +0, R3	MOVW @R1, R3, +0, R3	MOVW @R1, R4, +0, R4	MOVW @R1, R4, +0, R4	MOVW @R1, R5, +0, R5	MOVW @R1, R5, +0, R5	MOVW @R1, R6, +0, R6	MOVW @R1, R6, +0, R6	MOVW @R1, R7, +0, R7	MOVW @R1, R7, +0, R7
+A	MOVW @R2, R0, +0, R0	MOVW @R2, R0, +0, R0	MOVW @R2, R1, +0, R1	MOVW @R2, R1, +0, R1	MOVW @R2, R2, +0, R2	MOVW @R2, R2, +0, R2	MOVW @R2, R3, +0, R3	MOVW @R2, R3, +0, R3	MOVW @R2, R4, +0, R4	MOVW @R2, R4, +0, R4	MOVW @R2, R5, +0, R5	MOVW @R2, R5, +0, R5	MOVW @R2, R6, +0, R6	MOVW @R2, R6, +0, R6	MOVW @R2, R7, +0, R7	MOVW @R2, R7, +0, R7
+B	MOVW @R3, R0, +0, R0	MOVW @R3, R0, +0, R0	MOVW @R3, R1, +0, R1	MOVW @R3, R1, +0, R1	MOVW @R3, R2, +0, R2	MOVW @R3, R2, +0, R2	MOVW @R3, R3, +0, R3	MOVW @R3, R3, +0, R3	MOVW @R3, R4, +0, R4	MOVW @R3, R4, +0, R4	MOVW @R3, R5, +0, R5	MOVW @R3, R5, +0, R5	MOVW @R3, R6, +0, R6	MOVW @R3, R6, +0, R6	MOVW @R3, R7, +0, R7	MOVW @R3, R7, +0, R7
+C	MOVW @R0+0, R0, +0, R0	MOVW @R0+0, R0, +0, R0	MOVW @R0+0, R1, +0, R1	MOVW @R0+0, R1, +0, R1	MOVW @R0+0, R2, +0, R2	MOVW @R0+0, R2, +0, R2	MOVW @R0+0, R3, +0, R3	MOVW @R0+0, R3, +0, R3	MOVW @R0+0, R4, +0, R4	MOVW @R0+0, R4, +0, R4	MOVW @R0+0, R5, +0, R5	MOVW @R0+0, R5, +0, R5	MOVW @R0+0, R6, +0, R6	MOVW @R0+0, R6, +0, R6	MOVW @R0+0, R7, +0, R7	MOVW @R0+0, R7, +0, R7
+D	MOVW @R1+0, R0, +0, R0	MOVW @R1+0, R0, +0, R0	MOVW @R1+0, R1, +0, R1	MOVW @R1+0, R1, +0, R1	MOVW @R1+0, R2, +0, R2	MOVW @R1+0, R2, +0, R2	MOVW @R1+0, R3, +0, R3	MOVW @R1+0, R3, +0, R3	MOVW @R1+0, R4, +0, R4	MOVW @R1+0, R4, +0, R4	MOVW @R1+0, R5, +0, R5	MOVW @R1+0, R5, +0, R5	MOVW @R1+0, R6, +0, R6	MOVW @R1+0, R6, +0, R6	MOVW @R1+0, R7, +0, R7	MOVW @R1+0, R7, +0, R7
+E	MOVW @R2+0, R0, +0, R0	MOVW @R2+0, R0, +0, R0	MOVW @R2+0, R1, +0, R1	MOVW @R2+0, R1, +0, R1	MOVW @R2+0, R2, +0, R2	MOVW @R2+0, R2, +0, R2	MOVW @R2+0, R3, +0, R3	MOVW @R2+0, R3, +0, R3	MOVW @R2+0, R4, +0, R4	MOVW @R2+0, R4, +0, R4	MOVW @R2+0, R5, +0, R5	MOVW @R2+0, R5, +0, R5	MOVW @R2+0, R6, +0, R6	MOVW @R2+0, R6, +0, R6	MOVW @R2+0, R7, +0, R7	MOVW @R2+0, R7, +0, R7
+F	MOVW @R3+0, R0, +0, R0	MOVW @R3+0, R0, +0, R0	MOVW @R3+0, R1, +0, R1	MOVW @R3+0, R1, +0, R1	MOVW @R3+0, R2, +0, R2	MOVW @R3+0, R2, +0, R2	MOVW @R3+0, R3, +0, R3	MOVW @R3+0, R3, +0, R3	MOVW @R3+0, R4, +0, R4	MOVW @R3+0, R4, +0, R4	MOVW @R3+0, R5, +0, R5	MOVW @R3+0, R5, +0, R5	MOVW @R3+0, R6, +0, R6	MOVW @R3+0, R6, +0, R6	MOVW @R3+0, R7, +0, R7	MOVW @R3+0, R7, +0, R7

Table B.9-20 XCH Ri, ea Instruction (First Byte = 7EH)

[illegible]

540

Table B.9-21 XCHW RWi, ea Instruction (First Byte = 7F_H)

[illegible]

INDEX

**The index follows on the next page.
This is listed in alphabetic order.**

Index

Numerics

16-bit free-run timer (x 1).....	216
16-bit free-run timer count timing	232
16-bit free-run timer operations.....	231
16-bit I/O timer block diagram	218
16-bit I/O timer registers.....	219
16-bit input capture operations.....	236
16-bit output compare operations	233
16-bit output compare timing.....	234
16-bit reload register (TMRLR).....	245
16-bit reload timer (with event count function), block diagram of.....	240
16-bit reload timer (with event count function), overview of	240
16-bit reload timer (with event count function), register of	241
16-bit timer register (TMR)	245
1M-bit flash memory program, of example.....	451
1M-bit flash memory sector configuration of	427
1M-bit flash memory, feature of.....	426
8/16-bit PPG interrupt	264
8/16-bit PPG operation.....	264
8/16-bit PPG operation modes	266
8/16-bit PPG, block diagram of	253
8/16-bit PPG, overview of	252
8/16-bit PPG, register in	255

A

A/D converter block diagram.....	290
A/D converter, caution on using	289
A/D converter, overview of	288
A/D converter, registers of	291
access mode	128
accumulator (A)	35
acknowledgment bit	401
ADB, additional bank register.....	30
ADCR1 and ADCR2.....	297
ADCS1 and ADCS2	292
additional bank register(ADB)	30
address match detection function, block diagram of.....	414
address match detection function, operation of	417
address match detection function, system configuration example of.....	418

Addressing.....	482
ADER	158
arbitration.....	391
ARSR.....	136
automatic ready function selection register (ARSR)	136
avoiding being subject to the notes.....	52

B

bank method, addressing by.....	29
bank registers	43
bank selection prefixes	46
BAP.....	76
basic configuration of MB90F583C/CA serial programming connection	458
bit format.....	407
block diagram for IEBus™ controller.....	347
block diagram for MB90580C series.....	6
broadcast	392
broadcast bit	394
broadcast control bit read register (DCRR).....	367
broadcast control bit set register (DCWR)	353
broadcast reception	375
broadcast reception (occurrence of reception interrupt)	378
buffer address pointer (BAP)	76
bus control signal selection register (ECSR)	139
bus mode	128

C

Calculating the Execution Cycle Count.....	499
calculating pulse width/period	206
caution on using the A/D converter.....	289
CCR	38
CDCR.....	316
characteristic of PWC timer	180
chip deletion, deleting the data from the flash memory	446
CKSCR	97
clearing the timer	201
CLK-bsynchronous mode, values set in registers in	339
clock division control register (CDCR)	316
clock generator, notes as to.....	84
clock monitor functions, block diagram of.....	410

clock output permission register (CLKR)	411	dedicated registers	33
clock selection bits	213	delayed interrupt generating module (DIRR),	
clock selection register (CKSCR)	97	register in	284
CMR	47	delayed interrupt generating module, block diagram	
CMRH	356	of	284
CMRL	358	delayed interrupt generating module,	
command register (higher 8 bits) (CMRH)	356	operation of	285
command register (lower 8 bits) (CMRL)	358	delayed interrupt request latch, note on use of	
command sequence table	430	285
common register bank prefix (CMR)	47	deleting data for the flash memory	426
communication address	392	DERR	368
communication modes	391	Description of Instruction Presentation Items and	
communication prescaler	334	Symbols	502
communication prescaler, operation of	318	determining the priority for using the bus	
communication, end of	339	(arbitration)	391
communication, start of	339	DEWR	355
compare register (OCCP0 and OCCP1)	225	Direct Addressing	484
condition code register (CCR)	38	direct page register (DPR)	42
continuous measurement mode	205	DIRR	284
control field	397	DIV A, Ri instruction	51
control status registers (ADCS1 and ADCS2)	292	division rate control register (DIVR)	191
control status register (FMCS)	428	DIVR	191
control status registers (ICS23 and ICS01)	229	DIVW A, RWi instruction	51
control status register (OCS0 to OCS1)	226	DPR	42
conversion data protection function	307	DQ3	439
count clock and maximum period	203	DQ5	438
count clock selection	197	DQ6	436
count clock, notes on selecting	269	DQ7	434
counter operation statuses	250	DTB, data bank register	30
CPU operation function, intermittent	107	DTP operation	278
		DTP request	280
D		DTP/external interrupt circuit operation	
D/A control registers (DACR0 and DACR1)	313	procedure	281
D/A converter block diagram	311	DTP/external interrupt circuit, block diagram of	
D/A converter data registers (DAT0 and DAT1)		275
.....	312	DTP/external interrupt circuit, registers in	274
D/A converter registers	310	DTP/external interrupt enable register (ENIR)	276
D/A converter, operation of	314	DTP/external interrupt source register (EIRR)	276
DACR0 and DACR1	313	division period measurement mode	213
DAT0 and DAT1	312		
data bank register (DTB)	30	E	
data counter (DCT)	74	each bus mode, memory space for	131
data field	399	each bus mode, recommended setting sample of	
data polling flag (DQ7)	434	memory space for	132
data registers (ADCR1 and ADCR2)	297	ECSR	139
DCRR	367	Effective Address Field	483, 501
DCT	74	EI ² OS activation in pause mode, example of	305
DCWR	353	EI ² OS activation in single mode, example of	301
DDR _x	154	EI ² OS activation in successive mode,	
		example of	303

INDEX

EI ² OS status register (ISCS).....	74
EI ² OS, conversion with	300
EIRR.....	276
ENIR.....	276
entire flash memory, block diagram of	427
error, transmission operation at occurrence of.....	388
example of program patch processing	419
Execution Cycle Count.....	498
expanded intelligent I/O service (EI ² OS), configuration of	69
expanded intelligent I/O service (EI ² OS), execution time of	79
expanded intelligent I/O service (EI ² OS), operational flow of	77
expanded intelligent I/O service (EI ² OS), overview of	68
expanded intelligent I/O service descriptor (ISD)	73
extended intelligent I/O service (EI ² OS).....	333
extended intelligent I/O service (EI ² OS) function and interrupts	247
external address output control register (HACR)	138
external bus pin control circuit, block diagram of external memory access	134
external bus pin control circuit, registers for external memory access	135
external clock	335
external event count.....	246
external interrupt operation	278
external interrupt request and a DTP request, switching between.....	280
external interrupt request level	282
external memory access (external bus pin control circuit)	134
external memory access (external bus pin control circuit), block diagram of	134
external memory access (external bus pin control circuit), registers for	135
external memory access control signal	142
F	
F ² MC-16LX Instruction List	505
features of IEBus TM controller.....	346
features of MB90580C series.....	2
five flags (PE, ORE, FRE, RDRF, and TDRE)	340
flag change suppression prefix (NCC)	48
flash memory register.....	426
flash memory writing and deletion, detailed explanation of.....	442
flash memory, writing data in	444
flash microcomputer programmer (when power is supplied from a programmer), example of minimum connection with	468
flash microcomputer programmer (when user power supply is used), example of minimum connection with.....	466
flash microcomputer programmer, of system configuration	461
flowchart of pulse-width measurement operation	211
flowchart of timer mode operation.....	204
FMCS.....	428
FPT-100P-M05 package, outside dimensions of.....	7
FPT-100P-M06 package, outside dimensions of.....	8
FTP-100P-M05, pin layout of.....	9
FTP-100P-M06, pin layout of.....	10
G	
general-purpose registers	44
group broadcast	392
H	
HACR.....	138
hardware components, initial value in.....	265
hardware interrupt request during writing to the internal resource area.....	60
hardware interrupts, example of procedure for using	65
hardware interrupts, notes on the use of	61
hardware interrupts, operating flow for	64
hardware interrupts, operation of	62
hardware interrupts, overview of.....	59
hardware interrupts, structure of	59
hardware sequence flag.....	432
hardware standby mode, releasing.....	106
hardware standby mode, transition to.....	106
hold function.....	146
I	
I (interrupt enable flag).....	38
I/O circuit types	19
I/O map	472
I/O port block diagram.....	149
I/O port overview	148
I/O port registers	151
I/O register address pointer (IOA).....	74
ICR.....	70
ICS23 and ICS01	229

IEBus™ controller, block diagram for	347	low power-consumption mode, status transition in	119
IEBus™ controller, features of	346	low-power consumption control circuit, block diagram of	94
IEBus™ controller, registers of	348	low-power consumption control circuit, operation of	100
IEBus™ protocol operation, overview of	390	low-power consumption control circuit, overview of	92
input pull-up resistor setting registers (RDR0, RDR1, and RDR6)	157	low-power consumption mode control register (LPMCR)	95
ILM	40	LPMCR	95
Indirect Addressing	490	LRRH and LRRL	365
initialization	216	M	
initialization routine	380	machine clock, initializing	110
initializing the machine clock	110	main and PLL clocks, switching between	109
input capture (x 4)	217	main clock and subclock, switching between	109
input capture data register (IPCP0 to IPCP3)	229	main clock, setting the oscillation stabilization time for	108
input capture input timing	237	main routine	379
input pin function (for the internal clock mode)	248	master address field	395
Instruction Types	481	master address read registers (MARH and MARL)	366
internal clock operations	246	master or slave transmission (occurrence of transmission interrupt)	376
internal timer	334	master reception	374
interrupts, Extended Intelligent I/O Service (EI ² OS) function and	247	master reception (occurrence of reception interrupt)	376
interrupt causes	55	master reception routine	383
interrupt control register (ICR)	70	master transmission	371
interrupt enable flag (I)	38	master transmission routine	381
interrupt flag setting timing in operating modes	340	MAWH and MAWL	351
interrupt level mask register (ILM)	40	MB90580C series, block diagram for	6
interrupt processing routine	379	MB90580C series, features of	2
interrupt request generation	202, 207	MB90F583C/CA serial programming connection, basic configuration of	458
interrupt stop instruction	60	measurement mode and measurement operation	208
interrupt suppression instructions	49	measurement result data	205
interrupt suppression instruction, restrictions on	49	measurement termination flag in timer mode	212
interrupt vectors	57	memory access mode overview	128
Interrupts, overview of	54	memory space	28
interval interrupt function	164, 178	memory space for each bus mode, recommended setting sample of	132
IOA	74	memory space, allocating multiple-byte data in	32
IPCP0 to IPCP3	229	minimum input pulse width	206, 213
ISCS	74	mode 1), application of UART (during operation in	343
ISD	73	mode data	130
L			
linear addressing methods	29		
local address set registers (MAWH and MAWL)	351		
lock address and slave status , transmission of	373		
lock address, reading the	406		
lock read registers (LRRH and LRRL)	365		
locking and unlocking	406		

INDEX

mode is changed, PWCR and timer values when	213
mode pin setting and corresponding modes	129
models available	5
multiple interrupts	60
multiple-byte data, access of	32

N

N (negative flag)	38
NCC	48
negative flag (N)	38
notes on the reception of control bits from master unit	372

O

OCCP0 and OCCP1	225
OCS0 to OCS1	226
ODR4	156
one-shot operation mode	202
operating mode	128
operations	178
operation mode selection	198
operation of communication prescaler	318
operation of D/A converter	314
oscillation clock frequency	460
output compare (x 2)	216
output pin function	249
outside dimensions of FPT-100P-M05 package	7
outside dimensions of FPT-100P-M06 package	8
overflow flag (V)	39
overview of IEBus™ protocol operation	390

P

PACSR	415
PADR0 and PADR1	415
parity bit	400
pause mode	300
pause mode, example of EI ² OS activation in	305
PC	41
PCB, program bank register	30
PDRx	153
peripheral devices connected externally when DTP is used, Conditions of	281
pin functions	11
pin layout of FTP-100P-M05	9
pin layout of FTP-100P-M06	10
port 4 output pin register (ODR4)	156
port data direction register (DDRx)	154
port data register (PDRx)	153

ports 5 analog input enable register (ADER)	158
PPG output operation	267
PPG0 operation mode control register (PPGC0)	256
PPG0/1 output pin control register (PPGOE)	261
PPG1 operation mode control register (PPGC1)	258
PPGOE	261
precautions on the handling of the device	22
prefix codes, in the case of consecutive	50
prefix instructions, restrictions on	49
PRL/PRLH	263
processor status (PS)	38
program address detection control status register (PACSR)	415
program address detection registers (PADR0 and PADR1)	415
program bank register, PCB	30
program counter (PC)	41
program patch processing, example of	419
PS	38
pseudo watch mode, releasing	103
pseudo watch mode, transition to	103
pulse output on pins, controlling	270
pulse width, relationship between the reloaded value and	268
pulse width/period measurement range	207
pulse-width measurement and starting and stopping timer	200
pulse-width measurement function	195
pulse-width measurement mode using continuous measurement mode	214
pulse-width measurement operation, flowchart of	211
PWC control status register (PWCSR)	184
PWC data buffer register (PWCR)	190
PWC noise filter register (RNCR)	192
PWC timer block diagram	181
PWC timer operation	180
PWC timer registers	182
PWC timer, characteristics of	180
PWCR	190
PWCR and timer values when the mode is changed	213
PWCSR	184
PWCSR, STRT and STOP bits of	212

R

RDB	369
RDR0, RDR1 and RDR6	157

read data buffer (RDB).....	369	serial input data register (SIDR0 to 4), configuration of.....	329
read, flash memory	443	serial mode register (SMR0 to 4).....	323
reading slave status (SSR)	404	serial output data register (SODR0 to 4)	329
reading the lock address.....	406	serial programming connection (when power is supplied from a programmer), example of	464
ready function	144	serial programming connection (when user power supply is used), example of.....	462
receiver operation	336	serial status register (SSR0 to 4).....	330
reception of control bits from master unit, notes on	372	setting the oscillation stabilization time for the main clock	108
register banks	45	SIDR0 to 4	329
register bank pointer (RP)	39	single measurement mode	205
register of IEBus™ controller	348	single measurement mode and continuous measurement mode	205
register value change.....	212	single mode	299
releasing the pseudo watch mode	103	single mode, example of EI ² OS activation in	301
reload operation mode.....	202	slave address field.....	396
reload registers (PRL/PRLH)	263	slave address set registers (SAWH and SAWL)	352
reload registers, write timing for.....	271	slave data transmission routine	382
reloaded value and pulse width, relationship between.....	268	slave reception	374
request level setting register (ELVR)	277	slave reception (occurrence of reception interrupt)	377
reset causes.....	85	slave status and lock Address, transmission of	373
reset input, registers not initialized by	88	slave transmission	372
reset is released, operation after	87	sleep mode, releasing	102
reset status, flash memory	443	sleep mode, transition to	102
restart during operation.....	213	SMR0 to 4.....	323
RNCR.....	192	SODR0 to 4	329
ROM mirror function selection module, block diagram of	422	software interrupts, notes on	67
ROM mirror function selection register (ROMM)	423	software interrupts, operation of	66
RP	39	software interrupt, overview of.....	66
S		software interrupt, structure of.....	66
S.....	38	SSB, system stack bank register.....	30
saving a register to the stack	61	SSP	37
SAWH and SAWL	352	SSR	404
SCR0 to 4	326	SSR0 to 4	330
sector configuration of the 1M-bit flash memory	427	stack flag (S).....	38
sector deletion from the flash memory, temporarily stopping	449	standby state, return from.....	281
sector deletion timer flag (DQ3)	439	start bit.....	394
Sector Deletion, flash memory from which any data item is deleted.....	447	starting and stopping timer and pulse-width measurement	200
sector deletion, restarting the flash memory	450	status register (higher 8 bits) (STRH)	361
sector from the flash memory, procedure for deleting	447	status register (lower 8 bits) (STRL)	363
serial clock input frequency.....	460	status transition	113
serial control register (SCR0 to 4).....	326	status transition in the low-power consumption mode (one clock system)	124

INDEX

status transition in the low power-consumption mode (two clocks system)	119
sticky bit flag (T)	38
stop mode, releasing	105
stop mode, transition to	105
STRH	361
STRL	363
STRT and STOP bits of PWCSR	212
Structure of Instruction Map	519
successive mode	299
successive mode, example of EI ² OS activation in	303
switching between the main and PLL clocks	109
switching between the main clock and subclock	109
system stack bank register (SSB)	30
system stack pointer (SSP)	37

T

T	38
TBTC	162
TCDTH and TCDTL	221
text length bit set register (DEWR)	355
text length bit read register (lower 8 bit) (DERR)	368
text length field	398
the device, precaution on the handling of	22
time-based timer block diagram	161
time-based timer control register (TBTC)	162
time-based timer operations	164
time-based timer register	160
timer clear	213
timer control status register (TCCS)	222
timer control status register (TMCSR)	242
timer data register (TCDTH and TCDTL)	221
timer function	194
timer mode operation, flowchart of	204
timer mode, measurement termination flag in	212
timer period	203
timer value and reload value	202
timing limit excess flag (DQ5)	438
TMCSR	242
TMR, 16-bit timer register	245
TMRLR, 16-bit reload register	245
toggle bit 2 flag (DQ2)	440
toggle bit flag (DQ6)	436
transfer data format	336, 338
transferring data or commands	405
transition to the pseudo watch mode	103

transmission data	403
transmission of slave status and lock Address	373
transmission operation at occurrence of error	388
transmission operation over multiple frames	386
transmission protocol	393
transmitter operation	337
two interrupt sources	340

U

UART (during operation in mode 1), application of	343
UART block diagram	321
UART operations	333
UART register	322
UART, feature of	320
undefined instruction, occurrence of exceptions because of executing	81
underflow operation	247
USB, user stack bank register	30
user power supply is used, example of minimum connection with flash microcomputer programmer	466
user power supply is used, example of serial programming connection	462
user stack bank register, USB	30
user stack pointer (USP)	37
USP	37

V

V	39
values set in register in the CLK-synchronous mode	339

W

watch mode, releasing	104
watch mode, transition to	104
watch timer block diagram	175
watch timer control register	174
watch timer control register (WTC)	176
watchdog timer block diagram	167
watchdog timer control register (WDTC)	168
watchdog timer register	166
watchdog timer reset, preventing	171
watchdog timer, activating	171
watchdog timer, clearing	171
watchdog, stopping	171
WDB	370
WDTC	168
write data buffer (WDB)	370

writing data for flash memory	426	Z	
writing data in flash memory, procedure for	444	Z	39
WTC	176	zero flag (Z)	39

CM44-10111-3E

FUJITSU MICROELECTRONICS • CONTROLLER MANUAL

F²MC™-16LX

16-BIT MICROCONTROLLER

MB90580C Series

HARDWARE MANUAL

July 2008 the 3rd edition

Published **FUJITSU MICROELECTRONICS LIMITED**

Edited Business & Media Promotion Dept.
