



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.

F²MC-16LX
16-BIT MICROCONTROLLER
MB90550A/B Series
HARDWARE MANUAL

F²MC-16LX

16-BIT MICROCONTROLLER

MB90550A/B 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 Readership

Thank you for your continued preference for Fujitsu semiconductor products.

The MB90550A/B 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 MB90550A/B Series.

Note: F²MC is the abbreviation of FUJITSU Flexible Microcontroller.

■ Trademark

Embedded AlgorithmTM is a trademark of Advanced Micro Devices, Inc.

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

■ Licence

Purchase of Fujitsu I²C components conveys a license under the Philips I²C Patent Rights to use, these components in an I²C system provided that the system conforms to the I²C Standard Specification as defined by Philips.

■ Structure of This Manual

This manual consists of the following 24 chapters:

Chapter 1 Overview

This chapter gives an overview of the features and basic specification of the MB90550A/B Series.

Chapter 2 CPU

This chapter describes the internal configuration of the CPU of the F²MC-16LX Series and provides a specification of the hardware incorporated in the MB90550A/B Series.

Chapter 3 Interrupts

This chapter describes the functions and operation of interrupts.

Chapter 4 Generating and Resetting Clocks

This chapter describes the functions and operation for generating clock signals and resetting clocks.

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 internal and the external memory access mode of the F²MC-16LX Series.

Chapter 7 I/O Ports

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

Chapter 8 Time-Based Timer

This chapter describes the functions and operation of the time-base timer.

Chapter 9 Watchdog Timer

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

Chapter 10 16-Bit I/O Timer

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

Chapter 11 16-Bit Reload Timer (with the Event Count Function)

This chapter describes the functions and operation of the 16-bit reload timer containing the event count function.

Chapter 12 8/16-Bit PPG

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

Chapter 13 DTP/External Interrupt

This chapter describes the functions and operation of the DTP/external interrupt circuit and lists cautions in connection with its use.

Chapter 14 Delayed Interrupt Generating Module

This chapter describes the functions and operation of the delayed interrupt generation module and lists cautions in connection with its use.

Chapter 15 A/D Converter

This chapter describes the functions and operation of the A/D converter and lists cautions in connection with its use.

Chapter 16 Communication Prescaler Register

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

Chapter 17 UART

This chapter describes the functions and operation of the UART, its application, and lists cautions in connection with its use.

Chapter 18 I/O Extended Serial Interface

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

Chapter 19 I²C Interface

This chapter describes the functions and operation of the I²C interface.

Chapter 20 Clock Monitor Function

This chapter describes the clock monitor function.

Chapter 21 Address Match Detection Function

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

Chapter 22 ROM Mirror Function Selection Module

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

Chapter 23 1M-Bit Flash Memory

This chapter describes the functions and operation of the 1 Mb flash memory.

Chapter 24 Example of MB90F553A Serial Programming Connection

This chapter provides examples of MB90F553A serial programming connections.

Appendix

The appendix lists the I/O map, instructions, OTPROM programming, and cautions in connection with reset.

- 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	Available Models	5
1.3	Block Diagram	6
1.4	External Dimensions of the Package	7
1.5	Pin Assignment	9
1.6	Description of the Pin Functions	11
1.7	I/O Circuit Types	17
1.8	Cautions on Handling Devices	21
CHAPTER 2	CPU	25
2.1	Memory Space	26
2.2	Addressing	27
2.3	Allocating Multiple-byte Data in a Memory Space	30
2.4	Dedicated Registers	31
2.4.1	Accumulator (A)	33
2.4.2	User Stack Pointer (USP) and System Stack Pointer (SSP)	35
2.4.3	Processor Status (PS)	36
2.4.4	Program Counter (PC)	39
2.4.5	Direct Page Register (DPR)	40
2.4.6	Bank registers (PCB, DTB, USB, SSB, ADB)	41
2.5	General-purpose Registers	42
2.6	Prefix Codes	44
2.7	Interrupt Suppression Instructions and Prefix Codes	46
2.8	Notes on Using the "DIV A, Ri" and "DIVW A, RWi" Instructions	47
CHAPTER 3	INTERRUPTS	51
3.1	Overview of Interrupts	52
3.2	Interrupt Causes	53
3.3	Interrupt Vectors	56
3.4	Hardware Interrupts	58
3.4.1	Operation of Hardware Interrupts	61
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	80

CHAPTER 4	GENERATING AND RESETTING CLOCKS	81
4.1	Clock Generator	82
4.2	Clock Supply Map	83
4.3	Reset Causes	84
4.4	Operation after a Reset is Released	86
4.5	Registers not Initialized by Reset Input	87
CHAPTER 5	LOW-POWER CONSUMPTION CONTROL CIRCUIT	89
5.1	Overview of the Low-power Consumption Control Circuit	90
5.2	Low-power Consumption Mode Control Register (LPMCR)	93
5.3	Clock Selection Register (CKSCR)	95
5.4	Operation of the Low-power Consumption Control Circuit	98
5.4.1	Sleep Mode	100
5.4.2	Watch Mode	101
5.4.3	Stop Mode	103
5.4.4	Hardware Standby Mode	104
5.4.5	Pin status in the Sleep, Stop, Hold, Reset, and Hardware Standby Modes	105
5.5	Intermittent CPU Operation Function	108
5.6	Setting the Oscillation Stabilization Time	109
5.7	Machine Clock	110
CHAPTER 6	MEMORY ACCESS MODES	113
6.1	Memory Access Mode Overview	114
6.1.1	Mode Pins	115
6.1.2	Mode Data	116
6.1.3	Memory Space for Each Bus Mode	117
6.2	External Memory Access (External Bus Pin Control Circuit)	120
6.2.1	Registers for External Memory Access (External Bus Pin Control Circuit)	121
6.2.2	Automatic Ready Function Selection Register (ARSR)	122
6.2.3	External Address Output Control Register (HACR)	124
6.2.4	Bus Control Signal Selection Register (ECSR)	125
6.3	Operation of the External Memory Access Control Signals	128
6.3.1	Ready Function	130
6.3.2	Hold Function	132
CHAPTER 7	I/O PORTS	133
7.1	I/O Port Overview	134
7.2	I/O Port Block Diagram	135
7.3	I/O Port Registers	138
7.3.1	Port Data Registers (PDRx)	140
7.3.2	Port Data Direction Registers (DDRx)	142
7.3.3	Output Pin Register (ODR4)	143
7.3.4	Input Resistor Registers (RDR0 and RDR1)	144
7.3.5	Analog Input Enable Register (ADER)	145

CHAPTER 8	TIME-BASED TIMER	147
8.1	Overview of the Time-Based Timer	148
8.2	Time-Based Timer Control Register (TBTC)	149
8.3	Time-Based Timer Operations	151
CHAPTER 9	WATCHDOG TIMER	153
9.1	Overview of the Watchdog Timer	154
9.2	Watchdog Timer Control Register (WDTC)	155
9.3	Watchdog Timer Operations	157
CHAPTER 10	16-BIT I/O TIMER	159
10.1	Overview of the 16-Bit I/O Timer	160
10.2	16-Bit I/O Timer Block Diagram	162
10.3	16-Bit I/O Timer Registers	163
10.3.1	16-bit Free-run Timer	165
10.3.2	Output Compare	168
10.3.3	Input Capture	172
10.4	16-Bit Free-Run Timer Operations	174
10.5	16-Bit Output Compare Operations	176
10.6	16-Bit Input Capture Operations	179
CHAPTER 11	16-BIT RELOAD TIMER (WITH THE EVENT COUNT FUNCTION)	181
11.1	Overview of the 16-Bit Reload Timer (with the Event Count Function)	182
11.2	Registers of the 16-Bit Reload Timer (with the Event Count Function)	183
11.2.1	Timer Control Status Register (TMCSR)	184
11.2.2	16-Bit Timer Register (TMR) and 16-Bit Reload Register (TMRLR)	187
11.3	Clock Operations	188
11.4	Underflow Operation	189
11.5	I/O Pin Functions	190
11.6	Counter Operation Statuses	192
CHAPTER 12	8/16-BIT PPG	193
12.1	Overview of the 8/16-Bit PPG	194
12.2	Block Diagrams of the 8-Bit PPG	195
12.3	Registers in the 8/16-Bit PPG	197
12.3.1	PPG0 Operation Mode Control Register (PPGC0)	198
12.3.2	PPG1 Operation Mode Control Register (PPGC1)	200
12.3.3	PPG0/1 Output Pin Control Register (PPGE)	203
12.3.4	Reload Registers (PRL/PRLH)	205
12.4	8/16-Bit PPG Operation	206
12.4.1	8/16-bit PPG Operation Modes	208
12.4.2	PPG Output Operation	209
12.4.3	Selecting a Count Clock	211
12.4.4	Controlling Pulse Output on Pins	212
12.4.5	Write Timing for the Reload Registers	213

CHAPTER 13 DTP/EXTERNAL INTERRUPT	215
13.1 Overview of the DTP/External Interrupt Circuit	216
13.2 Registers in the DTP/External Interrupt Circuit	218
13.3 Operation of DTP/External Interrupt Circuit	221
13.4 Notes on Using the DTP/External Interrupt Circuit	224
CHAPTER 14 DELAYED INTERRUPT GENERATING MODULE	227
14.1 Outline of the Delayed Interrupt Generating Module	228
14.2 Operation of the Delayed Interrupt Generating Module	229
CHAPTER 15 A/D CONVERTER	231
15.1 Overview of the A/D Converter	232
15.2 Registers of the A/D Converter	235
15.2.1 Control Status Registers (ADCS0 and ADCS1)	236
15.2.2 Data Register (ADCR1 and ADCR0)	240
15.3 Operation of A/D Converter	242
15.3.1 Example of EI ² OS Activation in Single Mode	244
15.3.2 Example of EI ² OS Activation in Successive Mode	246
15.3.3 Example of EI ² OS Activation in Pause Mode	248
15.4 Conversion Data Protection Function	250
CHAPTER 16 COMMUNICATION PRESCALER REGISTER	253
16.1 Overview of Communication Prescaler Register	254
16.2 Operation of Communication Prescaler Register	256
CHAPTER 17 UART	257
17.1 Overview of UART	258
17.2 UART Block Diagram	259
17.3 UART Registers	260
17.3.1 Serial Mode Register (SMR)	261
17.3.2 Serial Control Register (SCR)	263
17.3.3 Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)	266
17.3.4 Serial Status Register (SSR)	267
17.4 UART Operations	270
17.4.1 UART Clock Selection	271
17.4.2 Asynchronous (Start-stop Synchronous) Mode	273
17.4.3 CLK-synchronous Mode	275
17.4.4 Occurrence of Interrupt and Flag Setting Timing	277
17.5 Application of UART (During Operation in Mode 1)	280
CHAPTER 18 I/O EXTENDED SERIAL INTERFACE	283
18.1 Overview of the I/O Extended Serial Interface	284
18.2 Registers of the I/O Extended Serial Interface	286
18.2.1 Serial Mode Control Status Register (SMCS)	287
18.2.2 Serial Shift Data Register (SDR)	291
18.3 Operation of I/O Extended Serial Interface	292
18.3.1 Shift Clock	293

18.3.2	Operation States of the Serial I/O	295
18.3.3	Start/Stop Timing Of Shift Operation and Input/Output Timing	297
18.3.4	Interrupt Function of the I/O Extended Serial Interface	300
CHAPTER 19	I²C INTERFACE	301
19.1	Overview of the I ² C Interface	302
19.2	Block Diagram and Structure of the I ² C Interface	303
19.3	Registers of the I ² C Interface	305
19.3.1	Bus Status Register (IBSR)	306
19.3.2	Bus Control Register (IBCR)	309
19.3.3	Clock Control Register (ICCR)	312
19.3.4	Address Register (IADR)	314
19.3.5	Data Register (IDAR)	315
19.3.6	Port Selection Register (ISEL)	316
19.4	Operation of the I ² C Interface	317
19.4.1	Flow of the I ² C Interface Transmission	319
19.4.2	Flow of the I ² C Interface Modes	321
CHAPTER 20	CLOCK MONITOR FUNCTION	323
20.1	Overview of the Clock Monitor Functions	324
20.2	Clock Output Permission Register (CLKR)	325
CHAPTER 21	ADDRESS MATCH DETECTION FUNCTION	327
21.1	Overview of the Address Match Detection Function	328
21.2	Registers of the Address Match Detection Function	329
21.3	Operation of the Address Match Detection Function	331
21.4	Example of the Address Match Detection Function	332
21.5	Program Example of the Address Match Detection Function	334
CHAPTER 22	ROM MIRROR FUNCTION SELECTION MODULE	337
22.1	Overview of the ROM Mirror Function Selection Module	338
22.2	ROM Mirror Function Selection Register (ROMM)	339
CHAPTER 23	1M-BIT FLASH MEMORY	341
23.1	Overview of the 1M-Bit Flash Memory	342
23.2	Block Diagram and Sector Configuration of the Entire Flash Memory	343
23.3	Writing and Deletion Modes	345
23.4	Flash Memory Control Status Register (FMCS)	347
23.5	Activating the Automatic Algorithm of the Flash Memory	349
23.6	Confirming the Automatic Algorithm Execution Status	350
23.6.1	Data Polling Flag (DQ7)	352
23.6.2	Toggle Bit Flag (DQ6)	354
23.6.3	Timing Limit Excess Flag (DQ5)	355
23.6.4	Sector Deletion Timer Flag (DQ3)	356
23.7	Detailed Explanations of Flash Memory Writing and Deletion	357
23.7.1	Setting the Flash Memory in the Read or Reset Status	358
23.7.2	Writing Data in the Flash Memory	359

23.7.3	Deleting all Data Items from the Flash Memory (Chip Deletion)	361
23.7.4	Deleting any Data Item from the Flash Memory (Sector Deletion)	362
23.7.5	Temporarily Stopping the Sector Deletion from the Flash Memory	364
23.7.6	Restarting the Flash Memory Sector Deletion	365
23.8	Example of the 1M-Bit Flash Memory Program	366

CHAPTER 24 EXAMPLE OF MB90F553A SERIAL PROGRAMMING CONNECTION

.....		371
24.1	Basic Configuration of MB90F553A Serial Programming Connection	372
24.2	Example of Serial Programming Connection (When User Power Supply Is Used)	375
24.3	Example of Serial Programming Connection (When Power Is Supplied from a Writer)	377
24.4	Example of Minimal Connection with the Flash Microcontroller Programmer (When User Power Supply Is Used)	379
24.5	Example of Minimal Connection with the Flash Microcontroller Programmer (When Power Is Supplied from a Writer)	381

APPENDIX **383**

APPENDIX A	I/O MAP	384
APPENDIX B	Instructions	392
B.1	Instruction Types	393
B.2	Addressing	394
B.3	Direct Addressing	396
B.4	Indirect Addressing	402
B.5	Execution Cycle Count	410
B.6	Effective address field	413
B.7	How to Read the Instruction List	414
B.8	F ² MC-16LX Instruction List	417
B.9	Instruction Map	431
APPENDIX C	Programming the OTPROM	453

INDEX..... **455**

Main changes in this edition

Page	Changes (For details, refer to main body.)	
392 to 452	APPENDIX B Instructions	Changed the entire part of "APPENDIX B Instructions"

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

Reference: Main changes (Rev.4 → Rev.5)

Page	Changes (For details, refer to main body.)	
-	-	Pin names were changed. (TOUT→ TOT)
24	CHAPTER 1 OVERVIEW 1.8 Cautions on Handling Devices	"○ Notes on operation in PLL clock mode" was changed.
105	CHAPTER 5 LOW-POWER CONSUMPTION CONTROL CIRCUIT 5.4.5 Pin status in the Sleep, Stop, Hold, Reset, and Hardware Standby Modes	"Table 5.4-2 Status of Each Pin in the Single Chip Mode" was changed.
162	CHAPTER 10 16-BIT I/O TIMER 10.2 16-Bit I/O Timer Block Diagram	"Figure 10.2-1 16-bit I/O Timer Block Diagram" was changed.
167	CHAPTER 10 16-BIT I/O TIMER 10.3.1 16-bit Free-run Timer	"[Bit 2] CLR" was changed.
218	CHAPTER 13 DTP/EXTERNAL INTERRUPT 13.2 Registers in the DTP/External Interrupt Circuit	"Note:" of "■ Interrupt/DTP Enable Register (ENIR)" was added.
219		"Notes:" of "■ Interrupt/DTP Source Register (EIRR)" was changed.
236	CHAPTER 15 A/D CONVERTER 15.2.1 Control Status Registers (ADCS0 and ADCS1)	"Note:" was changed.
		"[Bit 14] INT (Interrupt)" was changed.
243	CHAPTER 15 A/D CONVERTER 15.3 Operation of A/D Converter	"Figure 15.3-1 Example of Flow from Activation of A/D Conversion to Conversion Data Transfer (Successive Mode)" was changed.
264	CHAPTER 17 UART 17.3.2 Serial Control Register (SCR)	"[Bit 10] REC (Receiver error clear)" was changed.
287	CHAPTER 18 I/O EXTENDED SERIAL INTERFACE 18.2.1 Serial Mode Control Status Register (SMCS)	"Figure 18.2-2 Serial Mode Control Status Register (SMCS)" was changed.
306	CHAPTER 19 I ² C INTERFACE 19.3.1 Bus Status Register (IBSR)	"[Bit 6] RSC (Repeated start condition)" was changed.

Reference: Main changes (Rev.4 → Rev.5)

342	CHAPTER 23 1M-BIT FLASH MEMORY 23.1 Overview of the 1M-Bit Flash Memory	"■ Features of the 1M-bit Flash Memory" was changed.
429	APPENDIX B Instructions B.8 F ² MC-16LX Instruction List	"Table B.8-17 6 Accumulator Operation Instructions (Byte, Word)" was changed.

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

CHAPTER 1 OVERVIEW

This chapter provides an overview of the MB90550A/B Series.

- 1.1 Features
- 1.2 Available Models
- 1.3 Block Diagram
- 1.4 External Dimensions of the Package
- 1.5 Pin Assignment
- 1.6 Description of the Pin Functions
- 1.7 I/O Circuit Types
- 1.8 Cautions on Handling Devices

1.1 Features

The MB90550A/B Series is a general-purpose high-performance 16-bit microcontroller that was designed for devices used in various industries, OA devices, and devices for process control requiring high-speed real-time processing. In addition to inheriting the F²MC-8 Series AT architecture, the MB90550A/B Series instruction system adds support of high-level language instructions, expanded addressing modes, enhanced multiplication and division instructions, and provides substantial bit processing capabilities. By employing a 32-bit accumulator, this system also enables long-word data processing.

■ Features

The features of the MB90550A/B Series are shown below.

○ Minimum time for instruction execution

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

○ Maximum memory space

16 MB

○ Instruction system optimized for the controller

Data types that can be handled: bit/bytes/read/long word

Standard addressing modes: 23 types

Enhanced high-precision arithmetic operations by using a 32-bit accumulator

Enhanced signed multiplication and division instructions as well as an enhancement of the functions for the reti command.

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

- Use of the system stack pointer
- Symmetrical instruction set and barrel shift instructions

○ Incorporation of an address match detection function (for two address pointers)

○ Improved execution speed

4-byte queue

○ Powerful interrupt functions

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

- **Data transfer function**
 - Intelligent I/O services: Up to 16 channels
 - DTP request input: 8 channels
- **Internal ROM**
 - EPROM version, Flash version: 128 KB
 - Mask ROM version: 64 KB/128 KB
- **Internal RAM**
 - EPROM version, Flash version: 4 KB
 - Mask ROM version: 2 KB/4 KB
- **General-purpose ports**

Up to 83 ports (allowing input pull-up register setting: 16 ports, allowing open-drain setting: 8 ports, I/O open-drain: 6 ports)
- **A/D converter**
 - RC sequential approximation type): 8 channels
 - Resolution: 8 or 10 bits selectable, Conversion time 26.3 μ s [minimum]
- **UART: 1 channel**
- **Extended I/O serial interface: 2 channels**
- **I²C interface: 2 channels**

One of the two channels can be switched between terminal input/output. [For two systems]
- **16 bit reload timer: 2 channels**
- **8/16-bit PPG: 3 channels**

With function for switching between 8 bits \times 2 channels and 16 bits \times 1 channel mode
- **16-bit I/O timer configuration**
 - Input capture \times 4 channels
 - Output compare \times 4 channels
 - Free-run timer \times 1 channel
- **Incorporation of a clock monitor function**

Outputs a clock signal with segments of 2^1 to 2^8
- **Time-base and watchdog timer: 18 bits**

- **Low-power consumption mode**
 - Sleep
 - Stop
 - Hardware standby mode
 - CPU intermittent operation mode function
- **Package**
 - QFP-100
 - LQFP-100
- **CMOS technology**
- **Reduction in radiation noise (MB90550B Series)**

1.2 Available Models

Table 1.2-1 shows the models of the MB90550A/B Series. Characteristics other than the ROM/RAM sizes are common to all models.

■ Available Models

Table 1.2-1 Models of the MB90550A/B Series

Model name	ROM size	RAM size	Remark
MB90V550A	-	6Kbytes	Device for evaluation
MB90P553A	128Kbytes	4Kbytes	OTPROM
MB90F553A	128Kbytes	4Kbytes	Flash ROM
MB90T553A	-	4Kbytes	External ROM
MB90553A/B	128Kbytes	4Kbytes	Mask ROM
MB90T552A	-	2Kbytes	External ROM
MB90552A/B	64Kbytes	2Kbytes	Mask ROM

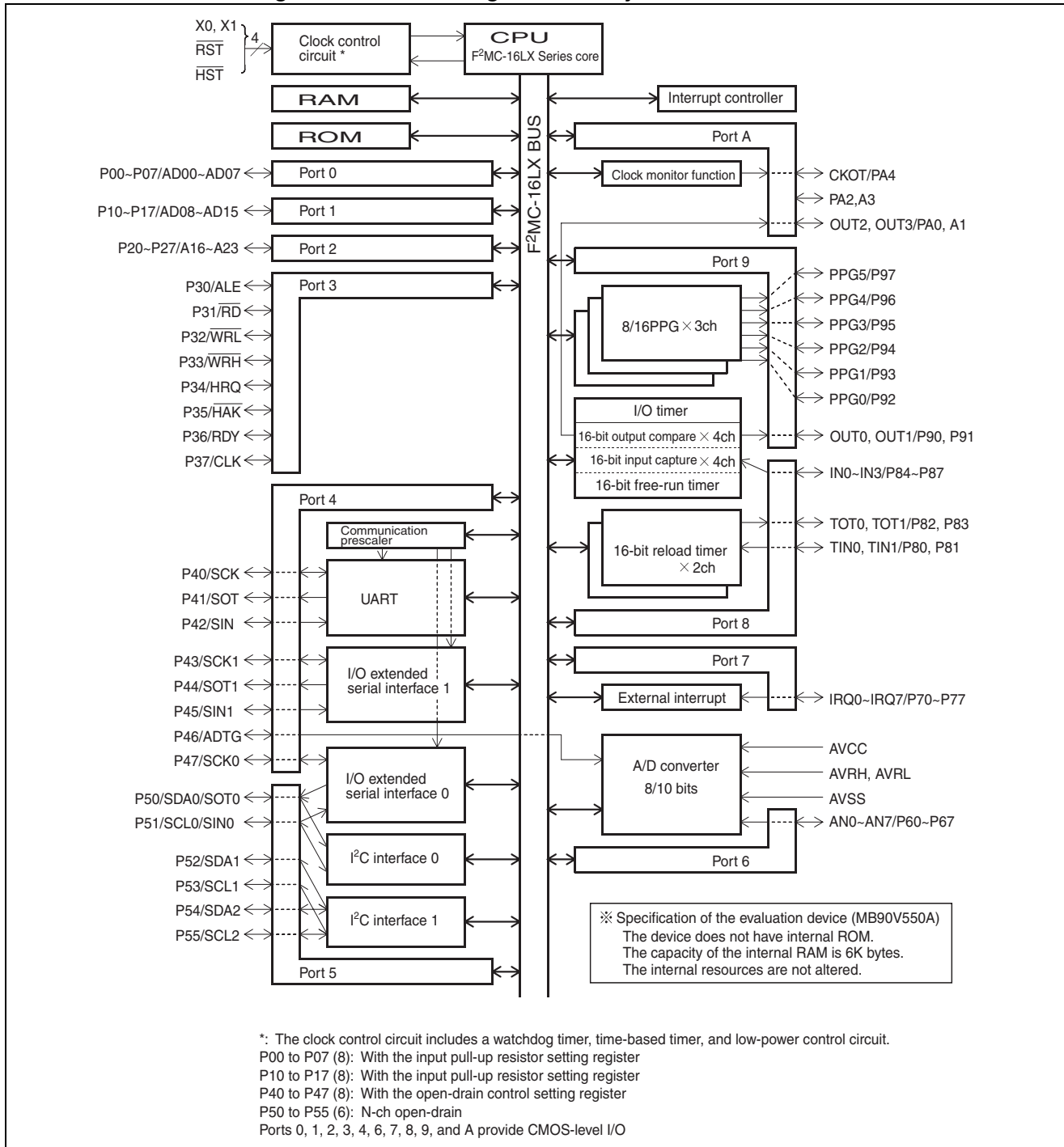
- Purchase of Fujitsu I²C components conveys a license under the Philips I²C Patent Rights to use, these components in an I²C system provided that the system conforms to the I²C Standard Specification as defined by Philips.

1.3 Block Diagram

Figure 1.3-1 shows a block diagram of the system architecture.

■ Block Diagram of the System Architecture

Figure 1.3-1 Block Diagram of the System Architecture



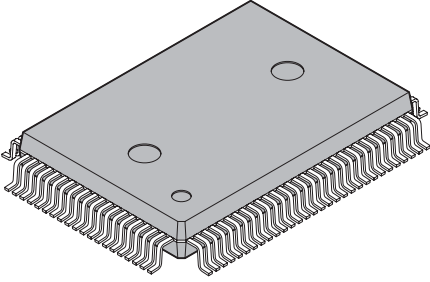
1.4 External Dimensions of the Package

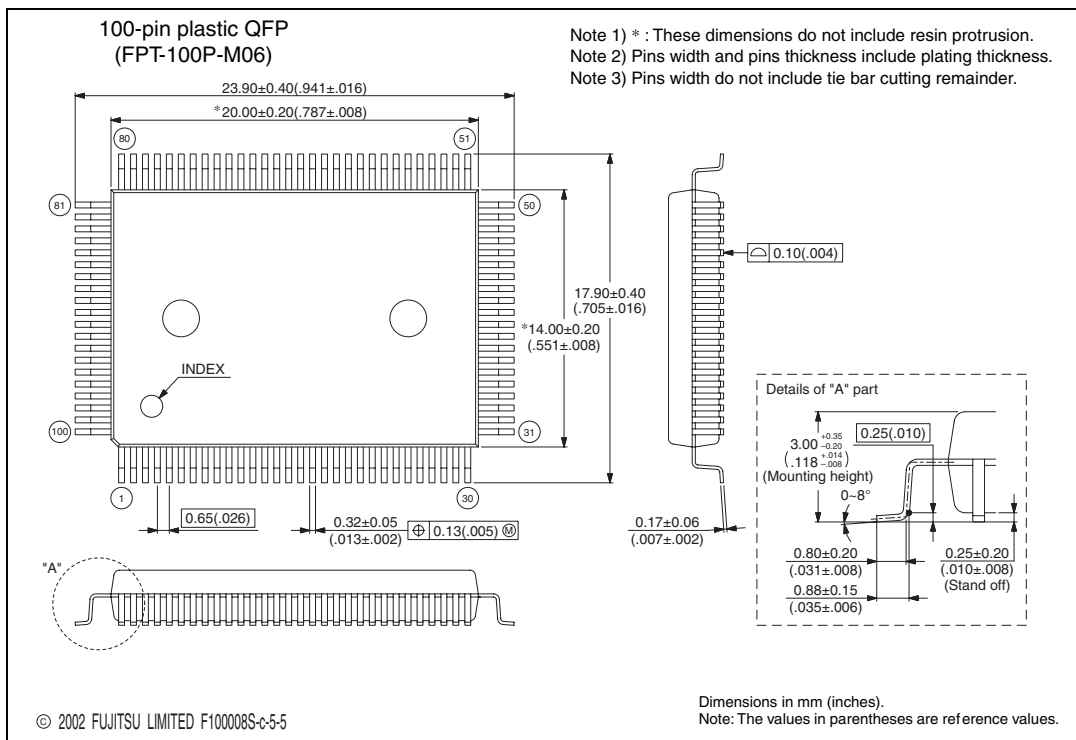
Figure 1.4-1 shows the external dimensions of the QFP-100 package and Figure 1.4-2 shows those of the LQFP-100 package.

Note that these dimensions are for reference purposes only. Contact us for the official values, if required.

■ External Dimensions of the FPT-100P-M06 Package

Figure 1.4-1 QFP-100 External Dimensions

 <p>100-pin plastic QFP</p> <p>(FPT-100P-M06)</p>	Lead pitch	0.65 mm
	Package width × package length	14.00 × 20.00 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	3.35 mm MAX
	Code (Reference)	P-QFP100-14×20-0.65

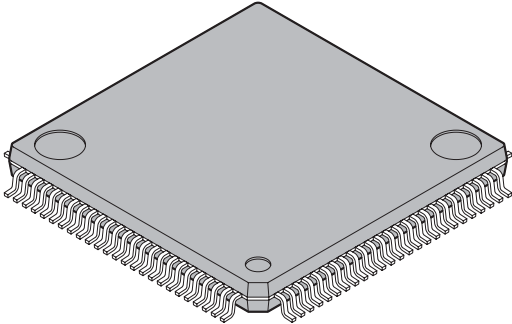


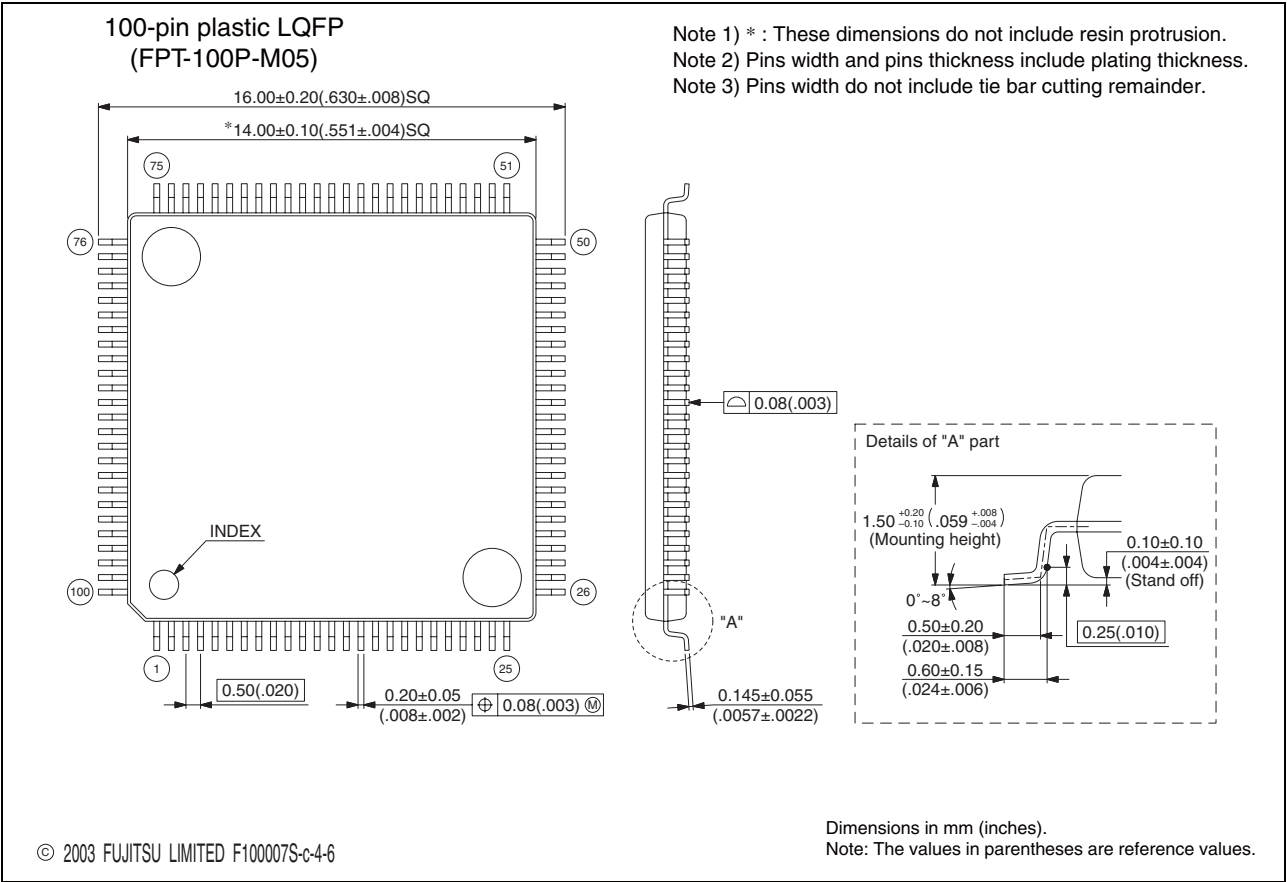
Please confirm the latest Package dimension by following URL.

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

■ External Dimensions of the FPT-100P-M05 Package

Figure 1.4-2 LQFP-100 External Dimensions

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



Please confirm the latest Package dimension by following URL.

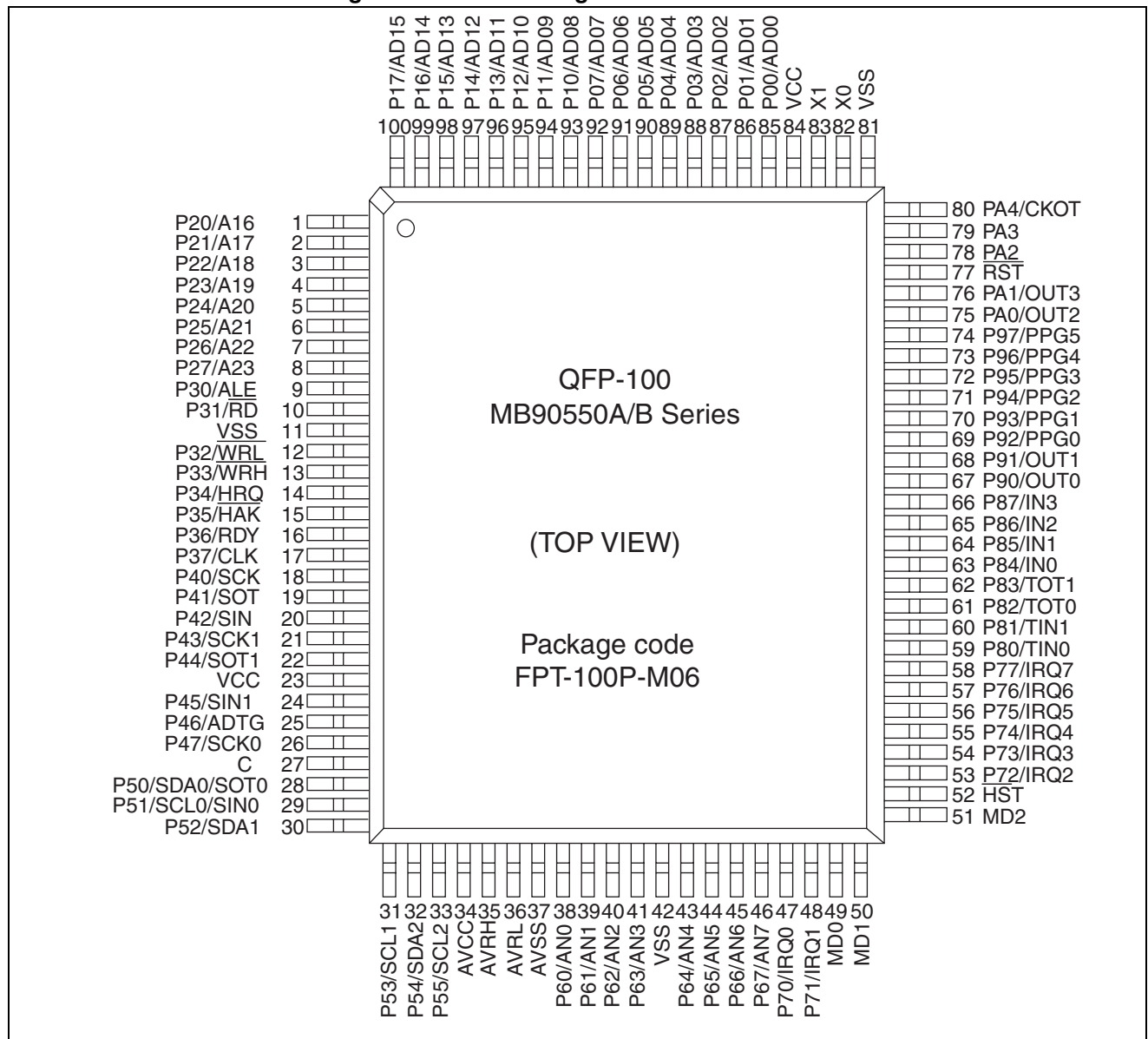
<http://edevic.fujitsu.com/fj/DATASHEET/ef-ovpklv.html>

1.5 Pin Assignment

Figure 1.5-1 shows the pin arrangement of the QFP-100 and Figure 1.5-2 shows that of the LQFP-100.

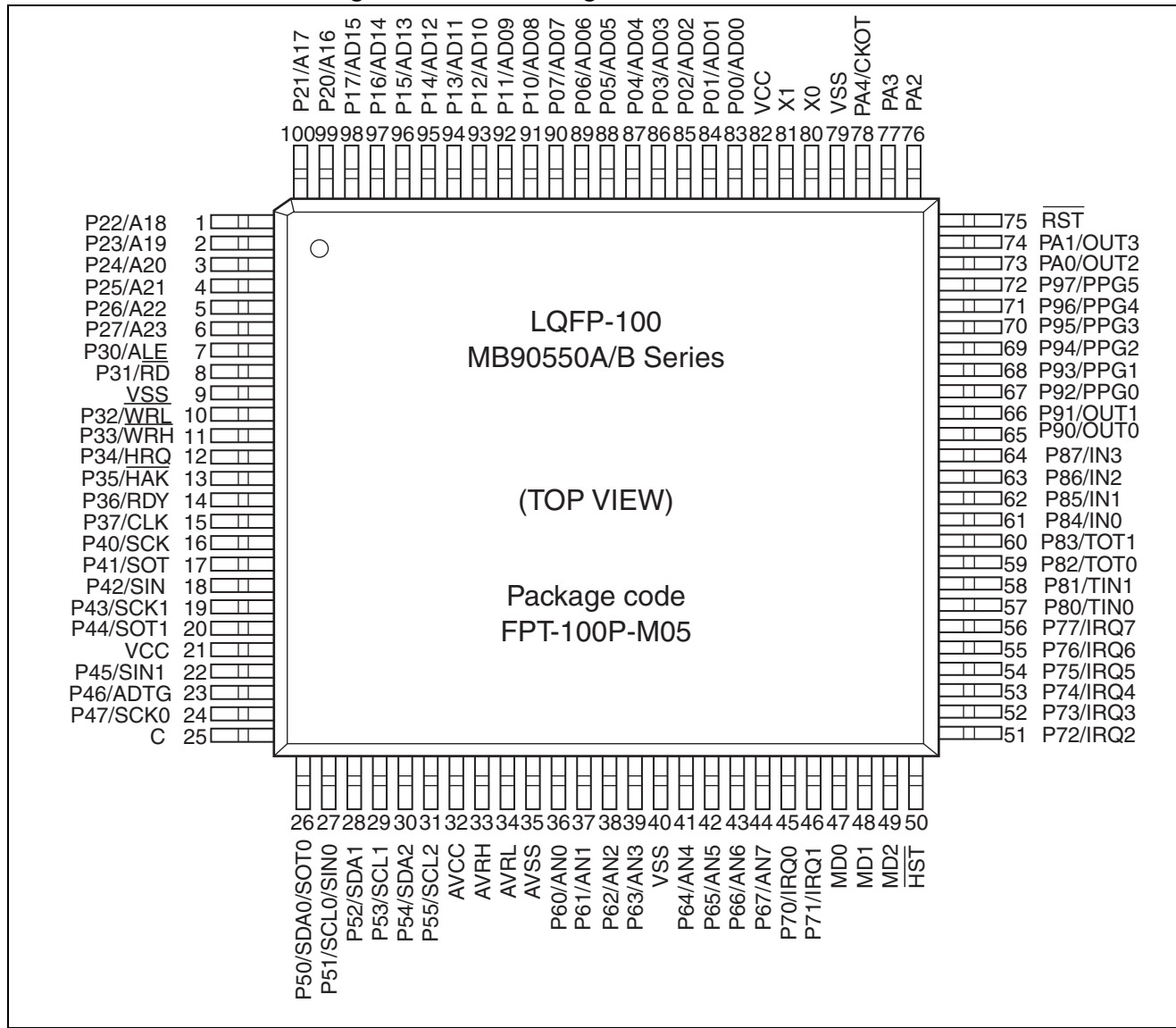
■ Pin Arrangement of the FTP-100P-M06

Figure 1.5-1 Pin Arrangement of the QFP-100



■ Pin Arrangement of the FTP-100P-M05

Figure 1.5-2 Pin Arrangement of the LQFP-100



1.6 Description of the Pin Functions

Table 1.6-1 lists the functions of the pins.

■ Description of the Pin Functions

Table 1.6-1 Description of the Pin Functions (1/6)

QFP	LQFP	Pin name	Circuit type	Description of function
82	80	X0	A	Oscillation pin
83	81	X1	A	Oscillation pin
77	75	$\overline{\text{RST}}$	B	Reset input pin
52	50	$\overline{\text{HST}}$	C	Hardware standby input pin
85 to 92	83 to 90	P00 to P07	D (CMOS)	General-purpose input/output port. A pull-up resistor can be added (RD07 to RD00 = 1) depending on the pull-up resistor setting register (RDR0). D07 to D00 = 1: Invalid during output setup
		AD00 to AD07		Act as lower data output and lower address I/O (AD00 to AD07) pins in external bus mode.
93 to 100	91 to 98	P10 to P17	D (CMOS)	General-purpose input/output port. A pull-up resistor can be added (RD17 to RD10 = 1) depending on the pull-up resistor setting register (RDR1). D17 to D10 = 1: Invalid during output setup
		AD08 to AD15		Act as the higher data I/O and middle address output (AD08 to AD15) pins in 16-bit external bus mode.
1 to 8	99, 100, 1 to 6	P20 to P27	E (CMOS)	General-purpose input/output port This function is enabled in single-chip mode or external bus mode when the corresponding bit of the external address output control register (HACR) is "1".
		A16 to A23		Output pin for external address bus (A16 to A23) This function is enabled in external bus mode when the corresponding bit of the external address output control register (HACR) is "0".
9	7	P30	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		ALE		Address latch enable output pin. This function is enabled in the mode in which the external bus is valid.

Table 1.6-1 Description of the Pin Functions (2/6)

QFP	LQFP	Pin name	Circuit type	Description of function
10	8	P31	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		$\overline{\text{RD}}$		Read strobe output pin for the data bus. This function is enabled in the mode in which the external bus is valid.
12	10	P32	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		$\overline{\text{WRL}}$		Write strobe output pin for the lower 8 bits of the data bus. This function is enabled in the mode in which the external bus is valid.
13	11	P33	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		$\overline{\text{WRH}}$		Write strobe output pin for the lower 8 bits of the data bus. This function is enabled in the mode in which the external bus is valid.
14	12	P34	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		HRQ		Hold request input pin. This function is enabled in the mode in which the external bus is valid.
15	13	P35	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		$\overline{\text{HAK}}$		Hold acknowledge output pin. This function is enabled in the mode in which the external bus is valid.
16	14	P36	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		RDY		Ready input pin. This function is enabled in the mode in which the external bus is valid.
17	15	P37	E (CMOS)	General-purpose I/O port. This function is enabled in single-chip mode.
		CLK		CLK output pin. This function is enabled in the mode in which the external bus is valid.

Table 1.6-1 Description of the Pin Functions (3/6)

QFP	LQFP	Pin name	Circuit type	Description of function
18	16	P40	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD40 = 1) by the open-drain control setting register (ODR4)(D40 = 0: Invalid during input setup).
		SCK		UART serial clock I/O pin. This function is enabled when clock output specification of the UART is allowed.
19	17	P41	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD41 = 1) by the open-drain control setting register (ODR4)(D41 = 0: Invalid during input setup).
		SOT		UART serial data output pin. This function is enabled when data output specification of the UART is allowed.
20	18	P42	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD42 = 1) by the open-drain control setting register (ODR4)(D42 = 0: Invalid during input setup).
		SIN		UART serial data input pin. As this input pin can be used whenever the UART performs an input operation, be sure to use another function to stop the output unless for a special purpose.
21	19	P43	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD43 = 1) by the open-drain control setting register (ODR4)(D43 = 0: Invalid during input setup).
		SCK1		Extended I/O serial clock I/O pin. This function is enabled when extended I/O serial clock output is allowed.
22	20	P44	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD44 = 1) by the open-drain control setting register (ODR4)(D44 = 0: Invalid during input setup).
		SOT1		Extended I/O serial data output pin. This function is enabled when extended I/O serial data output is allowed.
24	22	P45	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD45 = 1) by the open-drain control setting register (ODR4)(D45 = 0: Invalid during input setup).
		SIN1		Extended I/O serial data input pin. As this input pin can be used whenever an extended I/O serial input operation is performed, be sure to use another function to stop the output unless for a special purpose.

Table 1.6-1 Description of the Pin Functions (4/6)

QFP	LQFP	Pin name	Circuit type	Description of function
25	23	P46	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD46 = 1) by the open-drain control setting register (ODR4)(D46 = 0: Invalid during input setup).
		ADTG		Pin for A/D converter external trigger input. As this input pin can be used whenever an A/D converter external trigger input pin performs an input operation, be sure to use another function to stop the output unless for a special purpose.
26	24	P47	F (CMOS/H)	General-purpose I/O port. It is used as the open-drain output port (OD47 = 1) by the open-drain control setting register (ODR4)(D47 = 0: Invalid during input setup). D47 = 0: Disabled at input setting
		SCK0		Extended I/O serial clock I/O pin. This function is enabled when extended I/O serial clock output is allowed.
27	25	C		Power supply stabilization capacity pin. Connect externally a ceramic condenser of about 0.1 μ F.
28	26	P50	G (NchOD/H)	N-ch open-drain type I/O port.
		SDA0		Data I/O pin for an I ² C interface. This function is enabled when operation of an I ² C interface is allowed. For operation of an I ² C interface, set the port output to Hi-Z (PDR = 1).
		SOT0		Extended I/O serial data output pin. This function is enabled when extended I/O serial data output is allowed.
29	27	P51	G (NchOD/H)	N-ch open-drain type I/O port.
		SCL0		Clock I/O pin for an I ² C interface. This function is enabled when operation of an I ² C interface is allowed. For operation of an I ² C interface, set the port output to Hi-Z (PDR = 1).
		SIN0		Extended I/O serial data input pin. As this input pin can be used whenever an extended I/O serial input operation is performed, be sure to use another function to stop the output unless for a special purpose.
30,32	28,30	P52,P54	G (NchOD/H)	N-ch open-drain type I/O port.
		SDA1,SDA2		Data I/O pin for an I ² C interface. This function is enabled when operation of an I ² C interface is allowed. For operation of an I ² C interface, set the port output to Hi-Z (PDR = 1).

Table 1.6-1 Description of the Pin Functions (5/6)

QFP	LQFP	Pin name	Circuit type	Description of function
31,33	29,31	P53,P55	G (NchOD/H)	N-ch open-drain type I/O port.
		SCL1,SCL2		Clock I/O pin for an I ² C interface. This function is enabled when operation of an I ² C interface is allowed. For operation of an I ² C interface, set the port output to Hi-Z (PDR = 1).
38 to 41, 43 to 46	36 to 39, 41 to 44	P60 to P67	H (CMOS/H)	General-purpose I/O port.
		AN0 to AN7		Analog signal output pins of the A/D converter. This function is enabled when analog input specification is allowed.
47,48, 53 to 58	45,46, 51 to 56	P70 to P77	I (CMOS/H)	General-purpose I/O port.
		IRQ0 to IRQ7		External interrupt request input pin. Since this input pin can be used whenever an external interrupt is allowed, be sure to use another function to stop the output unless for a special purpose.
59,60	57,58	P80,P81	J (CMOS/H)	General-purpose I/O port.
		TIN0,TIN1		Event input pin for the reload timer. As this input pin can be used whenever a reload timer input operation is performed, be sure to use another function to stop the output unless for a special purpose.
61,62	59,60	P82,P83	J (CMOS/H)	General-purpose I/O port.
		TOT0,TOT1		Output pin for the reload timer. This function is enabled when output specification for the reload timer is allowed.
63 to 66	61 to 64	P84 to P87	J (CMOS/H)	General-purpose I/O port.
		IN0 to IN3		Input capture trigger input pin. Since this function can be used whenever an input operation by input capture is performed, be sure to use another function to stop the output unless for a special purpose.
67,68	65,66	P90,P91	J (CMOS/H)	General-purpose I/O port.
		OUT0,OUT1		Output compare event output pin.
69 to 74	67 to 72	P92 to P97	J (CMOS/H)	General-purpose I/O port.
		PPG0 to PPG5		PPG output pin. This function is enabled when the output specification of PPG is allowed.
75,76	73,74	PA0,PA1	J (CMOS/H)	General-purpose I/O port.
		OUT2,OUT3		Output compare event output pin.
78,79	76,77	PA2,PA3	J (CMOS/H)	General-purpose I/O port.

Table 1.6-1 Description of the Pin Functions (6/6)

QFP	LQFP	Pin name	Circuit type	Description of function
80	78	PA4	J (CMOS/H)	Output compare event output pin.
		CKOT		While the CKOT is operated, this pin is used as CKOT output.
34	32	AVcc	-	Power supply pin for the A/D converter.
35	33	AVRH	-	External reference power supply pin for the A/D converter.
36	34	AVRL	-	External reference power supply pin for the A/D converter.
37	35	AVss	-	Power supply pin for the A/D converter.
49, 50	47, 48	MD0, MD1	C	Operating mode selection input pin. Connect it directly to Vcc or Vss.
51	49	MD2	K	Operating mode selection input pin. Connect it directly to Vcc or Vss. (MB90552A/552B/553A/553B/V550A)
			C	Operating mode selection input pin. Connect it directly to Vcc or Vss. (MB90P553A/F553A)
23,84	21,82	Vcc	-	Power supply (5 V) input pin
11,42,81	9,40,79	Vss	-	Power supply (0 V) input pin

1.7 I/O Circuit Types

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

■ I/O Circuit Types

Table 1.7-1 I/O Circuit Types (1/4)

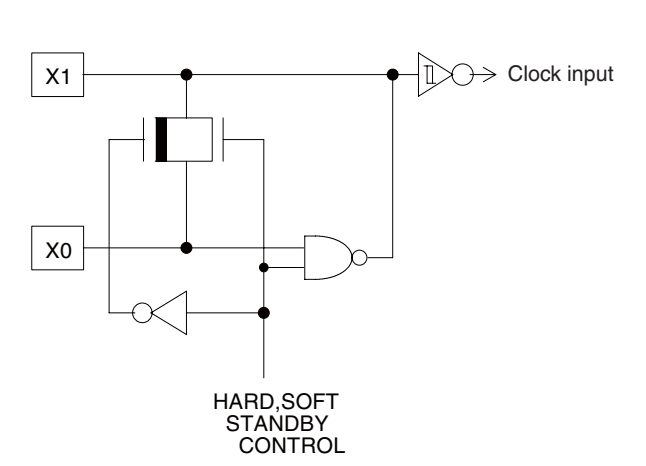
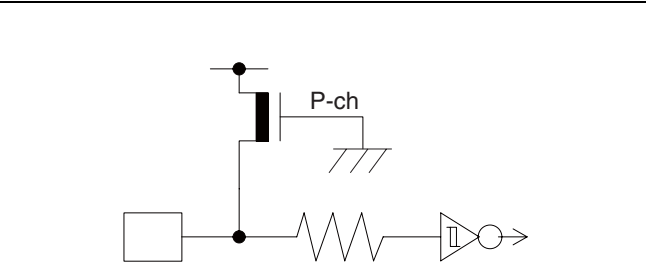
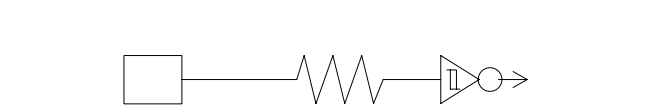
Classification	Circuit	Remark
A		<ul style="list-style-type: none">• 3MHz to 16MHz• Oscillation feedback resistor: approx. 1 MΩ
B		<ul style="list-style-type: none">• CMOS level hysteresis input• With pull-up: resistor of approx. 50 kΩ
C		CMOS level hysteresis input

Table 1.7-1 I/O Circuit Types (2/4)

Classification	Circuit	Remark
D		<ul style="list-style-type: none">• CMOS level output• CMOS level input• With standby control• With input pull-up resistor control: resistor of approx. 50 kΩ
E		<ul style="list-style-type: none">• CMOS level output• CMOS level input• With standby control
F		<ul style="list-style-type: none">• CMOS level output• CMOS level hysteresis input• With open-drain control

Table 1.7-1 I/O Circuit Types (3/4)

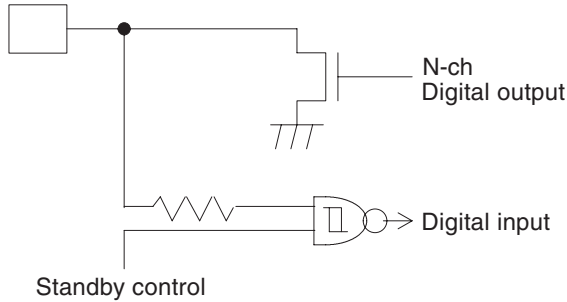
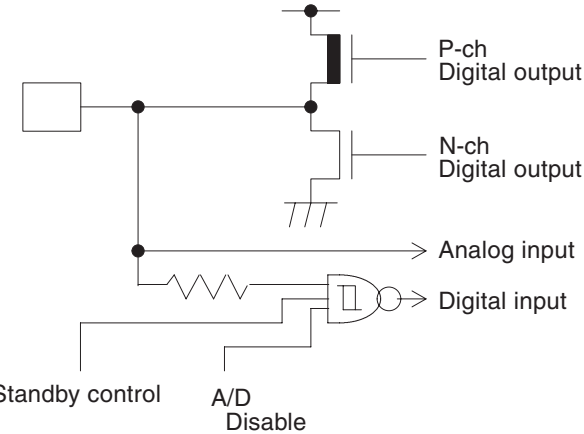
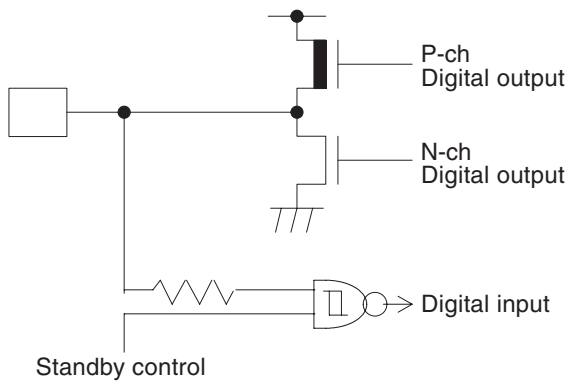
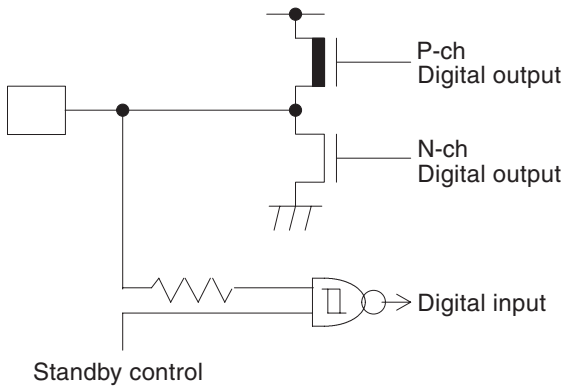
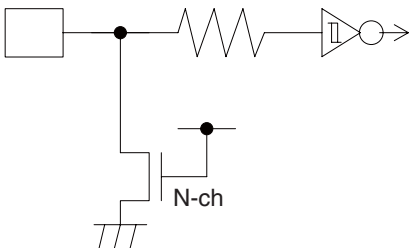
Classification	Circuit	Remark
G	 <p>N-ch Digital output</p> <p>Digital input</p> <p>Standby control</p>	<ul style="list-style-type: none"> • N channel open-drain output • CMOS level hysteresis input • With standby control <p>Note: Unlike usual CMOS I/O pins, this pin contains no Pch transistor, and so current does not flow to Vcc even if voltage is externally applied to this pin when IC power supply is off.</p>
H	 <p>P-ch Digital output</p> <p>N-ch Digital output</p> <p>Analog input</p> <p>Digital input</p> <p>Standby control</p> <p>A/D Disable</p>	<ul style="list-style-type: none"> • CMOS level output • CMOS level hysteresis input • With standby control • Analog input
I	 <p>P-ch Digital output</p> <p>N-ch Digital output</p> <p>Digital input</p> <p>Standby control</p>	<ul style="list-style-type: none"> • CMOS level output • CMOS level hysteresis input • With standby control

Table 1.7-1 I/O Circuit Types (4/4)

Classification	Circuit	Remark
J		<ul style="list-style-type: none">• CMOS level output• CMOS level hysteresis input• With standby control
K		<ul style="list-style-type: none">• CMOS level hysteresis input• With pull-down Resistor: approx. 50 kΩ

1.8 Cautions on Handling Devices

When handling MB90550A/B devices, pay particular attention to the following points:

- Prevention of latchup
 - Stabilization of power supply voltage
 - Handling unused pins
 - Cautions in connection with using an external clock
 - Handling power supply pins (Vcc/Vss)
 - Crystal oscillator circuit
 - Procedure for turning on the A/D converter power supply and the analog input
 - Recover from standby
 - Cautions on turning on the power supply
 - Handling power supply pins of the A/D converter
 - Undefined output from port0 or port1
 - Using the "DIV A, Ri" and "DIVW A, RWi" instructions
 - Using REALOS
 - Notes on operation in PLL clock mode
-

■ Cautions on Handling Devices

○ Prevention of latchup

In CMOS ICs, latchup may occur when:

- A voltage higher than Vcc or lower than Vss is applied to an input or output pin.
- A voltage which exceeds the rating is applied between Vcc and Vss
- The AVcc power supply is turned on before Vcc voltage is applied

If latchup occurs, power supply current increases rapidly and elements may be damaged by heat. Pay particular attention when using the device to prevent this.

For the same reason, make sure that the analog power supply current does not exceed the digital power supply current.

○ Stabilization of power supply voltage

If the power supply voltage varies acutely even within the operation assurance range of the Vcc power supply voltage, a malfunction may occur. The Vcc power supply voltage must therefore be stabilized. As stabilization guidelines, stabilize the power supply voltage so that Vcc ripple fluctuations (peak to peak value) in the commercial frequencies (50 Hz to 60 Hz) fall within 10% of the standard Vcc power supply voltage and the transient fluctuation rate becomes 0.1 V/ms or less in instantaneous fluctuation for power supply switching.

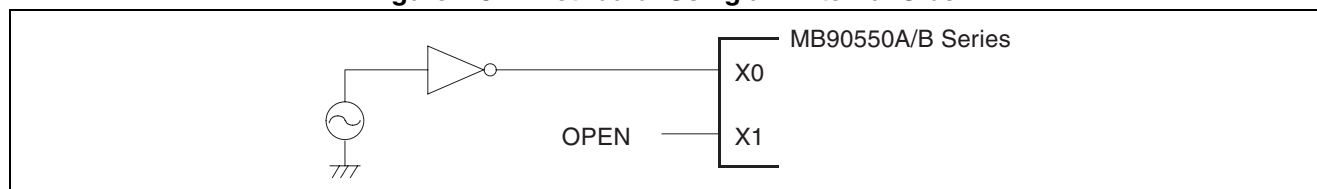
○ Handling unused pins

Leaving unused input pins open may cause malfunctions or permanent damage due to latchup of the device. To prevent this problem, use pull-up or pull-down resistors of 2 kΩ or more. If an input/output pin is not used, set it to output and leave the pin open, or set it to input and handle it as an input pin.

○ Cautions on using an external clock

When using an external clock, use only the X0 pin and leave the X1 pin open.

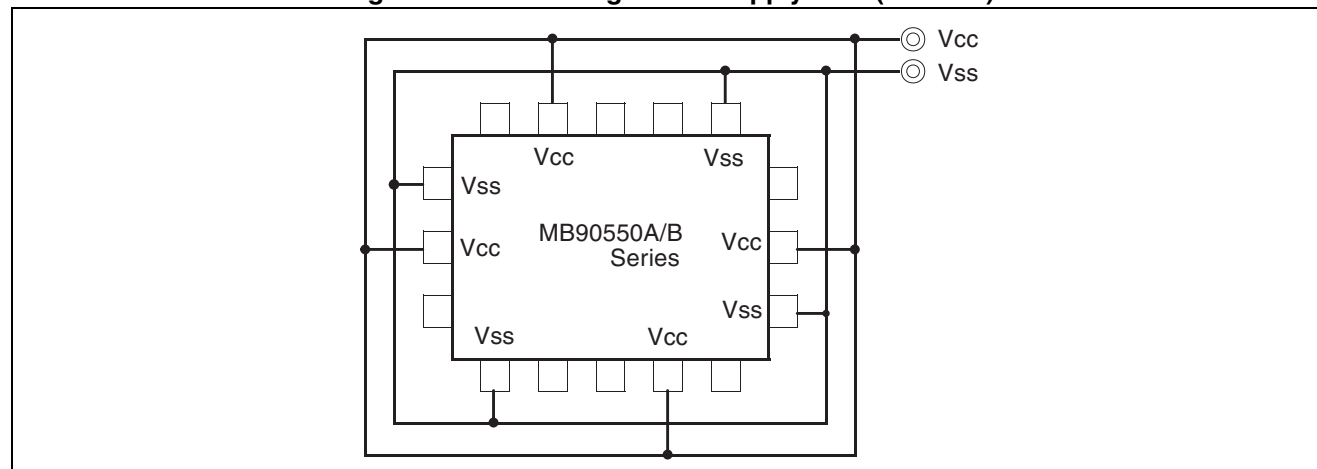
Figure 1.8-1 Method of Using an External Clock



○ Handling power supply pins (Vcc/Vss)

If there are multiple power supply pins (Vcc/Vss), pins to which the same voltage is applied are designed to be connected to each other in order to prevent malfunctions such as latchup. Confirm that all of these pins are externally connected to the power supply or ground to reduce undesired radiation, prevent malfunction of the strobe signal because of ground level increase, and maintain the total output current standard.

Figure 1.8-2 Handling Power Supply Pins (Vcc/Vss)



Also, connect the Vcc and Vss pins of this device to a current supply with the lowest possible impedance.

In addition, it is recommended to connect a condenser of about 0.1 μ F between Vcc and Vss as by-pass condenser for this device.

○ Crystal oscillator circuit

Noise around the X0 or X1 pins may cause device malfunction. When designing a printed-circuit board, confirm that the X0 and X1 pins, a crystal oscillator (ceramic oscillator), and by-pass condenser to the ground are located as close to the device as possible, and that the number of cables crossing other cables is as low as possible.

Also, it is highly recommended to use a printed-circuit board artwork surrounding the X0 and X1 pins to stabilize the operation of the device.

○ Procedure for turning on the A/D converter power supply and the analog input

Confirm that the digital power supply (Vcc) is turned on before turning the power supplies of the A/D converter (AVcc, AVRH, AVRL) and applying analog input (AN0 to AN7).

Moreover, when shutting down power supplies, turn off the digital power supply only after turning off the power supplies of the A/D converter and the analog input. Confirm that AV_{RH} does not exceed AVcc when turning the power supplies on or off (The analog power supply and the digital power supply can be turned on or off at the same time).

○ Recover from standby

If the power supply voltage becomes lower than the standby RAM retain voltage at standby, the device may not recover from standby. In this case, it can be recovered by applying reset from an external reset pin.

○ Cautions on turning on the power supply

Maintain a voltage rising time of 50 μ s or more (between 0.2 V and 2.7 V) when turning on the power supply in order to prevent the mounted step-down circuit from malfunctioning.

○ Handling power supply pins of the A/D converter

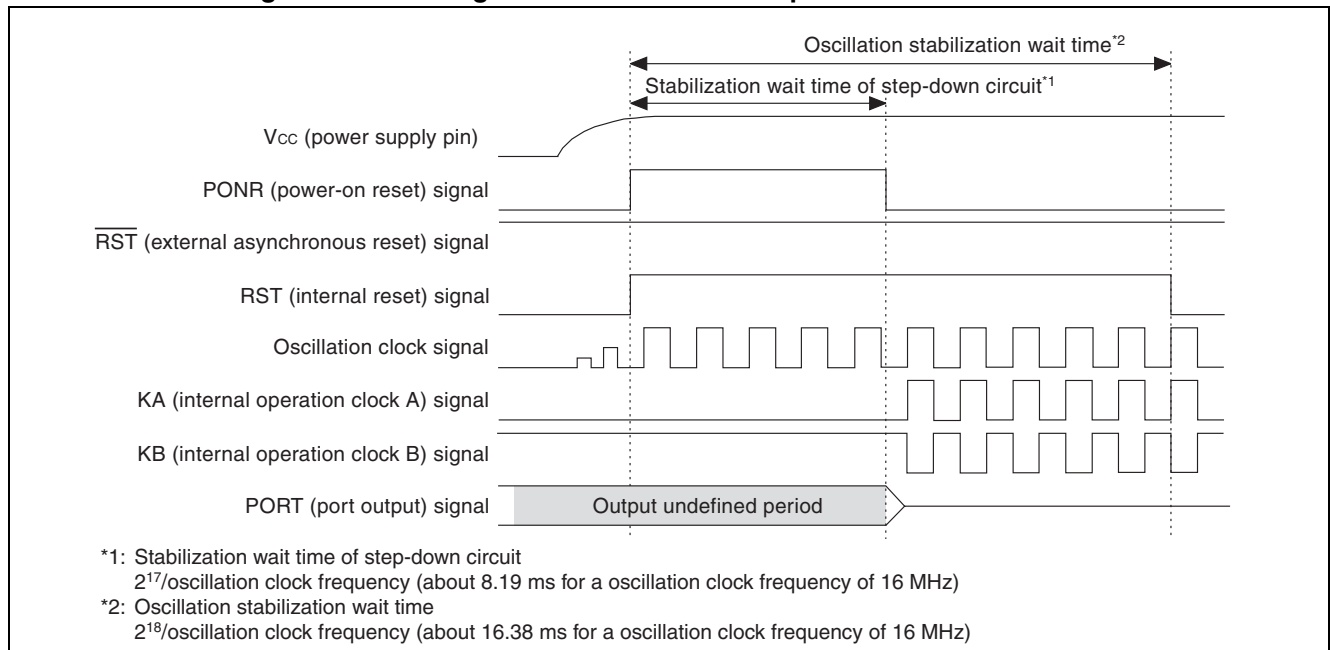
Even if an A/D converter is not used, connect the pins as follows:

$AV_{CC} = V_{CC}$, $AV_{SS} = AV_{RH} = AV_{RL} = V_{SS}$

○ Undefined output from port0 or port1

After power-on, output from port0 and port1 is undefined during the oscillation stabilization wait time (during power-on reset) of the step-down circuit. Note that the timing is as shown in Figure 1.8-3 (target type: MB90V550A, MB90F553A, MB90T553A, MB90S553A, MB90T552A, and MB90S552A). The type of device without a step-down circuit does not have undefined output because there is no oscillation stabilization wait time for a step-down circuit (target type: MB90P553A).

Figure 1.8-3 Timing Chart of Undefined Output from Port0 and Port1



○ Using the "DIV A, Ri" and "DIVW A, RWi" instructions

Use the signed multiplication/division instructions "DIV A, Ri" and "DIVW A, RWi" after setting the corresponding bank registers (DTB, ADB, USB, and SSB) to 00_H.

If the corresponding bank register (DTB, ADB, USB, or SSB) is set to a value other than 00_H, the remainder obtained from instruction execution results is not saved in the register of the instruction operand.

For details, see Section "2.8 Notes on Using the "DIV A, Ri" and "DIVW A, RWi" Instructions".

○ **Using REALOS**

During use of REALOS, the expanded intelligent I/O service (EI²OS) cannot be used.

○ **Notes on operation in PLL clock mode**

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

CHAPTER 2 CPU

This chapter describes the functions and operation of the CPU.

- 2.1 Memory Space
- 2.2 Addressing
- 2.3 Allocating Multiple-byte Data in a Memory Space
- 2.4 Dedicated Registers
- 2.5 General-purpose Registers
- 2.6 Prefix Codes
- 2.7 Interrupt Suppression Instructions and Prefix Codes
- 2.8 Notes on Using the "DIV A, Ri" and "DIVW A, RWi" Instructions

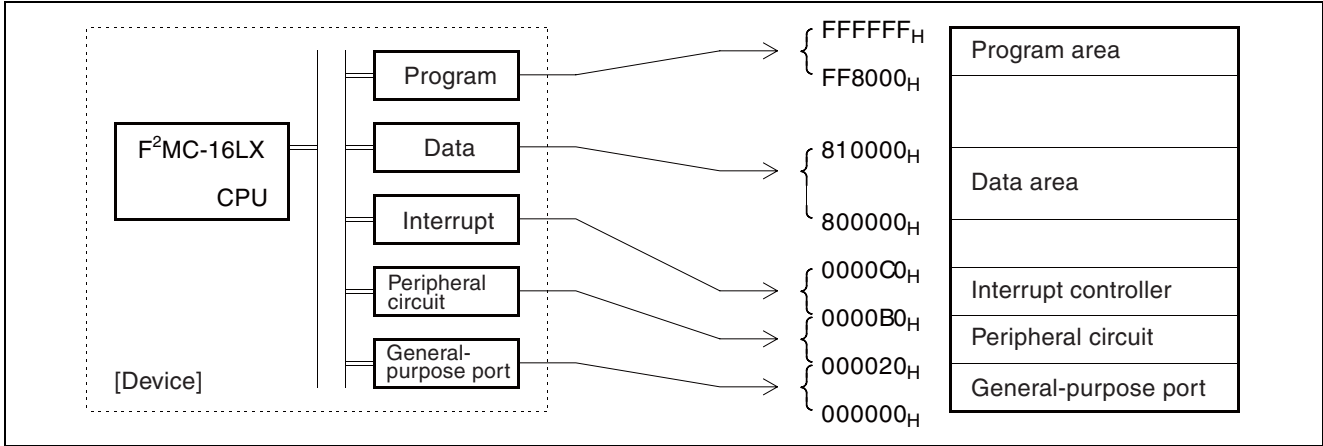
2.1 Memory Space

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-8L 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

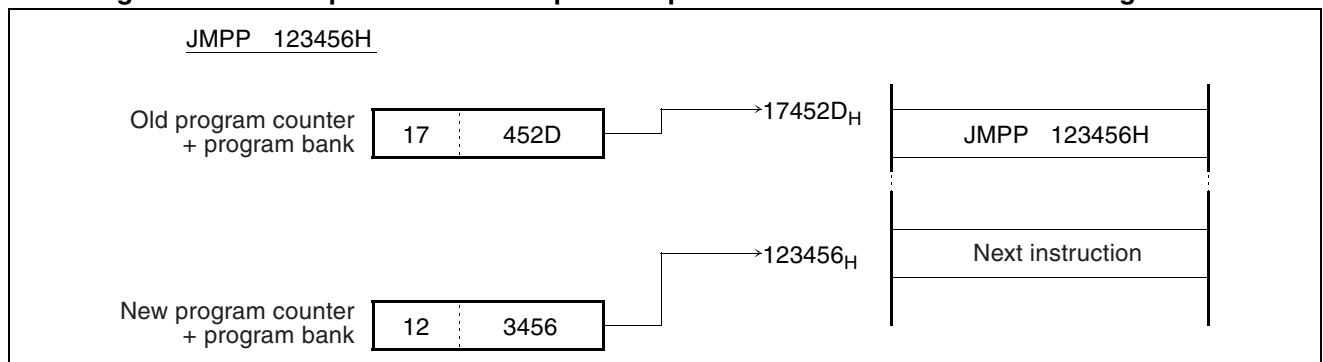
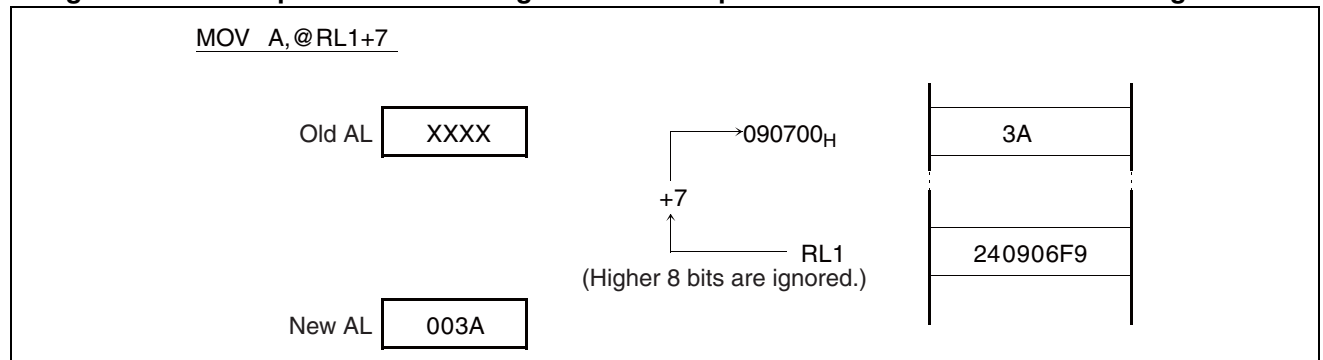


Figure 2.2-2 Example of the 32-bit Register-Indirect Specification in the Linear Addressing Method



■ Addressing by the Bank Method

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

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

The 64K-byte 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 data bank register (ADB): Initial reset value 00_H**

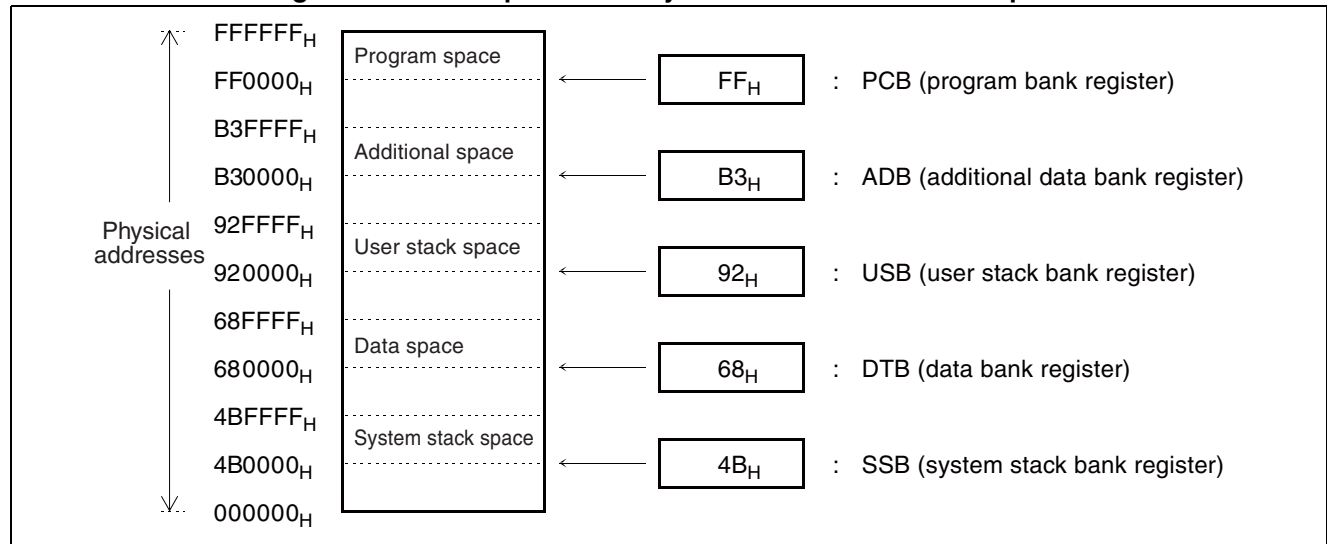
The 64 KB bank specified by ADB is called additional (AD) space. The additional space mainly contains data that overflowed from the DT space.

To increase instruction code efficiency, a default space is determined for instructions of each addressing mode, as listed below. 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 to 00FFFF_H). The PC space is allocated in the bank specified by the reset vector.

Table 2.2-1 Default Spaces

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

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

2.3 Allocating Multiple-byte Data in a Memory Space

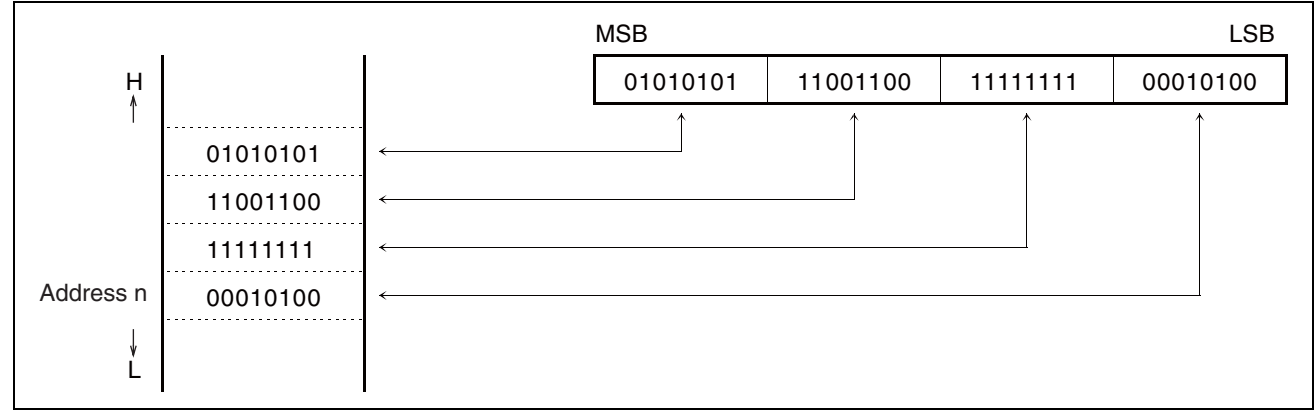
In a memory space, the lower eight bits of multiple-byte data are stored at address n . The remaining bits are stored at the addresses $n + 1$, $n + 2$, $n + 3$, ... in that order.

■ Allocating Multiple-byte Data in a Memory Space

As shown in Figure 2.3-1, 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.

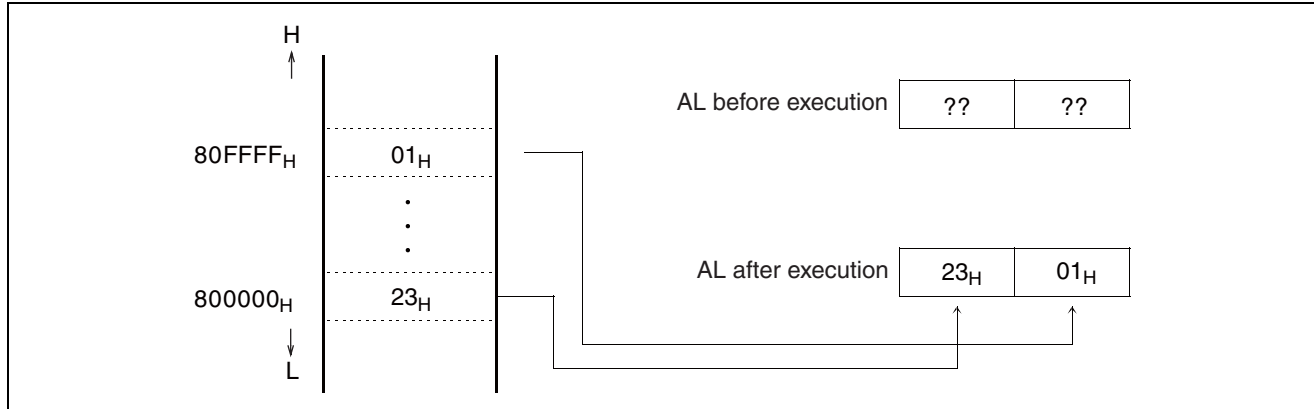
Figure 2.3-1 Example of Allocating Multiple-byte Data in a Memory Space



■ Access of Multiple-byte Data

As shown in Figure 2.3-2, 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 2.3-2 Example of Accessing Multiple-byte Data (Execution of `MOVW A, 080FFFFH`)



2.4 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.

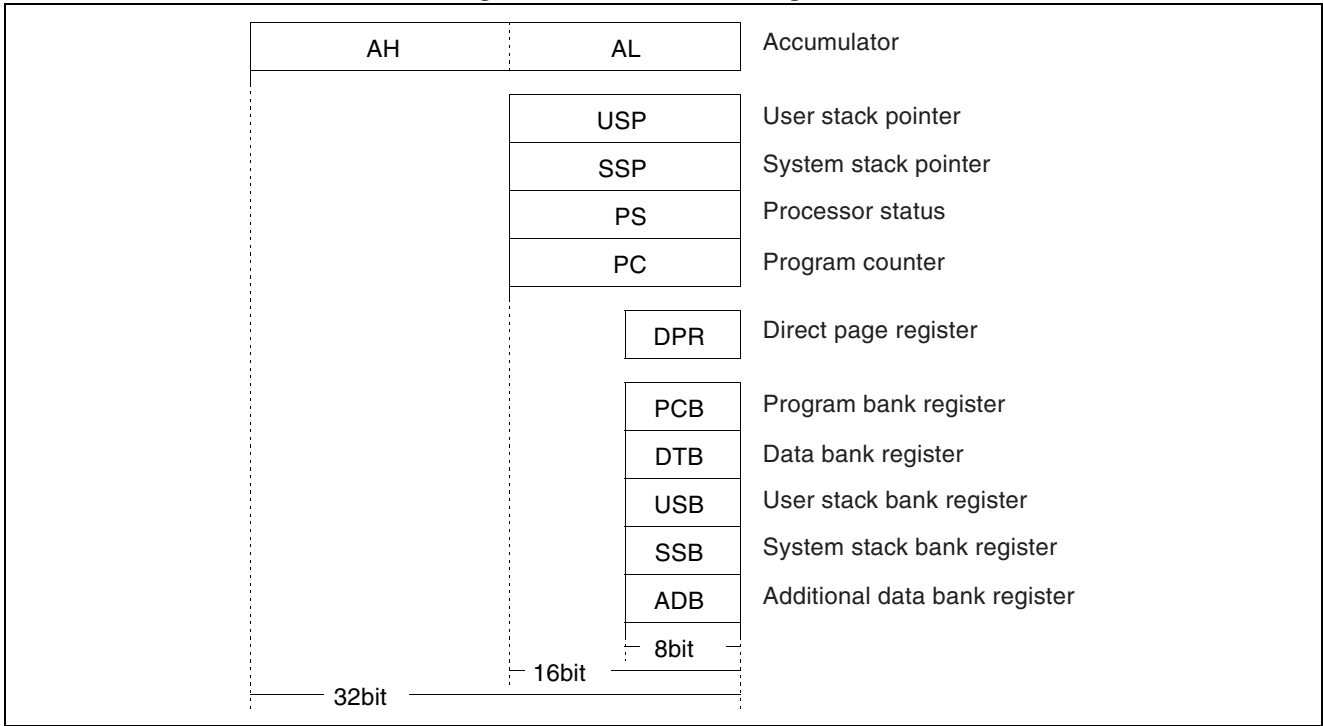
- **Additional data bank register (ADB)**

An 8-bit register indicating the AD space.

- **Direct page register (DPR)**

An 8-bit register indicating the direct page.

Figure 2.4-1 Dedicated Registers



2.4.1 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-8, 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 2.4-2 Example of 32-bit Data Transfer

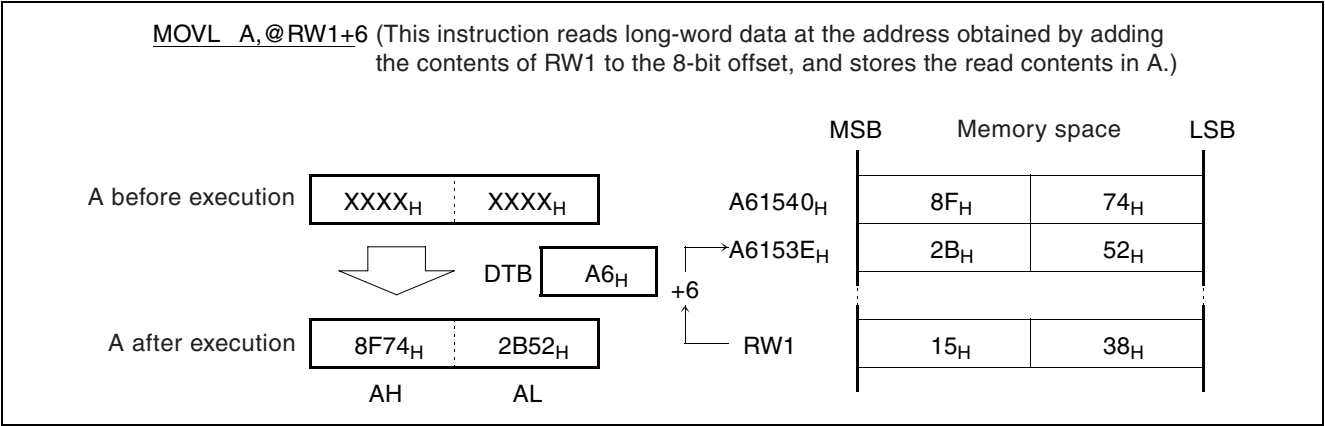
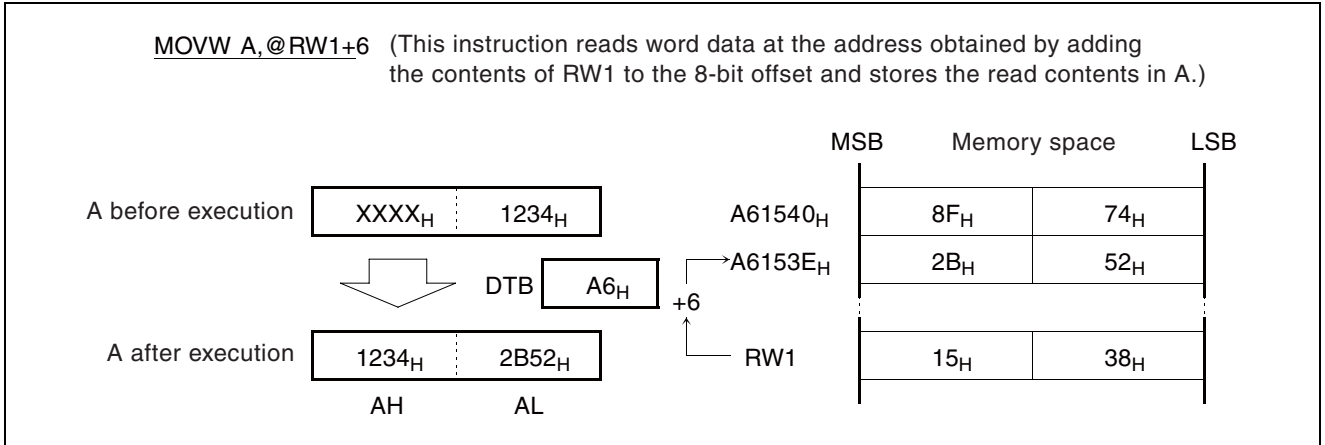


Figure 2.4-3 Example of AL-AH Transfer



As shown in Figure 2.4-4, 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 (see Figure 2.4-5).

Figure 2.4-4 Example of Zero Extension

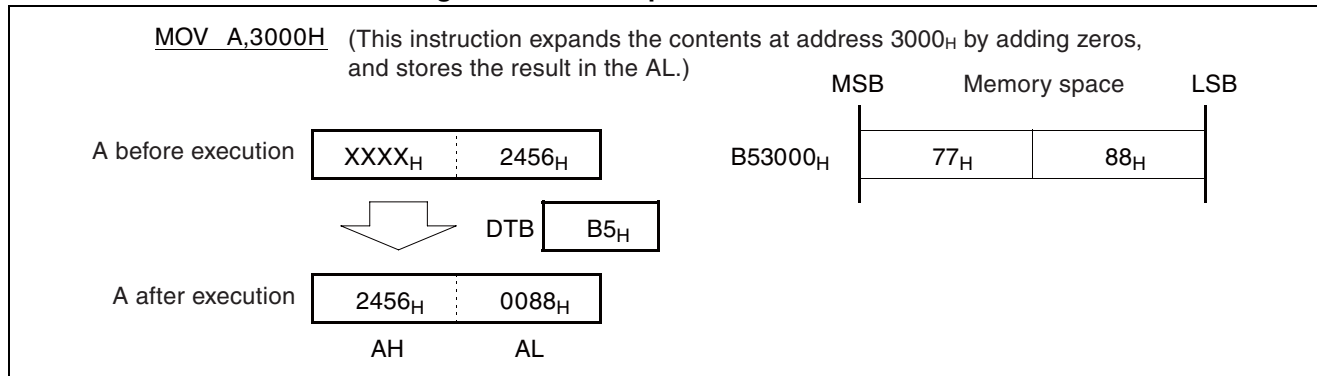
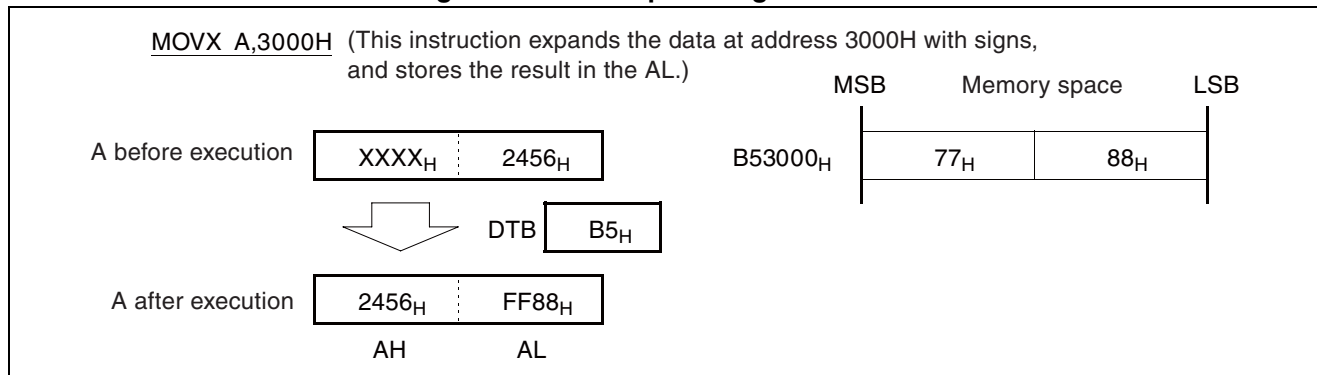


Figure 2.4-5 Example of Sign Extension



2.4.2 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 2.4-6 and Figure 2.4-7). 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, and 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 2.4-6 Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is "0")

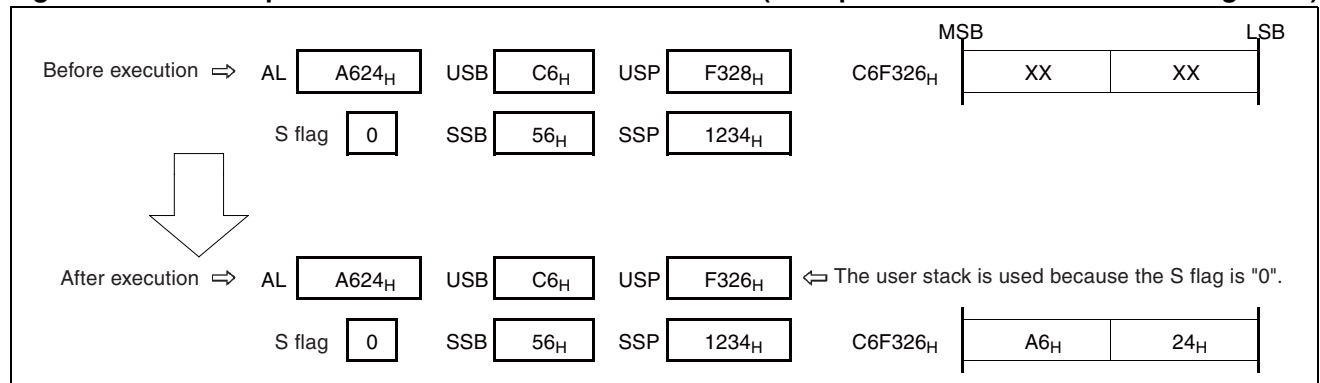
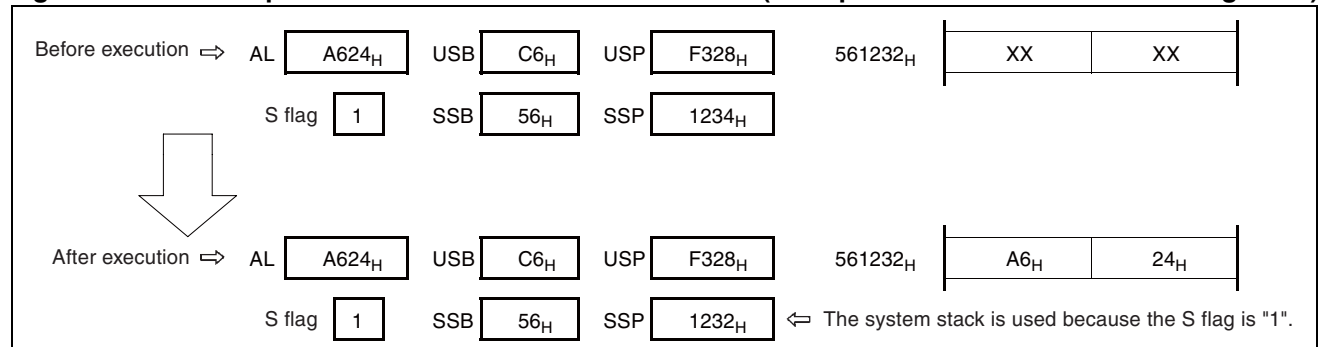


Figure 2.4-7 Stack Operation Instructions and Stack Pointers (Example of PUSHW A when the S Flag is "1")



Note:

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

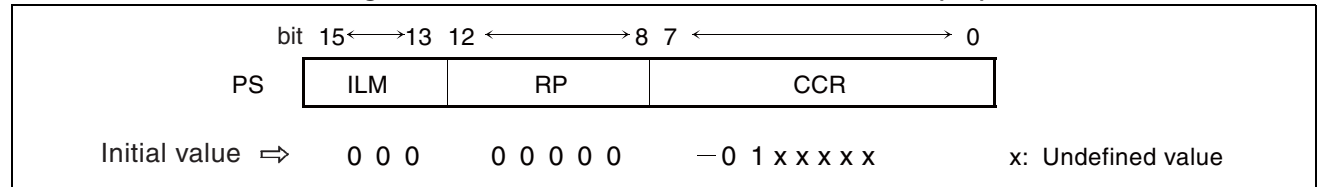
2.4.3 Processor Status (PS)

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

■ Processor Status (PS)

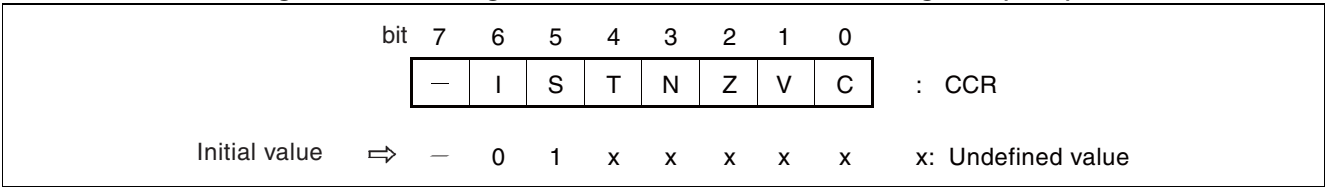
The higher bytes of the processor status (PS) consist of the register bank pointer (RP), which indicates the starting address of the register bank, and the interrupt level mask register (ILM). The lower bytes of the PS consist of the condition code register (CCR) formed by flags to be set or reset by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

Figure 2.4-8 Structure of the Processor Status (PS)



■ Condition Code Register (CCR)

Figure 2.4-9 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)**

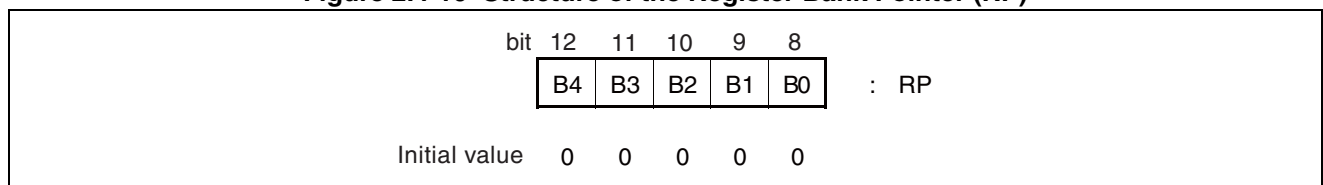
This flag is set when a carry-up or carry-down is generated from the MSB as a result of an arithmetic operation. It is cleared if no carry-up or carry-down occurs.

■ **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 2.4-10 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 Table 2.4-1, 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 2.4-11 Structure of the Interrupt Level Register (ILM)

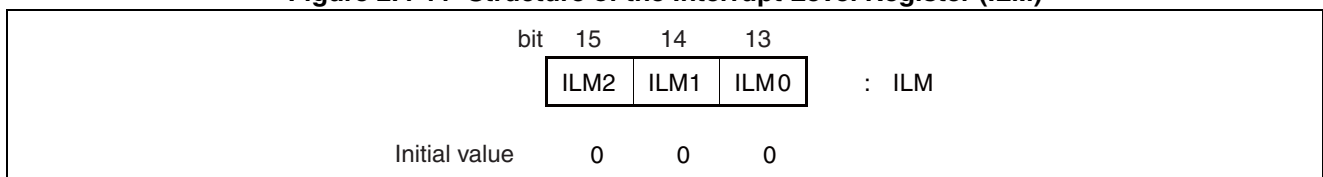


Table 2.4-1 Level Hierarchy of the Levels Indicated by the Interrupt Level Mask Register (ILM)

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

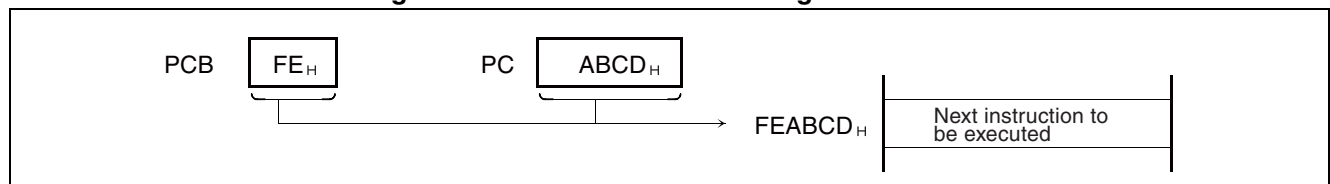
2.4.4 Program Counter (PC)

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 2.4-12 Structure of the Program Counter



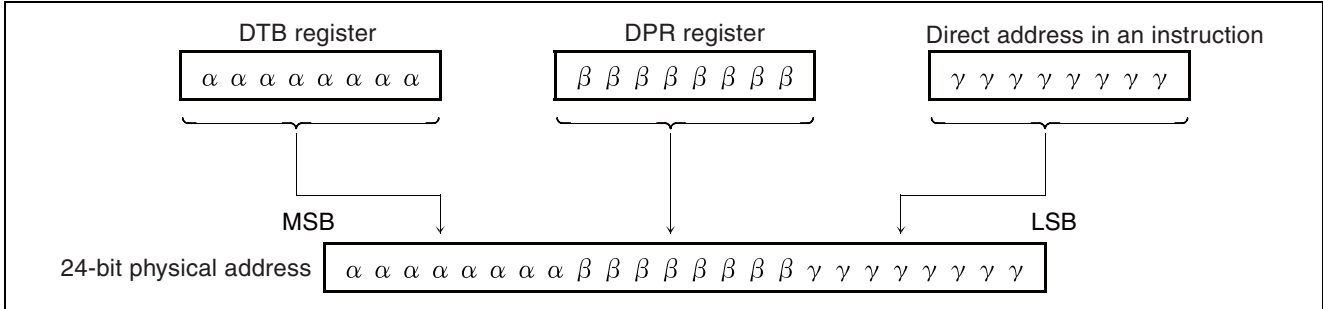
2.4.5 Direct Page Register (DPR)

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 2.4-13 shows physical address generation by direct addressing.

Figure 2.4-13 Generating a Physical Address by Direct Addressing



2.4.6 Bank registers (PCB, DTB, USB, SSB, ADB)

Using bank addressing, each bank register is set with the highest eight bits of a 24-bit address. Bank registers can be classified into the following five types:

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

The bank registers indicate the memory banks in which program, data, user stack, system stack, and additional spaces are allocated.

■ Program Bank Register (PCB)

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

This register is rewritten at execution of the JMPP, CALLP, RETP, or RETI instruction that branches to all 16 MB space, at execution of the software interrupt instruction, and at an occurrence of a hardware interrupt or exception handling interrupt.

■ Data Bank Register (DTB)

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

■ User Stack Bank Register (USB) and System Stack Bank Register (SSB)

The user stack bank register (USB) and system stack bank register (SSB) specify the stack (SP) space. The value of the stack flag (CCR:S) is the basis for determining which of the bank registers is used.

■ Additional Data Bank Register (ADB)

The additional data bank register (ADB) specifies the additional (AD) space.

■ Settings and Data Access of Bank Registers

Each of the bank registers has a length of eight bits. The program bank register (PCB) is initialized to FF_H by a reset, while the other bank registers are initialized to 00_H.

The program bank register (PCB) is a read-only register. The other bank registers are read-write registers.

For information on bank register operations, see Section "2.1 Memory Space".

2.5 General-purpose Registers

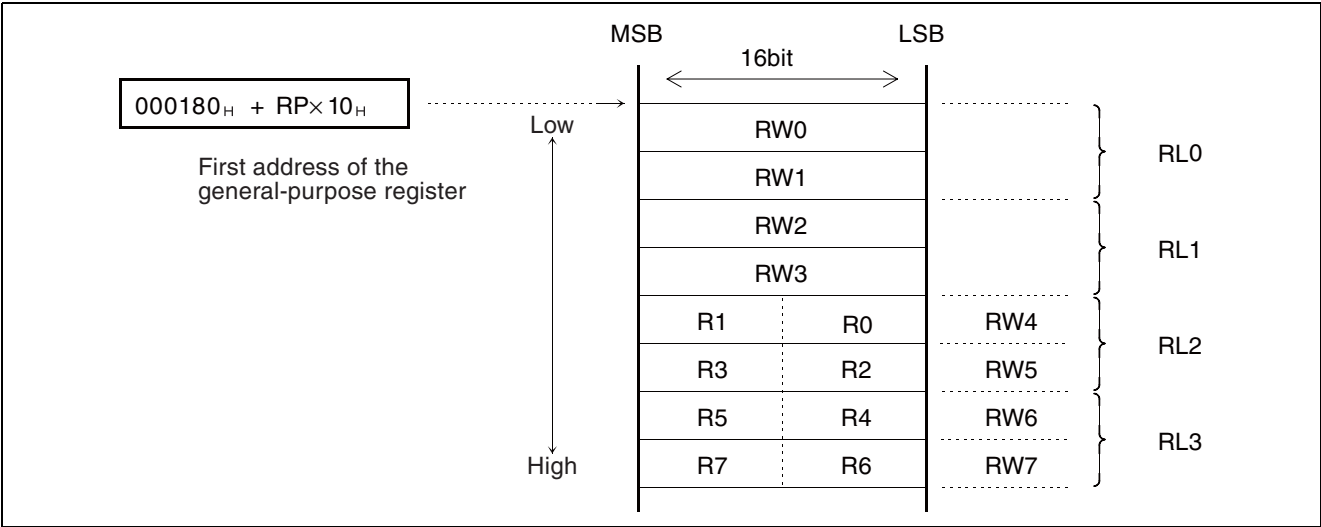
Similar to ordinary memory, the user can specify the use of general-purpose registers. General-purpose registers are the same as dedicated registers in the sense that they coexist with the RAM in the address space of the CPU and that they can be accessed without specifying an address.

■ General-purpose Registers

The general-purpose registers of the F²MC-16LX are located at 000180_H to 00037F_H (maximum) of the main storage. 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:

- 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 2.5-1 General-purpose Registers



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

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

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

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

■ 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 2.5-1, 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. RL0 to RL3 can also be used as linear pointers for directly accessing all regions of the memory space.

Table 2.5-1 Functions of Register Banks

Register	Function
R0 to R7	Used as operands of instructions.
RW0 to RW7	Used as pointers and operands of instructions.
RL0 to RL3	Used as long pointers and operands of instructions.

Notes:

- R0 is also used as a barrel shift counter and normalize instruction counter.
 - RW0 is also used as a string instruction counter.
-

2.6 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 2.6-1 lists the bank selection prefixes and the memory spaces selected by them.

Table 2.6-1 Bank Selection Prefixes

Bank selection prefix	Selected space
PCB	Program 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.

2.7 Interrupt Suppression Instructions and Prefix Codes

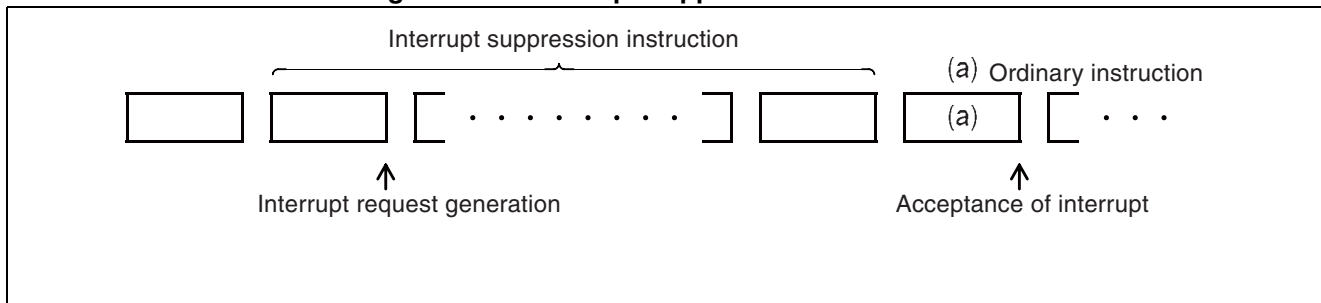
The following 10 types of interrupt suppression instructions do not detect whether there is a hardware interrupt request and ignore any such interrupt request.

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

■ Interrupt Suppression Instructions

As shown in Figure 2.7-1, 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.

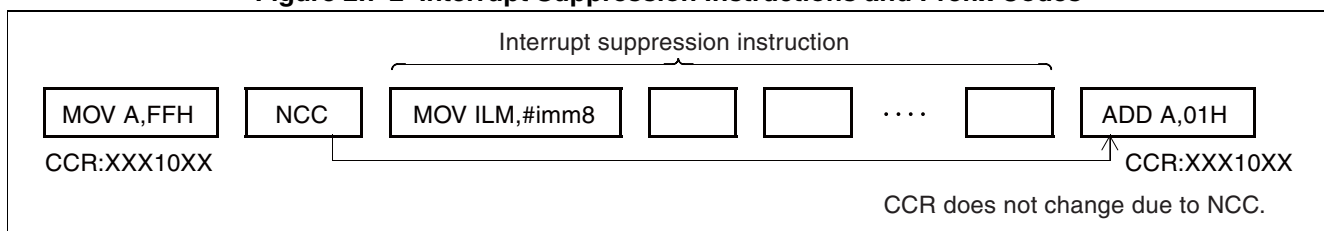
Figure 2.7-1 Interrupt Suppression Instructions



■ Restrictions on Interrupt Suppression and Prefix Instructions

As shown in Figure 2.7-2, 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.

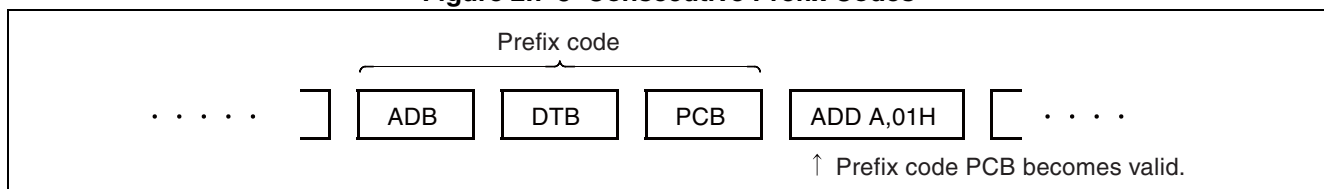
Figure 2.7-2 Interrupt Suppression Instructions and Prefix Codes



■ In the Case of Consecutive Prefix Codes

As shown in Figure 2.7-3, 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 2.7-3 Consecutive Prefix Codes



2.8 Notes on Using the "DIV A, Ri" and "DIVW A, RWi" Instructions

Before using the "DIV A, Ri" or "DIVW A, RWi" instruction, set the corresponding bank register to 00_H.

■ Using the "DIV A, Ri" and "DIVW A, RWi" Instructions

Table 2.8-1 Using the "DIV A, Ri" and "DIVW A, RWi" Instructions (i=0 to 7) (1/2)

Instruction	Name of bank register affected by execution of instruction on left	Address at which remainder is stored
DIV A, R0	DTB	(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + 8 _H : Low-order 16 bits)
DIV A, R1		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + 9 _H : Low-order 16 bits)
DIV A, R4		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + C _H : Low-order 16 bits)
DIV A, R5		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + D _H : Low-order 16 bits)
DIVW A, RW0		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + 0 _H : Low-order 16 bits)
DIVW A, RW1		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + 2 _H : Low-order 16 bits)
DIVW A, RW4		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + 8 _H : Low-order 16 bits)
DIVW A, RW5		(DTB: High-order 8 bits) + (0180 _H + RP × 10 _H + A _H : Low-order 16 bits)
DIV A, R2	ADB	(ADB: High-order 8 bits) + (0180 _H + RP × 10 _H + A _H : Low-order 16 bits)
DIV A, R6		(ADB: High-order 8 bits) + (0180 _H + RP × 10 _H + E _H : Low-order 16 bits)
DIVW A, RW2		(ADB: High-order 8 bits) + (0180 _H + RP × 10 _H + 4 _H : Low-order 16 bits)
DIVW A, RW6		(ADB: High-order 8 bits) + (0180 _H + RP × 10 _H + E _H : Low-order 16 bits)

Table 2.8-1 Using the "DIV A, Ri" and "DIVW A, RWi" Instructions (i=0 to 7) (2/2)

Instruction	Name of bank register affected by execution of instruction on left	Address at which remainder is stored
DIV A, R3	USB SSB ^{*1}	(USB ^{*2} : High-order 8 bits) + (0180 _H + RP × 10 _H + B _H : Low-order 16 bits)
DIV A, R7		(USB ^{*2} : High-order 8 bits) + (0180 _H + RP × 10 _H + F _H : Low-order 16 bits)
DIVW A, RW3		(USB ^{*2} : High-order 8 bits) + (0180 _H + RP × 10 _H + 6 _H : Low-order 16 bits)
DIVW A, RW7		(USB ^{*2} : High-order 8 bits) + (0180 _H + RP × 10 _H + E _H : Low-order 16 bits)

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

*2: If the S bit of the CCR register is "0"

If the value of the affected bank register (DTB, ADB, USB, SSB) is 00_H, the remainder after division is saved in the register of the instruction operand. If the value of the bank register is not 00_H, the high-order 8-bit part of the address is specified in the bank register corresponding to the instruction operand register; the low-order 16-bit part of the address is identical to the address of the instruction operand register. The remainder is saved in the bank register set with the high-order eight bits.

[Example]

If "DIV A, R0" is executed when DTB=053_H and RP=03_H, the R0 address will be 0180_H + RP (03_H) × 10_H + 08_H (address corresponding to R0) = 0001B8_H.

Since the bank register specified by "DIV A, R0" is the data bank register (DTB), the remainder is stored at address 05301B8_H, which is obtained by adding the bank address, 053_H.

References:

- For information about bank registers, see Section "2.4.6 Bank registers (PCB, DTB, USB, SSB, ADB)".
 - For information about the registers of Ri and RWi, see Section "2.5 General-purpose Registers".
-

■ **Working Around the Conditions Described in the above Notes**

Compilers that do not generate the instructions listed in Table 2.8-1 and assemblers that replace the listed instructions with an equivalent sequence of instructions are available. Using such compilers and assemblers, programs can be developed while working around the conditions described in notes on using the "DIV A, Ri" and "DIVW A, RWi" instructions. Use the following compilers and assemblers for the MB90550A/B Series:

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

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 lists interrupt causes, interrupt vectors, and interrupt control registers.

■ Interrupt Causes

Table 3.2-1 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers (1/2)

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	Y2	#11	FFFFD0 _H	ICR00	0000B0 _H
Time-base timer	N	#12	FFFFCC _H		
DTP0 (external interrupt 0)	Y2	#13	FFFFC8 _H	ICR01	0000B1 _H
DTP4/5 (external interrupt 4/5)	Y2	#14	FFFFC4 _H		
DTP1 (external interrupt 1)	Y2	#15	FFFFC0 _H	ICR02	0000B2 _H
8/16-bit PPG0 counter borrow	N	#16	FFFFBC _H		
DTP2 (external interrupt 2)	Y2	#17	FFFFB8 _H	ICR03	0000B3 _H
8/16-bit PPG1 counter borrow	N	#18	FFFFB4 _H		
DTP3 (external interrupt 3)	Y2	#19	FFFFB0 _H	ICR04	0000B4 _H
8/16-bit PPG2 counter borrow	N	#20	FFFFAC _H		
Extended I/O serial 0	Y2	#21	FFFFA8 _H	ICR05	0000B5 _H
8/16-bit PPG3 counter borrow	N	#22	FFFFA4 _H		
Extended I/O serial 1	Y2	#23	FFFFA0 _H	ICR06	0000B6 _H
16-bit free run-timer (I/O timer) overflow	Y2	#24	FFFF9C _H		
16-bit reload timer 0	Y2	#25	FFFF98 _H	ICR07	0000B7 _H
DTP6/7 (external interrupt 6/7)	Y2	#26	FFFF94 _H		
16-bit reload timer 1	Y2	#27	FFFF90 _H	ICR08	0000B8 _H
8/16-bit PPG4/5 counter borrow	N	#28	FFFF8C _H		

Table 3.2-1 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers (2/2)

Interrupt cause	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Input capture (ch.0) incorporation (input timer)	Y2	#29	FFFF88 _H	ICR09	0000B9 _H
Input capture (ch.1) incorporation (input timer)	Y2	#30	FFFF84 _H		
Input capture (ch.2) incorporation (input timer)	Y2	#31	FFFF80 _H	ICR10	0000BA _H
Input capture (ch.3) incorporation (input timer)	Y2	#32	FFFF7C _H		
Output compare (ch.0) unification (output timer)	Y2	#33	FFFF78 _H	ICR11	0000BB _H
Output compare (ch.1) unification (output timer)	Y2	#34	FFFF74 _H		
Output compare (ch.2) unification (output timer)	Y2	#35	FFFF70 _H	ICR12	0000BC _H
Output compare (ch.3) unification (output timer)	Y2	#36	FFFF6C _H		
UART has been sent	Y2	#37	FFFF68 _H	ICR13	0000BD _H
I ² C interface 0	N	#38	FFFF64 _H		
UART has been received	Y1	#39	FFFF60 _H	ICR14	0000BE _H
I ² C interface 1	N	#40	FFFF5C _H		
Flash memory status	N	#41	FFFF58 _H	ICR15	0000BF _H
Delay interrupt	N	#42	FFFF54 _H		

Y1: An interrupt request flag is cleared by the EI²OS interrupt clear signal. There is a stop request.

Y2: 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.

■ Notes in Connection with Using the EI²OS Feature by Extended I/O Serial 2

For a resource in which the same interrupt number has two possible interrupt causes, both interrupt request flags are cleared by the EI²OS interrupt clear signal. Therefore, if the EI²OS feature is used when either of the two causes is effective, the other interrupt feature cannot be used. Set the interrupt request permission bit of the corresponding resource to "0" and address the problem by software polling. (See Table 3.2-2.)

Table 3.2-2 When the EI²OS Feature is Used by Extended I/O Serial 2

Interrupt factor	Interrupt number	Interrupt control register	Resource interrupt request
Extended I/O serial 1	# 23	ICR06	Allowed
16-bit free-run timer (I/O timer) overflow	# 24		Prohibited

3.3 Interrupt Vectors

Table 3.3-1 shows lists the interrupt vectors.

■ Interrupt Vectors

Table 3.3-1 List of Interrupt Vectors (1/2)

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 _H	#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>
INT 11	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Not used	#11	A/D
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
INT 14	FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Not used	#14	DTP4/DTP5
INT 15	FFFFC0 _H	FFFFC1 _H	FFFFC2 _H	Not used	#15	DTP1
INT 16	FFFFBC _H	FFFFBD _H	FFFFBE _H	Not used	#16	PPG0 borrow
INT 17	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Not used	#17	DTP2
INT 18	FFFFB4 _H	FFFFB5 _H	FFFFB6 _H	Not used	#18	PPG1 borrow
INT 19	FFFFB0 _H	FFFFB1 _H	FFFFB2 _H	Not used	#19	DTP3
INT 20	FFFFAC _H	FFFFAD _H	FFFFAE _H	Not used	#20	PPG2 borrow
INT 21	FFFFA8 _H	FFFFA9 _H	FFFFAA _H	Not used	#21	Extended I/O serial 0
INT 22	FFFFA4 _H	FFFFA5 _H	FFFFA6 _H	Not used	#22	PPG3 borrow
INT 23	FFFFA0 _H	FFFFA1 _H	FFFFA2 _H	Not used	#23	Extended I/O serial 1
INT 24	FFFF9C _H	FFFF9D _H	FFFF9E _H	Not used	#24	16-bit free-run timer
INT 25	FFFF98 _H	FFFF99 _H	FFFF9A _H	Not used	#25	16-bit reload timer 0
INT 26	FFFF94 _H	FFFF95 _H	FFFF96 _H	Not used	#26	DTP6/DTP7
INT 27	FFFF90 _H	FFFF91 _H	FFFF92 _H	Not used	#27	16-bit reload timer 1
INT 28	FFFF8C _H	FFFF8D _H	FFFF8E _H	Not used	#28	PPG4/PPG5 borrow

Table 3.3-1 List of Interrupt Vectors (2/2)

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 29	FFFF88 _H	FFFF89 _H	FFFF8A _H	Not used	#29	Input capture #0
INT 30	FFFF84 _H	FFFF85 _H	FFFF86 _H	Not used	#30	Input capture #1
INT 31	FFFF80 _H	FFFF81 _H	FFFF82 _H	Not used	#31	Input capture #2
INT 32	FFFF7C _H	FFFF7D _H	FFFF7E _H	Not used	#32	Input capture #3
INT 33	FFFF78 _H	FFFF79 _H	FFFF7A _H	Not used	#33	Output compare #0
INT 34	FFFF74 _H	FFFF75 _H	FFFF76 _H	Not used	#34	Output compare #1
INT 35	FFFF70 _H	FFFF71 _H	FFFF72 _H	Not used	#35	Output compare #2
INT 36	FFFF6C _H	FFFF6D _H	FFFF6E _H	Not used	#36	Output compare #3
INT 37	FFFF68 _H	FFFF69 _H	FFFF6A _H	Not used	#37	UART has been sent
INT 38	FFFF64 _H	FFFF65 _H	FFFF66 _H	Not used	#38	I ² C0
INT 39	FFFF60 _H	FFFF61 _H	FFFF62 _H	Not used	#39	UART has been received
INT 40	FFFF5C _H	FFFF5D _H	FFFF5E _H	Not used	#40	I ² C1
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 43	FFFF50 _H	FFFF51 _H	FFFF52 _H	Not used	#43	None
:	:	:	:	:	:	:
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, A, 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.

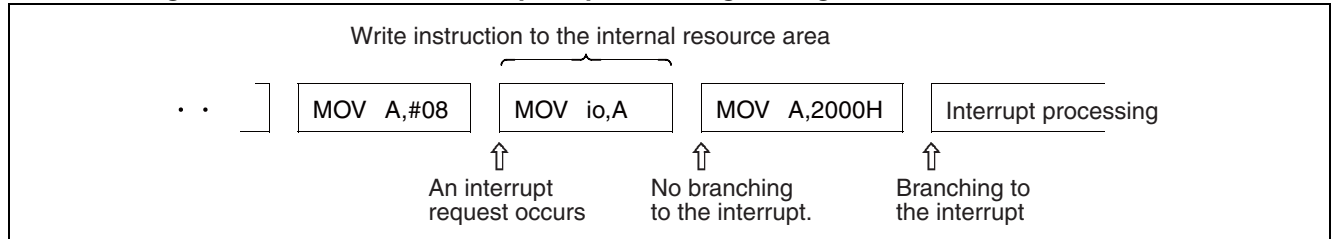
Each interrupt is defined by the control register for an internal resource, the ICR for the interrupt controller, and the contents of CCR in the CPU. For using a hardware interrupt, it is necessary to define these three structure parts in advance on the software level. See section "3.6.1 Interrupt Control Register (ICR)" for the ICR.

The table of interrupt vectors referenced during interrupt processing is allocated to FFFC00_H to FFFFFFF_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

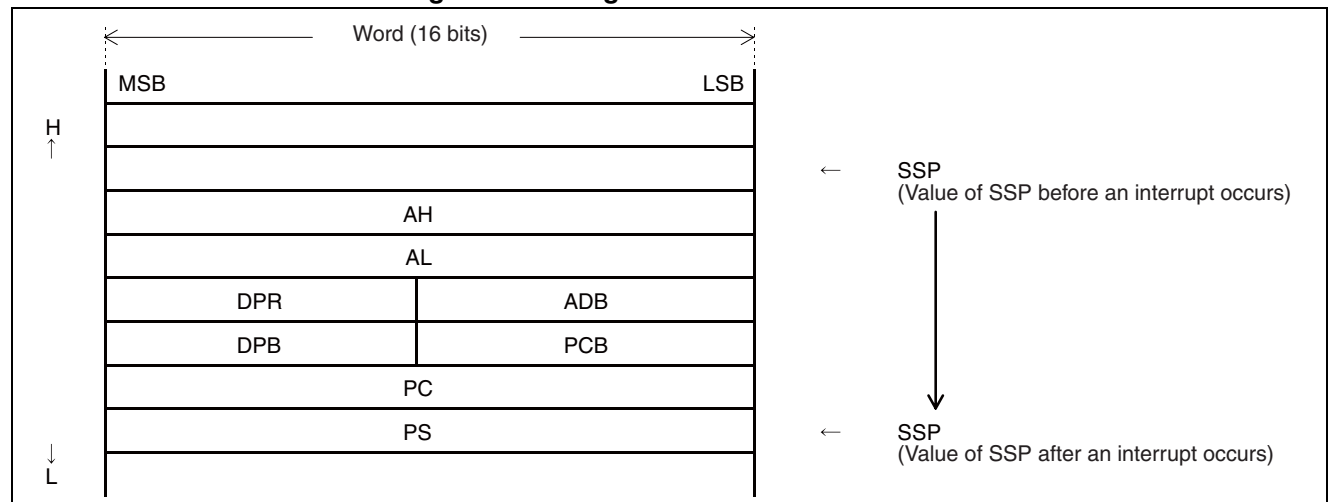
See Section "2.7 Interrupt Suppression Instructions and Prefix Codes".

■ 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 at an Interrupt

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 returning 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 returning 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.

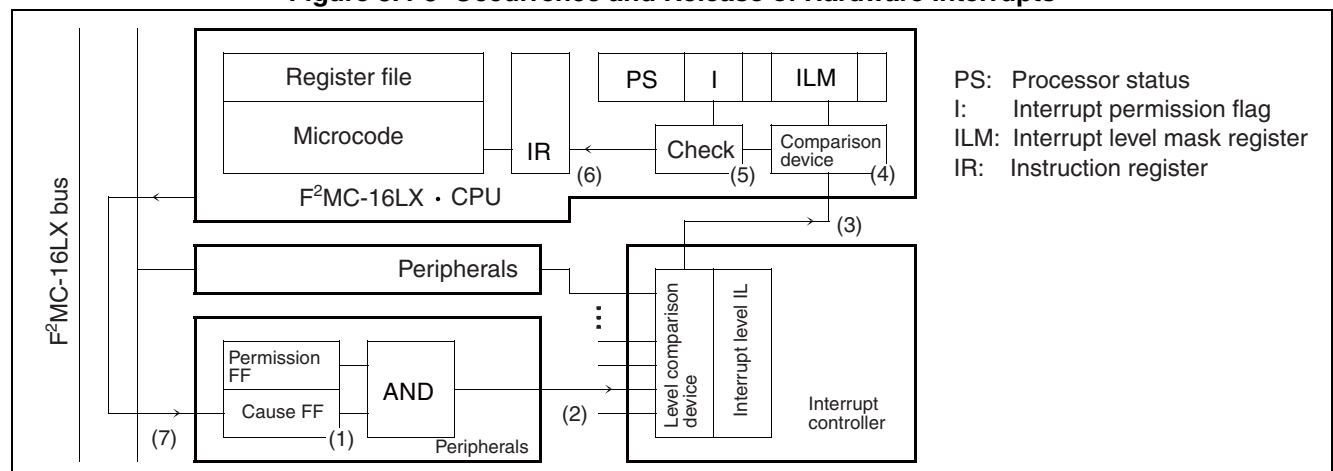
■ Operations of Hardware Interrupts

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 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 the interrupt request evaluates the priority of simultaneous interrupt requests and notifies the interrupt level for the corresponding interrupt 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.

■ Processing Time for a Hardware Interrupt

After an interrupt request occurs, receiving the interrupt and executing the interrupt processing routine takes the time required to wait for the interrupt request sample and the interrupt handling time.

○ Time to wait for the interrupt request sample

This represents the time between the occurrence of an interrupt request up to the completion of the currently executed instruction. Whether an interrupt request has occurred is determined by sampling for interrupt requests in the last cycle of each instruction. This leads to wait time.

The time to wait for an interrupt request sample is longest immediately after the POPW with the longest execution cycle or when one of the instructions PW0 to PW7 (45 machine cycles) is started.

○ Interrupt handling time (time required to perform interrupt processing)

Interrupt start : $24 + 6 \times (\text{machine cycles according to Table 3.4-1})$

Interrupt recover : $15 + 6 \times (\text{machine cycles according to Table 3.4-1})$ (RETI instruction)

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

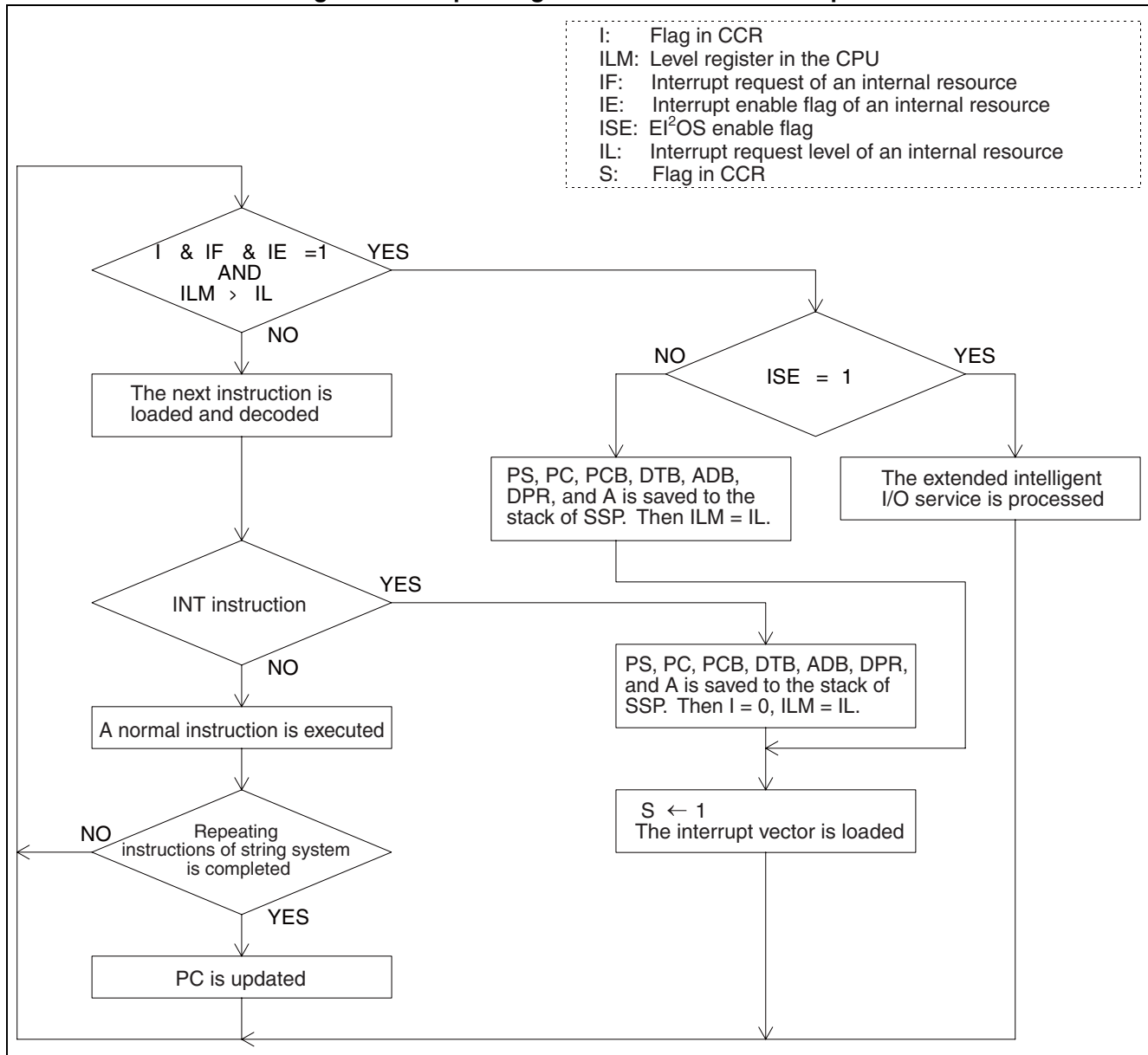
Address to which the stack pointer points	Corrective value of the number of cycles
External area 8-bit data bus	+4
External area even-numbered address	+1
External area odd-numbered address	+4
Internal area even-numbered address	0
Internal area odd-numbered address	+2

3.4.2 Operating Flow for Hardware Interrupts

Figure 3.4-4 shows the flow of operation for hardware interrupts.

■ Operating Flow for Hardware Interrupts

Figure 3.4-4 Operating Flow for Hardware Interrupts

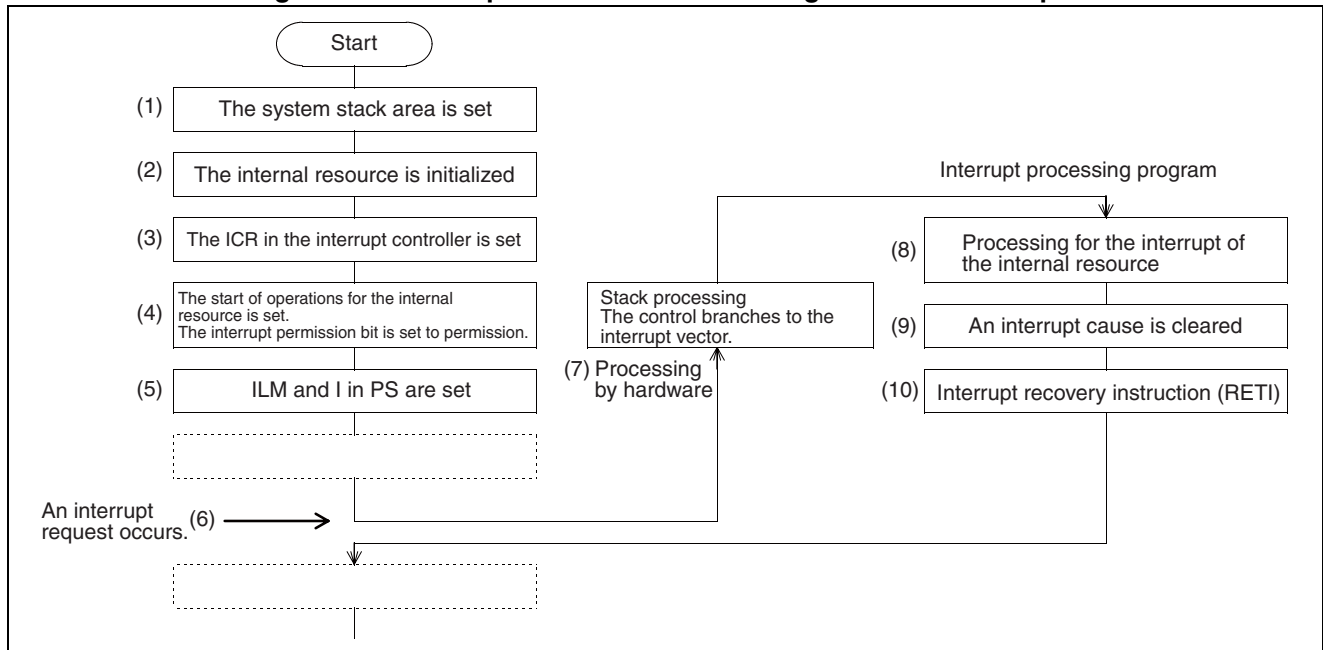


3.4.3 Example of Procedure for Using Hardware Interrupts

Figure 3.4-5 shows an example of a procedure for using hardware interrupts.

■ Example of Procedure for Using Hardware Interrupts

Figure 3.4-5 Example of Procedure for Using Hardware Interrupts



The usage of the items (1) to (10) in Figure 3.4-5 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

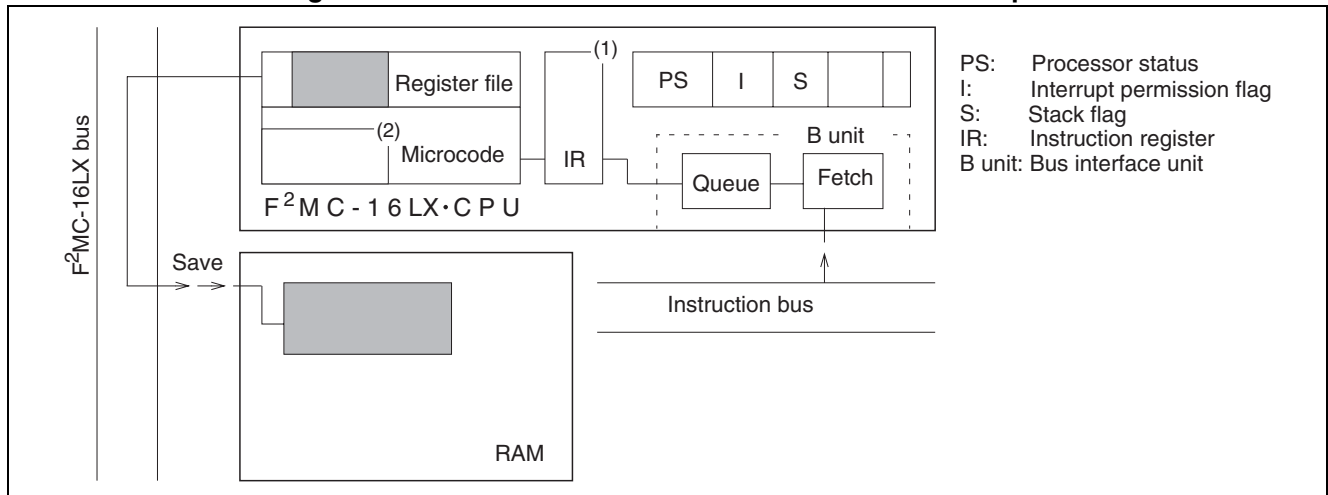
Microcode: Interrupt processing step

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

As shown in Table 3.3-1, software interrupts share the interrupt vector area with hardware interrupts. For example, interrupt request number INT 13 is used at the same time by external interrupt #0 from a hardware interrupt and INT #13 from a software interrupt. Therefore, external interrupt #0 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 Hardware Interrupts

The meanings of the items (1) to (3) in Figure 3.5-1 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.

■ 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 to specify whether a buffer address is to be updated or left unchanged.
- Capability to specify whether an I/O register address is to be updated or left unchanged when the buffer address is updated.

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

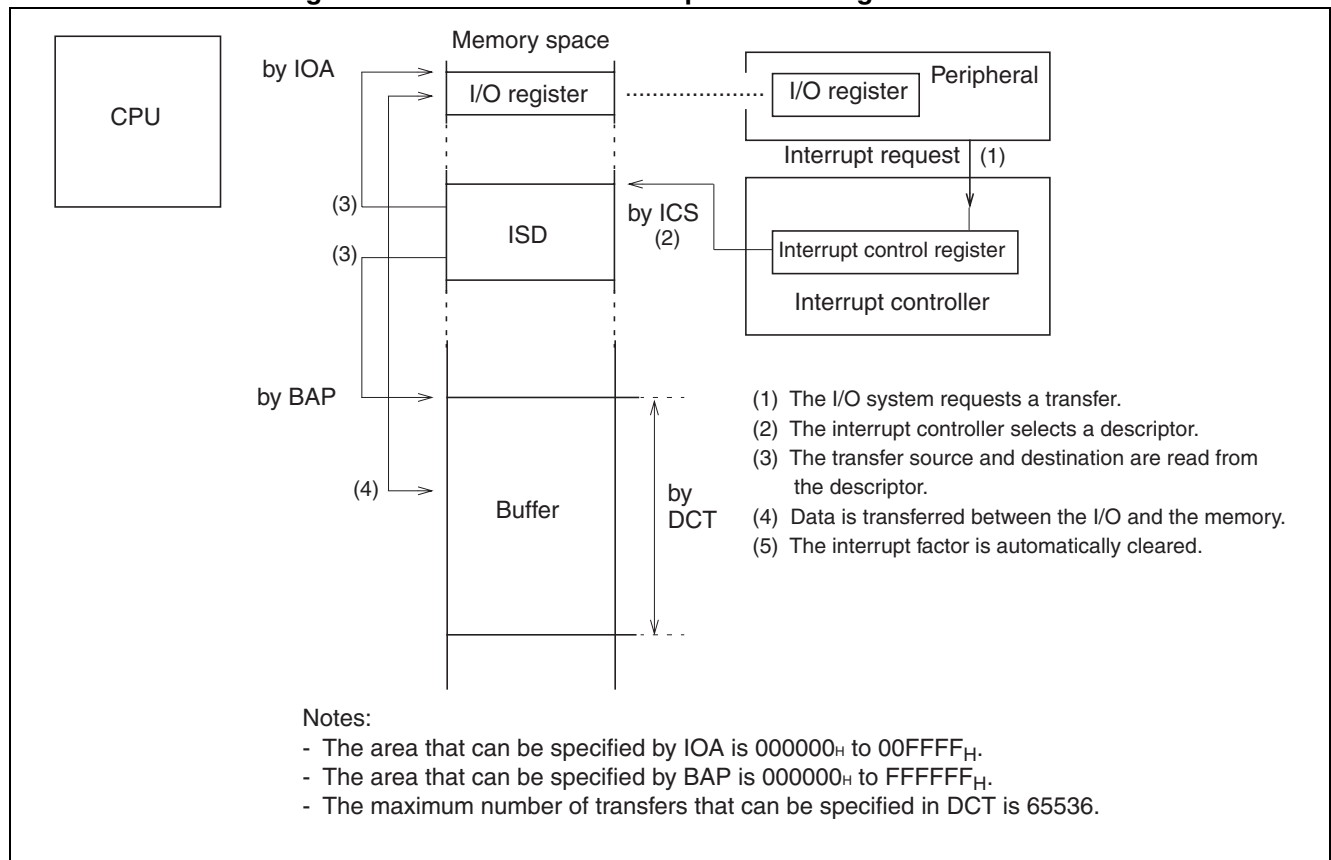
This register is located in the interrupt controller and indicates an ISD address.

○ 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.

Note:

During use of REALOS, the expanded intelligent I/O service (EI²OS) cannot be used.

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.

Microcode: Notes the steps for EI²OS 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)		bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
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)	
		bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: B0 _H to BF _H			—	—	S1	S0	ISE	IL2	IL1	IL0	When reading
Read/write ⇒			(-)	(-)	(R)	(R)	(R)	(R)	(R)	(R)	
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.

[bit15 to bit12, bit7 to bit4] 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 expended intelligent I/O service descriptor in the memory. The ICS is initialized to "0000_B" 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

[bit13, bit12, bit5, bit4] S1 and S0

S1 and S0 are the EI²OS termination status bits.

S1 and S0 are read only and allow to determine the termination condition for the termination of the EI²OS. They are initialized to "00_B" 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

[bit11, bit3] ISE

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.

[bit10 to bit8, bit2 to bit0] IL2, IL1, and IL0

The IL2, IL1, and IL0 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

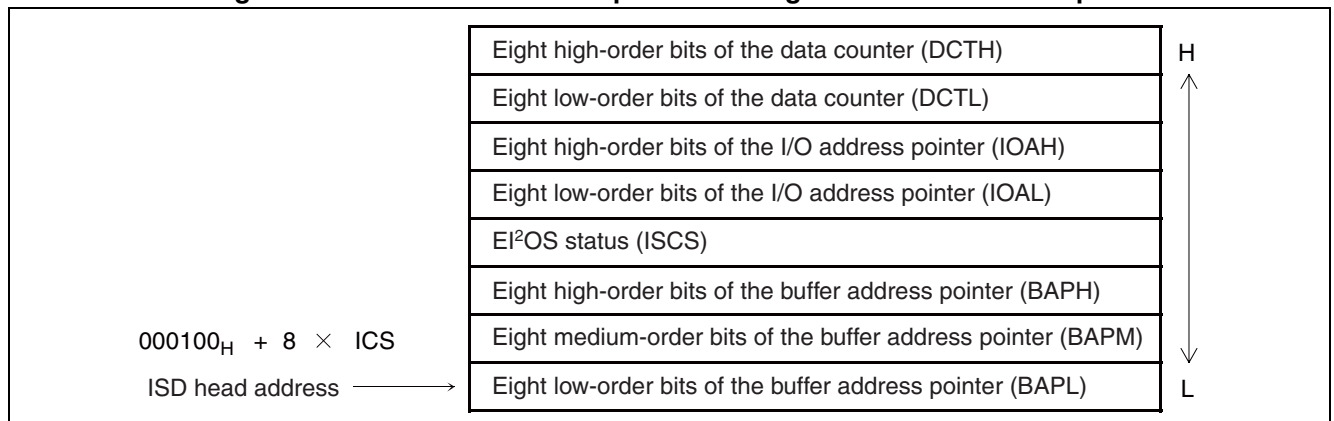
■ Expanded Intelligent I/O Service Descriptor (ISD)

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

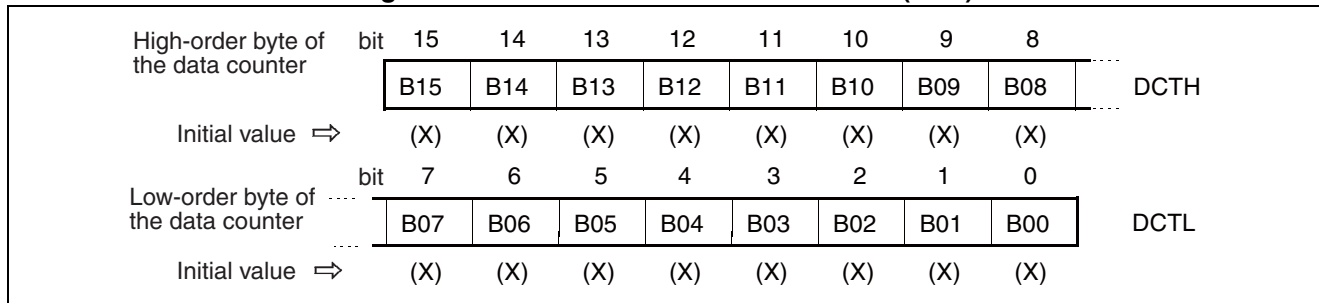
Figure 3.6-3 shows the structure of the expanded intelligent I/O service descriptor.

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



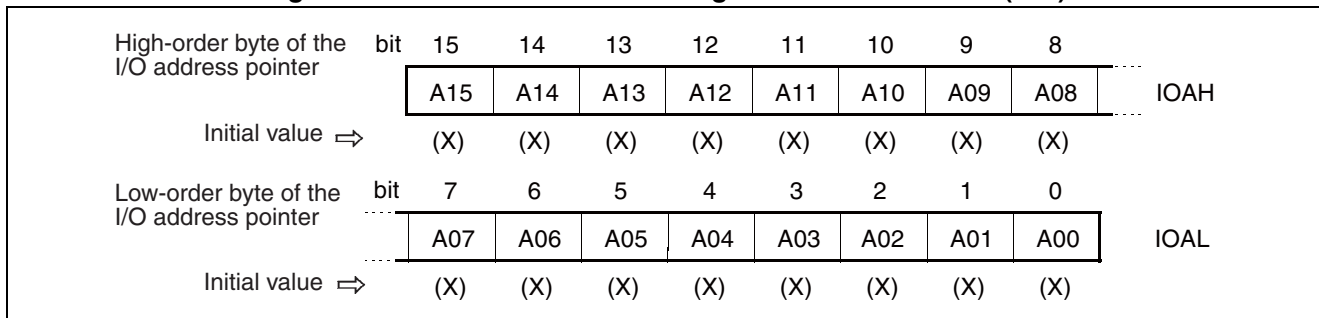
■ Data Counter (DCT)

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)

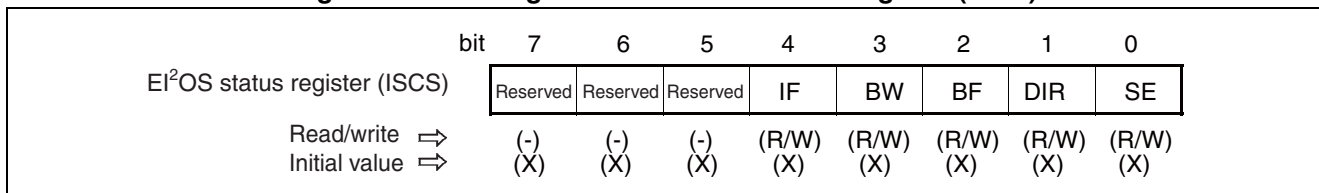
■ I/O Register Address Pointer (IOA)

The I/O register address pointer (IOA) is a 16-bit register that indicates the lower address (A15 to A00) 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)

■ 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)

[bit7 to bit5]

Reserved bits. Always set these bits to "0" when setting the ISCS.

[bit4] 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.

[bit3] 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

[bit2] 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 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.

[bit1] 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 ---> I/O address pointer

[bit0] 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 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 shows the operational flow of the expanded intelligent I/O service (EI²OS), while Figure 3.6-8 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)

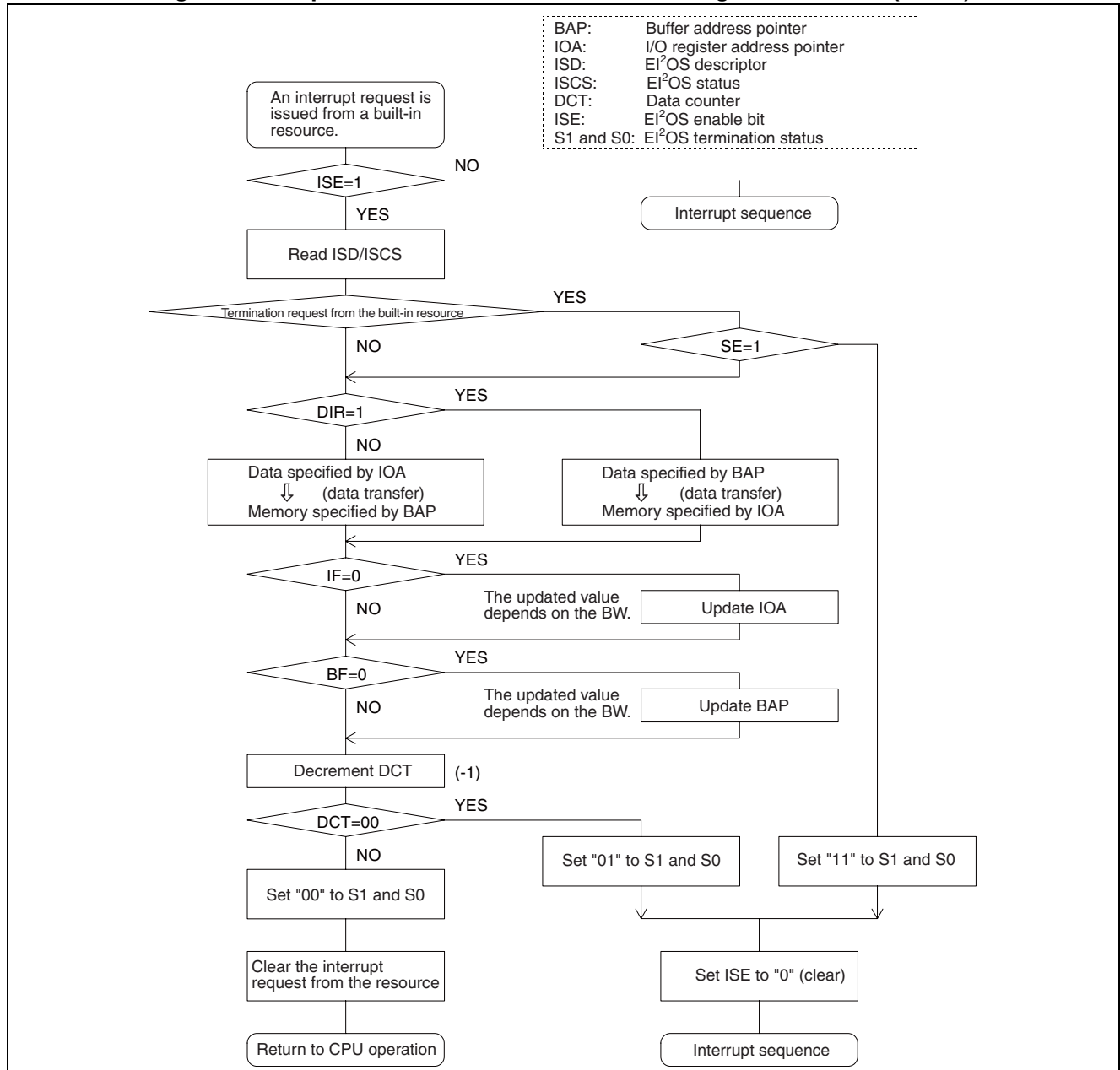
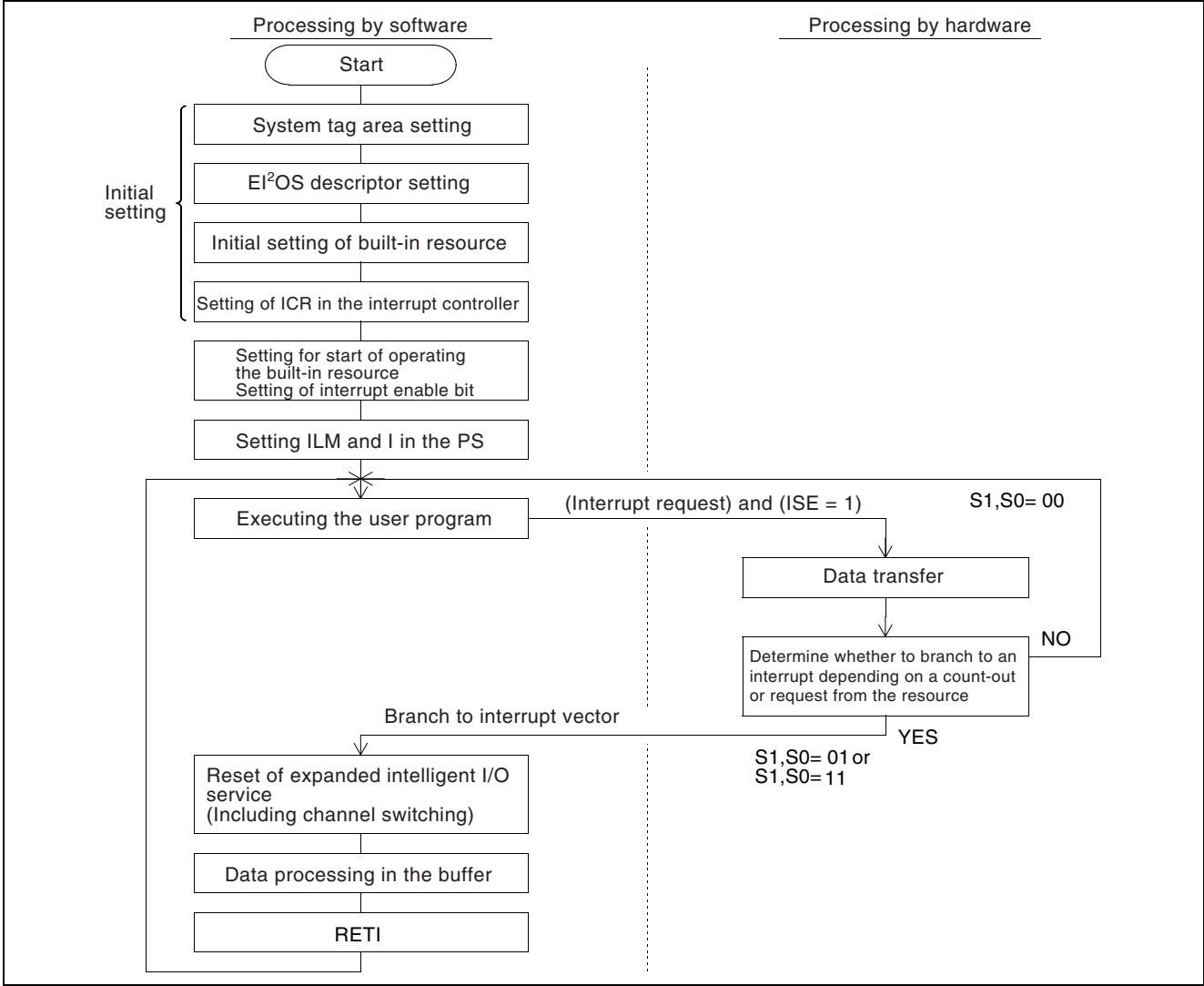


Figure 3.6-8 Procedural Flow of the Expanded Intelligent I/O Service (EI²OS)



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 + Table 3.6-10) machine cycles

Table 3.6-9 Execution Time of EI²OS During Data Transfer

Bit SE of ISCS		Set to "0"		Set to "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 × Table 3.4-1) machine cycles

- In case of a count-out

(Table 3.6-9 + Table 3.6-10 + (21 + 6 × Table 3.4-1)) 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

Even: Word transfer for even address

8: Word transfer for 8-bit external bus

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 Clock Supply Map
- 4.3 Reset Causes
- 4.4 Operation after a Reset is Released
- 4.5 Registers not Initialized by Reset Input

4.1 Clock Generator

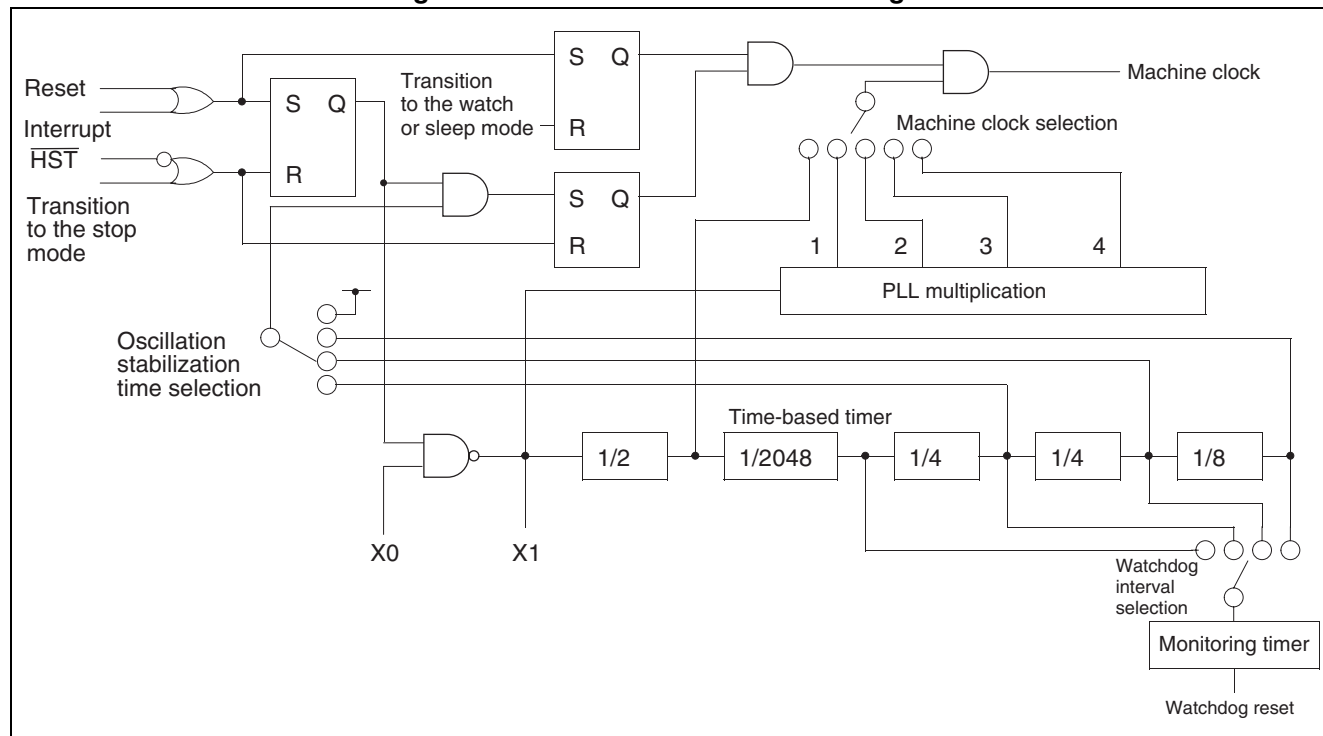
The clock generator controls internal clock operations such as the sleep, watch, and stop modes and the PLL clock multiplication function. This internal clock is called the machine clock. One cycle of the machine clock is used as a machine cycle. The clock generated by OSC oscillation is called the main 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 8 MHz. Any frequency less than this frequency cannot be specified.

Figure 4.1-1 Clock Generator Block Diagram

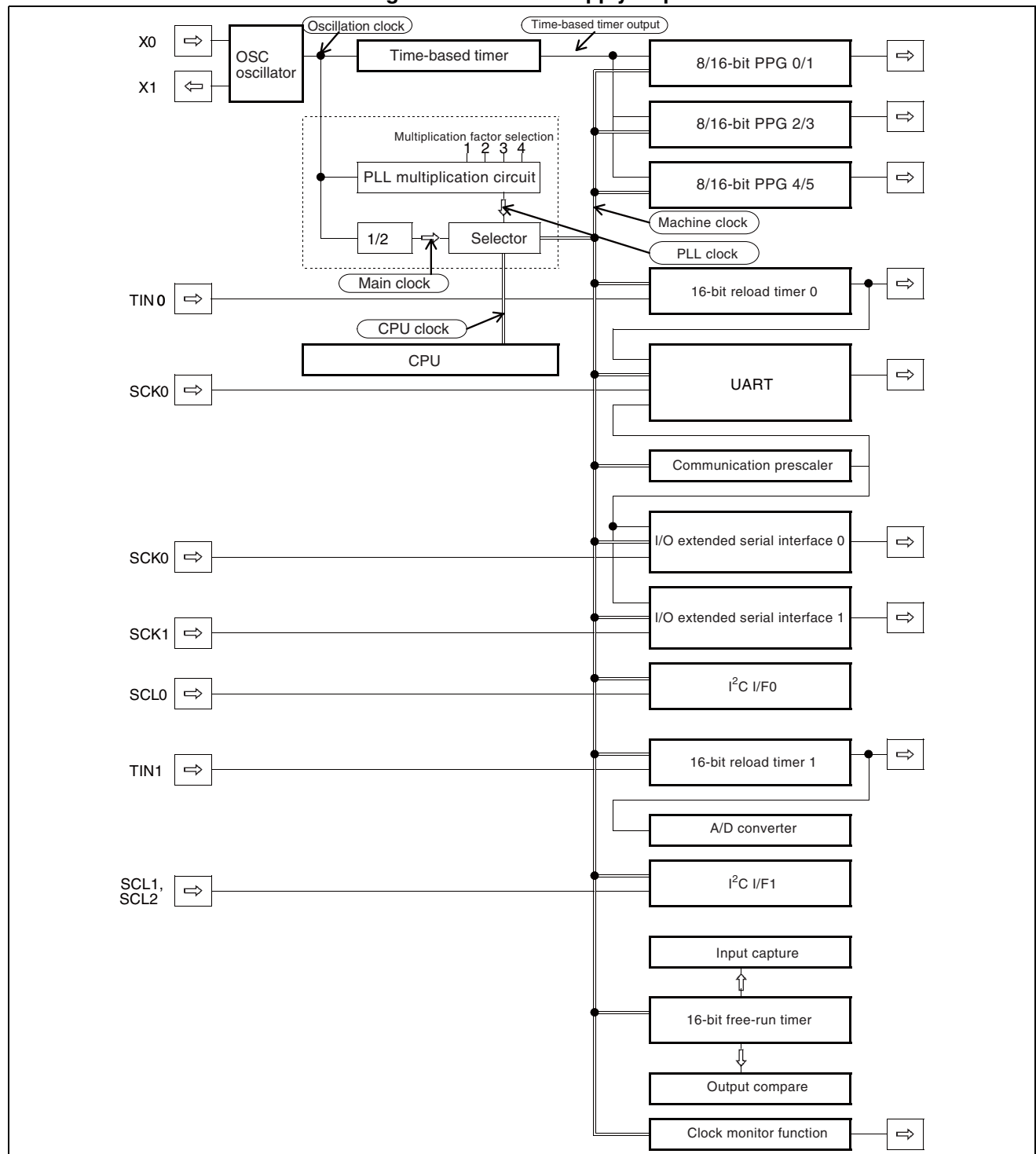


4.2 Clock Supply Map

Figure 4.2-1 shows a clock supply map.

■ Clock Supply Map

Figure 4.2-1 Clock Supply Map



4.3 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.

Notes:

- 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 $\overline{\text{WRX}}$ become inactive.

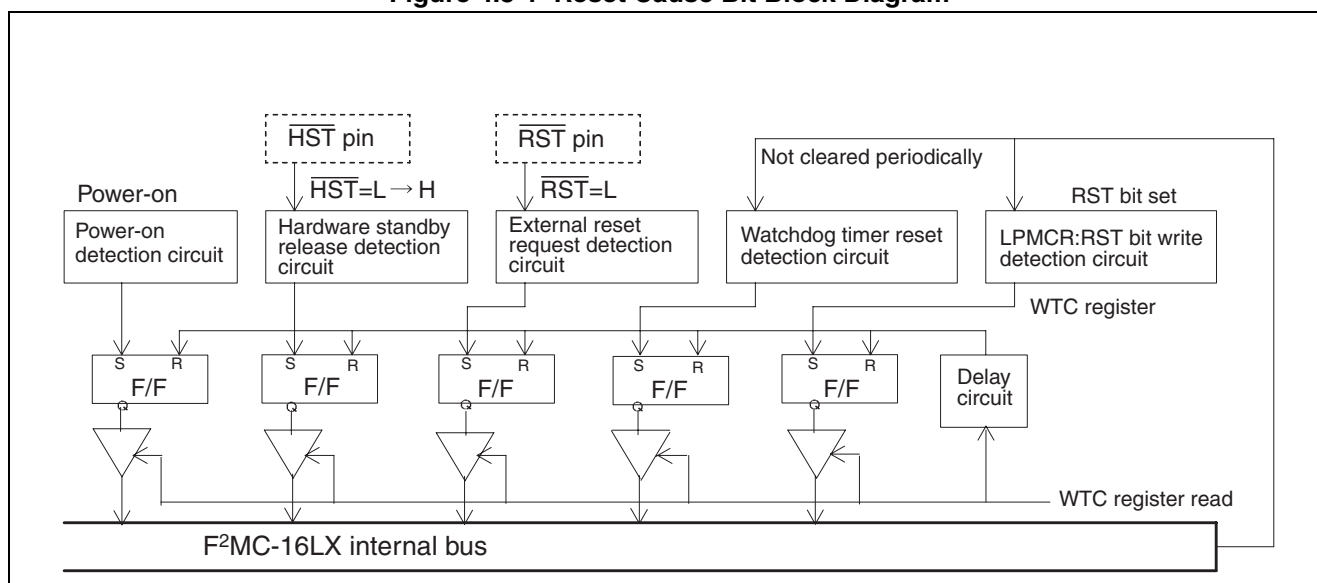
Table 4.3-1 Reset Causes

Reset	Cause	Machine clock	Watchdog timer	Oscillation stabilization time
Power-on	Power supply startup.	Main clock	Stopped	Required
Hardware standby	The $\overline{\text{HST}}$ pin input "L" level.	Main clock	Stopped	Required
Watchdog timer	The watchdog timer overflows.	Main clock	Stopped	Required
External pin	The $\overline{\text{RST}}$ pin input "L" level.	Holds previous status.	Holds previous status.	Not required
Software	Write "0" to LPMCR register RST bit.	Holds previous status.	Holds previous status.	Not required

- 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 2¹⁸ 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 4.3-1 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 4.3-2 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".

4.4 Operation after a Reset is Released

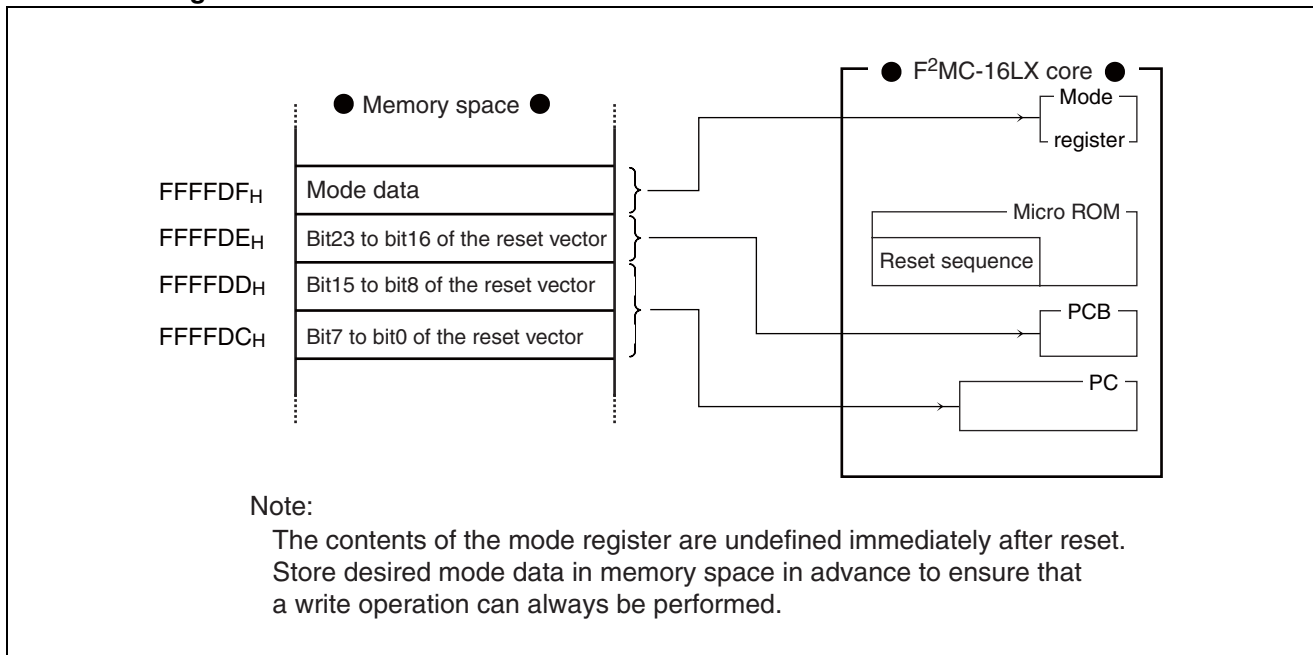
When a reset cause is removed, the F²MC-16LX immediately outputs the address at which the reset vector is stored and fetches the reset vector and mode data. A 4-byte area at FFFFDC_H to FFFFDF_H is allocated for the reset vector and mode data. The reset vector and mode data are transferred to corresponding 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 4.4-1 Locations and Destinations of the Reset Vector and of Mode Data



4.5 Registers not Initialized by Reset Input

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

Table 4.5-1 lists registers not initialized by each reset cause.

■ Registers not Initialized by Reset Input

Table 4.5-1 Registers not Initialized by Reset Input

Type of reset	CKSCR					LPMCR		WDTC
	WS1	WS0	MCS	CS1	CS0	CG1	CG0	WTE
Power-on reset	Y	Y	Y	Y	Y	Y	Y	Y
Hardware standby ($\overline{\text{HST}}$)	N	N	Y	N	N	Y	Y	Y
Watchdog reset	N	N	Y	N	N	Y	Y	Y
External reset ($\overline{\text{RST}}$)	N	N	N	N	N	N	N	N
Software reset	N	N	N	N	N	N	N	N

Y: Initialized

N: Not initialized (retains the status before reset.)

[Bit explanation]

- WS1 and WS0: Sets the oscillation stabilization time for the main clock.
- MCS: Sets the machine clock. (0: PLL clock, 1: Main clock)
- CS1 and CS0: Sets the multiplication factor for the PLL clock.
- CG1 and CG0: Sets intermittent CPU operation.
- WTE: Sets watchdog timer start.

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 $\overline{\text{RST}}$ 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 $\overline{\text{HST}}$ plus $\overline{\text{RST}}$ (connecting $\overline{\text{HST}}$ and $\overline{\text{RST}}$ with a jumper) is recommended.

Table 4.5-2 lists registers that are not initialized by reset input using $\overline{\text{HST}}$ plus $\overline{\text{RST}}$. Note that the operation status after the reset is released differs depending on the reset input type, $\overline{\text{HST}}$ plus $\overline{\text{RST}}$ reset input, or only $\overline{\text{RST}}$ input, as listed in Table 4.5-2.

Table 4.5-2 Registers not Initialized by Reset Input

Type of reset	CKSCR					LPMCR		WDTC
	WS1	WS0	MCS	CS1	CS0	CG1	CG0	WTE
$\overline{\text{RST}}$ and $\overline{\text{HST}}$ connection	N	N	Y	N	N	Y	Y	Y

Y: Initialized

N: Not initialized (retains the status before reset).

Figure 4.5-1 Operation Transition by Reset Input

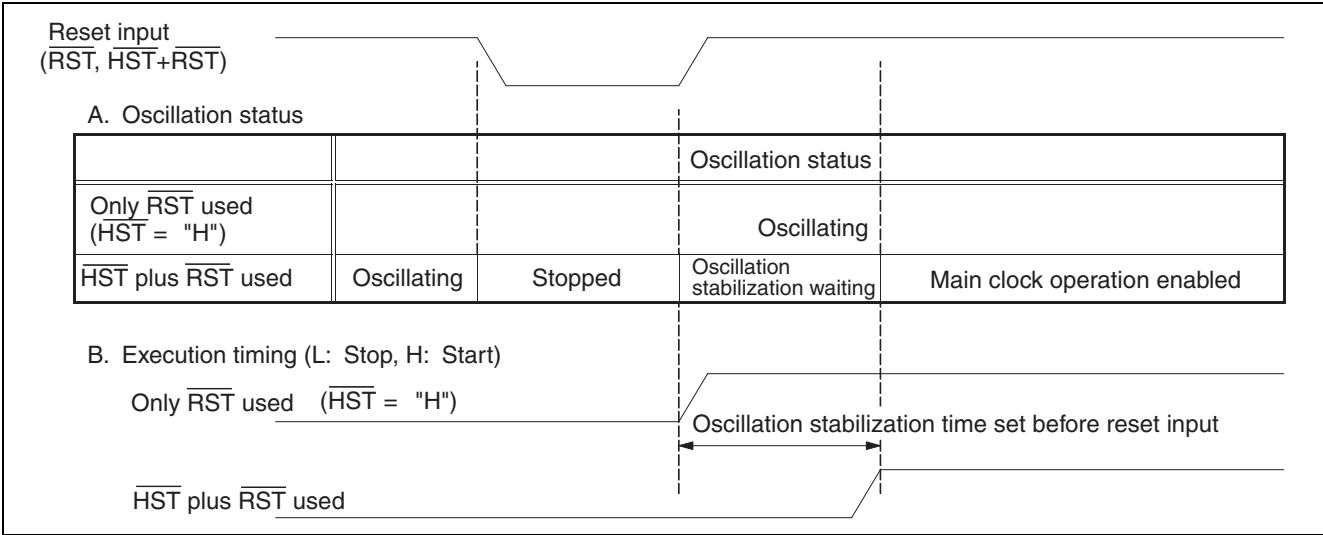
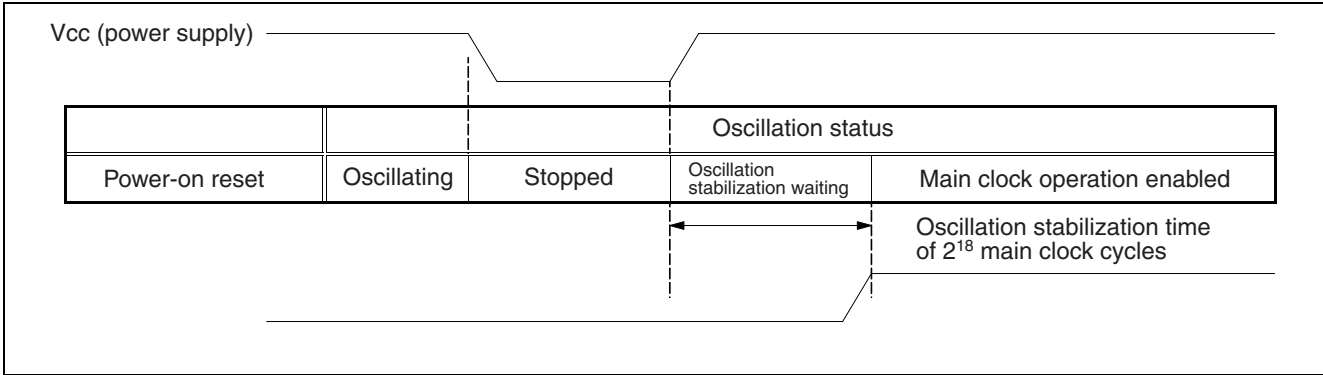


Figure 4.5-2 Transition at a Power-on Reset



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
- 5.7 Machine Clock

5.1 Overview of the Low-power Consumption Control Circuit

The operating modes are as follows:

- PLL clock mode
- PLL sleep mode
- Watch mode
- Main clock mode
- Main sleep mode
- Stop 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.

Note:

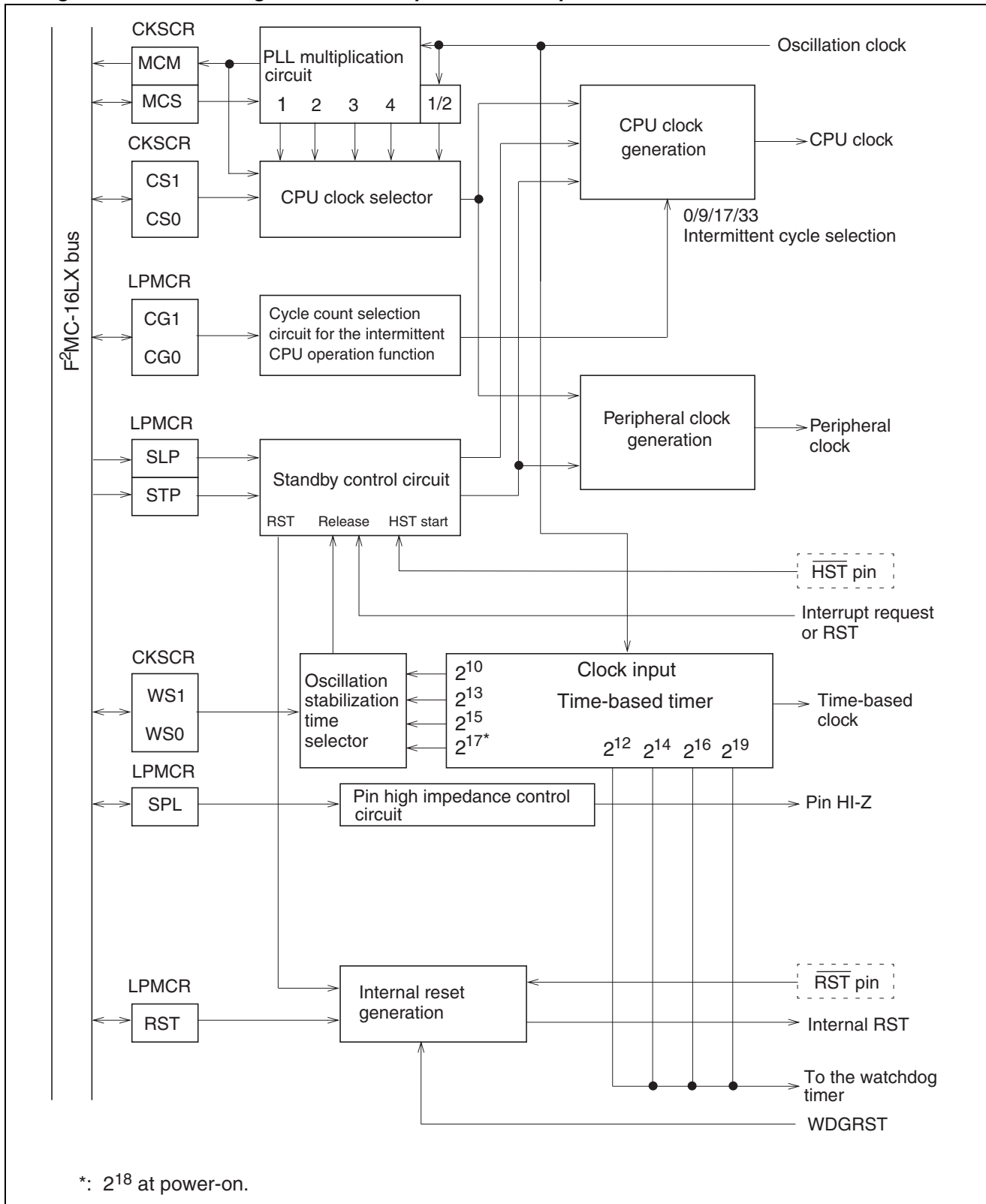
During switching of the clock mode, do not attempt to switch to another clock mode or the low power consumption mode until switching is completed. To verify that switching has completed, check the MCM bit of the clock selection register (CKSCR).

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

Low-power consumption mode control register		bit	7	6	5	4	3	2	1	0	
Address: 0000A0 _H			STP	SLP	SPL	RST	Reserved	CG1	CG0	Reserved	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	15	14	13	12	11	10	9	8	
Address: 0000A1 _H			Reserved	MCM	WS1	WS0	Reserved	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 Address: 0000A0 _H	bit	7	6	5	4	3	2	1	0	LPMCR
		STP	SLP	SPL	RST	Reserved	CG1	CG0	Reserved	
	Read/write ⇒	(W)	(W)	(R/W)	(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 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,#immRi
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	

[bit7] STP

Writing "1" in the STP bit causes a transition to the watch mode (when CKSCR:MCS is "0") or stop mode (when CKSCR:MCS is "1"). 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".

[bit6] 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". Simultaneously writing "1" in the STP and SLP bits causes a transition to the watch or stop mode. This bit is a write only bit. The read value is always "0".

[bit5] 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.

[bit4] 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".

[bit3] Reserved

Be sure to set this bit to "1".

[bit2, bit1] 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.

Table 5.2-2 Settings of the Bits for Setting the Clock Temporary Stop Cycle Count (CG1 and CG0)

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)

[bit0] Reserved

Be sure to set this bit to "0".

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	bit	15	14	13	12	11	10	9	8	
Address: 0000A1 _H		Reserved	MCM	WS1	WS0	Reserved	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)	

[bit15] Reserved

Be sure to set this bit to "1".

[bit14] 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. When MCS is "0" and MCM is "1", the PLL clock is in the oscillation stabilization wait state. The oscillation stabilization time for the PLL clock is fixed to 2^{13} main clock cycles.

[bit13, bit12] WS1, WS0

These bits set the oscillation stabilization time for the main clock required 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 Clock Selection Register (bit13, bit12)

WS1	WS0	Oscillation stabilization time (OSC oscillation: 4 MHz)
0	0	About 256 μ s (2^{10} OSC oscillation cycles)
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) *

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

[bit11] Reserved

Be sure to set this bit to "1".

[bit10] MCS

This bit specifies whether the main or the PLL clock is to be selected as the machine clock. Writing "0" selects the PLL clock. Writing "1" selects the main clock. When this bit is "1", writing "0" automatically clears the time-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.)

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

Note:

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.

[bit9, bit8] CS1 and CS0

These bits 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 Clock Selection Register (bit13 , bit12)

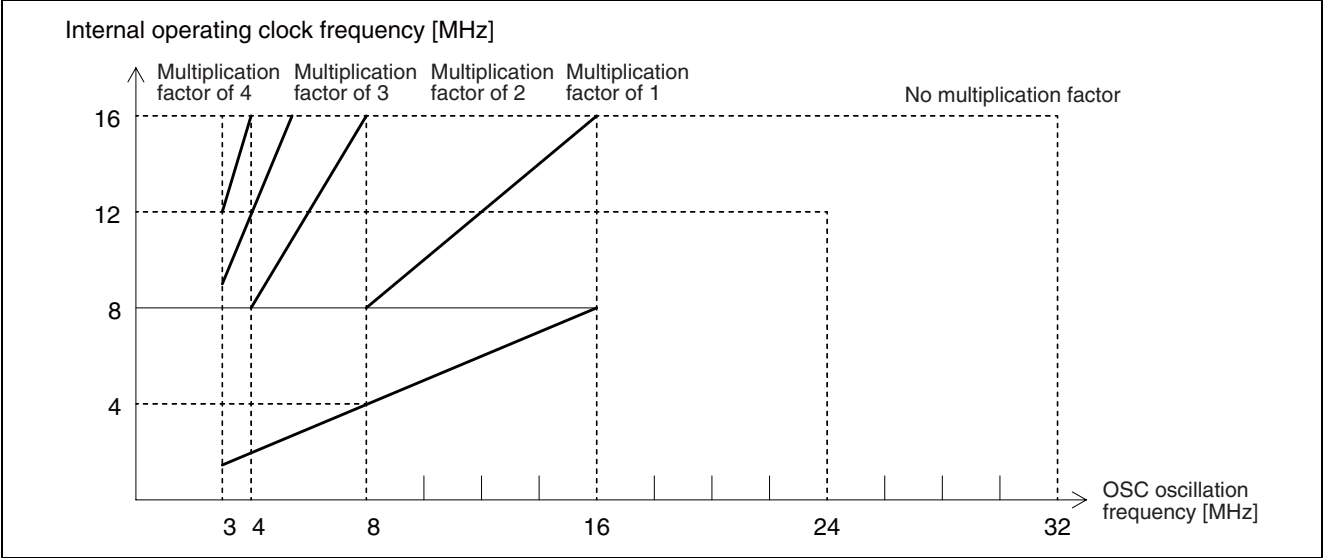
CS1	CS0	Multiplication factor	Internal operating clock (obtained by multiplying OSC oscillation by the multiplication factor)		
			When the OSC oscillation frequency is 4 MHz	When the OSC oscillation frequency is 8 MHz	When the OSC oscillation frequency is 16 MHz
0	0	1	Setting prohibited	8 MHz	16 MHz
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 lists the operating states in the low-power consumption mode. Table 5.4-1 shows the status transition diagram in the low-power consumption mode.

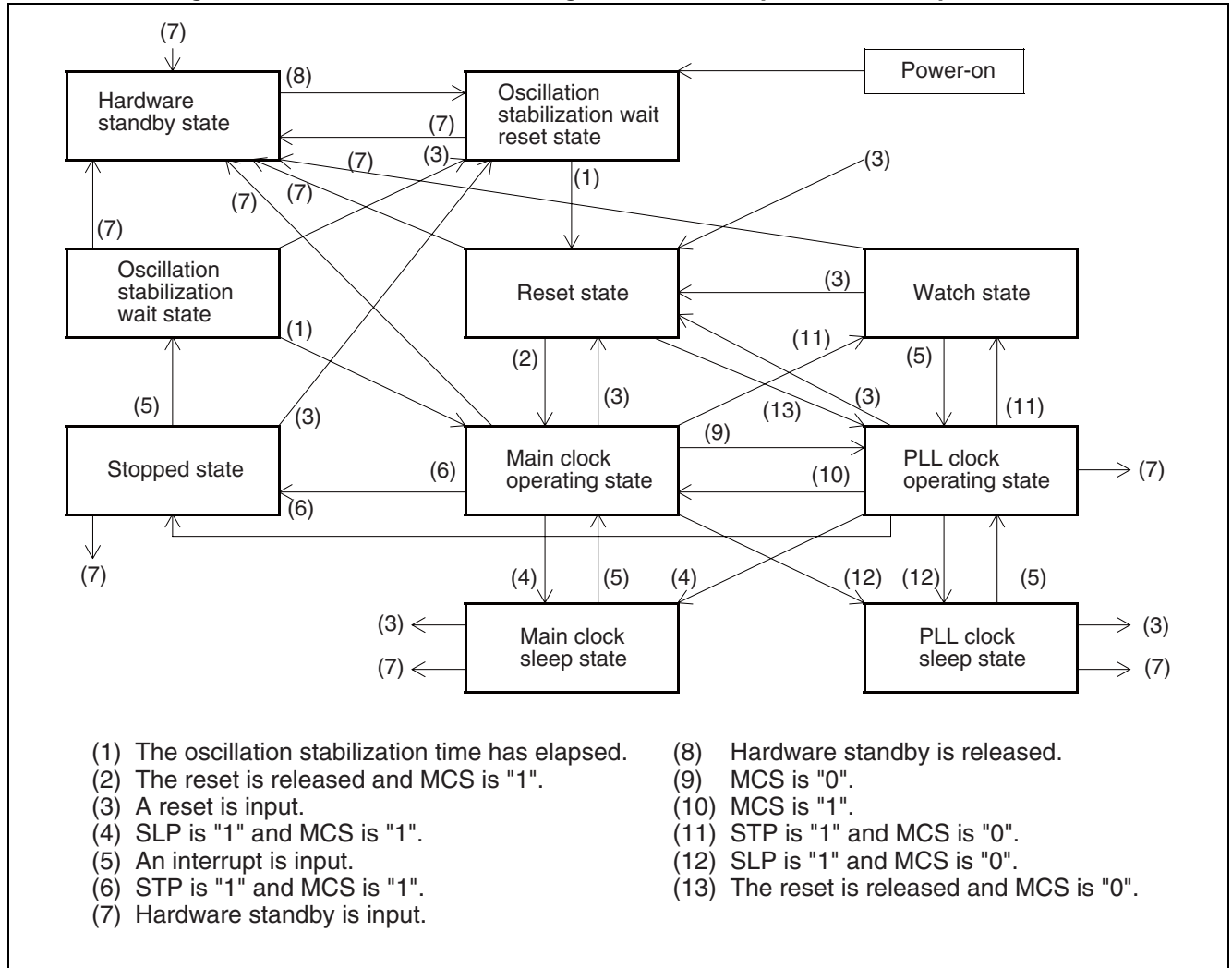
■ Operation of the Low-power Consumption Control Circuit

Table 5.4-1 Operating States in the Low-power Consumption Mode

	Transition condition	Oscillation/ time-based timer	CPU	Peripherals (machine clock)	Pins	Released by
Main sleep	MCS=1 SLP=1	Operating	Stopped	Operating	Operating	Reset interrupt
PLL sleep	MCS=0 SLP=1	Operating	Stopped	Operating	Operating	Reset interrupt
Watch (SPL=0)	MCS=0 SLP=1	Operating	Stopped	Stopped	Retained	Reset interrupt
Watch (SPL=1)	MCS=0 SLP=1	Operating	Stopped	Stopped	HI-Z	Reset interrupt
Stop (SPL=0)	MCS=1 SLP=1	Stopped	Stopped	Stopped	Retained	Reset interrupt
Stop (SPL=1)	MCS=1 SLP=1	Stopped	Stopped	Stopped	HI-Z	Reset interrupt
Hardware standby	$\overline{\text{HST}}=\text{L}$	Stopped	Stopped	Stopped	HI-Z	$\overline{\text{HST}}=\text{H}$

Note:

During switching of the clock mode, do not attempt to switch to another clock mode or the low power consumption mode until switching is completed. To verify that switching has completed, check the MCM bit of the clock selection register (CKSCR).

Figure 5.4-1 Status Transition Diagram in the Low-power Consumption Mode

5.4.1 Sleep Mode

Writing "1" in the SLP bit and "0" in the STP bit in the low-power consumption mode control register sets the standby control circuit to the sleep mode. In the sleep mode, only the clock supplied to the CPU is stopped. The CPU stops and the peripheral circuits continue operation.

■ Transition to the Sleep Mode

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

In the sleep mode, the contents of dedicated registers such as the accumulator and internal RAM are retained. The external bus hold function also operates in the sleep mode. When a hold request is issued, the external bus enters the hold state.

■ Releasing the Sleep Mode

The standby control circuit releases the sleep mode when a reset is input or an interrupt occurs. After this circuit releases the sleep mode by a reset cause, it enters the reset state. When a peripheral circuit or another device generates an interrupt request at interrupt level 7 or higher in the sleep mode, the standby control circuit releases the sleep mode. After the sleep mode is released, the interrupt is processed in the same way as normal interrupts. The interrupt enable state may be set by the settings of the I flag, ILM, and interrupt control register (ICR). In this case, the CPU executes interrupt processing according to the external interrupt cause. In the interrupt disable state, the CPU continues processing from the instruction following the instruction causing the sleep mode.

Note:

Normally, the CPU executes interrupt processing after executing the instruction following the instruction causing the sleep mode. If a transition to the sleep mode and acceptance of an external bus hold request occur simultaneously, the CPU may execute interrupt processing before executing the next instruction.

5.4.2 Watch Mode

In the watch mode, operation of something other than the OSC oscillation and time-based timer is stopped, and most chip functions stop. Whether to retain the status of each I/O pin immediately before the watch mode or set it to high impedance in the watch mode can be specified. Specify this using the SPL bit in the low-power consumption mode control register.

■ Transition to the Watch Mode

Under the following condition, writing "1" in the STP bit in the low-power consumption mode control register sets the standby control circuit to the watch mode:

- The MCS bit in the clock selection register is "0".

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

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

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 $\overline{\text{HAK}}$ 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 is input or an interrupt occurs. When this circuit releases the watch mode by a reset cause, it enters the reset state. At return from the watch mode, the standby control circuit first releases the watch mode, then waits for PLL clock oscillation to become stable. The MCS bit is not cleared by an external reset. For this reason, if the reset period is shorter than the oscillation stabilization time for the PLL clock, the reset sequence is performed using the main clock. In this case, the oscillation stabilization time for the PLL clock is 2^{13} main clock cycles to as many main clock cycles as the number obtained by multiplying 3 by 2^{13} . It depends on the time-based timer status because the timer is not cleared.

When a peripheral circuit or another device generates an interrupt request at interrupt level 7 or higher in the watch mode, the standby control circuit releases the watch mode. After the watch mode is released, the interrupt is processed in the same way as normal interrupts. The interrupt enable state may be set by the settings of the I flag, ILM, and interrupt control register (ICR). In this case, the CPU executes interrupt processing according to the external interrupt cause. In the interrupt disable state, the CPU continues processing from the next instruction before the watch mode.

Note:

Normally, the CPU executes interrupt processing after executing the instruction following the instruction causing the watch mode. If a transition to the watch mode and acceptance of an external bus hold request occur simultaneously, the CPU may execute interrupt processing before executing the next instruction.

After the watch mode is released, the standby control circuit waits for PLL clock oscillation to become stable. When the PLL clock is not used, set the MCS bit to "1" using the instruction immediately after the reset or interrupt destination.

To release the external interrupt clock mode, set the external interrupt request level to "H". The "L"-level interrupt setting may cause malfunctions. The clock mode cannot be released by an edge interrupt request.

5.4.3 Stop Mode

In the stop mode, OSC oscillation is stopped and all chip functions stop. Therefore, data can be retained with the lowest power consumption.

Retention of the status of each I/O pin immediately before the stop mode or a high impedance setting in the stop mode can be specified using the SPL bit in the LPMCR.

■ Transition to the Stop Mode

Under the following conditions, writing "1" in the STP bit in the low-power consumption mode control register sets the standby control circuit to the stop mode:

- The MCS bit in the clock selection register is "1".

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

In the stop mode, the contents of dedicated registers such as the accumulator and internal RAM are retained. In the stop 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 stop mode, the \overline{HAK} signal may not become low with the bus set to high impedance.

■ Releasing the Stop Mode

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

At return from the stop mode, the standby control circuit first enters the oscillation stabilization wait state, then releases the stop mode. For this reason, the reset sequence is also performed after the oscillation stabilization time has elapsed when the stop mode is released by a reset cause.

When a peripheral circuit or another device generates an interrupt request at interrupt level 7 or higher in the stop mode, the standby control circuit releases the stop mode. After the stop mode is released, the interrupt is processed in the same way as normal interrupts when the oscillation stabilization time for the main clock has elapsed. The oscillation stabilization time is specified by the WS1 and WS0 bits in the CKSCR. The interrupt enable state may be set by the settings of the I flag, ILM, and interrupt control register (ICR). In this case, the CPU executes interrupt processing according to the external interrupt cause. In the interrupt disable state, the CPU continues processing from the next instruction before the stop mode.

Note:

Normally, the CPU executes interrupt processing after executing the instruction following the instruction causing the stop mode. If a transition to the stop mode and acceptance of an external bus hold request occur simultaneously, the CPU may execute interrupt processing before executing the next instruction.

To release the external interrupt stop mode, set the external interrupt request level to "H". The "L"-level interrupt setting may cause malfunctions. The stop mode cannot be released by an edge interrupt request.

5.4.4 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.

5.4.5 Pin status in the Sleep, Stop, Hold, Reset, and Hardware Standby Modes

Table 5.4-2 lists the status of each pin in the single chip mode. Table 5.4-3 lists the status of each pin in the external bus 16-bit data bus mode. Table 5.4-4 lists the status of each pin in the external bus 8-bit data bus mode.

■ Status of Each Pin in the Single Chip Mode

Table 5.4-2 Status of Each Pin in the Single Chip Mode

Pin name	Sleep	Stop ^{*6}		Hold	Reset	Hardware standby ^{*6}
		SPL=0	SPL=1			
P07 to P00, P17 to P10, P27 to P20, P37 to P30, P47 to P40, P57 to P50, P67 to P60, P77 to P70, P87 to P80, P97 to P90, PA4 to PA0	Status before the mode retained ^{*2}	Input interrupted ^{*4} /status before the mode retained	Input interrupted ^{*4} /output Hi-Z ^{*5}	This status does not occur.	Input disabled ^{*3} /output Hi-Z ^{*5}	Input interrupted ^{*4} /output Hi-Z ^{*5}
P77 to P70		Input enabled ^{*1}			Input enabled ^{*1}	

- ^{*1} "Input enabled" means that the input function is available. The pull-up or pull-down option or external input is required. When the pin is used as an output port, the status is the same as other ports.
- ^{*2} "Status before the mode retained" means that the status output immediately before the mode is output as is or input is enabled. Outputting the status output immediately before the mode as is means that output is performed according to the internal peripheral with output when it is operating or output as a port is retained.
- ^{*3} "Input disabled" means that operation of the input gate that is closest to the pin is enabled, but the input from the pin cannot internally be accepted because the internal circuit does not operate.
- ^{*4} In input interrupted status, input is masked, and "L" level transmitted internally.
- ^{*5} "Output Hi-Z" means that the pin driving transistor is placed in the drive prohibited state to set the pin to high impedance.
- ^{*6} The pull-up option is disconnected from each port pin in the stop or hardware standby mode.

Note:

In the stop and hardware standby modes, up to eight external signals can be input and the frequency of each signal must be 1 kHz or less.

■ Status of Each Pin in the External Bus 16-bit Data Bus Mode

Table 5.4-3 Status of Each Pin in the External Bus 16-Bit Data Bus Mode

Pin name	Sleep	Stop		Hold	Reset	Hardware standby
		SPL=0	SPL=1			
P07 to P00 (AD07 to AD00), P17 to P10 (AD15 to AD08)	Input disabled/ output Hi-Z	Input interrupted/ output Hi-Z	Input interrupted/ output Hi-Z	Input disabled/ output Hi-Z	Input disabled/ output Hi-Z	Input interrupted/ output Hi-Z
P27 to P20 (A23 to A16)	Output state * ₁	Output state * ₁			Output state * ₁	
P37 (CLK)	Input disabled/ output enabled * ₂	Input disabled/ output state * ₁		Input disabled/ output enabled * ₂	Input disabled/ output Hi-Z	
P36 (RDY)	Status before the mode retained	Input interrupted/ status before the mode retained		Input disabled/ output Hi-Z		
P35 (HAK)				"L" output		
P34 (HRQ)				"1" input		
P33 (WRH)	"H" output	"H" output		Input disabled/ output Hi-Z		
P32 (WRL)						
P31 (RD)						
P30 (ALE)	"L" output	"L" output		Status before the mode retained	Input disabled/ output Hi-Z	
P47 to P40, P55 to P50, P67 to P60, P87 to P80, P97 to P90, PA4 to PA0	Status before the mode retained	Input interrupted/ status before the mode retained				
P77 to P70		Input enabled			Input enabled	

*1 "Output state" means that the pin driving transistor is placed in the drive enabled state, but a fixed value, "H" or "L", is output because internal circuit operation stops. When the internal peripheral circuit is operating and the output function is used, output changes.

*2 "Output enabled" means that the pin driving transistor is placed in the drive enabled state and the operation status appears at the pin because internal circuit operation is enabled.

■ Status of Each Pin in the External Bus 8-bit Data Bus Mode

Table 5.4-4 Status of Each Pin in the External Bus 8-Bit Data Bus Mode

Pin name	Sleep	Stop		Hold	Reset	Hardware standby
		SPL=0	SPL=1			
P07 to P00 (AD07 to AD00)	Input disabled/ output Hi-Z	Input interrupted/ output Hi-Z	Input interrupted/ output Hi-Z	Input disabled/ output Hi-Z	Input disabled/ output Hi-Z	Input interrupted/ output Hi-Z
P17 to P10 (AD15 to AD08), P23 to P20 (A23 to A16)	Output state	Output state			Output state	
P37 (CLK)	Input disabled/ output enabled	Input disabled/ output state		Input disabled/ output enabled	Input disabled/ output Hi-Z	
P36 (RDY)	Status before the mode retained	Input interrupted/ status before the mode retained		Input disabled/ output Hi-Z		
P35 ($\overline{\text{HAK}}$)				"L" output		
P34 (HRQ)				"1" input		
P33				Status before the mode retained		
P32 ($\overline{\text{WR}}$)	"H" output	"H" output		Input disabled/ output Hi-Z	"H" output	
P31 ($\overline{\text{RD}}$)					"L" output	
P30 (ALE)	"L" output	"L" output		Status before the mode retained	Input disabled/ output Hi-Z	
P27 to P24, P47 to P40, P55 to P50, P67 to P60, P87 to P80, P97 to P90, PA4 to PA0	Status before the mode retained	Input interrupted/ status before the mode retained				
P77 to P70		Input enabled			Input enabled	

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

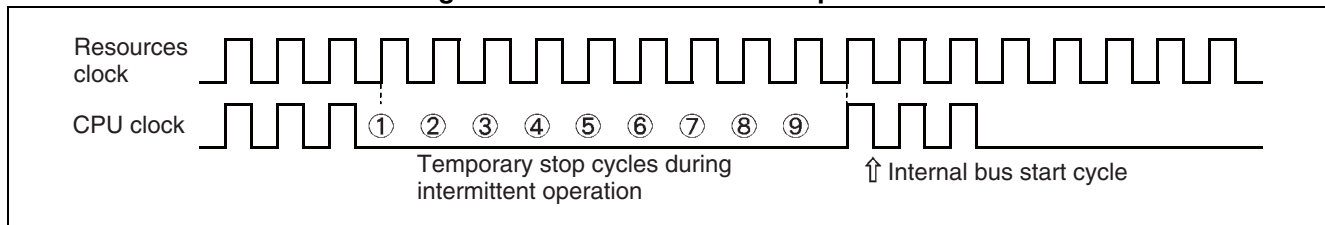
The intermittent CPU operation function can be used to perform processing with low-power consumption because the CPU execution speed is lowered by supplying the internal resources with a high-speed clock. Use the CG1 and CG0 bits to select the temporary stop cycle count for the clock supplied to the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

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

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

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.)

5.7 Machine Clock

Switch the machine clock by setting the MCS bit in the clock selection register (CKSCR).

■ Machine Clock Initialization

The MCS bit is not initialized by a reset caused by the $\overline{\text{RST}}$ pin or RST bit. The bit is initialized to "1" by a power-on, hardware standby, or watchdog reset.

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, 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 8 MHz, which is the minimum frequency that can be specified.

■ Switching the Machine Clock

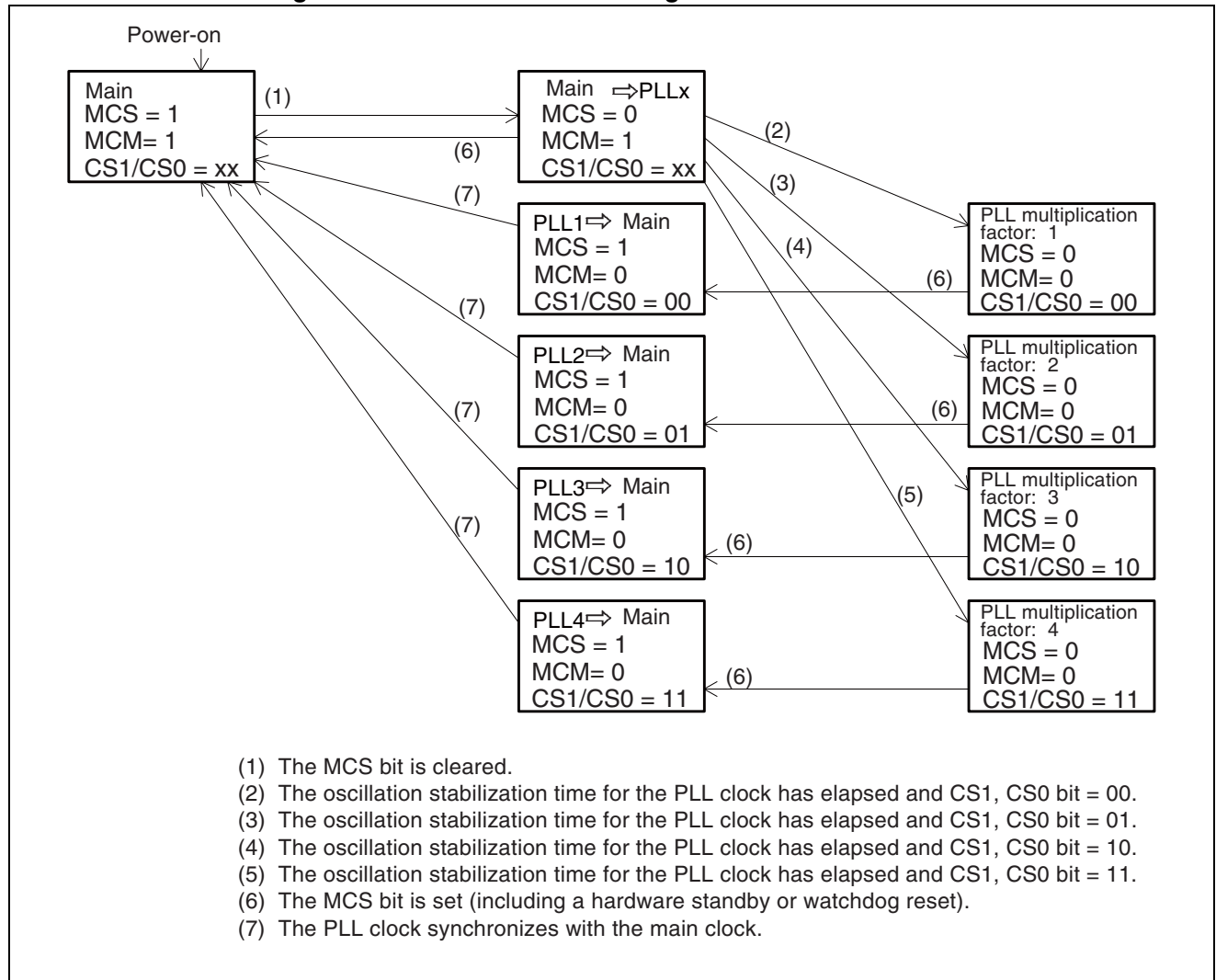
Writing to the MCS bit in the CKSCR register switches between the main and PLL clocks. Setting the MCS bit from "1" to "0" switches from the main clock to the PLL clock after the oscillation stabilization time for the PLL clock (2^{13} machine clock cycles) has elapsed.

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

When the MCS bit is rewritten, the machine clock is not immediately switched. Operate each peripheral depending on the machine clock after referencing the MCM bit to check that the machine clock has been switched.

Note:

During switching of the clock mode, do not attempt to switch to another clock mode or the low power consumption mode until switching is completed. To verify that switching has completed, check the MCM bit of the clock selection register (CKSCR).

Figure 5.7-1 Status Transition Diagram for Clock Selection

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. An operating mode is specified by mode setting pins (MDx). 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.

■ Mode Pins

Table 6.1-2 lists the relationship between mode pins and set 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

Notes:

- Even if external vector mode 0 is selected, the initial values of IOBS and LMBS of the bus control selection register are set to "1". Therefore, a 16-bit width is available in areas 0000C0_H to 0000FF_H and 002000_H to 7FFFFFF_H. To specify an 8-bit width for the areas, write "0" 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 description 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	bit	7	6	5	4	3	2	1	0
Address:FFFFDF _H		M1	M0	Reserved	Reserved	S0	Reserved	Reserved	Reserved

[bit7, bit6] 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)

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.

[bit3] 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)

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

This section describes the correspondence between access areas and physical addresses, depending on the specified bus mode.

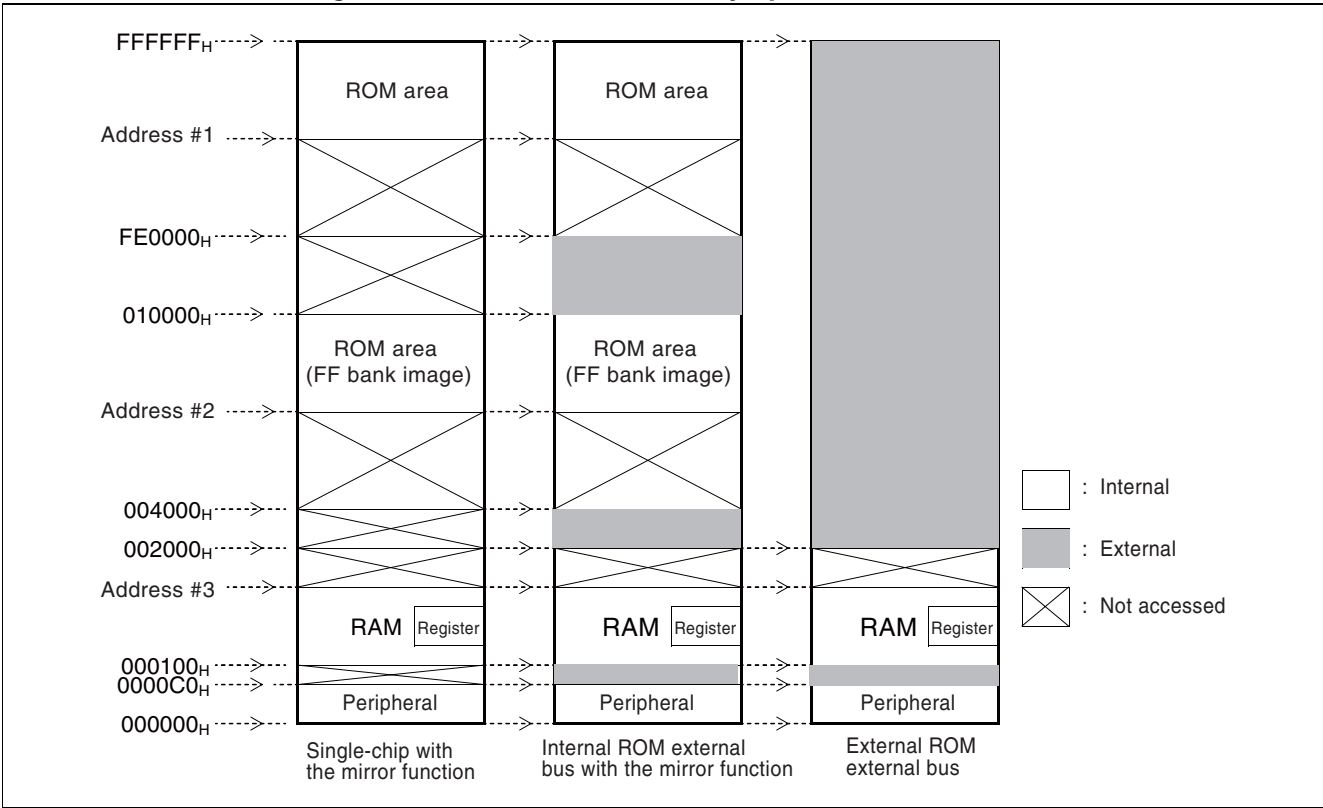
■ Memory Space for Each Bus Mode

As shown in Figure 6.1-2, 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 22 ROM MIRROR FUNCTION SELECTION MODULE" when ROM without the mirror function is selected.

Figure 6.1-2 MB90550A/B Memory Space for Each Mode



Part number	Address #1	Address #2	Address #3
MB90552A/B	FF000 _H	004000 _H	000900 _H
MB90553A/B	FE000 _H	004000 _H	001100 _H
MB90F553A	FE000 _H	004000 _H	001100 _H
MB90P553A	FE000 _H	004000 _H	001100 _H
MB90V550A	(FE000 _H)	004000 _H	001900 _H

■ Recommended Setting Sample of Memory Space for Each Bus Mode

Table 6.1-5 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 8-bit bus	0	1	1	0	1	0
Internal ROM, external 16-bit bus	0	1	1	0	1	1
External ROM, external 16-bit bus	0	0	1	1	0	1
External ROM, external 8-bit bus	0	0	0	1	0	0

Signals input to and output from external pins depend on the mode.

Table 6.1-6 Operation of External Pins Related to Modes

Pin	Function			
	Single chip	External bus extension		EPROM write
		8 bits	16 bits	
P07 to P00	Port	AD07 to AD00		D07 to D00
P17 to P10		AD15 to AD08	AD15 to AD08	A15 to A08
P27 to P20		AD23 to AD16*		A07 to A00
P30		ALE		A16
P31		\overline{RD}		\overline{CE}
P32		\overline{WRL}^*		\overline{OE}
P33		Port	\overline{WRH}^*	\overline{PGM}
P34		HRQ*		Not used
P35		\overline{HAK}^*		
P36		RDY*		
P37		CLK*		

*: The address high output pins, \overline{WRL} , \overline{WRH} , \overline{HAK} , HRQ, 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)

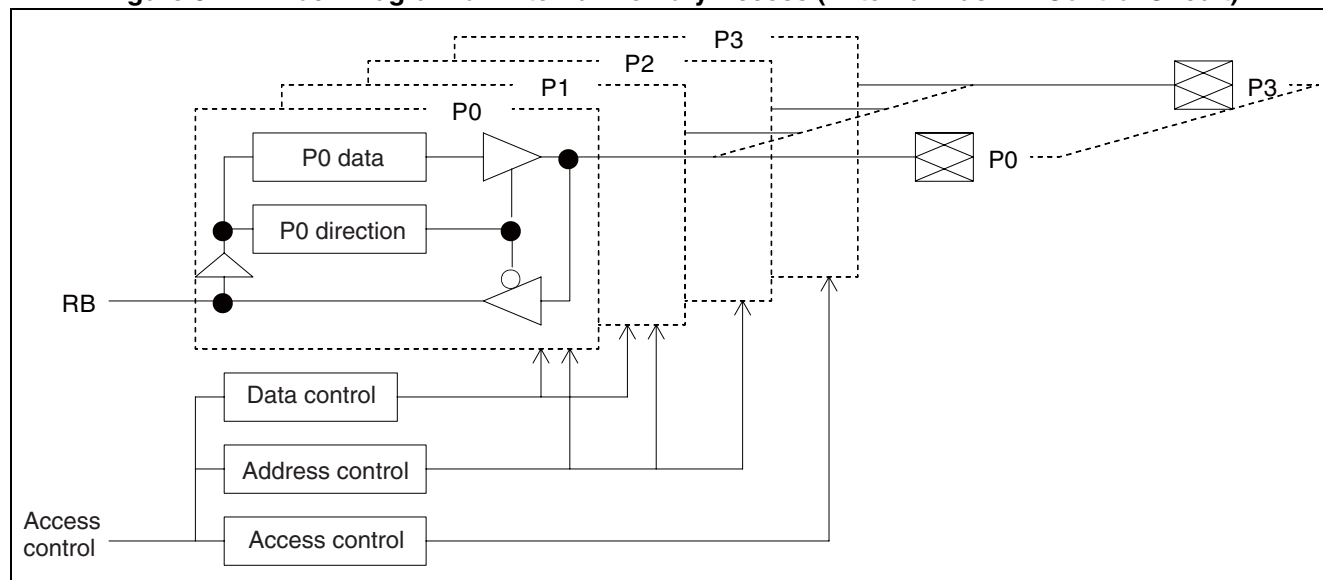
To access memory/peripheral circuits installed outside the device, the F²MC-16LX supplies the following address/data/control signals:

- 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 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		bit	15	14	13	12	11	10	9	8	
Address:0000A5 _H			IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write ⇒			(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
Initial valu ⇒			(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	
External address output control register		bit	7	6	5	4	3	2	1	0	
Address:0000A6 _H			E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇒			(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial valu ⇒			(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Bus control signal selection register		bit	15	14	13	12	11	10	9	8	
Address:0000A7 _H			CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇒			(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	
Initial valu ⇒			(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		bit 15	14	13	12	11	10	9	8	ARSR
Address:0000A5 _H		IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	
Read/write ⇒		(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
Initial value ⇒		(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	

[bit15, bit14] IOR1 and IOR0

The IOR1 and IOR0 bits specify an automatic wait function for external access to area 0000C0_H to 0000FF_H. The two bits are combined as listed in Table 6.2-1.

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	Inserts automatic 1-machine cycle wait at external access.
1	0	Inserts automatic 2-machine cycle wait at external access.
1	1	Inserts automatic 3-machine cycle wait at external access.

[bit13, bit12] HMR1 and HMR0

The HMR1 and HMR0 bits specify an automatic wait function for external access to area 800000_H to FFFFFFFF_H. The two bits are combined as listed in Table 6.2-2.

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

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

[bit9 , bit8] LMR1 and LMR0

The LMR1 and LMR0 bits specify an automatic wait function for external access to area 002000_H to 7FFFFFF_H. The two bits are combined as listed in Table 6.2-3.

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	Inserts automatic 1-machine cycle wait at external access.
1	0	Inserts automatic 2-machine cycle wait at external access.
1	1	Inserts automatic 3-machine cycle wait at external access.

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 Table 6.2-4.

■ External Address Output Control Register (HACR)

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

External address output control register		bit 7	6	5	4	3	2	1	0	HACR
Address:0000A6 _H		E23	E22	E21	E20	E19	E18	E17	E16	
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 (E23 to E16 bits)

E23 to E16	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	bit 15	14	13	12	11	10	9	8	
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)	(-)	

[bit15] CKE

The CKE bit controls output of the external clock (CLK) as listed in Table 6.2-5.

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.

[bit14] RYE

The RYE bit controls input of the external ready (RDY) as listed in Table 6.2-6.

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

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

[bit13] HDE

The HDE bit enables input/output of hold-related pins. The HDE bit controls hold request input (HRQ) and hold acknowledge output (HAK) as listed in Table 6.2-7.

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 ($\overline{\text{HAK}}$).

[bit12] 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.

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

[bit11] 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.

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

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

[bit10] WRE

The WRE bit controls output of the external write signal (in external data bus 16-bit mode, the $\overline{\text{WRH}}$ and $\overline{\text{WRL}}$ pins; and in external data bus 8-bit mode, the $\overline{\text{WRL}}$ pin) as listed in Table 6.2-10.

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 $\overline{\text{WRL}}$)

[bit9] 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.

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

Notes:

- In external data bus 16-bit mode, set P33 and P32 to input mode (set bit3, bit2 of DDR3 to DDR0) in order to enable the $\overline{\text{WRH}}$ and $\overline{\text{WRL}}$ functions with the WRE bit.
- In external data bus 8-bit mode, set P32 to input mode (set bit2 of DDR3 to DDR0) in order to enable the WRX function with the WRE bit.
- 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 DDR0 (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. 8-bit bus access in external data bus 16-bit mode enables reading and writing of 8-bit peripheral chips if 8-bit peripheral chips and 16-bit peripheral chips are connected together to the external bus.

■ 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 executed in external data bus 16-bit 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 $\overline{RD}/\overline{WRL}/\overline{WRH}$ 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 shows the timing chart of external data bus 8-bit mode access.
Figure 6.3-2 shows the timing chart of external data bus 16-bit mode access (for 16-bit bus width access and 8-bit bus width access).

Figure 6.3-1 Timing Chart of External Data Bus 8-bit Mode Access

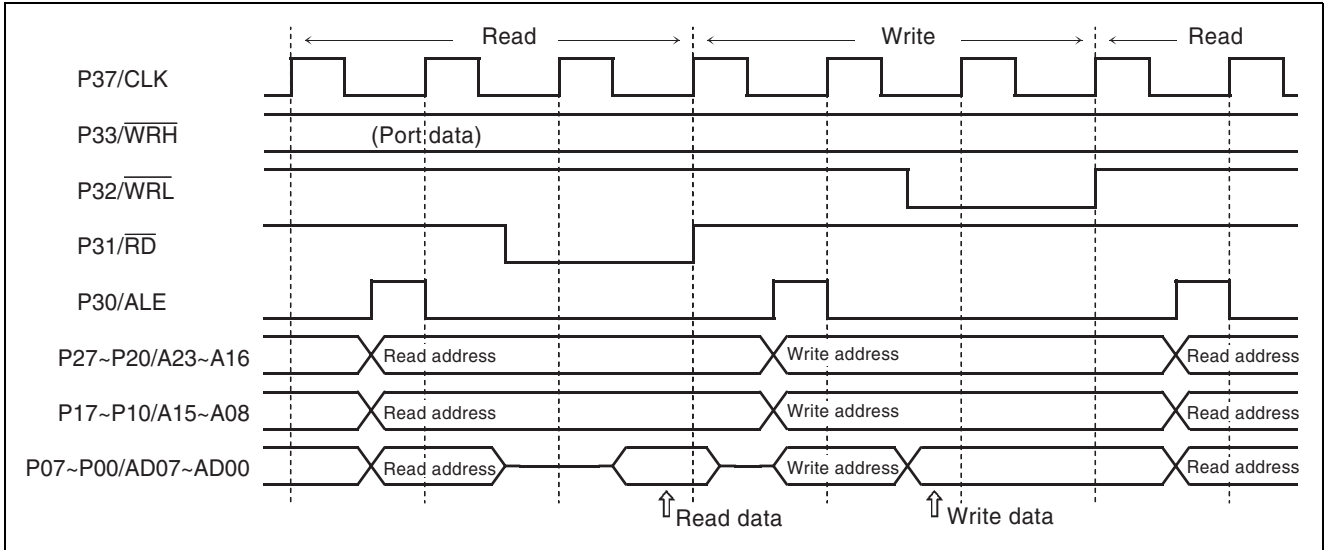
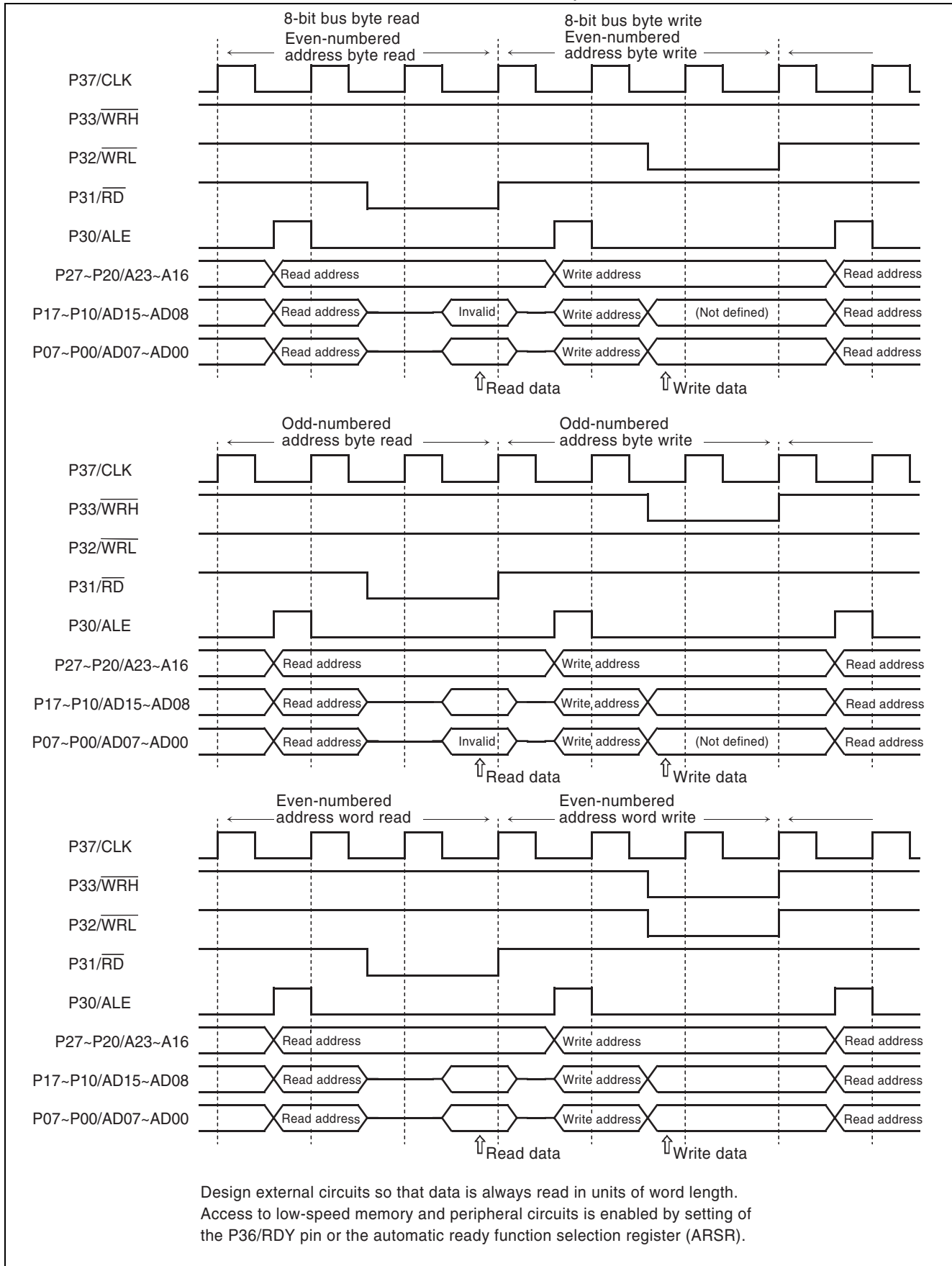


Figure 6.3-2 Timing Chart of External Data Bus 16-bit Mode Access (for 16-bit Bus Width Access and 8-bit Bus Width Access)

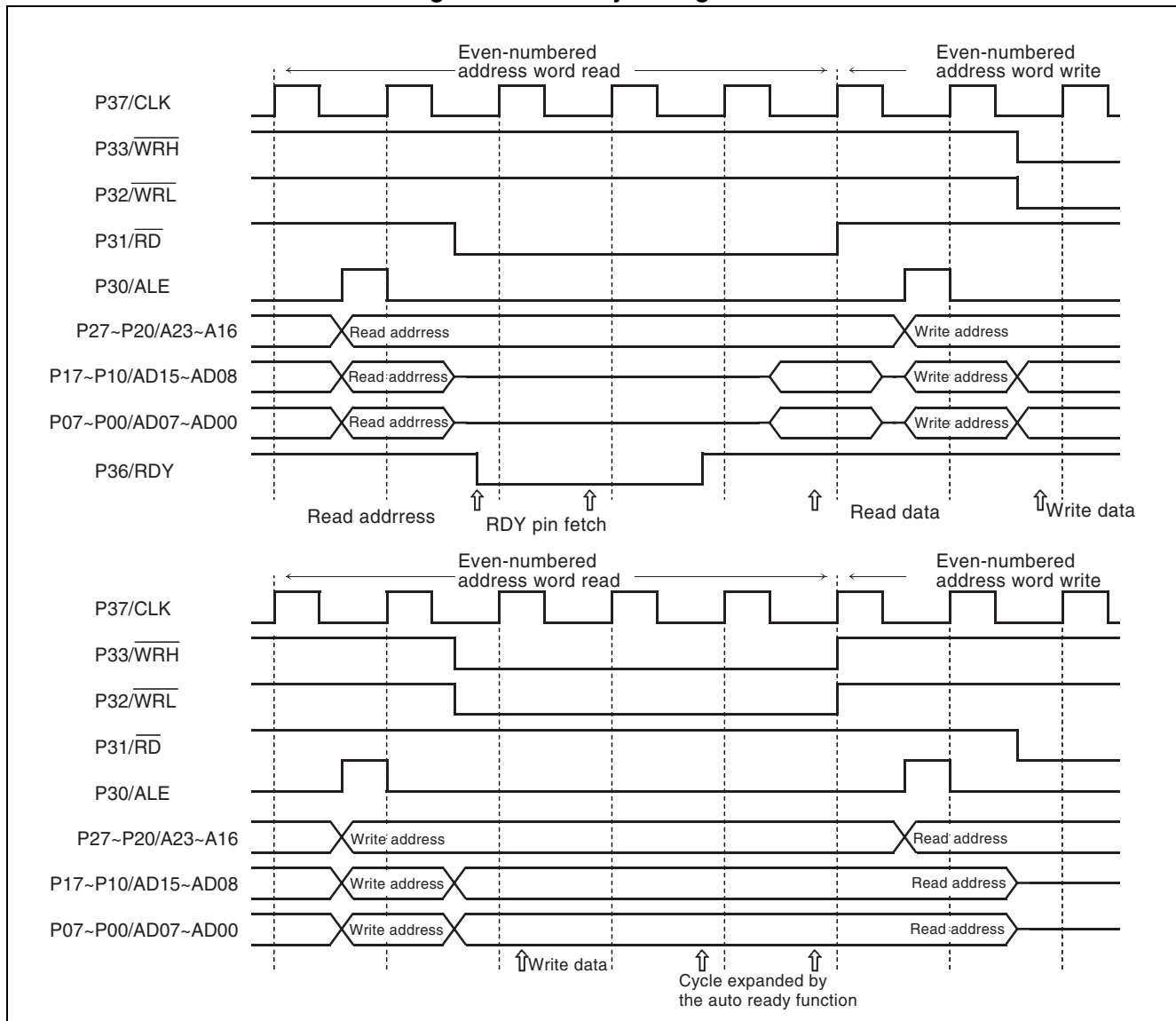
6.3.1 Ready Function

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



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 EPCR is set to "1", the external bus hold function specified by the P34/HRQ and P35/ $\overline{\text{HAK}}$ bits is enabled.

■ Hold Function

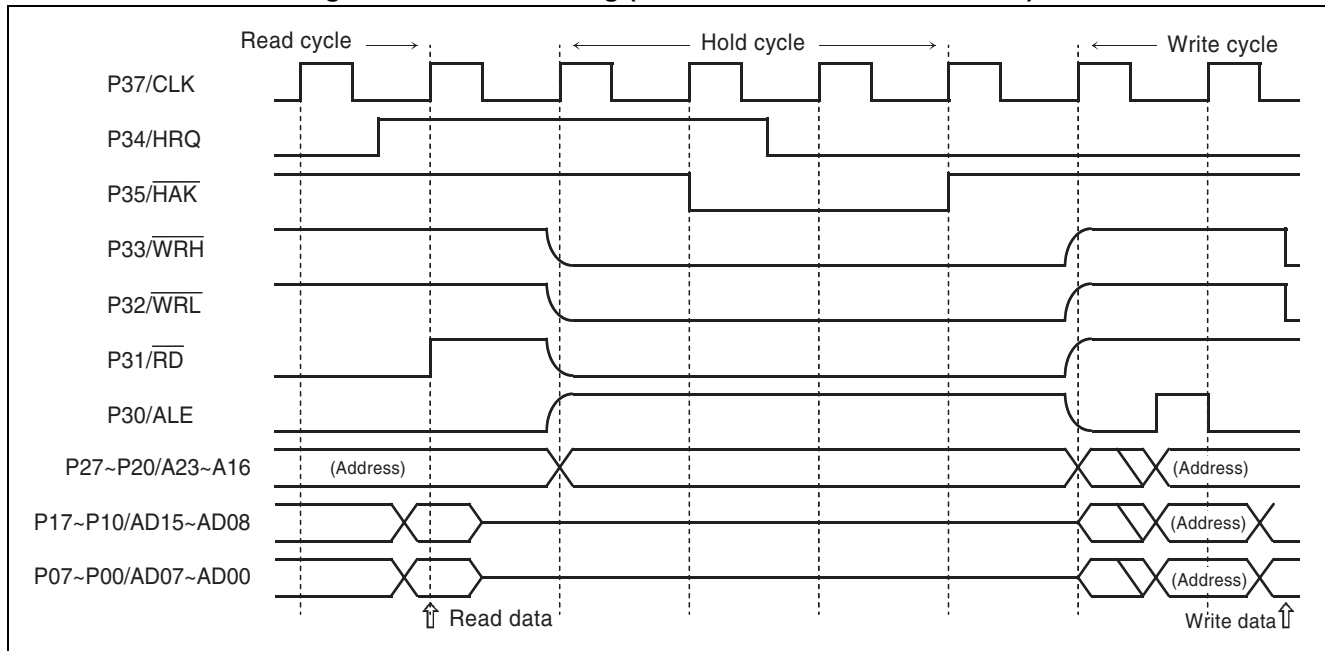
If the high level is applied to the P34/HRQ pin, the hold state is set up at termination of a CPU instruction (for a string instruction, at termination of 1-element data processing). The P35/ $\overline{\text{HAK}}$ pin outputs the low level to place the following pins in a high-impedance state:

- Address output: P23/A19 to P20/A16
- Address/data I/O: P17/D15 to P00/D00
- Bus control signal: P30/ALE, P31/ $\overline{\text{RD}}$, P32/ $\overline{\text{WRL}}$, 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/ $\overline{\text{HAK}}$ 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 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.

Input is set to I/O ports 0 to 4 and 6 to A when the data direction register indicates 0 and output is set to the same when the data direction register indicates 1. Port 5 is an open-drain port.

The MB90550A/B 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 to Figure 7.2-5 show the following I/O port block diagrams:

- Parallel port block diagram (Port 0 and Port 1)
- Parallel port block diagram (Port 2, Port 3, Port 7, Port 8, Port 9, and Port A)
- Parallel port block diagram (Port 4)
- Parallel port block diagram (Port 5)
- Parallel port block diagram (Port 6)

■ I/O Port Block Diagram

Figure 7.2-1 Parallel Port Block Diagram (Port 0 and Port 1)

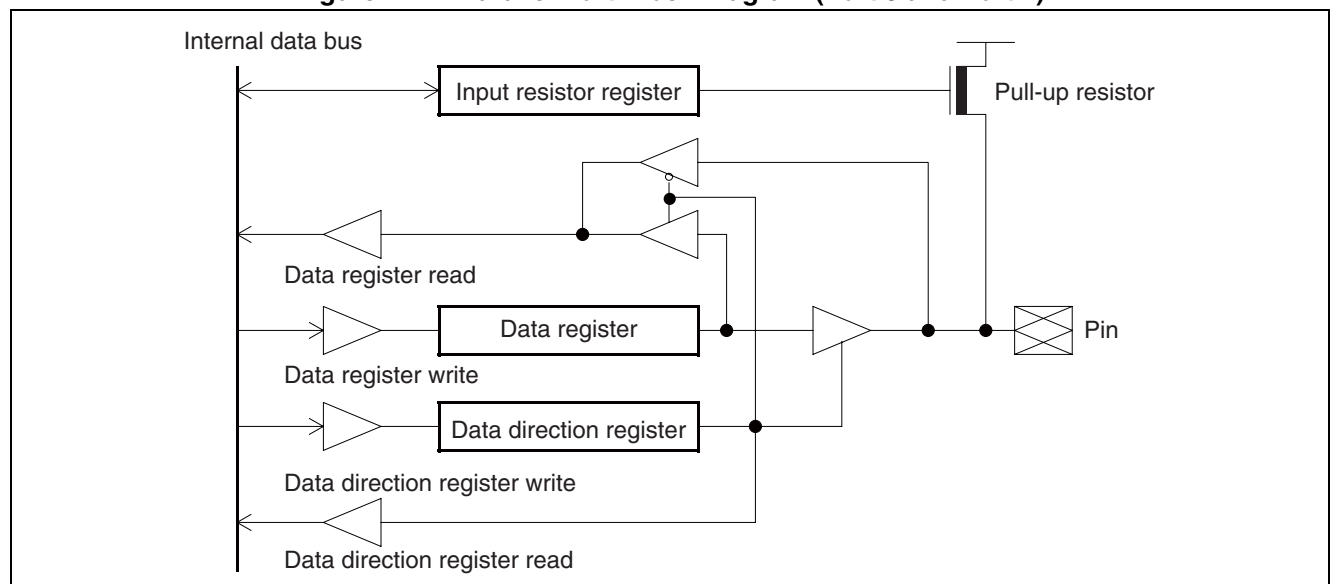


Figure 7.2-2 Parallel Port Block Diagram (Port 2, Port 3, Port 7, Port 8, Port 9, and Port A)

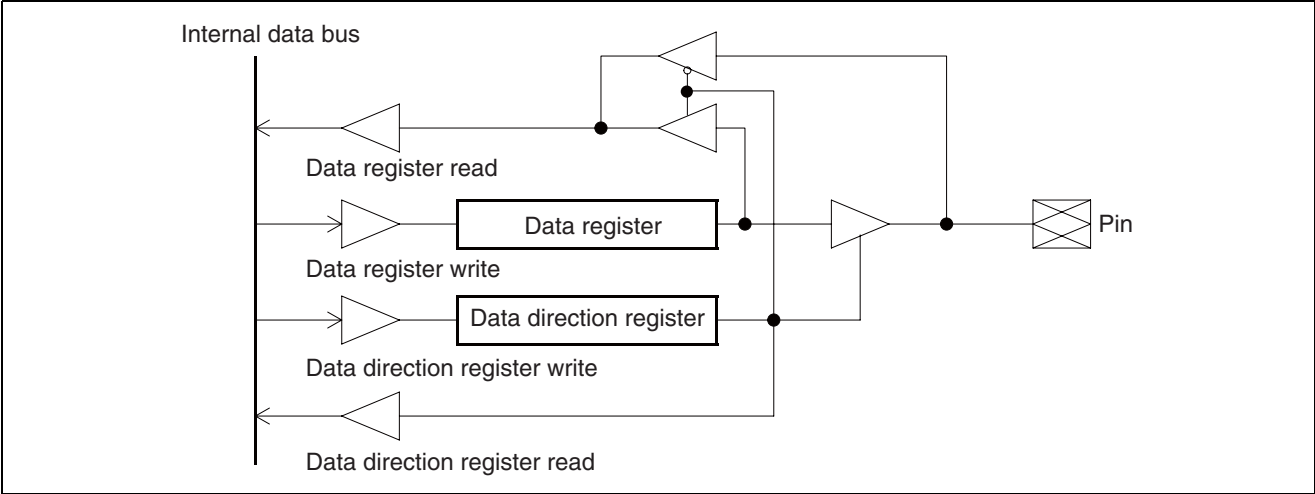


Figure 7.2-3 Parallel Port Block Diagram (Port 4)

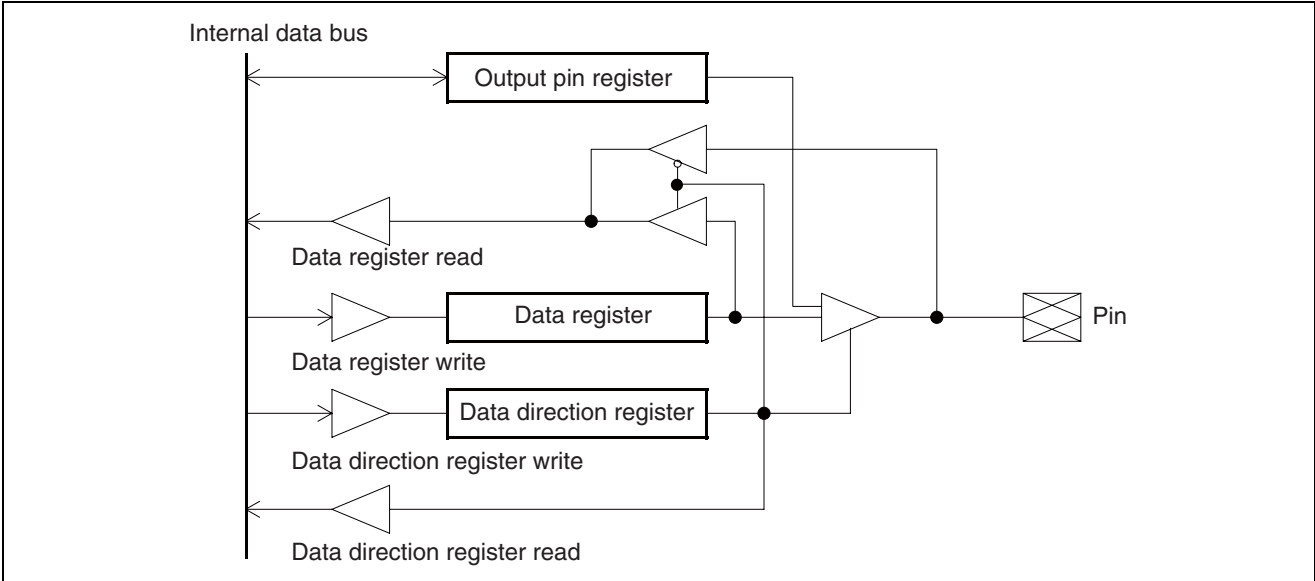


Figure 7.2-4 Parallel Port Block Diagram (Port 5)

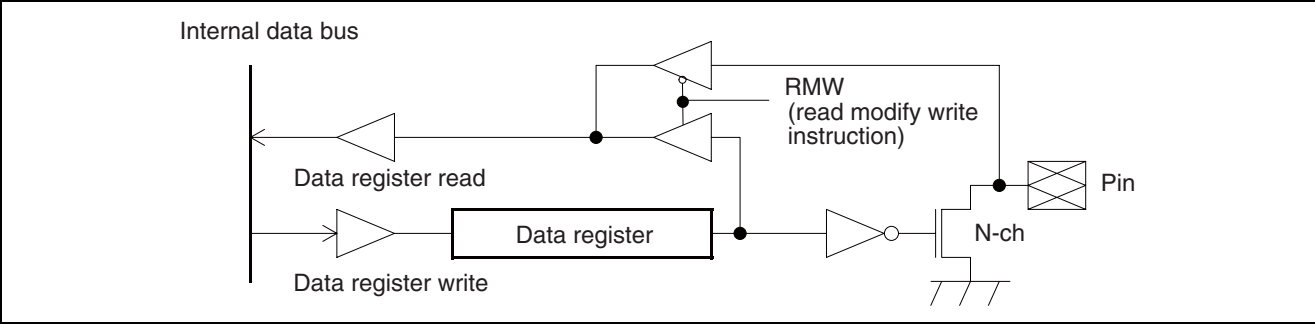
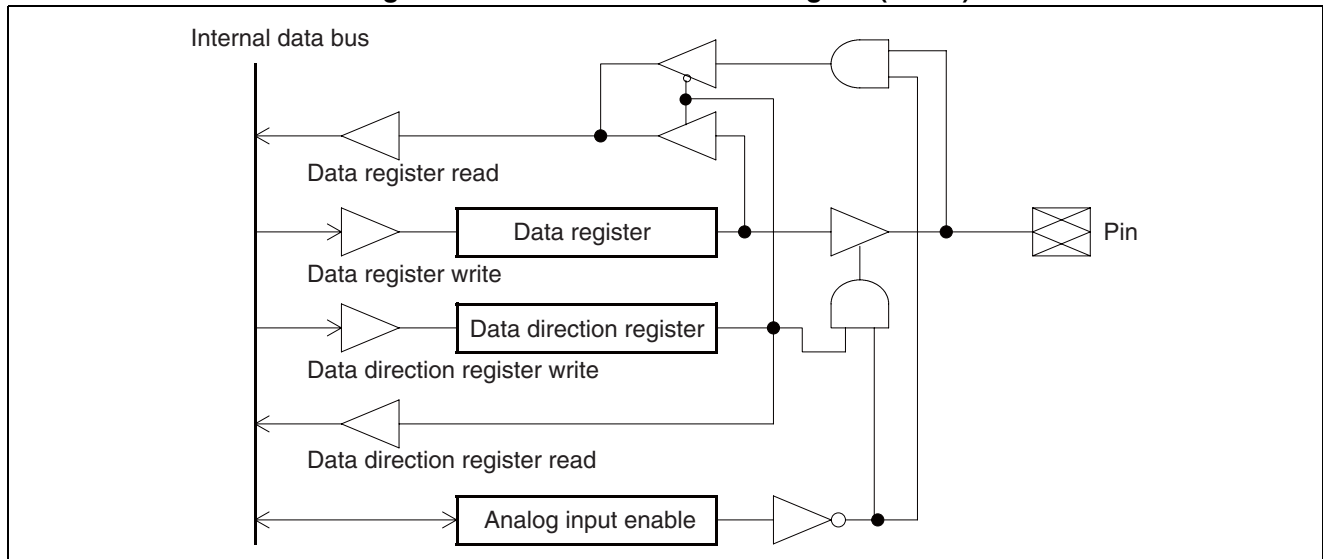


Figure 7.2-5 Parallel Port Block Diagram (Port 6)

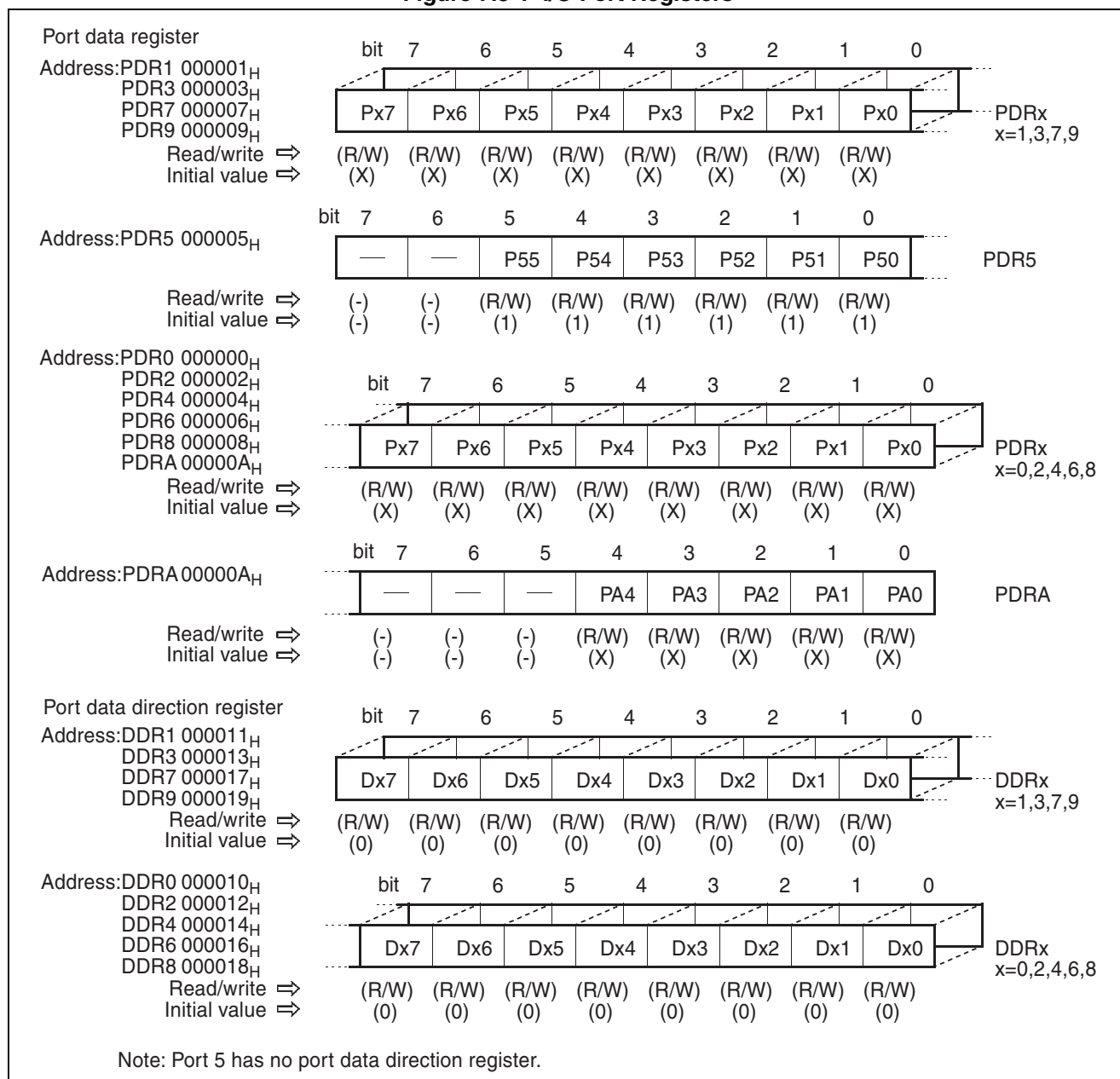
7.3 I/O Port Registers

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



(Continued)

(Continued)

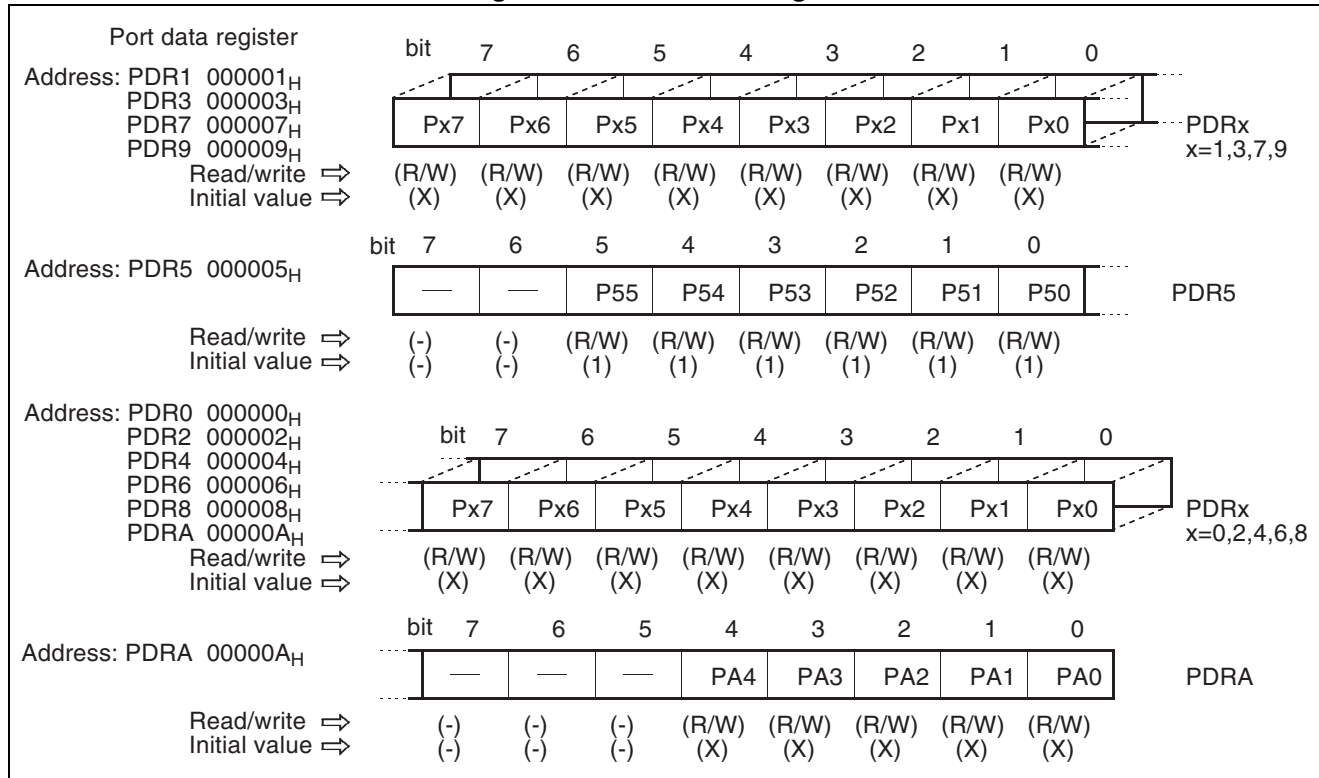
Address:DDRA00001A _H		bit	7	6	5	4	3	2	1	0	
			—	—	—	Dx4	Dx3	Dx2	Dx1	Dx0	DDRA
Read/write ⇒			(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒			(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	
Port 4 output pin register		bit	7	6	5	4	3	2	1	0	
Address:00001B _H			OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	ODR4
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)	
Port 0 input resistor register		bit	7	6	5	4	3	2	1	0	
Address:00001C _H			RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	RDR0
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)	
Port 1 input resistor register		bit	7	6	5	4	3	2	1	0	
Address:00001D _H			RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	RDR1
Read/write ⇒			(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒			(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

7.3.1 Port Data Registers (PDRx)

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

■ Port Data Register (PDRx)

Figure 7.3-2 Port Data Register



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

Note that the operation of port 5 R/W differs from that of another port.

Port 5 (P55 to P50) is a general-purpose I/O port of the open-drain output format.

When port 5 is used as an input port, to turn the open-drain output transistor off, the output data register must be set to "1" and an external pull-up resistor must be installed.

Moreover, port 5 operates as follows depending on the reading instruction:

- **Reading by a read modify write instruction**

The description of the output data register is read. Even if a pin is set to "0" externally, the contents of the bit not specified by the instruction do not change.

- **Reading by an instruction other than the above instruction**

The level of a pin can be read.

If port 5 is used as an output port, the value of a pin can be changed by writing a desired value in the corresponding output data register.

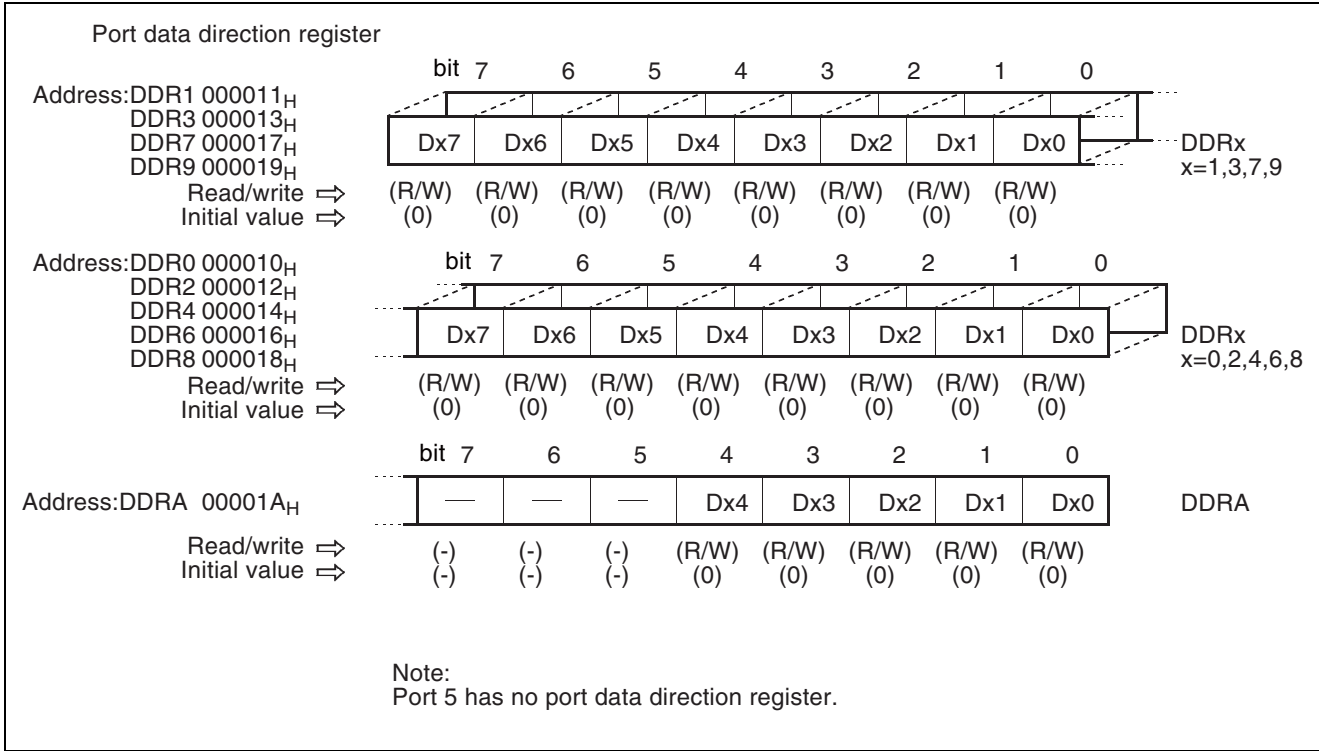
"0" is read from the pin corresponding to a bit set to "1" in the ADER register.

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)



A port data direction register (DDRx) controls each pin as listed in Table 7.3-1 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

7.3.3 Output Pin Register (ODR4)

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

■ Output Pin Register (ODR4)

Figure 7.3-4 Output Pin Register (ODR4)

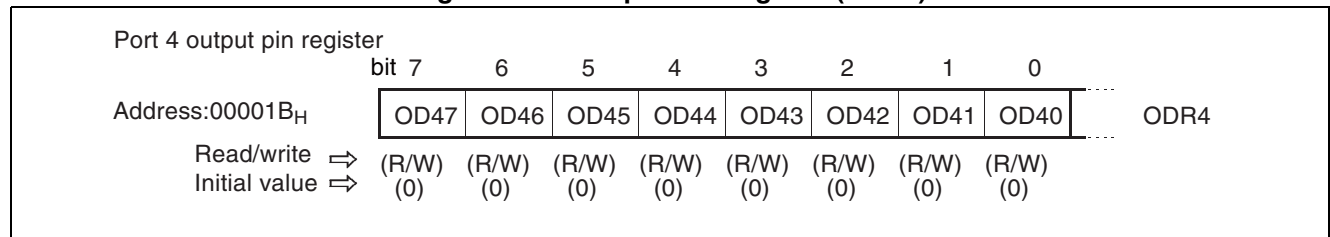


Table 7.3-2 Function of the Output Pin Register (ODR4)

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.

Notes:

- 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 Resistor Registers (RDR0 and RDR1)

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

■ Input Resistor Registers (RDR0 and RDR1)

Figure 7.3-5 Input Resistor Registers (RDR0 and RDR1)

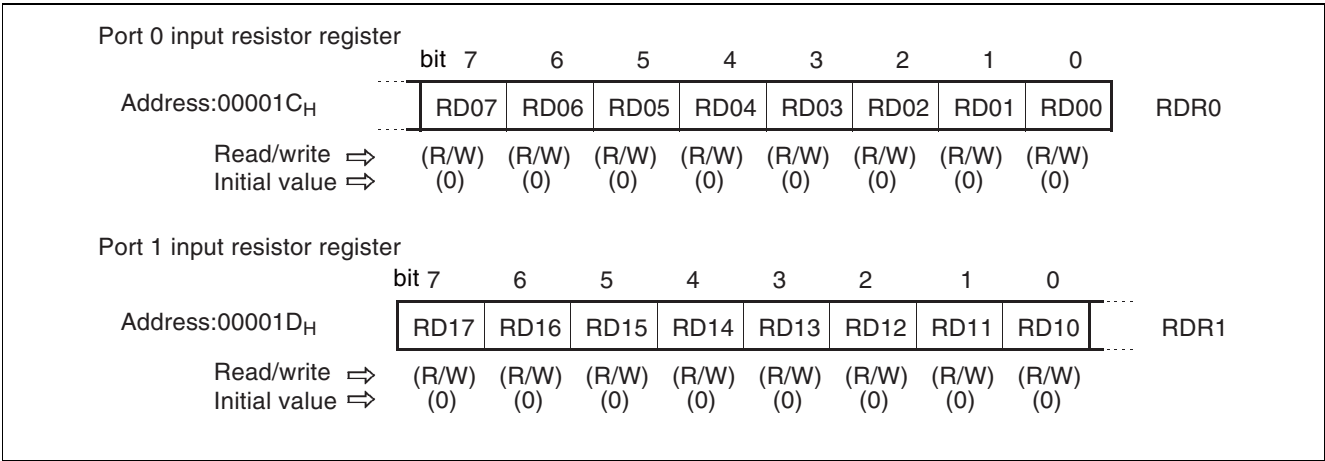


Table 7.3-3 Function of the Input Resistor Registers (RDR0 and RDR1)

RDR0, RDR1	Function
0	Without a pull-up resistor in an input mode [initial value]
1	With a pull-up resistor in an input mode

Notes:

- 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 Analog Input Enable Register (ADER)

The analog input enable register (ADER) controls each pin of port 6 as listed in Table 7.3-4.

■ Analog Input Enable Register (ADER)

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

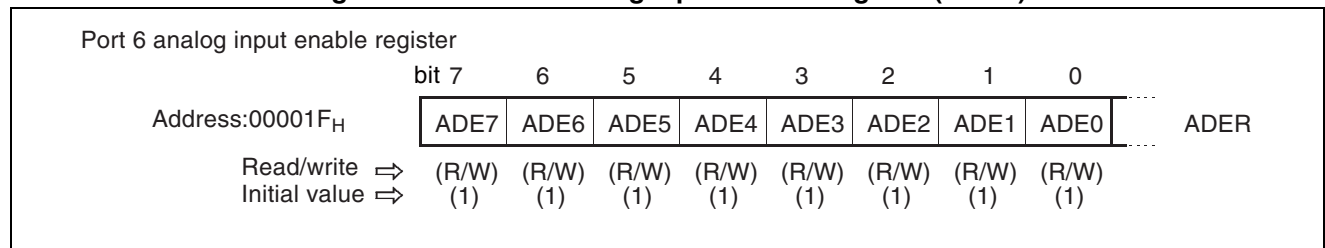


Table 7.3-4 Port 6 Analog Input Enable Register (ADER)

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 explains 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.

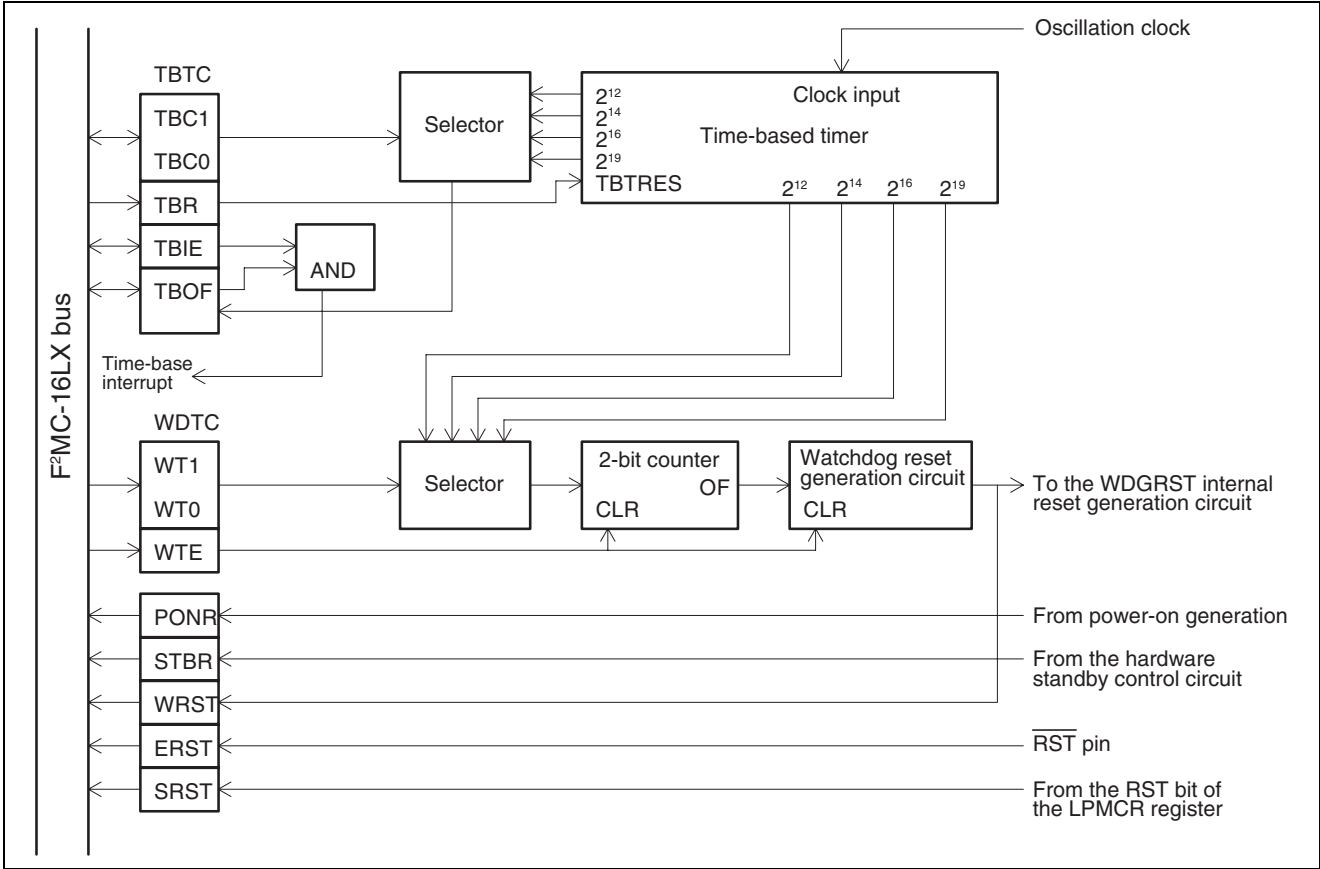
Time-based Timer Register

Figure 8.1-1 Time-based Timer Register

Time-based timer control register									
Address:0000A9 _H									
	bit	15	14	13	12	11	10	9	8
		Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0
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									
	bit	15	14	13	12	11	10	9	8
Address:0000A9 _H		Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0
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.

[bit15] Reserved

Bit15 is a reserved bit. When defining TBTC settings, be sure to set this bit to "1".

[bit12] 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.

[bit11] 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.

[bit10] 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.

[bit9, bit8] 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 explains 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.

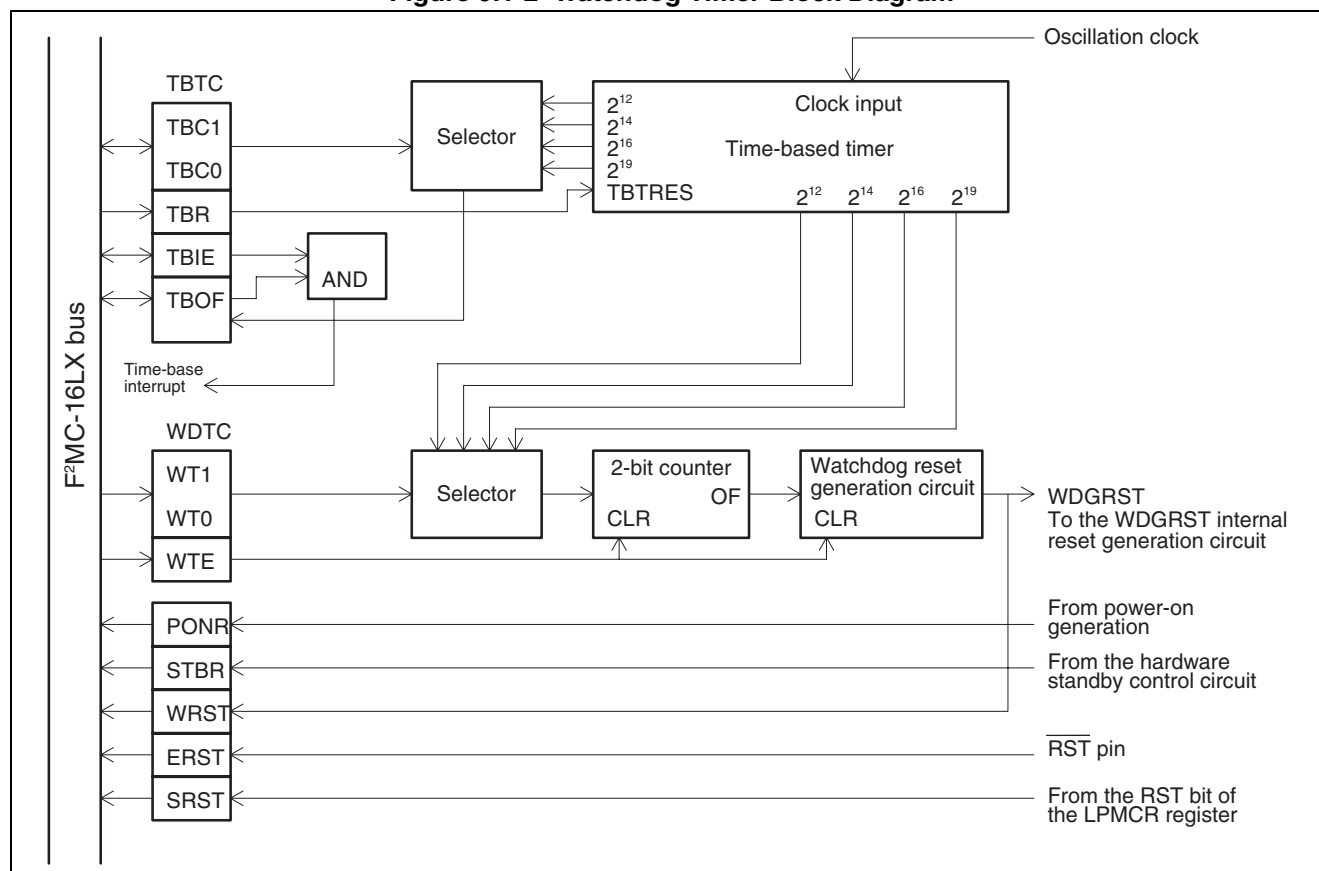
■ Watchdog Timer Register

Figure 9.1-1 Watchdog Timer Register

Watchdog control register		bit 7	6	5	4	3	2	1	0	
Address: 0000A8 _H		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	bit	7	6	5	4	3	2	1	0	
Address:0000A8 _H		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.

[bit7 to bit3] 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. After the WDTC register read operation, all bits are cleared to "0".

This is a read only register.

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.

[bit2] 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.

[bit1, bit0] WT1 and WT0

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.

Table 9.2-2 WT1 and WT0 (Interval Time Selection Bits)

WT1	WT0	Interval time (for oscillation clock frequency 4 MHz)		Number of oscillation clock (oscillation) cycles
		Minimum	Maximum	
0	0	Approx. 3.58 ms	Approx. 4.61 ms	$2^{14} \pm 2^{11}$
0	1	Approx. 14.33 ms	Approx. 18.43 ms	$2^{16} \pm 2^{13}$
1	0	Approx. 57.23 ms	Approx. 73.73 ms	$2^{18} \pm 2^{15}$
1	1	Approx. 458.75 ms	Approx. 589.82 ms	$2^{21} \pm 2^{18}$

Note:

Since the carry signal of the time-based timer is used as the count clock of interval time, the interval time of the watchdog timer may become longer when the time-based timer is cleared.

Note that the time-based timer is cleared and "0" is written to the TBR bit of the time-based timer control register (TBTC) during a transition from main clock mode to PLL clock mode.

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

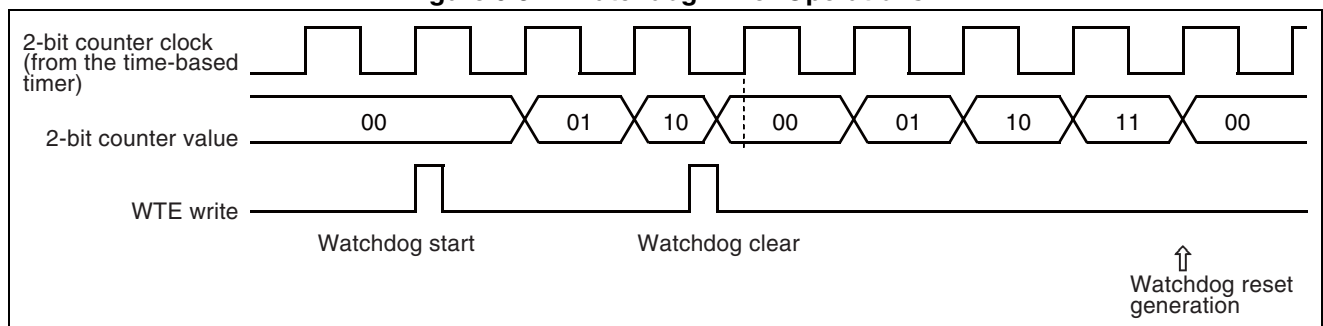
The watchdog timer is activated by writing "0" in the WTE bit of the WDTC register while the watchdog timer stops. At the same time, the reset generation interval of the watchdog timer is set by WT1 and WT0 bits. Only data at this activation is valid for interval setting.

■ 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.

Note that the time-based timer is cleared and "0" is written to the TBR bit of the time-based timer control register (TBTC) during a transition from main clock mode to PLL clock mode.

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 16-BIT I/O TIMER

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

- 10.1 Overview of the 16-Bit I/O Timer
- 10.2 16-Bit I/O Timer Block Diagram
- 10.3 16-Bit I/O Timer Registers
- 10.4 16-Bit Free-Run Timer Operations
- 10.5 16-Bit Output Compare Operations
- 10.6 16-Bit Input Capture Operations

10.1 Overview of the 16-Bit I/O Timer

The 16-bit I/O timer consists of one 16-bit free-run timer, four output 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 or match with compare register 0. (The compare match requires mode setting.)

○ Counter value

A counter value can be initialized to 0000_H by a reset, software clear, or match with compare register 0.

■ Output Compare (x 4)

An output compare 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 output pin is reversed by 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 consists of the capture register and control register corresponding to four independent external input pins. When an edge of the signal input from an external input pin is detected, a 16-bit free-run timer value can be retained in the capture register and an interrupt can be generated at the same time.

○ **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.**

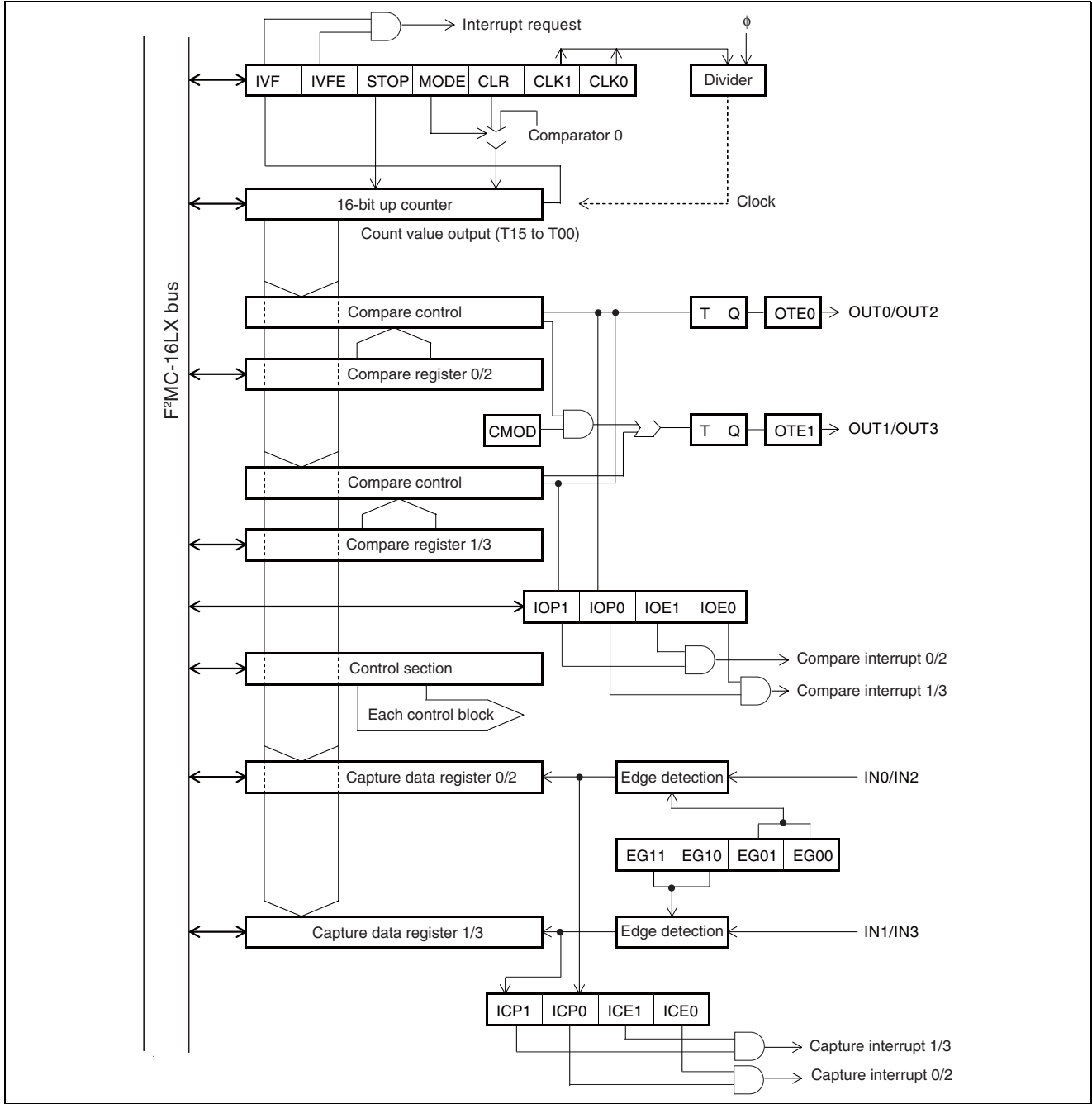
The extended intelligent I/O service can be activated by an interrupt of the input capture.

10.2 16-Bit I/O Timer Block Diagram

Figure 10.2-1 shows the 16-bit I/O timer block diagram.

■ 16-bit I/O Timer Block Diagram

Figure 10.2-1 16-bit I/O Timer Block Diagram



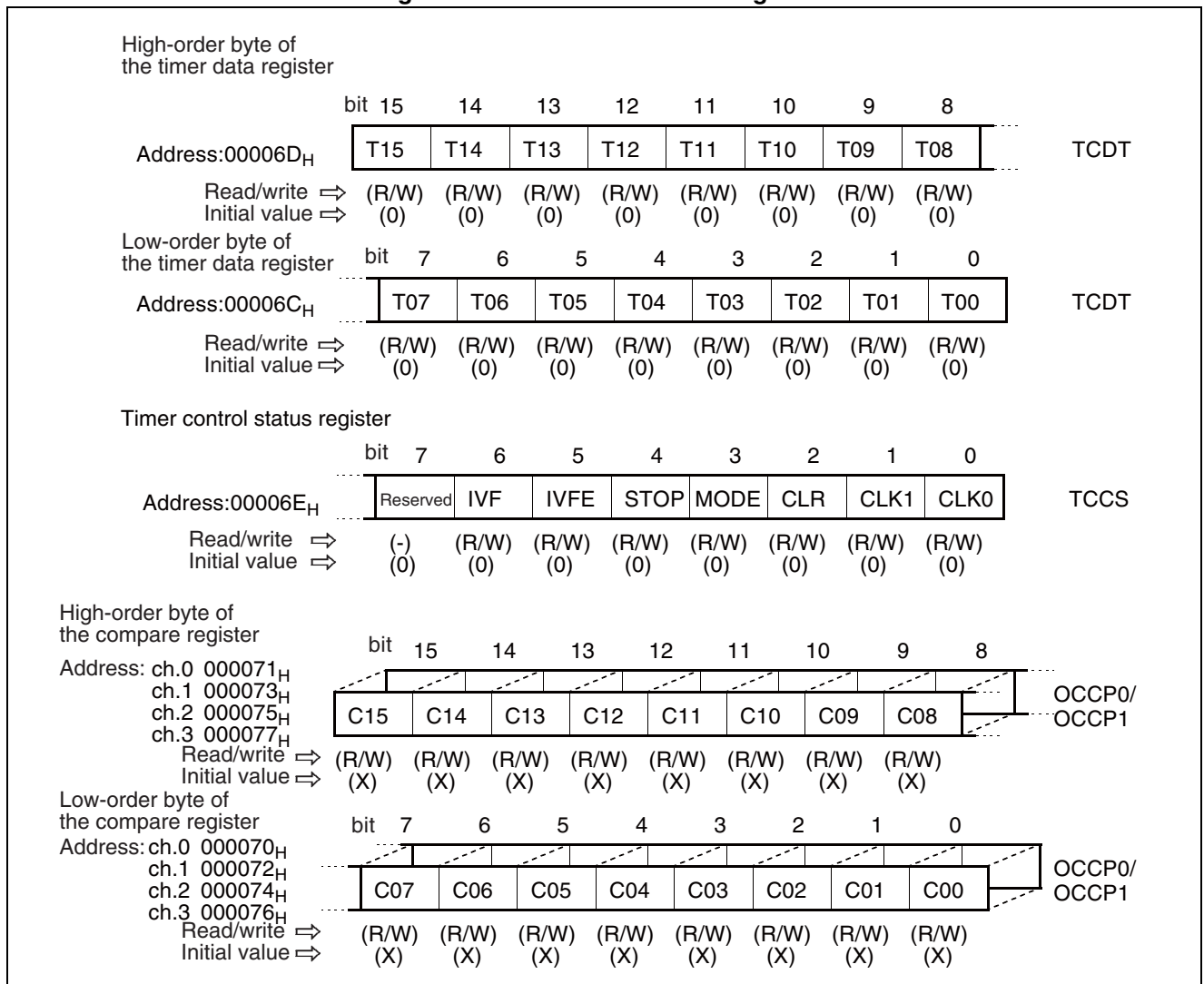
10.3 16-Bit I/O Timer Registers

The following six 16-bit I/O timer registers are supported:

- Timer data register (TCDT)
- Timer control status register (TCCS)
- Compare register (OCCP0/OCCP1)
- Compare control status register (OCS0 to OCS3)
- Input capture register (IPCP0 to IPCP3)
- Control status register (ICS01, ICS23)

■ 16-bit I/O Timer Registers

Figure 10.3-1 16-bit I/O Timer Registers



(Continued)

(Continued)

Compare control status register 1/3		bit 15	14	13	12	11	10	9	8	
Address: ch.1 000079 _H ch.3 00007B _H		—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1/OCS3
Read/write ⇒		(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒		(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	
Compare control status register 0/2		bit 7	6	5	4	3	2	1	0	
Address: ch.0 000078 _H ch.2 00007A _H		IOP1	IOP0	IOE1	IOE0	—	—	CST1	CST0	OCS0/OCS2
Read/write ⇒		(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇒		(0)	(0)	(0)	(0)	(-)	(-)	(0)	(0)	
High-order byte of the input capture register		bit 15	14	13	12	11	10	9	8	
Address: ch.0 000063 _H ch.1 000065 _H ch.2 000067 _H ch.3 000069 _H		CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	IPCO0 to IPCO3
Read/write ⇒		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
Low-order byte of the input capture register		bit 7	6	5	4	3	2	1	0	
Address: ch.0 000062 _H ch.1 000064 _H ch.2 000066 _H ch.3 000068 _H		CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	IPCO0 to IPCO3
Read/write ⇒		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
High-order byte of the control status register		bit 15	14	13	12	11	10	9	8	
Address: 00006B _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)	
Initial value ⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Low-order byte of the control status register		bit 7	6	5	4	3	2	1	0	
Address: 00006A _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)	
Initial value ⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

10.3.1 16-bit Free-run Timer

The following two 16-bit free-run timer registers are supported:

- Data register (TCDT)
- Control status register (ICCS)

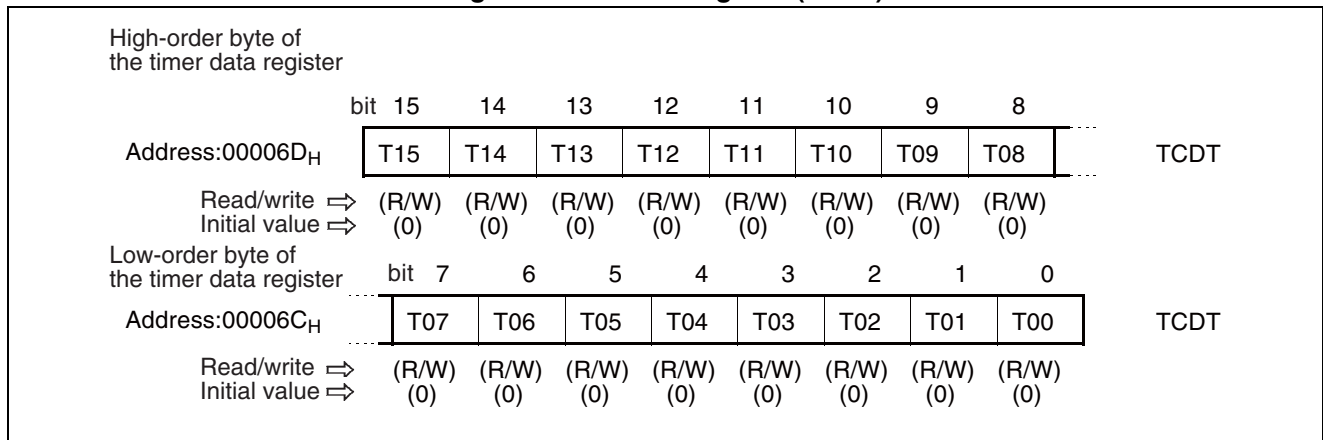
■ Data Register

The data register can read a count value of the 16-bit free-run timer. The counter value is cleared to "0000" 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 10.3-2 Data register (TCDT)



Note:

This register requires word accesses.

■ Control Status Register (ICCS)

Figure 10.3-3 Control Status Register (ICCS)

Timer control status register								
	bit 7	6	5	4	3	2	1	0
Address: 00006E _H	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)

TCCS

[bit7] Reserved bit

Bit7 is a reserved bit. When defining TBTC settings, be sure to set this bit to "0".

[bit6] 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 if the counter is cleared by a match with compare register 0 through mode setting, this bit is set to "1". At this time, if the IVFE bit is already set, an interrupt occurs.

This bit is cleared by writing "0". Writing "1" is meaningless. Value "1" can be read by a read modify instruction.

Table 10.3-1 Function of IVF (Interrupt Request Flag)

IVF	Function
0	No interrupt request (initial value)
1	Interrupt request

[bit5] IVFE

IVFE is an interrupt enable bit of the 16-bit free-run timer.

When this bit is "1", if the IVF bit is set to "1", an interrupt occurs.

Table 10.3-2 Function of IVFE (Interrupt Enable Bit)

IVFE	Function
0	Interrupt disabled (initial value)
1	Interrupt enabled

[bit4] 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 10.3-3 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.

[bit3] 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 and CLR bit. If it is "1", the counter value can be initialized by the reset, CLR bit, or a match with compare register 0 of the output compare.

Table 10.3-4 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 compare register 0

Note:

The counter value is initialized at the change point of the counter value.

[bit2] 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. After "1" is written, the counter value is not initialized to the following count clock when "0" is written in this bit.

Table 10.3-5 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.

[bit1, bit0] CLK1 and CLK0

CLK1 and CLK0 bits are used to select a count clock of the 16-bit free-run timer. Because a clock is changed immediately after a write to these bits, change the clock while the output compare and input capture are stopped.

Table 10.3-6 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 μs	0.5 μs	1 μs	4 μs
0	1	$\phi/16$	1 μs	2 μs	4 μs	16 μs
1	0	$\phi/64$	4 μs	8 μs	16 μs	64 μs
1	1	$\phi/256$	16 μs	32 μs	64 μs	256 μs

ϕ = Machine clock

10.3.2 Output Compare

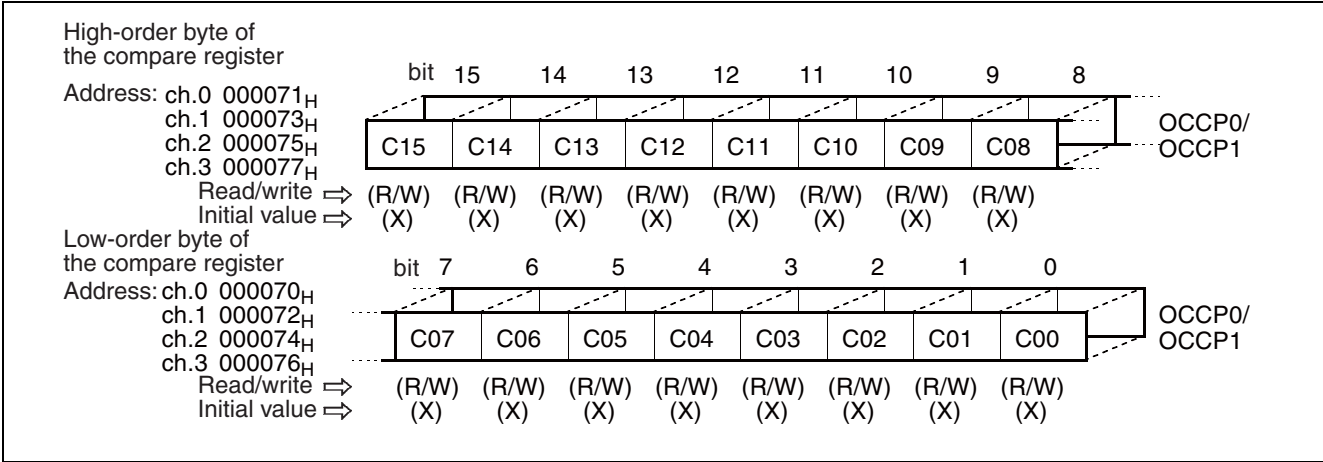
The output compare has the following two registers:

- Compare register
- Control status register

This section explains ch.0 and ch.1. For ch.2 and ch.3, consider that ch.0 corresponds to ch.2 and ch.1 to ch.3.

■ Compare Register (OCCP0 and OCCP1)

Figure 10.3-4 Compare Registers



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.

Note:

This register requires word accesses.

■ Control Status Register (OCS0 to OCS2)

Figure 10.3-5 Control Status Register

Compare control status register 1/3		bit	15	14	13	12	11	10	9	8	
Address: ch.1 000079 _H ch.3 00007B _H			—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1/OCS3
Read/write ⇒			(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒			(-)	(-)	(-)	(0)	(0)	(0)	(0)	(0)	
Compare control status register 0/2		bit	7	6	5	4	3	2	1	0	
Address: ch.0 000078 _H ch.2 00007A _H			IOP1	IOP0	IOE1	IOE0	—	—	CST1	CST0	OCS0/OCS2
Read/write ⇒			(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇒			(0)	(0)	(0)	(0)	(-)	(-)	(0)	(0)	

[bit12] 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 both a match with compare register 0 and 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 both a match with compare register 0 and a match with compare register 1.

[bit11, bit10] OTE1 and OTE0

OTE1 and OTE0 bits enable the pin output of the output compare.

Table 10.3-7 Function of OTE1 and OTE0 (Pin Output Enable Bits)

OTE1, OTE0	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.

[bit9, bit8] 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 10.3-8 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.

[bit7, bit6] IOP1 and IOP0

IOP1 and IOP0 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 (IOE1 and IOE0) are enabled, if IOP1 and IOP0 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 10.3-9 Function of IOP1 and IOP0 (Output Compare Interrupt Bits)

IOP1, IOP0	Function
0	No compare match [initial value]
1	Compare match

Note:

IOP1 corresponds to output compare 1 and IOP0 to output compare 0.

[bit5, bit4] IOE1 and IOE0

IOE1 and IOE0 are output compare interrupt enable bits. When these bits are "1", if the interrupt flags (IOP0 and IOP1) are set, an output compare interrupt occurs.

Table 10.3-10 Function of IOE1 and IOE0 (Output Compare Interrupt Enable Bits)

IOE1, IOE0	Function
0	Disables an output compare interrupt. [Initial value]
1	Enables an output compare interrupt.

Note:

IOE1 corresponds to output compare 1 and IOE0 to output compare 0.

[bit1, bit0] CST1 and CST0

CST1 and CST0 are used to enable the match operation with the 16-bit free-run timer.

Table 10.3-11 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.

- Before enabling the compare operation, set a compare register value.

Note:

CST1 corresponds to output compare 1 and CST0 to output compare 0.

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.

10.3.3 Input Capture

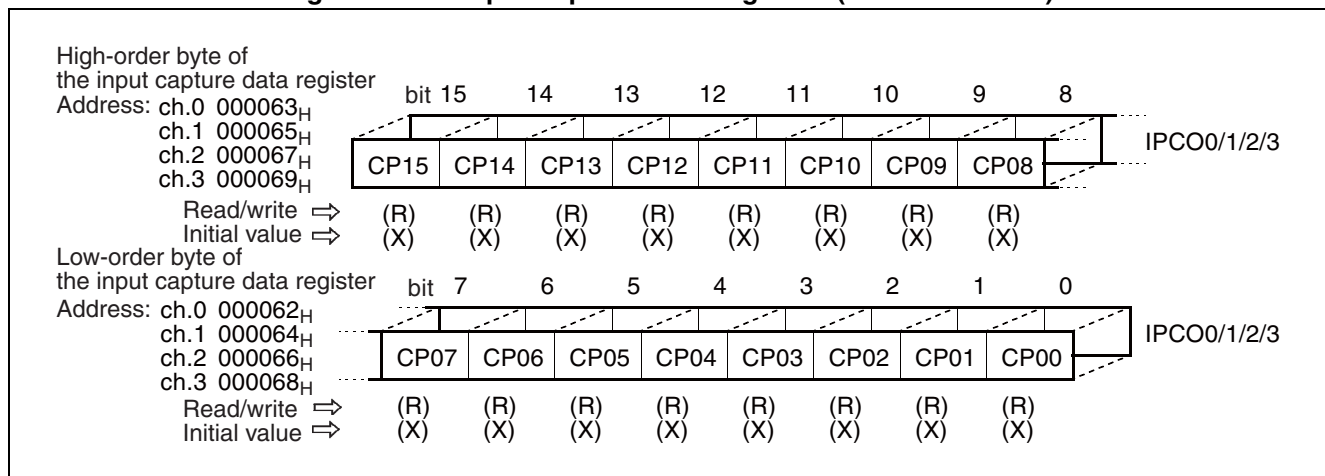
The input capture has the following two registers:

- Input capture data register (IPCO0 to IPCO3)
- Control status register (ICS23 and ICS01)

■ Input Capture Data Register (IPCO0 to IPCO3)

Input capture data registers (IPCO0 to IPCO3) are used to retain 16-bit free-run timer values when a valid edge of the corresponding external pin input waveform is detected.

Figure 10.3-6 Input Capture Data Registers (IPCO0 to IPCO3)

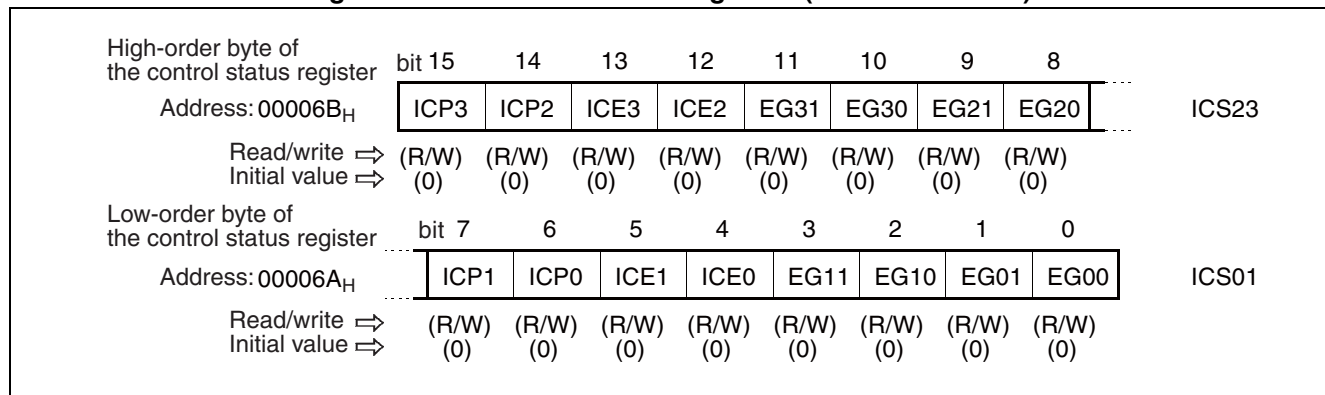


Note:

This register requires word access. No writing is possible.

■ Control Status Registers (ICS23 and ICS01)

Figure 10.3-7 Control Status Registers (ICS23 and ICS01)



Note:

This register requires byte access.

[bit15, bit14, bit7, bit6] ICPx (x: ch 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 interrupt enable bits (ICE0 and ICE1) are 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 10.3-12 Function of ICPx (Input Capture Interrupt Flag)

ICPx	Function
0	No valid edge detection [initial value]
1	Valid edge detection

[bit13, bit12, bit5, bit4] ICEx (x: ch number)

ICEx is an input capture interrupt enable bit. When this bit is "1", if the interrupt flags (ICP0 and ICP1) are set, an input capture interrupt occurs.





Table 10.3-13 Function of ICEx (Input Capture Interrupt Enable Bit)

ICEx	Function
0	Interrupt disabled [initial value]
1	Interrupt enabled

[bit11 to bit8, bit3 to bit0] EGx1 and EGx0 (x: ch number)

EGx1 and EGx0 bits specify the valid edge polarity of the external input. These bits can also allow an input capture operation.

Table 10.3-14 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 status) [initial value]
0	1	Rising edge detection 
1	0	Falling edge detection 
1	1	Both-edge detection  

10.4 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 operation.

■ 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 TCDT 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 10.4-1 Clearing the Counter by an Overflow

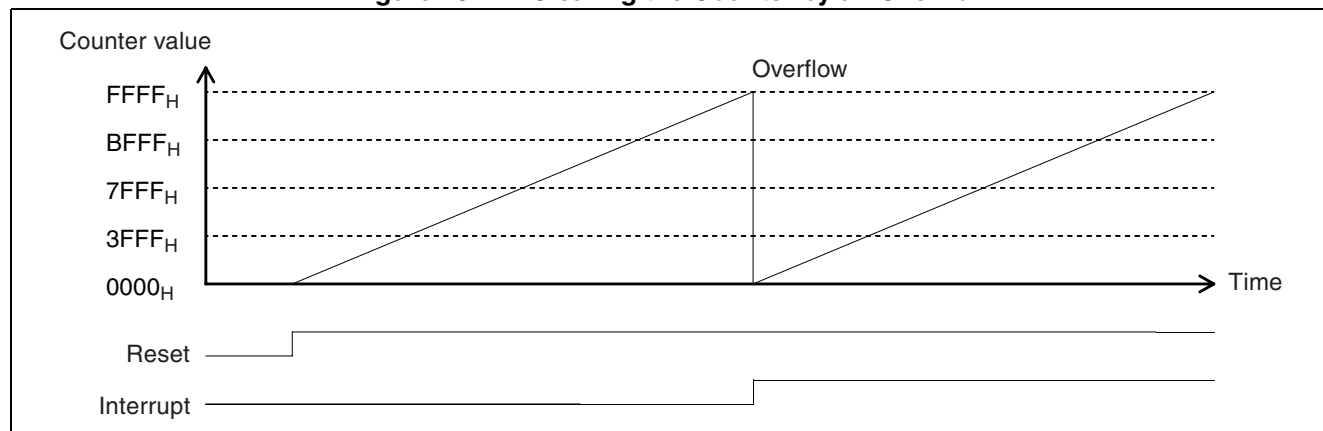
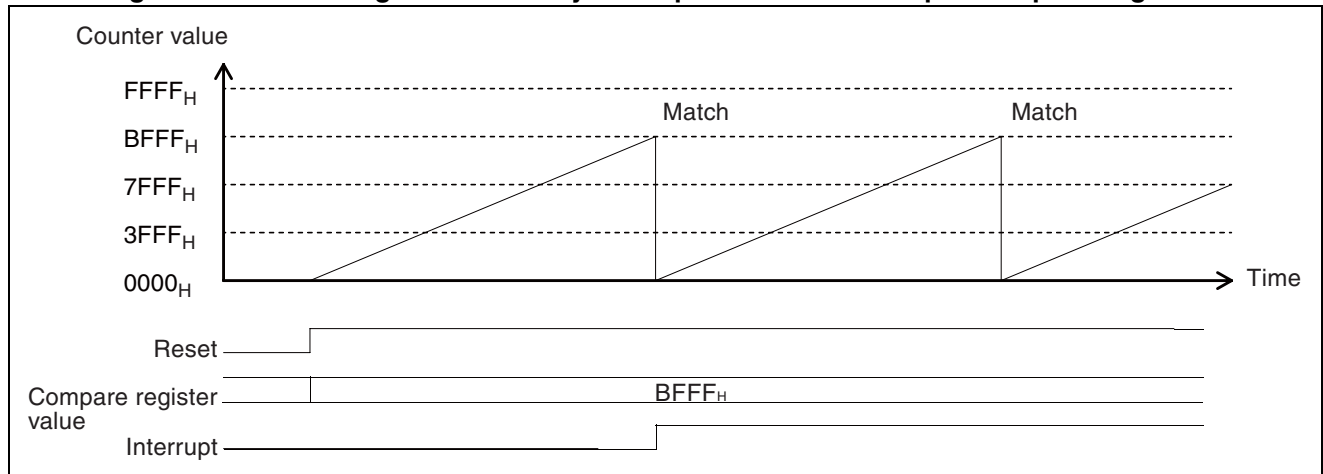
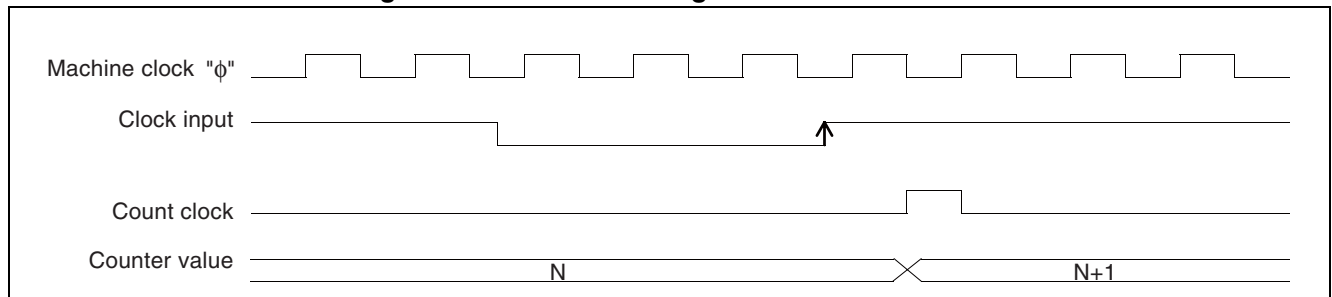


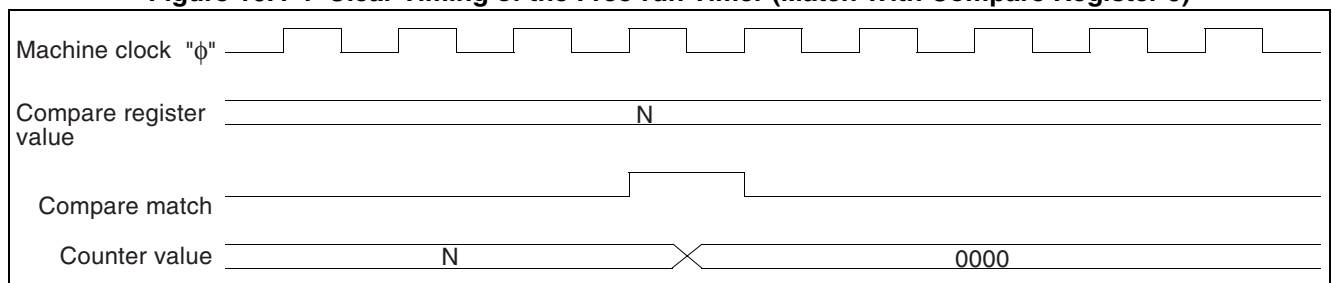
Figure 10.4-2 Clearing the Counter by a Compare Match with output Compare Register 0

■ 16-bit Free-run Timer Count Timing

The 16-bit free-run timer is counted up with an input clock.

Figure 10.4-3 Count Timing of the Free-run Timer

The counter can be cleared by a reset, software, or a match with compare register 0. The counter clear by the reset or software is executed simultaneously with clear generation. However, the counter clear by a match with compare register 0 is executed synchronously with the count timing.

Figure 10.4-4 Clear Timing of the Free-run Timer (Match With Compare Register 0)

10.5 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 reserve the output level.

■ 16-bit Output Compare Operations

Figure 10.5-1 Example of an Output Waveform when Compare Registers 0 and 1 are Used (When CMOD is "0")

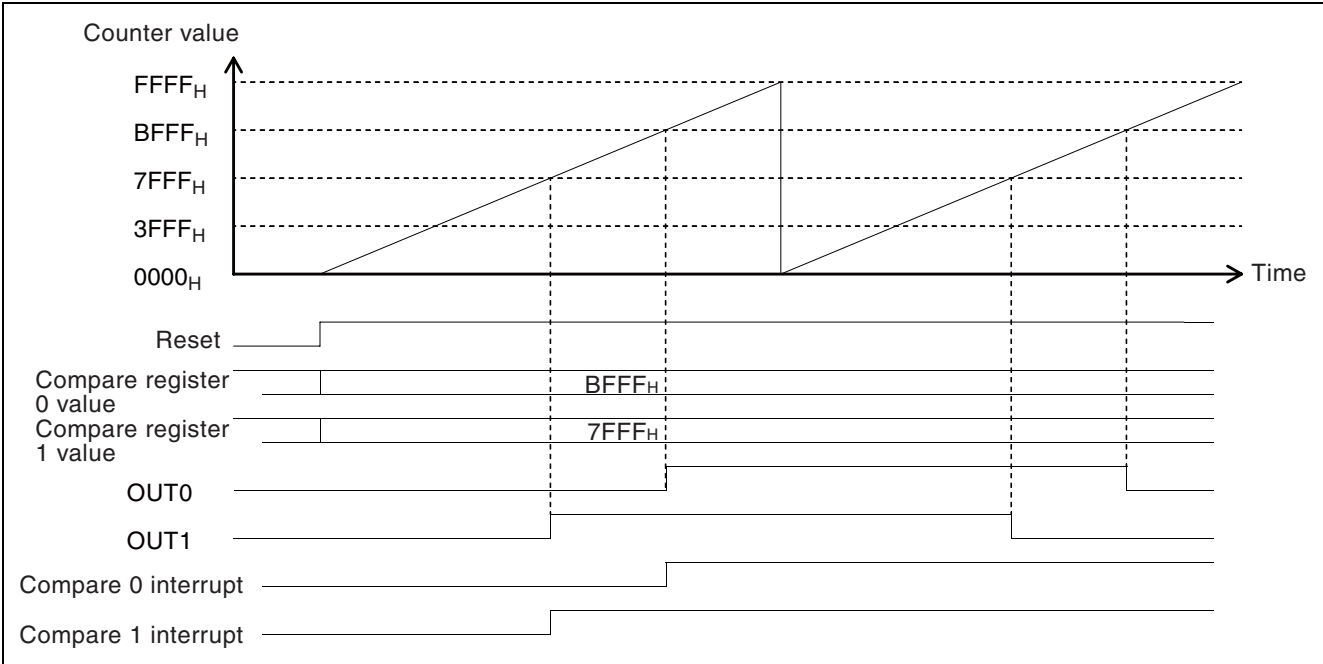
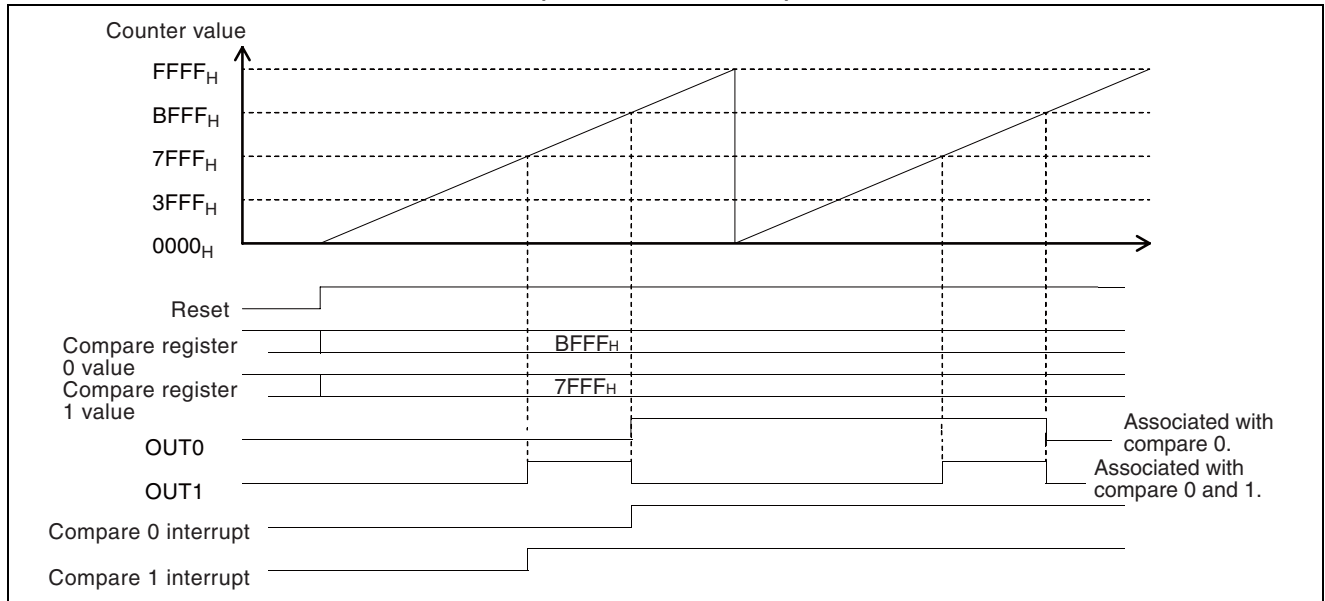


Figure 10.5-2 Example of an Output Waveform when Compare Registers 0 and 1 are Used (When CMOD is "1")



■ 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. A counter value at compare register rewriting is not compared.

Figure 10.5-3 Compare Operation at Compare Register Rewriting

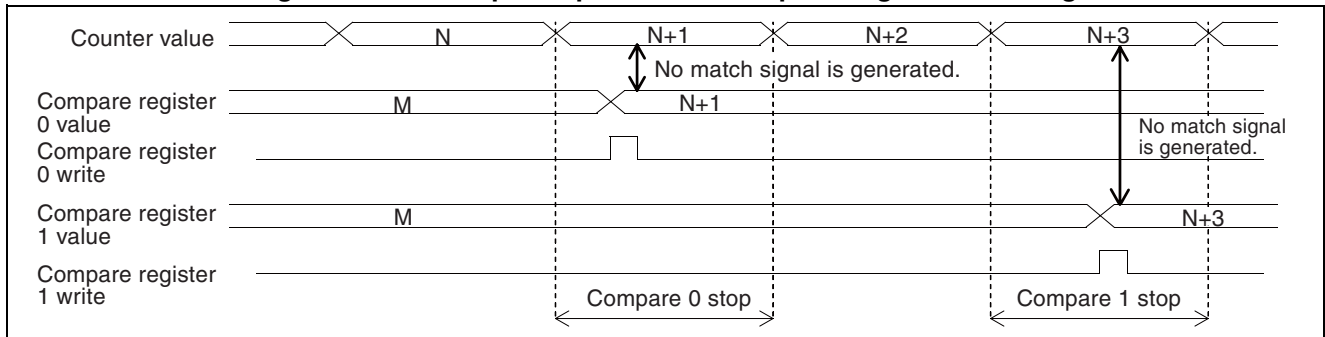


Figure 10.5-4 Interrupt Timing

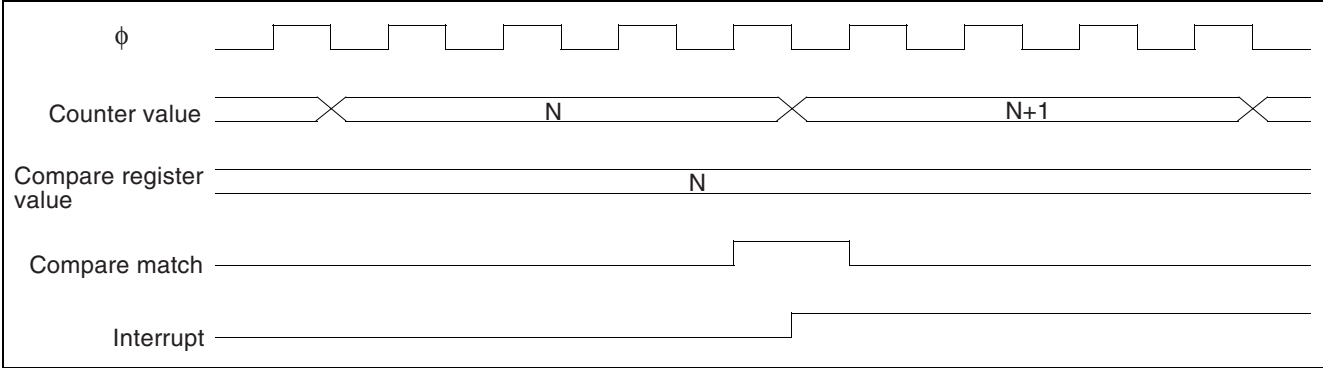
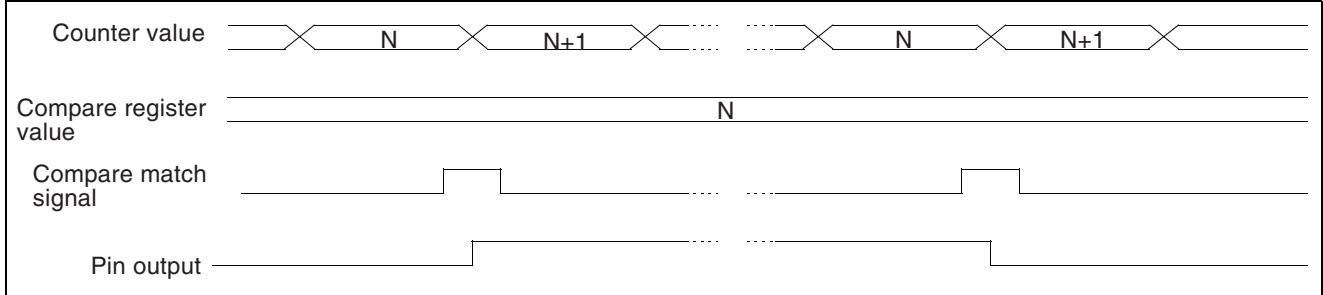


Figure 10.5-5 Output Pin Change Timing



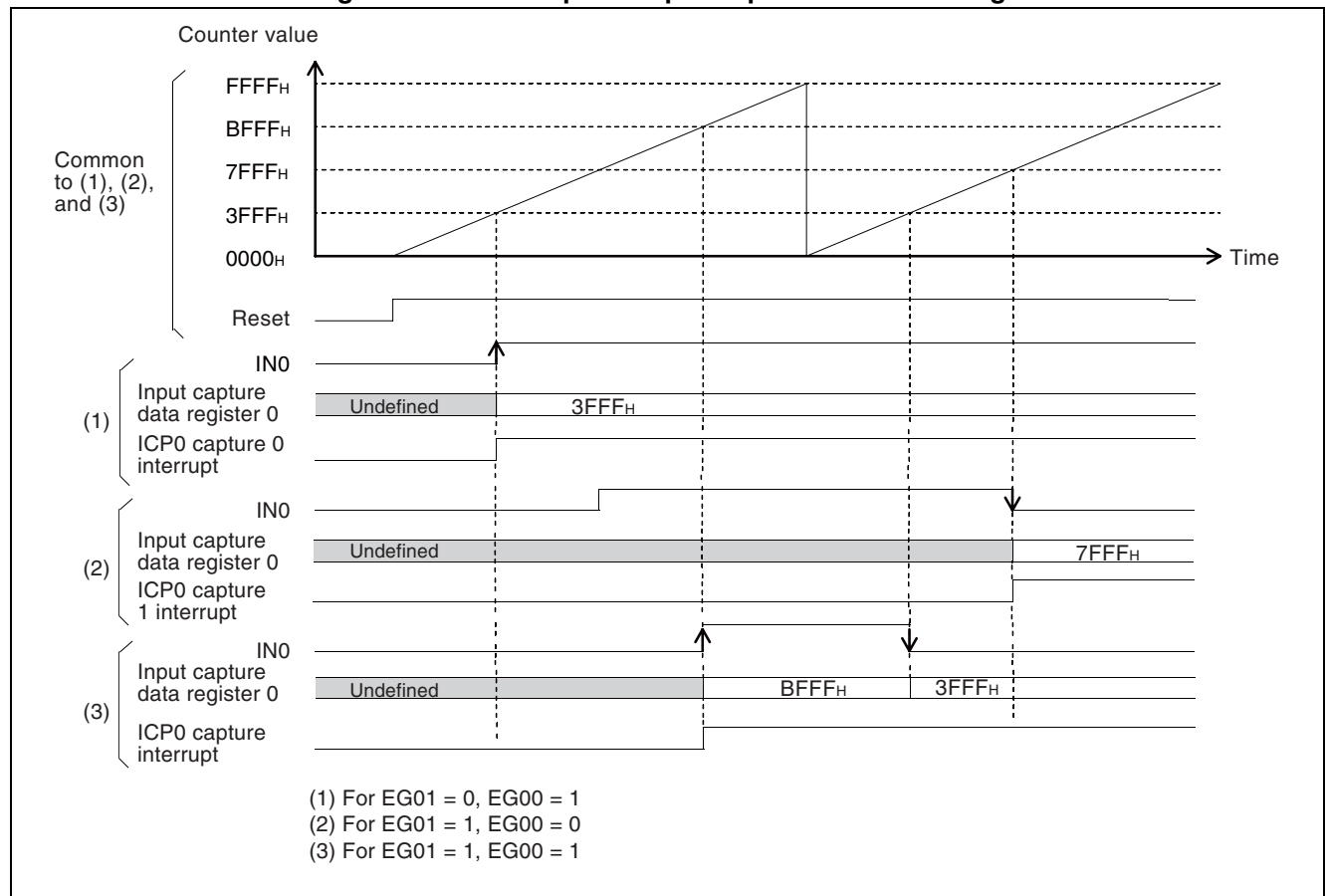
10.6 16-Bit Input Capture Operations

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 10.6-1 shows an example of the input capture take-in timings for 0ch. Other channels also perform the same operation.

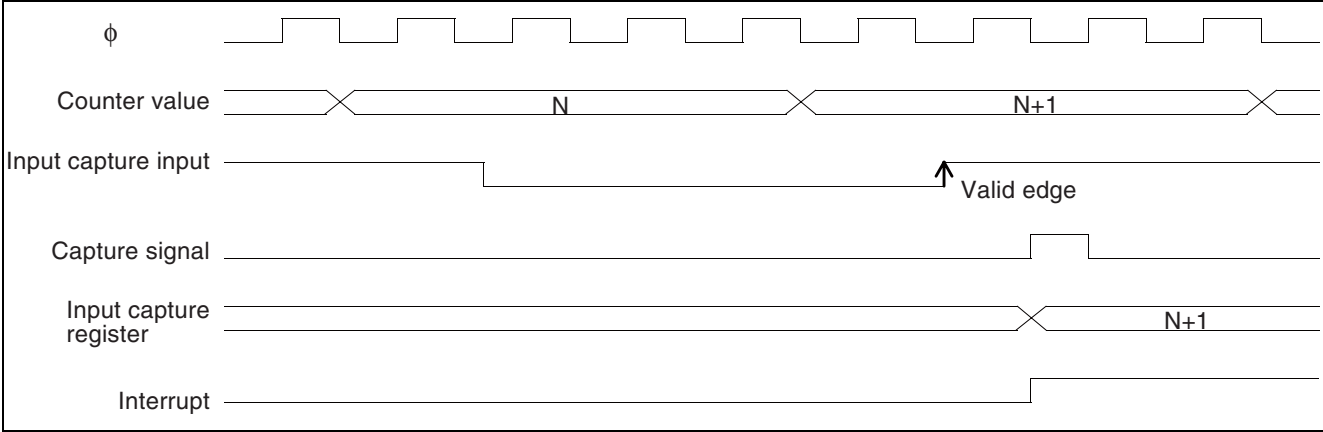
Figure 10.6-1 Example of Input Capture Take-in Timings



■ Input Capture Input Timing

Figure 10.6-2 shows the capture timing for an input signal when EGx1 is "0" and EGx0 is "1".

Figure 10.6-2 Capture Timing for Input Signals



CHAPTER 11 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.

- 11.1 Overview of the 16-Bit Reload Timer (with the Event Count Function)
- 11.2 Registers of the 16-Bit Reload Timer (with the Event Count Function)
- 11.3 Clock Operations
- 11.4 Underflow Operation
- 11.5 I/O Pin Functions
- 11.6 Counter Operation Statuses

■ Overview of the 16-bit Reload Timer (with the Event Count Function)

This series contains two channels of the 16-bit reload timer.

Figure 11.1-1 Block Diagram of the 16-bit Reload Timer (with the Event Count Function)



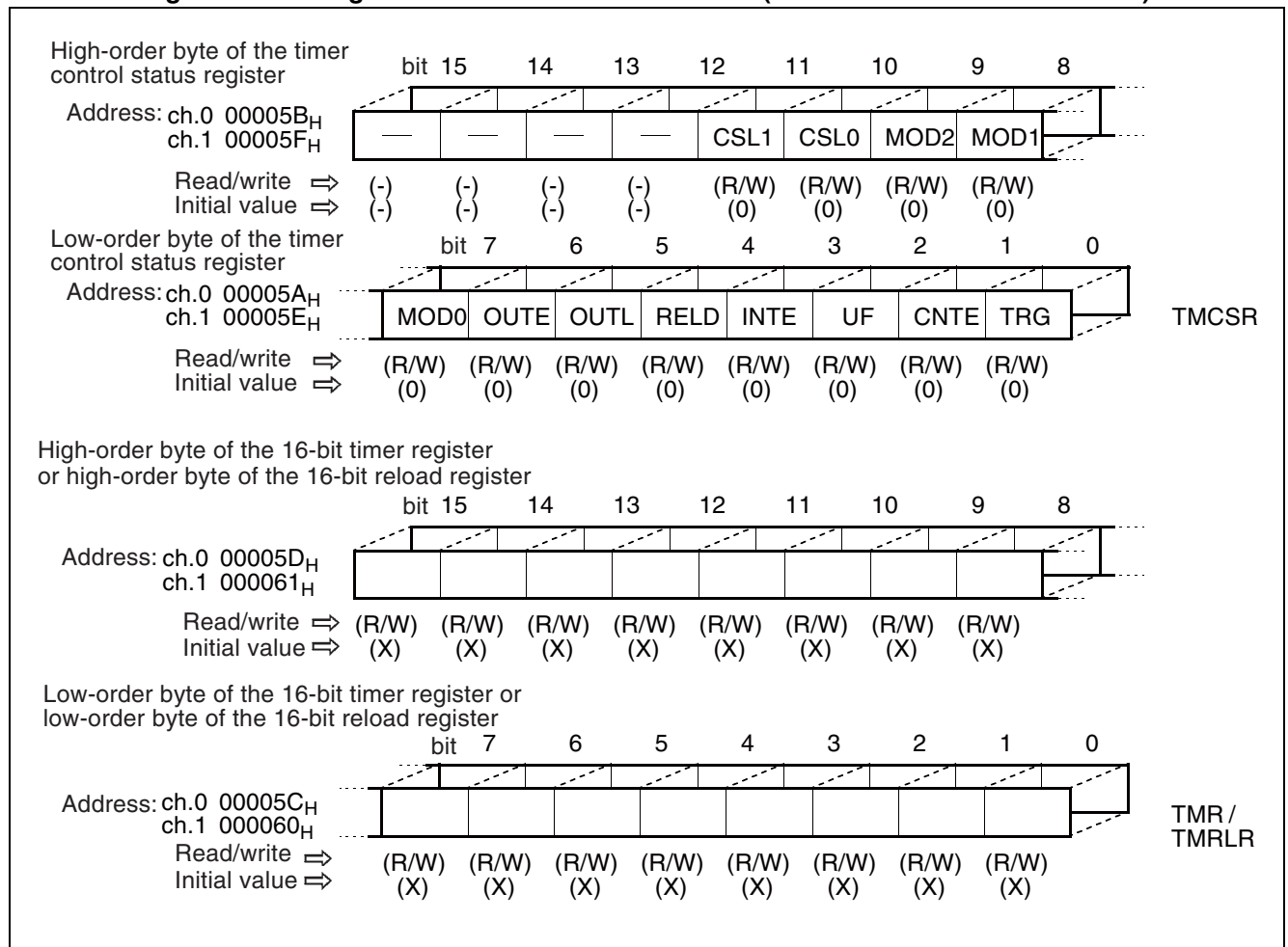
11.2 Registers of the 16-Bit Reload Timer (with the Event Count Function)

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 11.2-1 Registers of the 16-bit Reload Timer (with the Event Count Function)

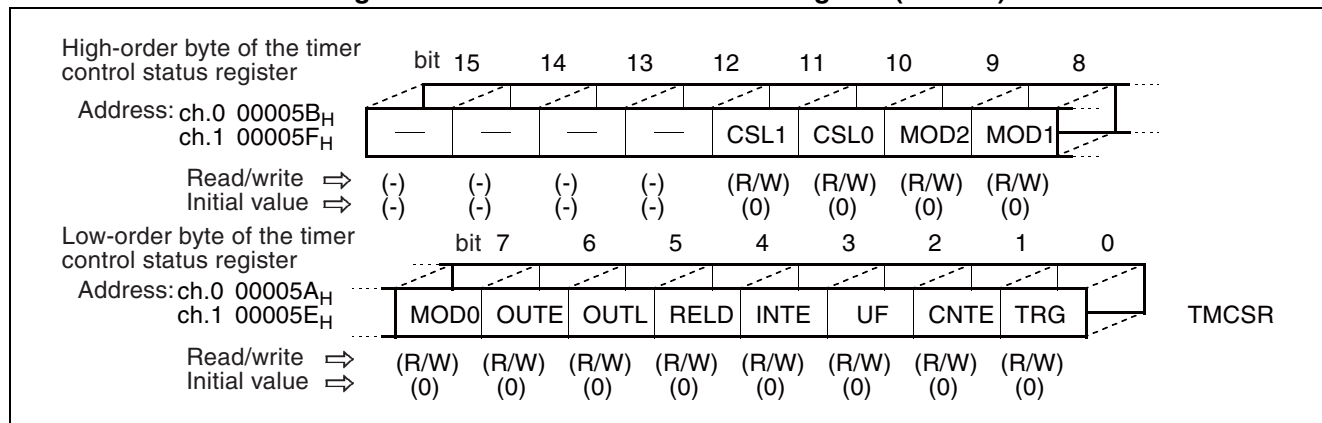


11.2.1 Timer Control Status Register (TMCSR)

The timer control status register (TMCSR) controls 16-bit timer operation modes and interrupts.

■ Timer Control Status Register (TMCSR)

Figure 11.2-2 Timer Control Status Register (TMCSR)



Note:

Rewrite a bit other than the UF, CNTE, and TRG bits when CNTE is "0".

[bit11 ,bit10] CSL1 and CSL0 (clock select 0, 1)

CSL1 and CSL0 bits are used to select a count clock. The following table lists the clock sources to be selected:

Table 11.2-1 Function of CSL1 and CSL0 (Count Clock Select Bits)

CSL1	CSL0	Clock source (machine cycle $\phi = 16 \text{ MHz}$)
0	0	$\phi/2^1$ (0.125 μs) [Initial value]
0	1	$\phi/2^3$ (0.5 μs)
1	0	$\phi/2^5$ (2.0 μs)
1	1	External event count mode

[bit9 to bit7] MOD2, MOD1, and MOD0

MOD2, MOD1, and MOD0 bits are used to set an operation mode and an I/O pin function.

The MOD2 bit is used to select an I/O function. If this bit is "0", the input pin (TIN) becomes a trigger input pin. If a valid edge is input, the contents of the reload register are loaded to the counter and the count operation continues. If this bit is "1", the gate counter mode is entered. The input pin (TIN) becomes a gate input and the count operation continues only while the active level is input.

The MOD1 and MOD0 bits are used to set a pin function in each mode.

Table 11.2-2 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, CSL1=00, 01, 10)	0	0	0	Trigger disabled	-
	0	0	1	Trigger input	Rising edge
	0	1	0		Falling edge
	0	1	1		Both edges
	1	×	0	Gate input	"L" level
	1	×	1		"H" level
Event count mode (CSL0, CSL1=11)	×	0	0	-	-
		0	1	Trigger input	Rising edge
		1	0		Falling edge
		1	1		Both edges

× : Any value

[bit6] 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.

[bit5] OUTL

The OUTL bit sets the output level of the TOT pin.

[bit4] 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 11.2-3 Function of RELD (reload operation enable bit)

OUTE	OUTL	RELD	Output waveform
0	×	×	General-purpose port
1	0	0	Rectangular wave of H during counting
1	0	1	Toggle output of L at count start
1	1	0	Rectangular wave of L during counting
1	1	1	Toggle output of H at count start

× : Any value

[bit3] INTE (Interrupt enable)

The INTE bit enables a timer interrupt request.

Table 11.2-4 Function of INTE (for Enabling a Timer Interrupt Request)

INTE	Function
0	Interrupt disabled
1	Interrupt enabled

[bit2] 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 intelligent I/O service. Writing "1" in this bit is meaningless. Value "1" is read at reading by read modify write instructions.

[bit1] 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.

[bit0] 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".

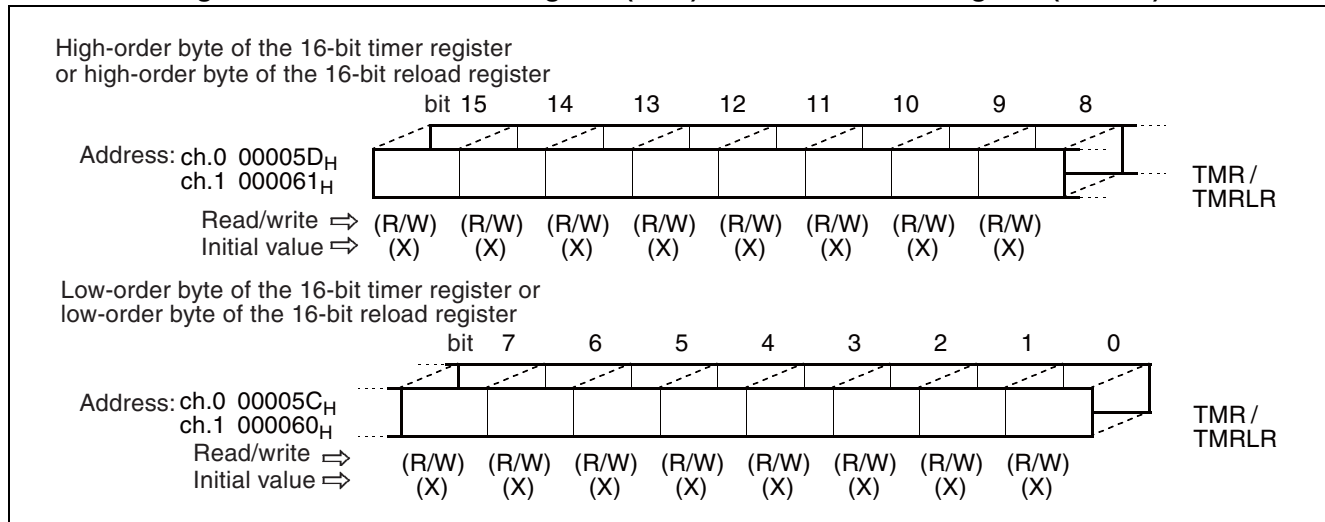
11.2.2 16-Bit Timer Register (TMR) and 16-Bit Reload Register (TMRLR)

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 11.2-3 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).

11.3 Clock Operations

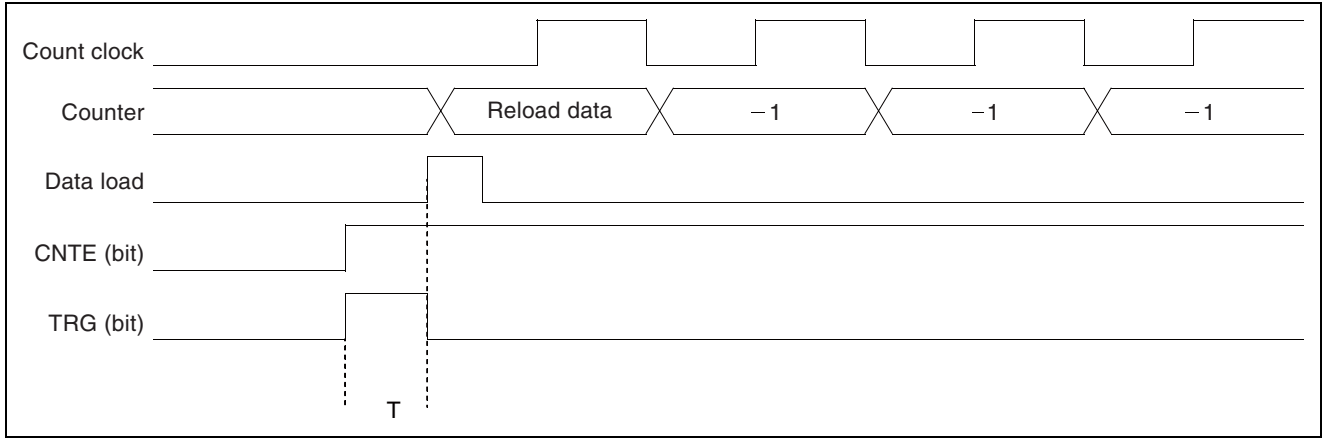
When the timer is operated with a divide-by clock of internal clocks, the divide-by clock can be selected, as a clock source, from 2^1 , 2^3 , and 2^5 divide-by clocks of the machine clock. The external input pin can be used as the trigger or gate input by the register setting.

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 11.3-1 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 4T (T: Machine cycle) to the TIN pin.

11.4 Underflow Operation

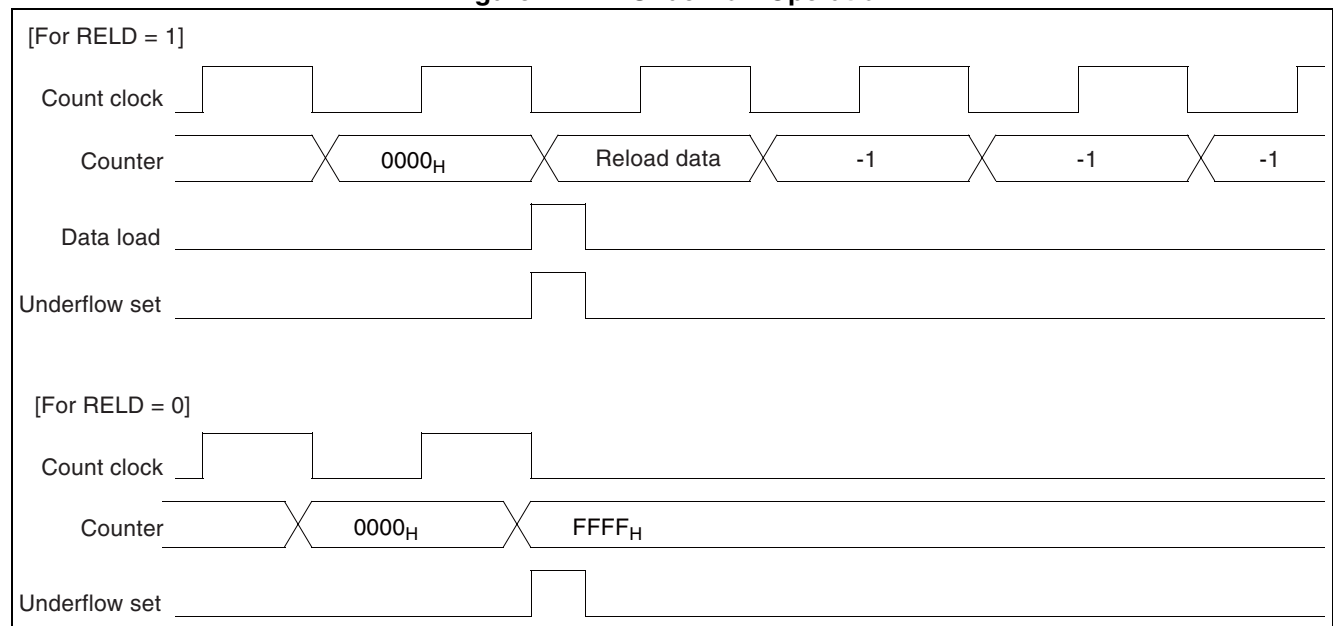
The 16-bit reload timer (with the event count function) defines the following case as an underflow: when a counter value changes 0000_H to FFFF_H.

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 11.4-1 Underflow Operation



■ Extended Intelligent I/O Service (EI²OS) Function and Interrupts

This timer has a circuit associated with EI²OS. Therefore, EI²OS can be activated by an underflow of this timer. For this product, both timers can use EI²OS.

11.5 I/O Pin Functions

If an internal clock is selected as a clock source, the TIN pin can be used either as a trigger or a 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.

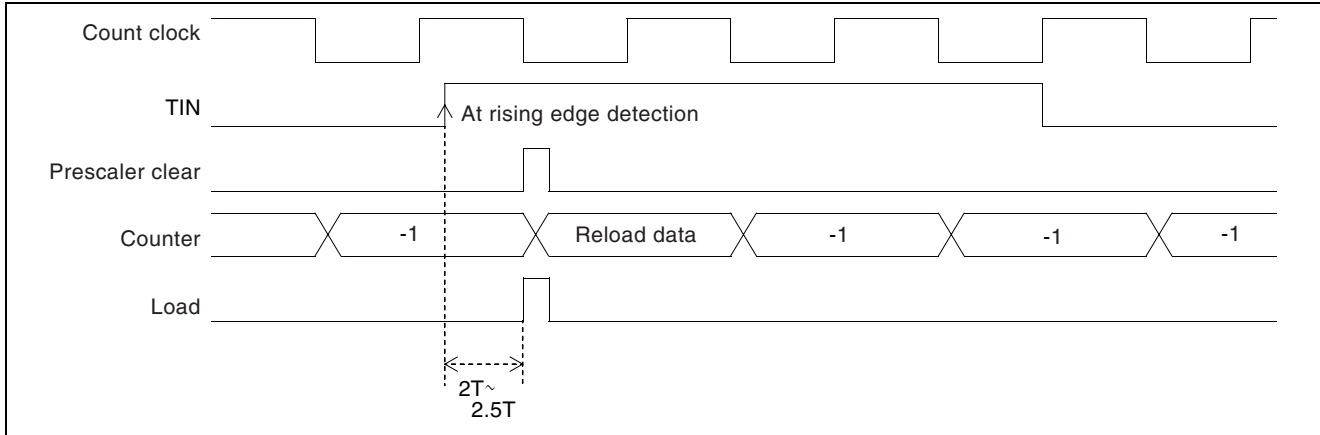
■ Input Pin Function (for the Internal Clock Mode)

If an internal clock is selected as the clock source, the TIN pin can be used as either a trigger or a gate input.

If the TIN pin is used as the trigger input, when an active edge is input as shown in Figure 11.5-1, the contents of the reload register are loaded to the counter. After the internal prescaler is cleared, the count operation starts.

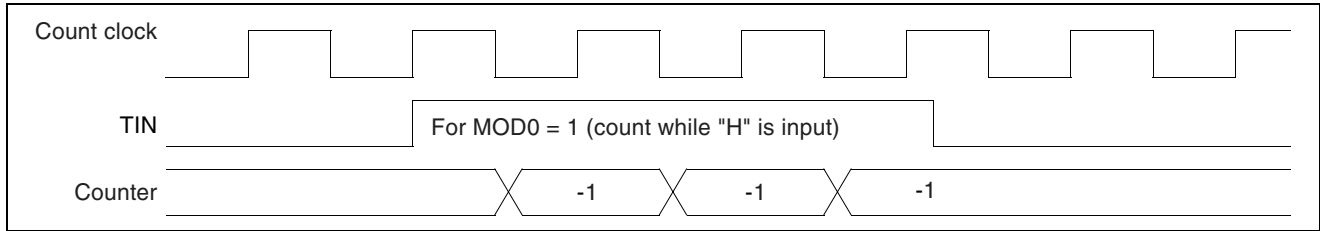
Input a pulse width of at least $2T$ (T : machine cycle) to TIN.

Figure 11.5-1 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 11.5-2. 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 $2T$ (T : machine cycle) to the TIN pin.

Figure 11.5-2 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 11.5-3 Output Pin Function (1)

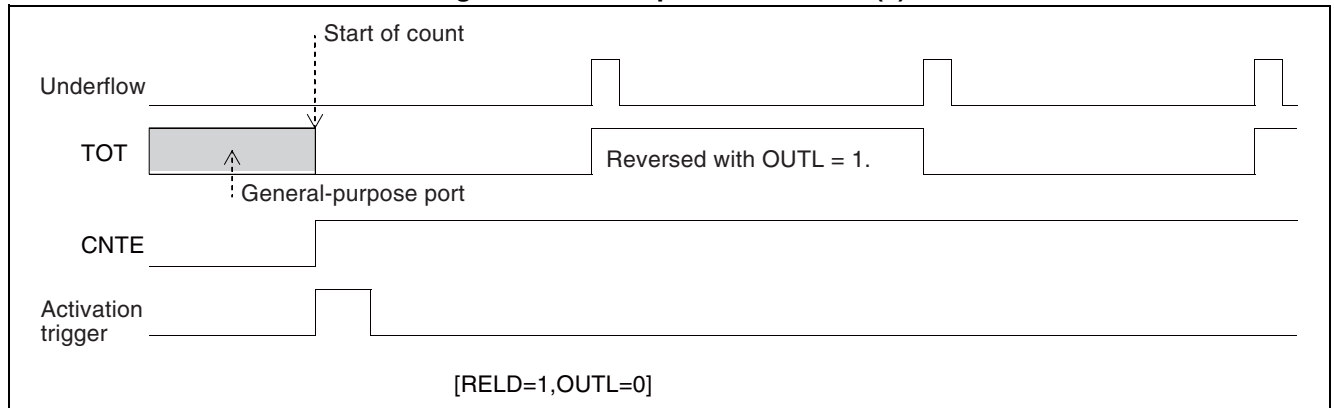
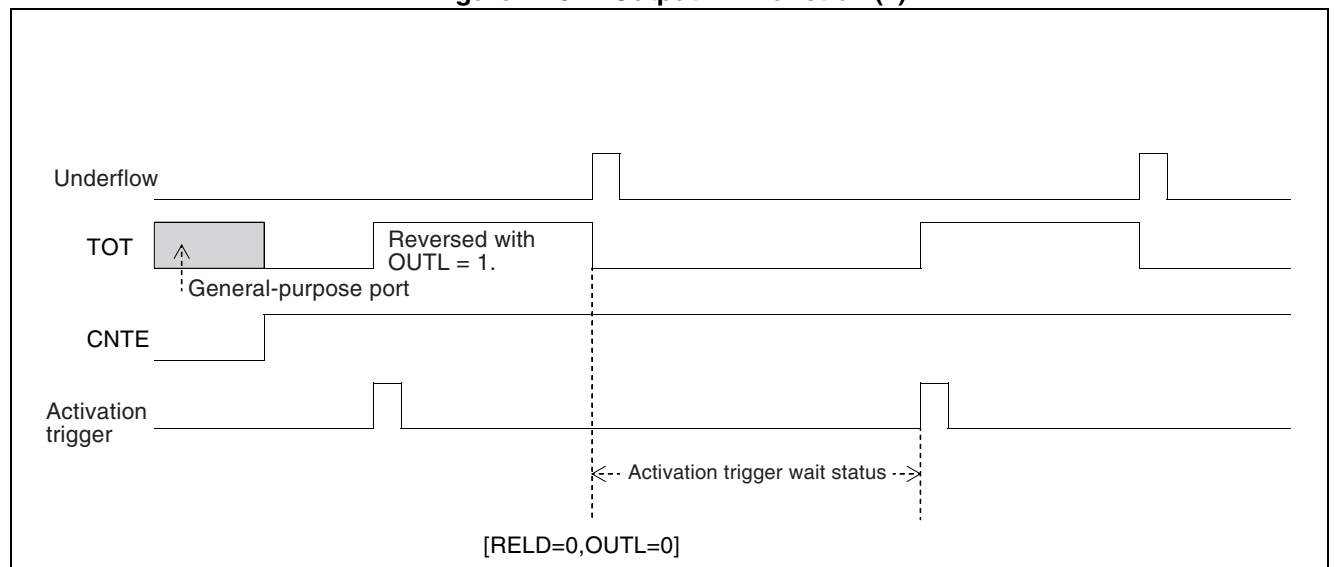


Figure 11.5-4 Output Pin Function (2)



11.6 Counter Operation Statuses

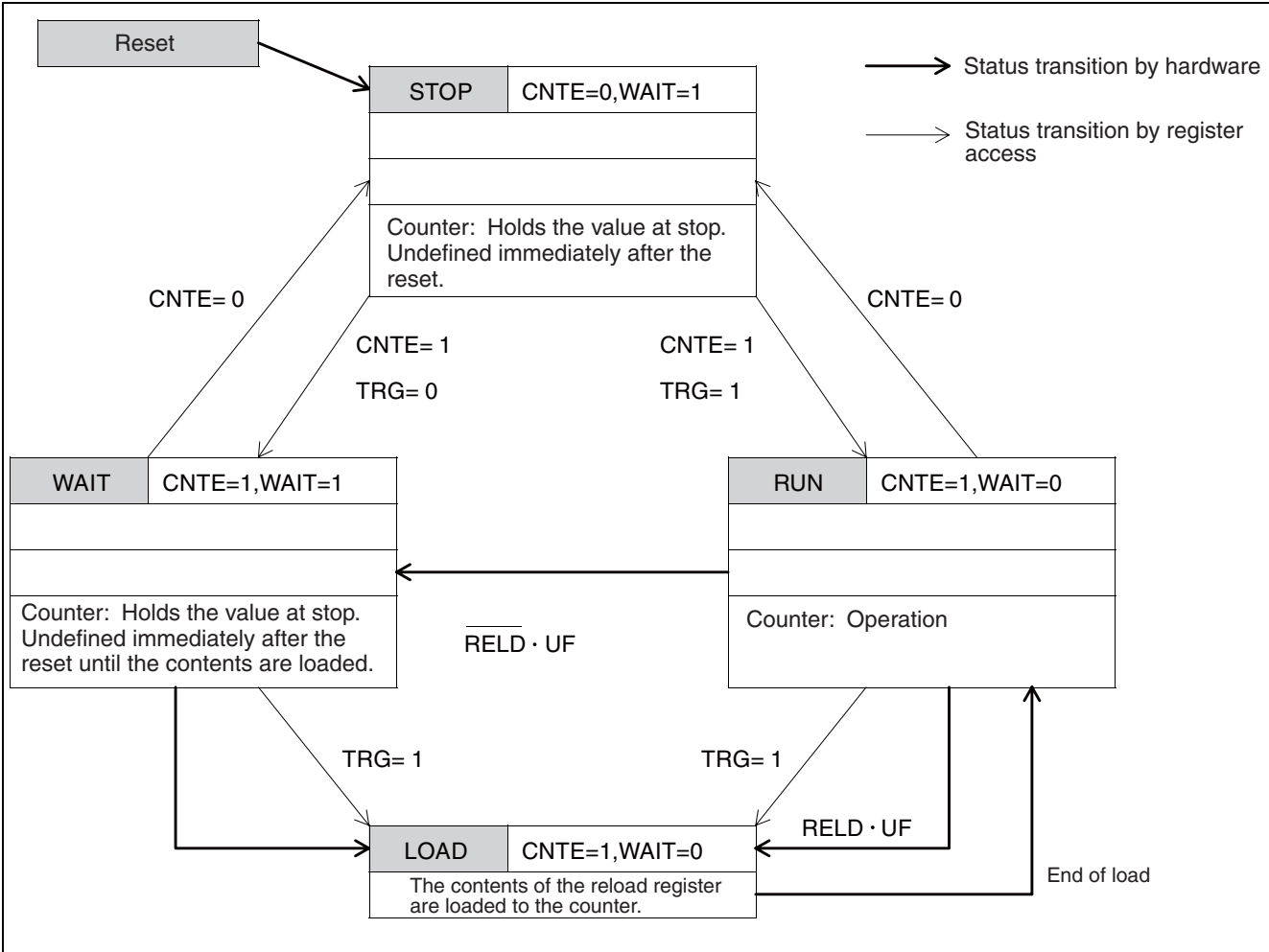
A counter status is determined by the CNTE bit of the control register and the WAIT signal of the internal signal. The following three statuses can be set:

- 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 11.6-1 shows the counter operation statuses.

Figure 11.6-1 Counter Status Transition



CHAPTER 12 8/16-BIT PPG

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

- 12.1 Overview of the 8/16-Bit PPG
- 12.2 Block Diagrams of the 8-Bit PPG
- 12.3 Registers in the 8/16-Bit PPG
- 12.4 8/16-Bit PPG Operation

12.1 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. The MB90550A/B Series contains three 8/16-bit PPGs.

■ Overview of the 8/16-bit PPG

The 8/16-bit PPG consists of two 8-bit down counters, four 8-bit reload registers, three 8-bit control registers, two external pulse output pins, and two interrupt outputs. 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. The PPG provides the following functions:

- **8-bit PPG output in 6-channel independent operation mode (8-bit PPG 2-ch mode)**

Allows 6-channel independent PPG output operation.

- **16-bit PPG output operation mode (16-bit PPG 1-ch mode)**

Allows 3-channel 16-bit PPG output operation.

- **Pair 8-bit-prescaler + 8-bit - PPG mode**

Allows 8-bit PPG output operation with an arbitrary cycle period by applying ch.0/ch.2/ch.4 output to the ch.1/ch.3/ch.5 clock input.

12.2 Block Diagrams of the 8-Bit PPG

Figure 12.2-1 shows a block diagram of the 8-bit PPG (ch.0/ch.2/ch.4), and Figure 12.2-2 shows a block diagram of the 8-bit PPG (ch.1/ch.3/ch.5).

■ Block Diagrams of the 8-bit PPG

Figure 12.2-1 Block Diagram of the 8-bit PPG (ch.0/ch.2/ch.4)

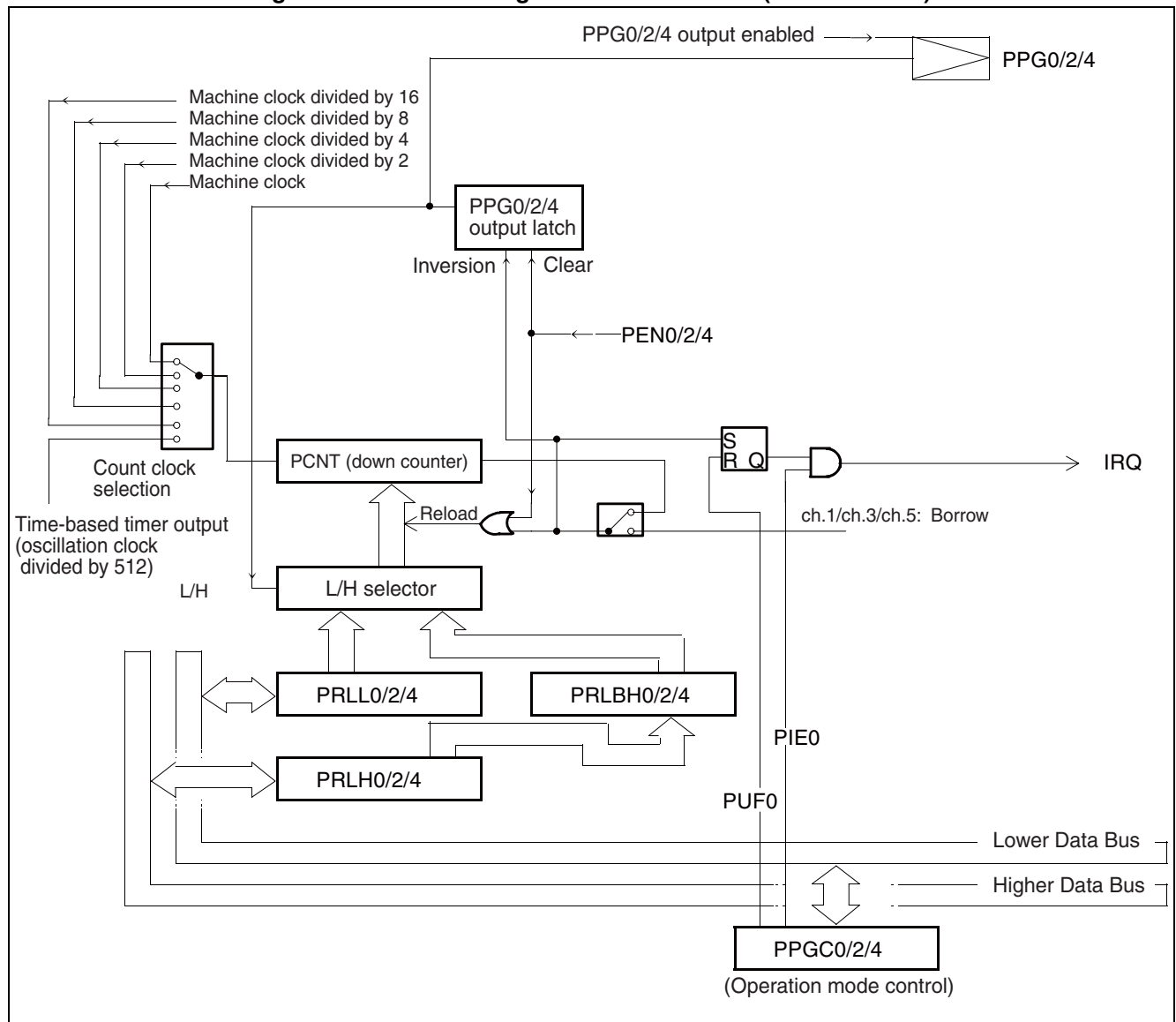
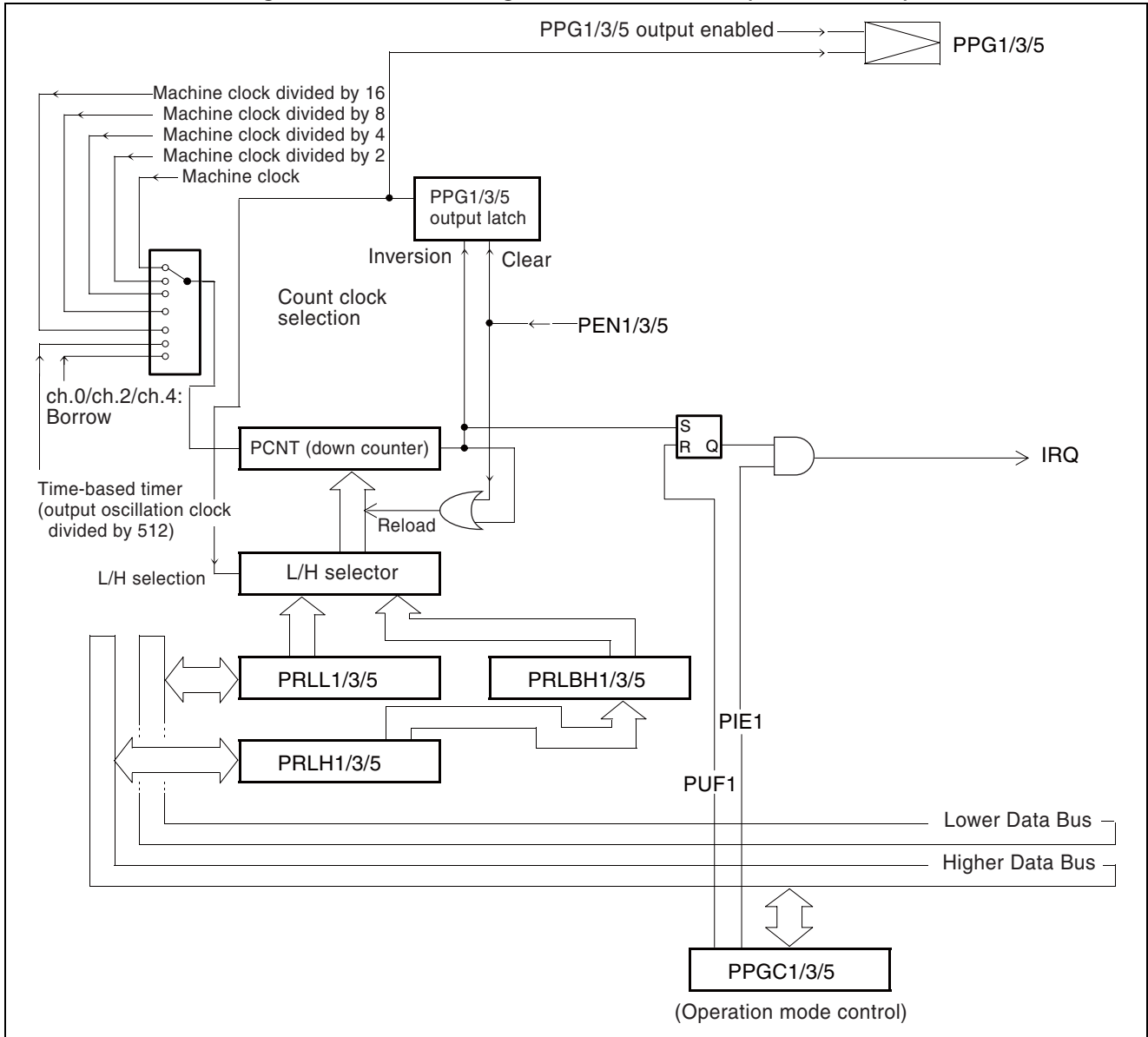


Figure 12.2-2 Block Diagram of the 8-bit PPG (ch.1/ch.3/ch.5)



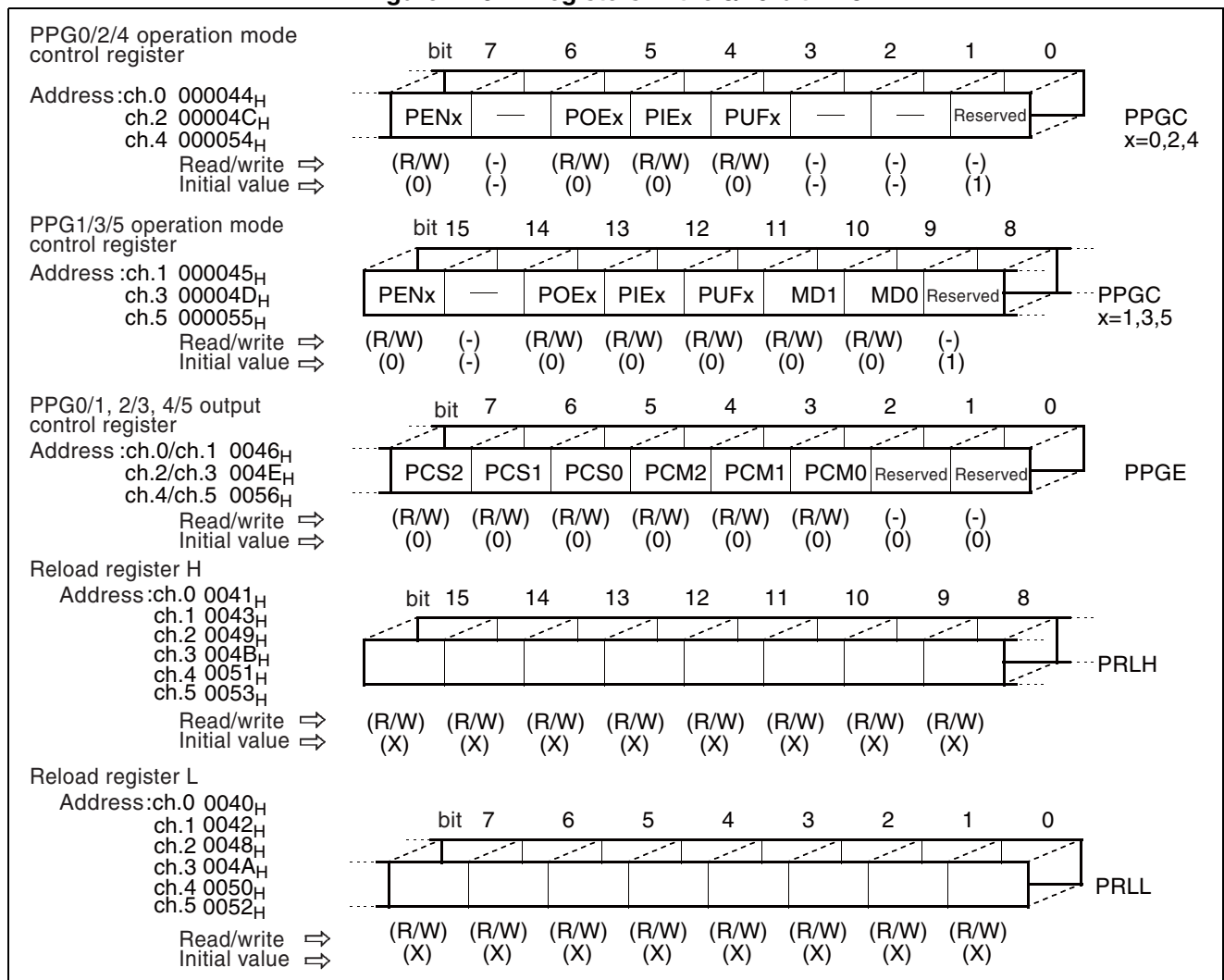
12.3 Registers in the 8/16-Bit PPG

The registers in the 8/16-bit PPG are classified into the following three types:

- PPG operation mode control register
- PPG output control register
- Reload register

■ Registers in the 8/16-bit PPG

Figure 12.3-1 Registers in the 8/16-bit PPG



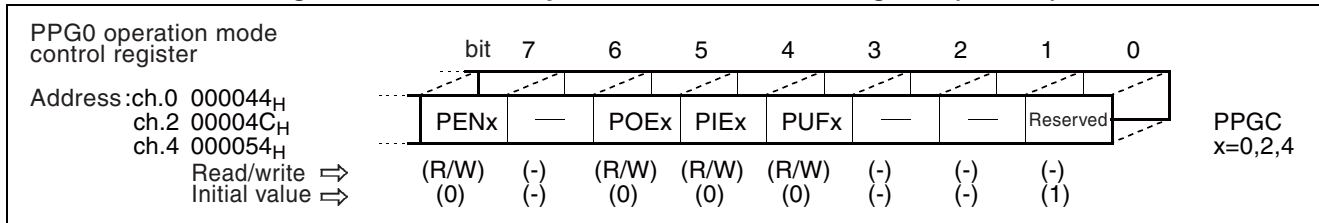
12.3.1 PPG0 Operation Mode Control Register (PPGC0)

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.

Here, ch.0 is used as an example. For ch.2 and ch.4, replace ch.0 in the explanation with ch.2 or ch.4. For ch.3 and ch.5 replace ch.1 in the explanation with ch.3 or ch.5.

■ PPG0 Operation Mode Control Register (PPGC0)

Figure 12.3-2 PPG0 Operation Mode Control Register (PPGC0)



[bit7] PEN0 (Ppg ENable)

The PEN0 bit selects the start of PPG operation and the operation mode of the PPG as shown in Table 12.3-1. Writing "1" to this bit causes the PPG to start counting.

Table 12.3-1 PEN0 (Operation Enable Bit) Function

PEN0	Function
0	Operation stopped (low level held output) [initial value]
1	PPG operation enabled

[bit5] POE0 (Ppg Output Enable)

The POE0 bit controls the pulse output external pin PPG0 as shown in Table 12.3-2.

Table 12.3-2 POE0 (PPG0 Pin Output Enable Bit) Function

POE0	Function
0	PPG output disabled (general-purpose port pin) [initial value]
1	PPG0 output enabled (PPG output pin)

[bit4] PIE0 (Ppg Interrupt Enable)

The PIE0 bit enables or disables PPG interrupts as shown in Table 12.3-3.

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.

Table 12.3-3 PIE0 (PPG Interrupt Enable Bit) Function

PIE0	Function
0	Interrupt disabled [initial value]
1	Interrupt enabled

[bit3] PUF0 (Ppg Underflow Flag)

The PUF0 bit is a PPG counter underflow bit. Table 12.3-4 shows the relationship between this bit and the 8-bit down counter PCNT.

Table 12.3-4 PUF0 (PPG Counter Underflow Bit) Function

PUF0	Function
0	Underflow not detected in down counter PCNT [initial value]
1	Underflow detected in down counter PCNT

In the 8-bit PPG 2-ch mode and the 8-bit prescaler + 8-bit PPG mode, an underflow caused when the ch.0 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 ch.1/ch.0 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.

[bit0] Reserved bit

Bit0 is a reserved bit. Whenever setting PPGC0, be sure to set this bit to "1".

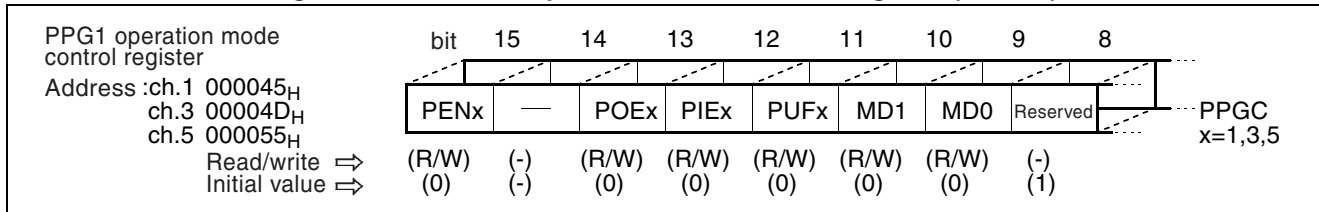
12.3.2 PPG1 Operation Mode Control Register (PPGC1)

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.

Here, ch.1 is used as an example. For ch.2 and ch.4, replace ch.0 in the explanation with ch.2 or ch.4. For ch.3 and ch.5 replace ch.1 in the explanation with ch.3 or ch.5.

■ PPG1 Operation Mode Control Register (PPGC1)

Figure 12.3-3 PPG1 Operation Mode Control Register (PPGC1)



[bit15] PEN1 (Ppg ENable)

The PEN1 bit selects the start of PPG operation and the operation mode of the PPG as shown in Table 12.3-5. Writing "1" to this bit causes the PPG to start counting.

Table 12.3-5 Operation Enable Bit (PEN1) Function

PEN1	Function
0	Operation stopped (low level held output) [initial value]
1	PPG operation enabled

[bit13] POE1 (Ppg Output Enable)

The POE1 bit controls the pulse output external pin PPG1 as shown in Table 12.3-6.

Table 12.3-6 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)

[bit12] PIE1 (Ppg Interrupt Enable)

The PIE1 bit enables or disables PPG interrupts as shown in Table 12.3-7. 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 12.3-7 PIE1 (PPG Interrupt Enable Bit) Function

PIE1	Function
0	Interrupt disabled [initial value]
1	Interrupt enabled

[bit11] PUF1 (Ppg Underflow Flag)

The PUF1 bit is a PPG counter underflow bit. Table 12.3-8 shows the relationship between this bit and the 8-bit down counter PCNT.

In the 8-bit PPG 2-ch mode and the 8-bit prescaler plus 8-bit PPG mode, an underflow caused when the ch.1 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 ch.1/ch.0 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 12.3-8 PUF1 (PPG Counter Underflow Bit) Function

PUF1	Function
0	Underflow not detected in down counter PCNT [initial value]
1	Underflow detected in down counter PCNT

[bit10, bit9] MD2, MD1 (ppg count MoDe)

The MD2 and MD1 bits select the operation mode of the PPG timer as shown in Table 12.3-9.

A reset initializes these bits to "00".

This bit can be read from and written to.

Table 12.3-9 MD2 and MD1 (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".

When setting these bits to "01", do not set the PEN0 bit of PPGC0 to "0" and the PEN1 bit of PPGC1 to "1". It is recommended that the PEN0 and PEN1 bits be set to "11" or "00" at the same time.

When setting these bits to "11", rewrite PPGC0/PPGC1 by a word transfer to set the PEN0 and PEN1 bits to "11" or "00" at the same time.

[bit8] Reserved bit

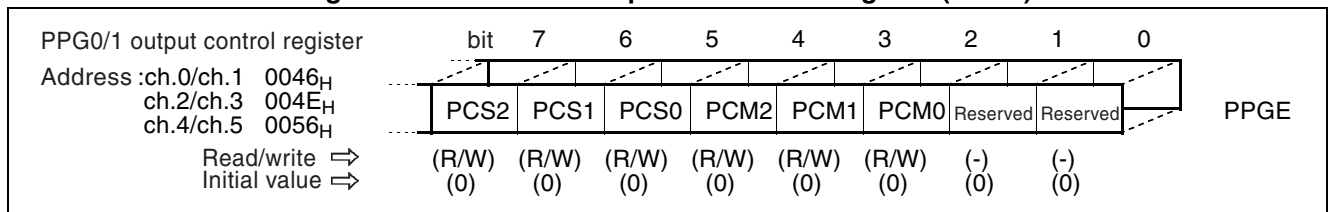
bit8 is a reserved bit. Whenever setting PPGC1, be sure to set this bit to "1".

12.3.3 PPG0/1 Output Pin Control Register (PPGE)

The PPG0/1 output pin control register (PPGE) is an 8-bit control register for 8/16-bit PPG pin output control.

■ PPG0/1 Output Pin Control Register (PPGE)

Figure 12.3-4 PPG0/1 Output Pin Control Register (PPGE)



[bit7 to bit5] PCS2 to PCS0 (Ppg Count Select)

PCS2 to PCS0 select the operation clock for the ch.1 down counter as shown in Table 12.3-10.

Table 12.3-10 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 ch.1 operates with the count clock signal received from ch.0. Therefore, the settings on PCS2 to PCS0 are ignored.

[bit4 to bit2] PCM2 to PCM0 (Ppg Count Mode)

The PCM2 to PCM0 bits select the operation clock of the ch.0 down counter as shown in Table 12.3-11.

Table 12.3-11 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)

[bit1 ,bit0] Reserved bits

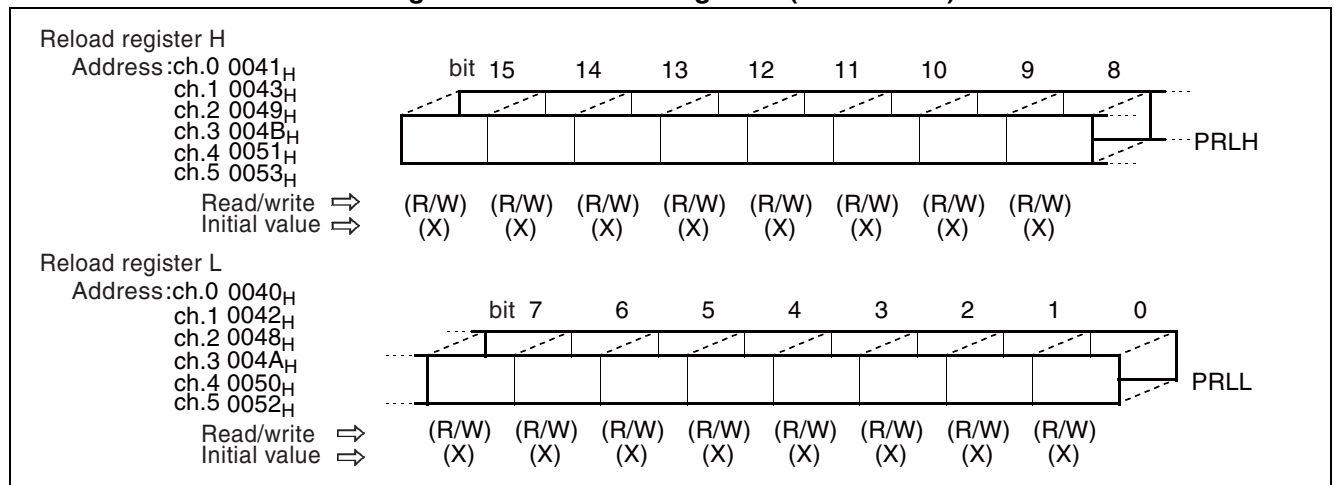
Bit1, bit0 are reserved bits. Whenever setting PPGE, be sure to set these bits to "0".

12.3.4 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 12.3-5 Reload Registers (PRLH/PRLH)



The reload registers (PRLH/PRLH) have the function shown in Table 12.3-12. Each register can be read from and written to.

Table 12.3-12 Reload Registers (PRLH/PRLH)

Register name	Function
0	Holds the reload value for lower data.
1	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 ch.0 may vary the PPG waveform on ch.1 from cycle to cycle. Therefore, it is recommended that the same value be set in PRLH and PRLH for ch.0.

12.4 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.

Here, ch.0 and ch.1 are used as an example. For ch.2 and ch.4, replace ch.0 in the explanation with ch.2 or ch.4. For ch.3 and ch.5, replace ch.1 in the explanation with ch.3 or ch.5.

■ 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 12.4-1.

Table 12.4-1 Relationship between Reload Operation and Pulse Output

Reloading	Status transition of output pins PPG0 and PPG1
PRLH --> PCNT	0 --> 1
PRLH --> PCNT	1 --> 0

When the PIE0 bit of the PPGC0 register is "1" and when the 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 --> 0X000001_B
- PPGC1 --> 00000001_B
- PPGOE --> XXXXXX00_B

○ Pulse output

- PPG0 --> "L"
- PPG1 --> "L"
- PE00 --> PPG0 output disabled
- PE10 --> PPG1 output disabled

○ Interrupt request

- IRQ0 --> "L"
- IRQ1 --> "L"

Hardware components other than the above are not initialized.

12.4.1 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 this operation mode, the two 8-bit PPG units are operated independently of each other.

The PPG0 pin is connected to the PPG output of ch.0, and the PPG1 pin is connected to the PPG output of ch.1.

○ 8-bit prescaler + 8-bit PPG mode

In this operation mode, an 8-bit PPG waveform with an arbitrary cycle period can be output by operating ch.0 as an 8-bit prescaler and counting ch.1 with ch.0 borrow output.

The PPG0 pin is connected to the prescaler output of ch.0, and the PPG1 pin is connected to the PPG output of ch.1.

○ 16-bit PPG 1-ch mode

In this operation mode, ch.0 and ch.1 interact, enabling 16-bit PPG operation. The PPG0 pin and PPG1 pin are both connected to 16-bit PPG output.

12.4.2 PPG Output Operation

In the 8/16-bit PPG, the PPG unit on ch.0 is activated and starts counting when the PEN0 bit of the PPGC0 register is set to "1". The PPG unit on ch.1 is activated and starts counting when the PEN1 bit of the PPGC1 register is set to "1". At the start of an operation, a counting operation is stopped by writing "0" to the PEN0 bit of the PPGC0 register or PEN1 bit of the PPGC1 register. After the counting operation stops, the pulse output is held low.

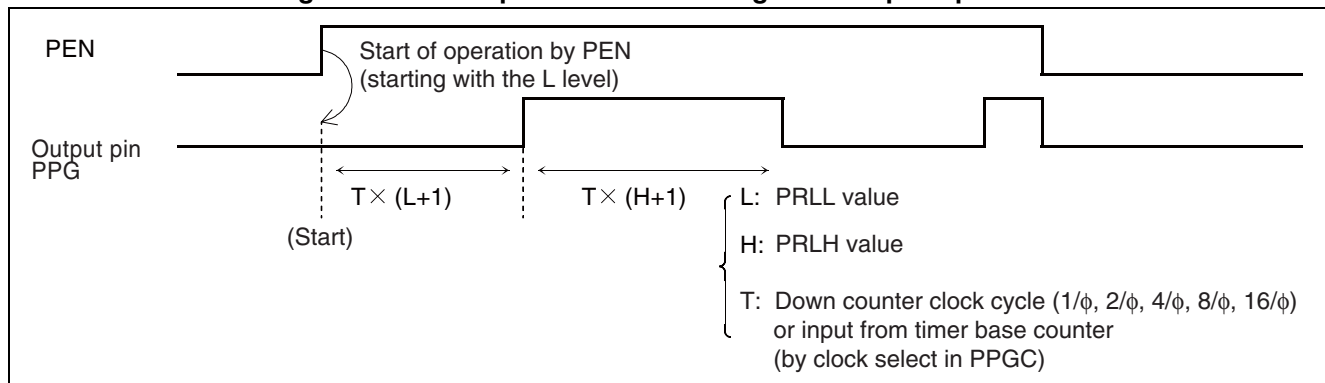
■ PPG Output Operation

In PPG output operation, observe the following two precautions:

- In the 8-bit prescaler + 8-bit PPG mode, do not enable ch.1 operation while ch.0 is in the stopped state.
- In the 16-bit PPG mode, perform start and stop control by manipulating the PEN0 bit of the PPGC0 register and the PEN1 bit of the PPGC1 register at the same time.

During PPG operation, a pulse waveform is output successively with an arbitrary frequency and an arbitrary duty cycle. Once the PPG starts outputting a pulse waveform, it does not stop until the operation is set to stop.

Figure 12.4-1 Output Waveform During PPG Output Operation



■ Relationship between the Reloaded Value and Pulse Width

As shown in Table 12.4-1, the pulse width of the output pulse signal is obtained by multiplying the value written in the reload register plus "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:

$$PI = T \times (L + 1)$$

$$Ph = T \times (H + 1)$$

L: PRL value

H: PRLH value

T: Input clock cycle

Ph: High-level pulse width

PI: Low-level pulse width

12.4.3 Selecting a Count Clock

As the count clock used for 8/16-bit PPG operation, machine clock and time-based counter inputs can be used. Six types of count clock input is selectable. The clock for ch.0 is selected by the PCM2 to PCM0 bits of the PPGE register, and the clock for ch.1 is selected by the PCS2 to PCS0 bits. The clock signal can be selected from the clocks produced by dividing the machine clock by 16 to 1 and the time-based timer input. In the 8-bit prescaler + 8-bit PPG mode and the 16-bit PPG 1-ch mode, however, the PPG unit on ch.1 operates with the count clock signal input from ch.0. Therefore, the value of the PCS1 bit in the PPGC1 register is ignored.

■ 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 plus 8-bit PPG mode, when ch.0 is operating and ch.1 is in the stopped state, starting ch.1 may shift the first count cycle.

12.4.4 Controlling Pulse Output on Pins

The pulse output generated by operating this module can be output on external pin PPG0/PPG1.

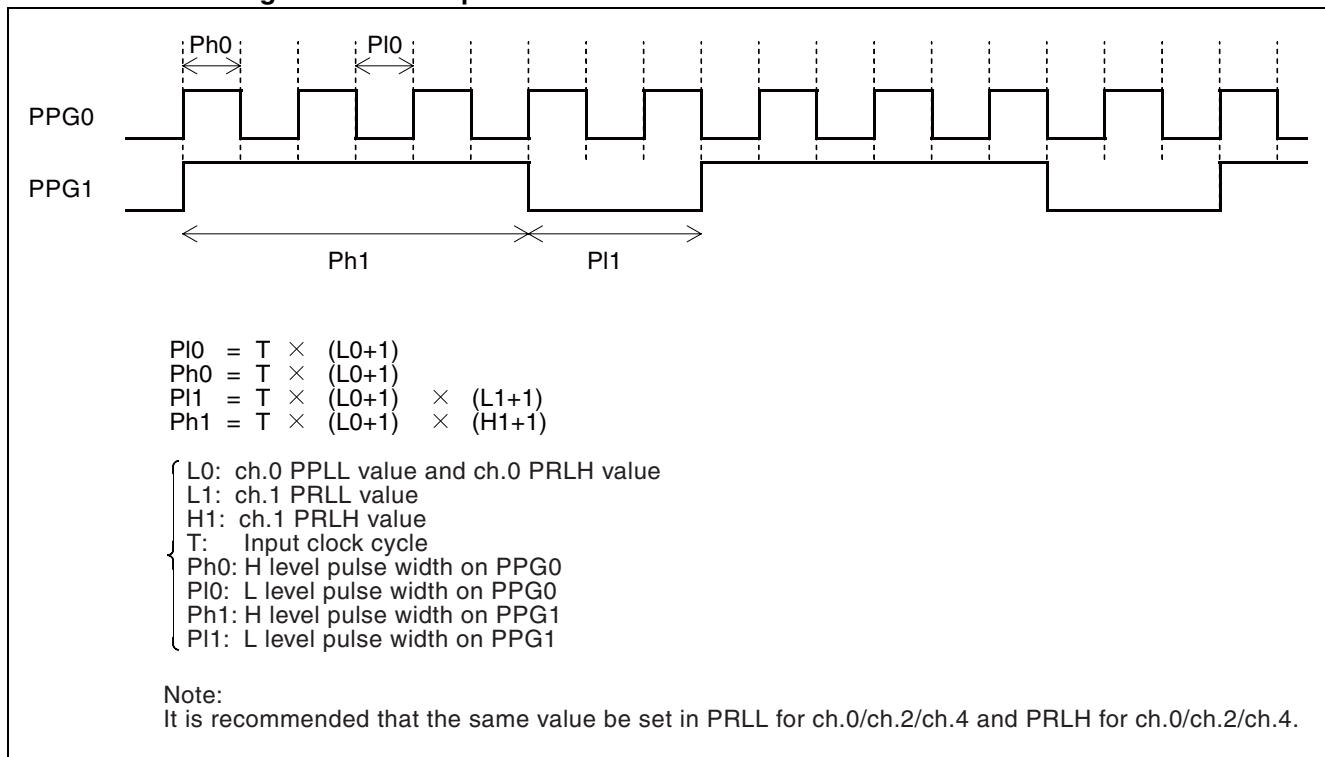
Output on the external pin PPG0 is enabled using the POE0 bit of the PPGC0 register, and output on the external pin PPG1 is enabled using the POE1 bit of the PPGC1 register. When these bits are 0 (the initial value), the pulse output does not appear on the external pins; these pins function as a general-purpose port. When these bits are set to "1", the pulse output appears on the external pins.

■ 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 12.4-2 gives an example of output waveforms in this mode.

Figure 12.4-2 Output Waveforms in 8-bit Prescaler + 8-bit PPG Mode

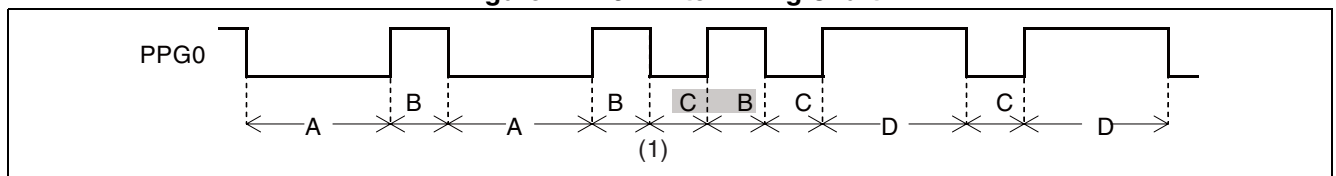


12.4.5 Write Timing for the Reload Registers

In all modes except the 16-bit PPG 1-ch 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 12.4-3 Write Timing Chart

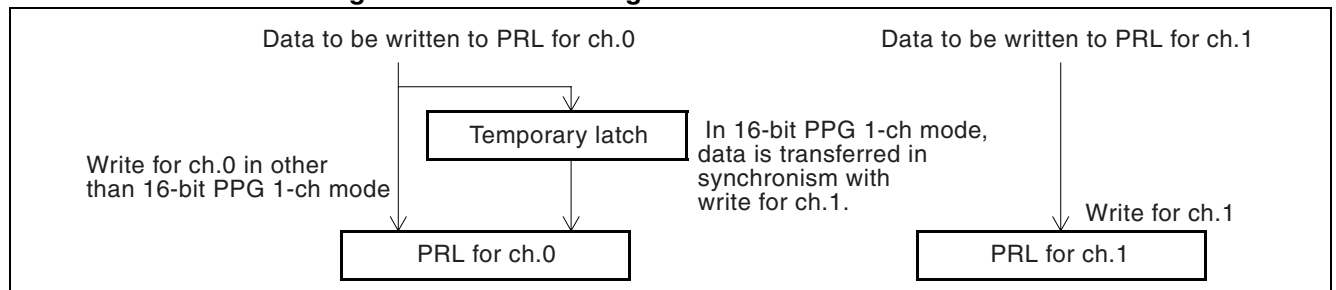


In the above timing chart, suppose that the PRL content is rewritten from A to C before (1), and that after (1), the PRLH content is rewritten from B to D. In such a case, a low for count C and a high for count B are output only once because at point (1), the PRL values include C in PRL and B in PRLH.

Similarly, in the 16-bit PPG mode, perform long-word transfer to write data in PRL for ch.0 and ch.1, or perform word transfer to write data in PRL in order from ch.0 to ch.1. In this mode, a write to PRL for ch.0 is performed temporarily. After a write to PRL for ch.1 has been performed, a write to PRL for ch.0 is performed.

In modes other than the 16-bit PPG mode, a write to PRL for ch.0 and ch.1 can be performed separately.

Figure 12.4-4 Block Diagram for the PRL Write Portion



CHAPTER 13 DTP/EXTERNAL INTERRUPT

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

- 13.1 Overview of the DTP/External Interrupt Circuit
- 13.2 Registers in the DTP/External Interrupt Circuit
- 13.3 Operation of DTP/External Interrupt Circuit
- 13.4 Notes on Using the DTP/External Interrupt Circuit

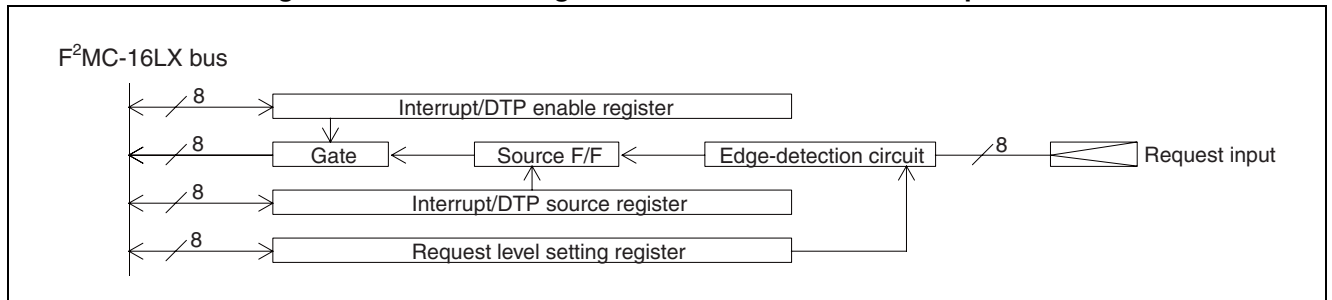
13.1 Overview of the DTP/External Interrupt Circuit

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 13.1-1 Registers in the DTP/External Interrupt Circuit

Interrupt/DTP source register		bit 15	14	13	12	11	10	9	8	
Address:000039 _H		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)	
Interrupt/DTP enable register		bit 7	6	5	4	3	2	1	0	
Address:000038 _H		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)	
High-order byte of the request level setting register		bit 15	14	13	12	11	10	9	8	
Address:00003B _H		LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	ELVR
Read/write ⇒		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Low-order byte of the request level setting register		bit 7	6	5	4	3	2	1	0	
Address:00003A _H		LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	ELVR
Read/write ⇒		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

■ Block Diagram of the DTP/External Interrupt Circuit**Figure 13.1-2 Block Diagram of the DTP/External Interrupt Circuit**

13.2 Registers in the DTP/External Interrupt Circuit

The ENIR register determines whether to use device pins as external interrupt/DTP request inputs to initiate the function of issuing a request to the interrupt controller.

■ Interrupt/DTP Enable Register (ENIR)

Figure 13.2-1 Interrupt/DTP Enable Register (ENIR)

Interrupt/DTP enable register								
	bit 7	6	5	4	3	2	1	0
Address:000038 _H	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
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)

When a bit in the interrupt/DTP enable register (ENIR) is set to "1", the corresponding pin is used as an external interrupt/DTP 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 external interrupt/DTP request input source is held, but no request is issued to the interrupt controller.

ENx corresponds to IRQx.

Note:

Please clear corresponding DTP/external interruption factor bit (EIRR:ER) immediately before DTP/external interruption is permitted (ENIR:EN=1).

■ Interrupt/DTP Source Register (EIRR)

Figure 13.2-2 Interrupt/DTP Source Register (EIRR)

Interrupt/DTP source register								
	bit 15	14	13	12	11	10	9	8
Address:000039 _H	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0
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)

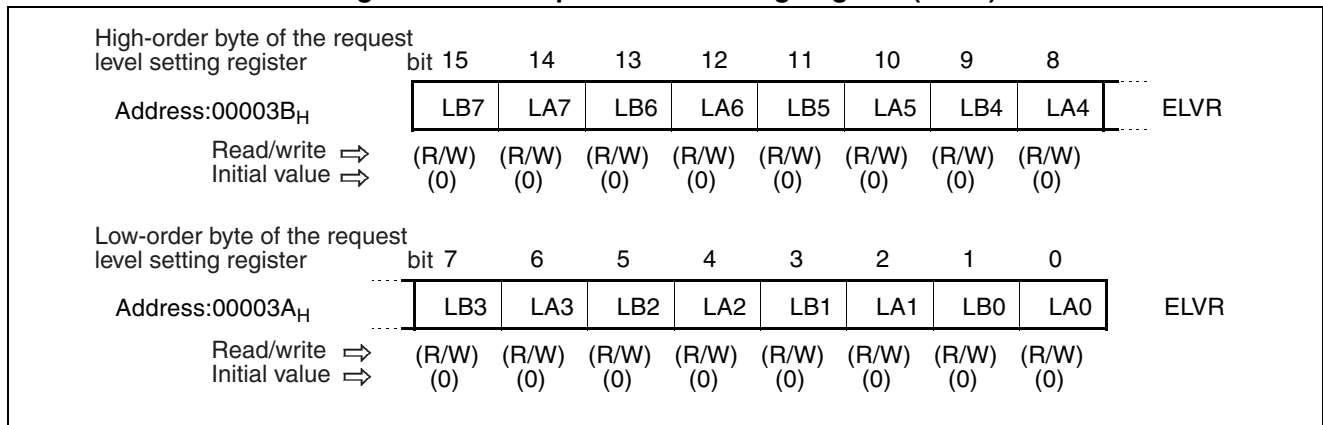
When the EIRR register is read from, it indicates that a corresponding external interrupt/DTP request is present. When the register is written to, the flip-flop content indicating the request is cleared. When "1" is read from a bit in this register, it indicates that an external interrupt/DTP request 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.

ERx corresponds to IRQx.

Notes:

- When more than one external interrupt request output is enabled (ENIR:EN7 to EN0 set to "1"), clear only the bit corresponding to the interrupt accepted by the CPU (the bit set to "1" among EN7 to EN0). Avoid clearing other bits unconditionally.
- When corresponding DTP external interrupt enable bit (ENIR: EN) is set to "1", the value of the DTP/external interrupt source bit (EIRR: ER) is valid. When the DTP/external interrupt has disabled (ENIR: EN=0), the DTP/external interrupt source bit might be set regardless of the existence of the DTP/external interrupt source.
- Please clear corresponding DTP/external interruption factor bit (EIRR:ER) immediately before DTP/external interruption is permitted (ENIR:EN=1).

■ Request Level Setting Register (ELVR)**Figure 13.2-3 Request Level Setting Register (ELVR)**

The ELVR register is used to select request detection. Two bits are assigned to each pin. The combinations of bit values and their meanings are listed below. When a request input is detected by level, the corresponding bits are set again even after cleared if the input is active.

LAx and LBx correspond to IRQx.

Table 13.2-1 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

Reference:

When a selected detection signal is applied to a DTP/external interrupt pin, an external interrupt request flag bit (ER7 to ER0) is set to "1" regardless of the setting in the DTP/external interrupt enable register (ENIR).

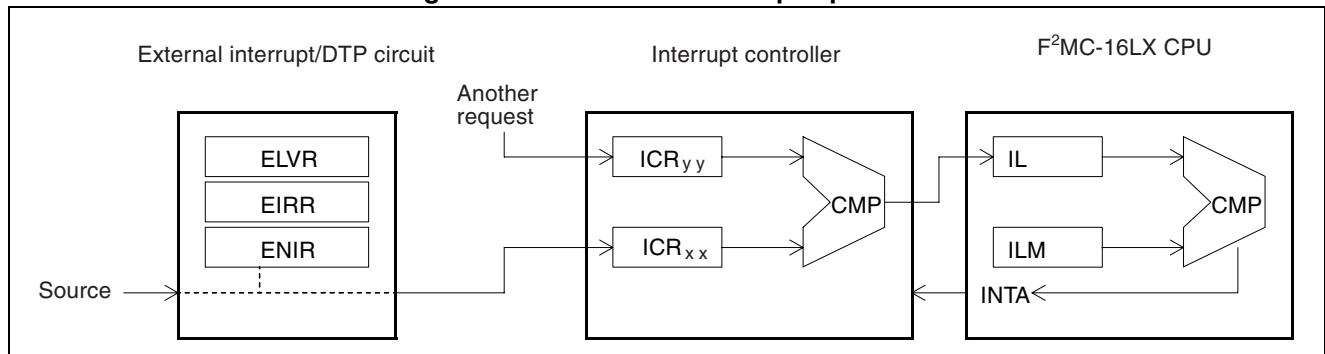
13.3 Operation of DTP/External Interrupt Circuit

When a request set in the ELVR register is applied to a corresponding pin after an external interrupt request setting, this resource issues an interrupt request signal to the interrupt controller. When the interrupt controller determines the priority of interrupts generated at the same time, it regards the interrupt from this resource as having the highest priority, and the interrupt controller issues an interrupt request to the F²MC-16LX CPU.

■ External Interrupt Operation

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 13.3-1 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

Before activating DTP, the intelligent I/O service must first be activated. This preprocess must include setting the address of a register allocated to a location 000000_H to 0000FF_H in the I/O address pointer in the extended intelligent I/O service descriptor and setting 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.

Figure 13.3-2 Timing for Canceling an External Interrupt Request when DTP Operation Terminates

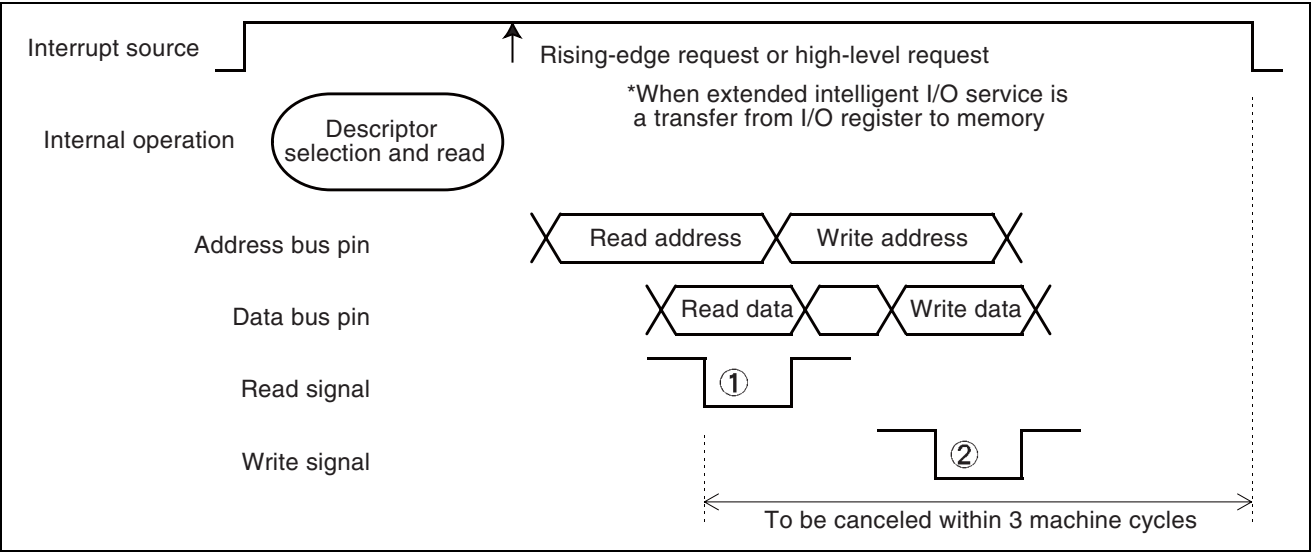
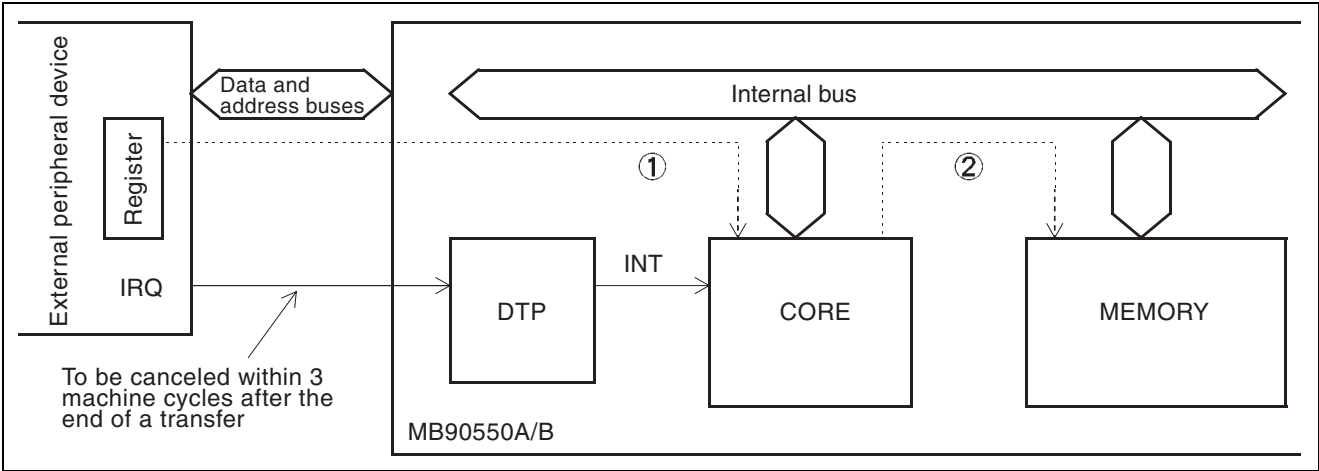


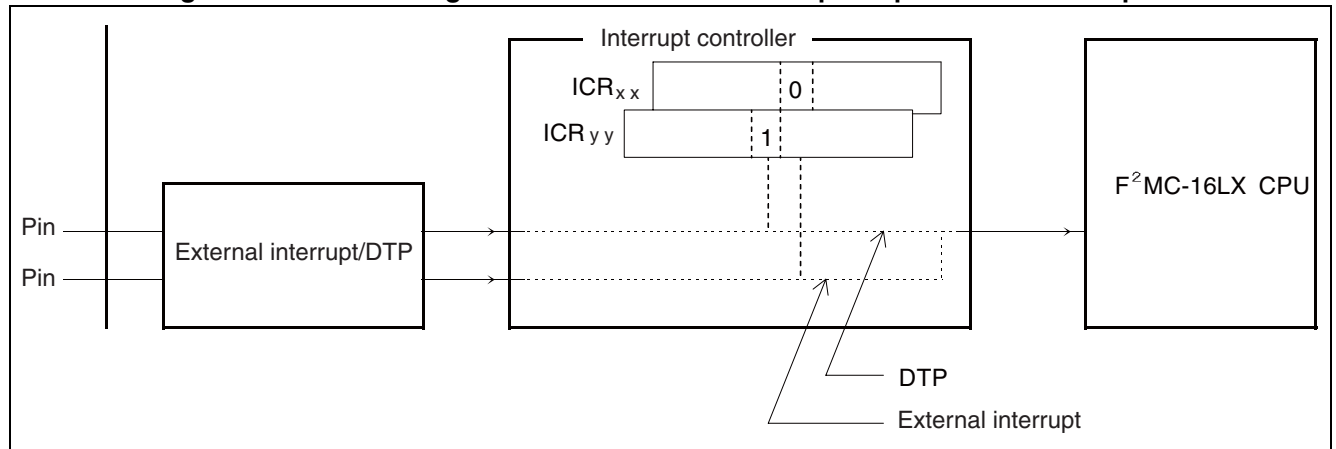
Figure 13.3-3 Schematic of a Sample Interface with an External Peripheral Device



■ **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 13.3-4 Switching between an External Interrupt Request and DTP Request

13.4 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
 - External interrupt/DTP 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.

■ External Interrupt/DTP Circuit Operation Procedure

When setting registers in the external interrupt/DTP circuit, proceed as follows.

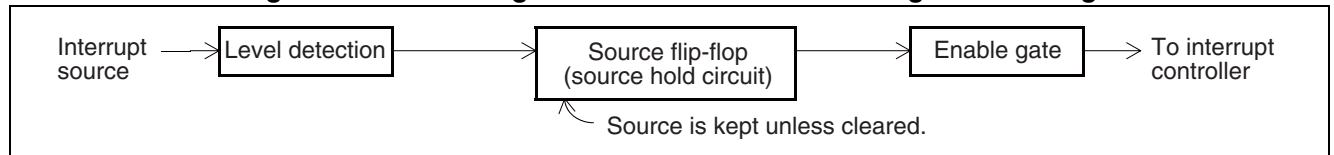
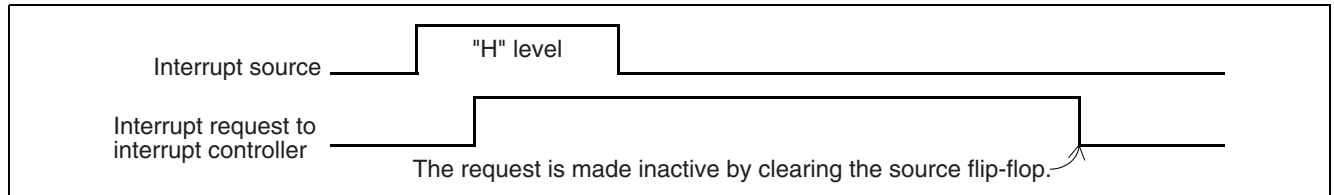
1. Disable a target bit in the enable register.
2. Set target bits in the request level setting register.
3. Clear a target bit in the source register.
4. Enable the target bit in the enable register. (Note that in steps 3 and 4, a word may be written to the register at a time.)

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 remains active because the source hold circuit exists internally.

To cancel the request to the interrupt controller, clear the external interrupt request flag bit to clear the source hold circuit.

Figure 13.4-1 Clearing the Source Hold Circuit During Level Setting**Figure 13.4-2 Interrupt Source and Interrupt Request to the Interrupt Controller when an Interrupt is Enabled**

CHAPTER 14 DELAYED INTERRUPT GENERATING MODULE

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

14.1 Outline of the Delayed Interrupt Generating Module

14.2 Operation of the Delayed Interrupt Generating Module

14.1 Outline 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.

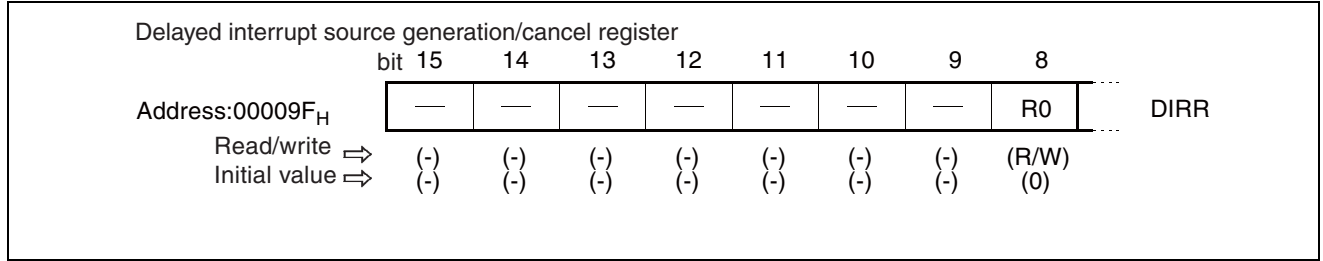
Register in the Delayed Interrupt Generating Module

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.

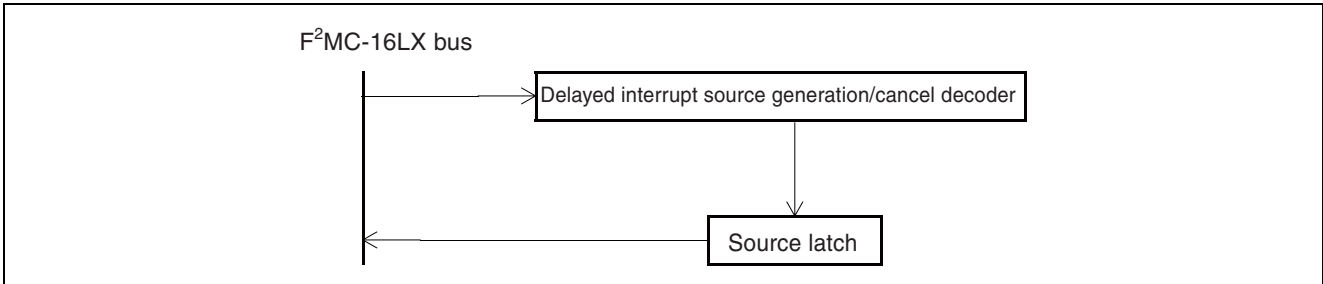
Either "0" or "1" may be written to the reserved bit area. For future expansion, it is recommended that the set bit and clear bit instructions be used to access this register.

Figure 14.1-1 Delayed Interrupt Source Generation/Cancel Register (DIRR)



Block Diagram of the Delayed Interrupt Generating Module

Figure 14.1-2 Block Diagram of the Delayed Interrupt Generating Module



14.2 Operation of the Delayed Interrupt Generating Module

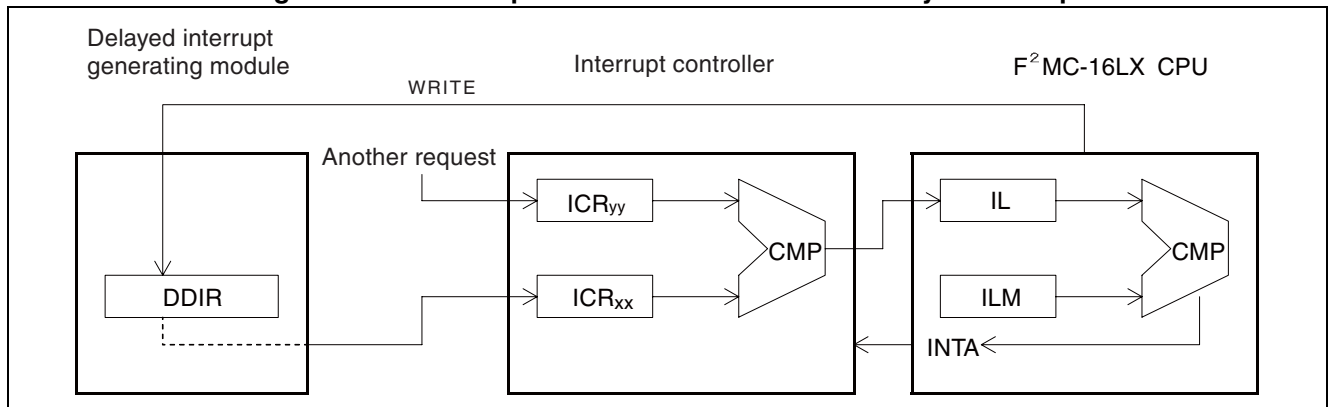
When software causes the CPU to write "1" to a bit of the DIRR register, the request latch in the delayed interrupt generating module is set, issuing an interrupt request to the interrupt controller.

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 14.2-1 Description of the Generation of a Delayed Interrupt



When "0" is written to an appropriate bit of the DDIR 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 15 A/D CONVERTER

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

- 15.1 Overview of the A/D Converter
- 15.2 Registers of the A/D Converter
- 15.3 Operation of A/D Converter
- 15.4 Conversion Data Protection Function

15.1 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 26.3 μ s per channel

- **Sampling time**

Either 64 machine cycles or 4096 machine cycles (minimum 4 μ s or 256 μ s) per channel can be selected.

Note that these conversion and sampling times are applied when the machine clock is set at 16 MHz.

- **Compare time**

Either 176 or 352 machine cycles per channel.

Note that 176 machine cycles are used at machine clock 8 MHz or smaller.

- **Adoption of RC type successive approximation conversion method with a sample and hold circuit.**

- **Resolution of 8 or 10 bits**

- **Analog input can be selected from eight channels by a program.**

- **Single conversion mode**

One channel can be selected and converted.

- **Scan conversion mode**

Successive multiple channels can be converted. Up to eight channels can be programmed.

- **Successive conversion mode**

Specified channels are repeatedly converted.

- **Pause conversion mode**

After conversion of a channel, pauses and waits for the next activation (enables synchronous conversion of conversion)

- Upon completion of A/D conversion, an interrupt request of A/D conversion completion for the CPU can be generated. The EI²OS can be started by the generation of this interrupt, and A/D conversion result data is transferred to memory. Therefore, the A/D converter is suitable for successive processing.
- A start up cause is selected from software, external trigger (falling edge), and timer (rising edge).

■ Cautions on Using the A/D Converter

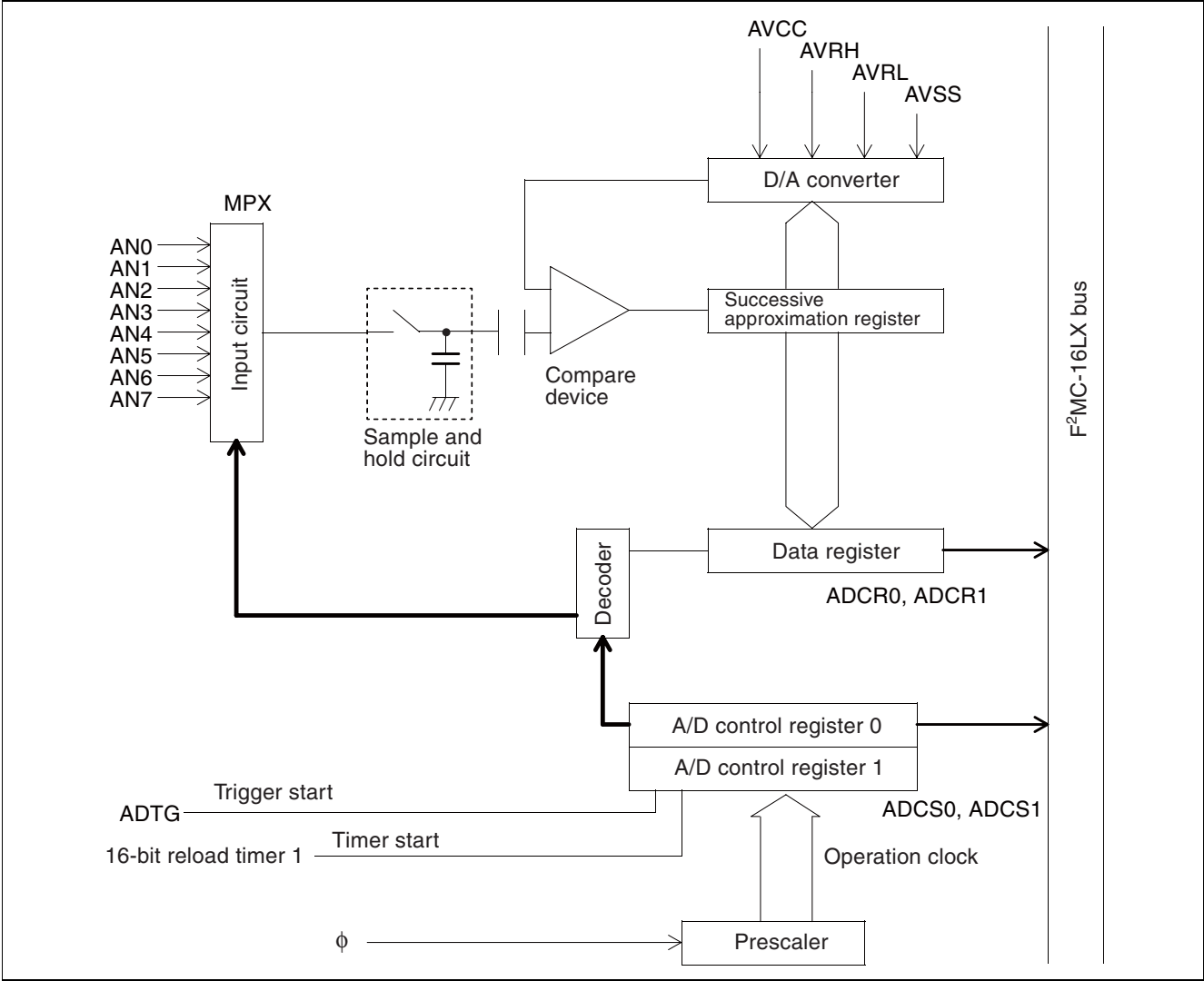
When starting the A/D converter using an external trigger or internal timer, the A/D startup cause is set at bits STS1 and STS0 in the ADCS1 register. At this time, confirm that the value entered by the external trigger or internal timer is inactive. If it is active, the converter may start operating. When setting bits STS1 and STS0, confirm that ADTG = 1 (input) and internal timer (16-bit reload timer) = 0 (output).

The bit of ADER register that corresponds to the pin used for analog input must be set to "1".

For details, see Section "7.3.5 Analog Input Enable Register (ADER)" in "CHAPTER 7 I/O PORTS".

■ Block Diagram of A/D Converter

Figure 15.1-1 Block Diagram of the A/D Converter



15.2 Registers of the A/D Converter

Figure 15.2-1 shows the registers of the A/D converter.

■ Registers of the A/D Converter

Figure 15.2-1 Registers of the A/D Converter

Higher byte of the control status register	bit 15	14	13	12	11	10	9	8	
Address:00003D _H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS1
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	bit 7	6	5	4	3	2	1	0	
Address:00003C _H	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS0
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	
Higher byte of the data register	bit 15	14	13	12	11	10	9	8	
Address:00003F _H	S10	ST1	ST0	CT1	CT0	—	D9	D8	ADCR1
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(-)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(1)	(-)	(X)	(X)	
Lower byte of the data register	bit 7	6	5	4	3	2	1	0	
Address:00003E _H	D7	D6	D5	D4	D3	D2	D1	D0	ADCR0
Read/write ⇒	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

15.2.1 Control Status Registers (ADCS0 and ADCS1)

The control status registers (ADCS0 and ADCS1) control the A/D converter and represent its status.

■ Control Status Registers (ADCS0 and ADCS1)

Do not rewrite data to ADCS0 during A/D conversion.

Figure 15.2-2 Control Status Registers (ADCS0 and ADCS1)

Higher byte of the control status register	bit	15	14	13	12	11	10	9	8	
Address:00003D _H		BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS1
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	bit	7	6	5	4	3	2	1	0	
Address:00003C _H		MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS0
Read/write ⇒		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Note:

Please do not rewrite ADCS1 while the A/D conversion is operating.

[bit15] BUSY (Busy flag and stop)

The BUSY bit is used to represent the operating status of the A/D converter.

○ **At read:**

This bit is set at activation of the A/D converter and cleared at termination. That is, the A/D converter pauses when this bit is 0 and operates when the bit is "1".

○ **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.

You cannot write 1 to the busy bit. In the read modify write (RMW) system instruction, "1" is read. In single mode, the bit is cleared upon A/D conversion completion. In successive and stop modes, the bit is not cleared until the converter is terminated by writing "0".

Note:

Do not perform a forced termination and activation at the same time with software. (BUSY = 0, STRT = 1)

[bit14] 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 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.

"1" is read in read modify write (RMW) system instruction.

Note:

When clearing this bit by writing "0", confirm that the A/D converter is not under operation.

[bit13] 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.

Table 15.2-1 Functions of INTE (Bit to Allow or not Allow an Interrupt)

INTE	Function
0	Interrupt prohibited [Initial value]
1	Interrupt allowed

[bit12] 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 setting is cleared by writing "0" or reset.

Note:

This bit is valid only when the EI²OS is used. See the conversion data protection function in the operation description.

[bit11, bit10] STS1 and STS0 (Start source select)

An A/D activation cause is selected by setting the STS1 and STS0 bits.

Table 15.2-2 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

In mode in which multiple activation causes are selected, the A/D converter is activated by each 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 is selected.

[bit9] 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.

Note:

Do not perform forcible stop and activation at the same time with software. (BUSY=0, STRT = 1)

[bit8] Reserved bit

Bit8 is a reserved bit. Whenever setting ADCS1, be sure to set this bit to "0".

[bit7, bit6] MD1 and MD0 (A/D converter mode set)

The MD1 and MD0 bits set the operation mode.

Table 15.2-3 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 during pause is resumed by the generation of an activation cause.

Notes:

- 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.

[bit5 to bit3] 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 15.2-4 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:

During A/D conversion, these bits can read conversion channel numbers. While A/D conversion is stopped, however, channel numbers previously converted by the A/D are read. The value read by this bit is the conversion channel number until A/D conversion is started. This value is initialized to "000" at reset.

[bit2 to bit0] ANE2, ANE1, and ANE0 (Analog end channel set)

The end channel of A/D conversion is set by ANE2, ANE1, and ANE0 bits.

Table 15.2-5 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

Notes:

- Setting the same channel as ANS2 to ANS0 selects one-channel conversion. (Single conversion)
- When successive mode or pause mode is set, the conversion returns to the start channel set by ANS2 to ANS0 upon completed conversion of channels set by these bits.
- When the channels set are ANS > ANE, conversion starts at ANS. When channels up to channel 7 are converted, the A/D returns to channel 0 and converts until ANE.
Example: Channel setting ANS = 6ch and ANE = 3ch in single mode
Operation Conversion channel 6ch --> 7ch --> 0ch --> 1ch --> 2ch --> 3ch

15.2.2 Data Register (ADCR1 and ADCR0)

In the data register (ADCR1 and ADCR0), resolution is selected and machine cycle is set.

■ Data Registers (ADCR1 and ADCR0)

Figure 15.2-3 Data Registers (ADCR1 and ADCR0)

Higher byte of the data register	bit 15	14	13	12	11	10	9	8	
Address:00003F _H	S10	ST1	ST0	CT1	CT0	—	D9	D8	ADCR1
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(-)	(R)	(R)	
Initial value ⇒	(0)	(0)	(0)	(0)	(1)	(-)	(X)	(X)	
Lower byte of the data register	bit 7	6	5	4	3	2	1	0	
Address:00003E _H	D7	D6	D5	D4	D3	D2	D1	D0	ADCR0
Read/write ⇒	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Note:

Read values of bit9 and bit8 of ADCR0 and ADCR1 are undefined, and the contents of bit11 to bit15 of ADCR1 are always "0".

[bit15] S10

S10 is the A/D resolution select bit. When rewriting the S10 bit, confirm that the A/D converter has not started conversion. The contents of ADCR are undefined if rewritten after conversion.

Table 15.2-6 Functions of S10

S10	Function
0	10-bit mode [Initial value]
1	8-bit mode

[bit14 ,bit13] ST1 and ST0 (Sampling time)

The ST1 and ST0 bits set the machine cycle number at sampling.

Table 15.2-7 ST1 and ST0 (Machine Cycle Setting Bits at Sampling)

ST1	ST0	Machine cycle at sampling	Sampling Time
0	0	64 machine cycle	4μs/machine clock 16 MHz
0	1	Reserved	
1	0	Reserved	
1	1	4096 machine cycle	256μs/machine clock 16 MHz

[bit12 ,bit11] CT1 and CT0 (Compare time)

The CT1 and CT0 bits set the machine cycle number at compare.

Table 15.2-8 CT1 and CT0 (Machine Cycle Number Setting Bits at Compare)

CT1	CT0	Machine cycle at compare	Compare time
0	0	176 machine cycle	22μs/machine clock 8 MHz
0	1	352 machine cycle	22μs/machine clock 16 MHz
1	0	Reserved	
1	1	Reserved	

Notes:

- To set these bits at "00", confirm that the machine clock is 8 MHz or slower.
- If the machine clock is faster than 8 MHz, conversion accuracy cannot be guaranteed.

[bit9 to bit0] 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 "15.3 Operation of A/D Converter".

These registers are undefined at reset.

Notes:

- Do not write data to this register during A/D operation.
- D9 and D8 are not used in 8-bit mode. Read values of D9 and D8 are undefined.

15.3 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-bit), the conversion data registers (ADCR1 and 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, ANE = 011

Start --> AN0 --> AN1 --> AN2 --> AN3 --> End

ANS = 010, ANE = 010

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), only the channel specified by ANS continues to be converted.

[Example]

ANS = 000, ANE = 011

Start --> AN0 --> AN1 --> AN2 --> AN3 --> AN0 --> Repeat

ANS = 010, ANE = 010

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 each 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 channels specified by ANS are converted.

[Example]

ANS = 000, ANE = 011

Start --> AN0 --> Stop --> Activate --> AN1 --> Stop --> Activate --> AN2 --> Stop --> Activate --> AN3 --> Stop --> Activate --> AN0 --> Repeat

ANS = 010, ANE = 010

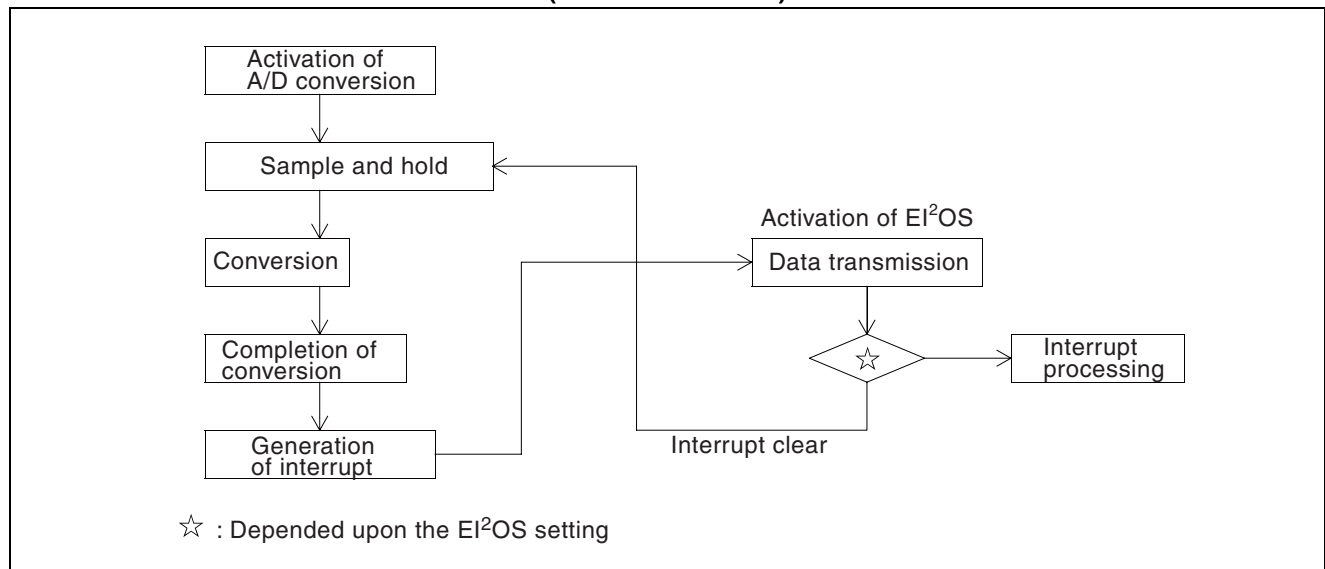
Start --> AN2 --> Stop --> Activate --> AN2 --> Stop --> Activate --> AN2 --> Repeat

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 15.3-1 Example of Flow from Activation of A/D Conversion to Conversion Data Transfer (Successive Mode)



15.3.1 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 200_H to 206_H sequentially
- Activate with software
- Maximum interrupt level

■ Example of EI²OS Activation in Single Mode

Table 15.3-1 Example of EI²OS Activation in Single Mode

Items set	Example of programming	Description of operation
Setting of EI ² OS	MOV ICR0, #08H	Setting of maximum interrupt, activation of EI ² OS at interrupt, and setting of descriptor address.
	MOV BAPL, #00H	Destination address of conversion data.
	MOV BAPM, #02H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Transfers word data and increments destination address after transmission. Transfer from I/O to memory.
	MOV IOAL, #3EH	Setting of result register of A/D converter.
	MOV IOAH, #00H	
	MOV DCTL, #03H	Transfers data three times with EI ² OS. The number of data transfers is the same as the number of conversions.
	MOV DCTH, #00H	
Setting of A/D converter	MOV ADCS0, #0BH	Single mode, start channel AN1, end channel AN3.
	MOV ADCS1, #A2H	Activation with software, start of A/D conversion
Other processing	-	-
EI ² OS termination interrupt sequence	MOV ADCS1, #80H	-
	RETI	Recover from interrupt.

ICR3: 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

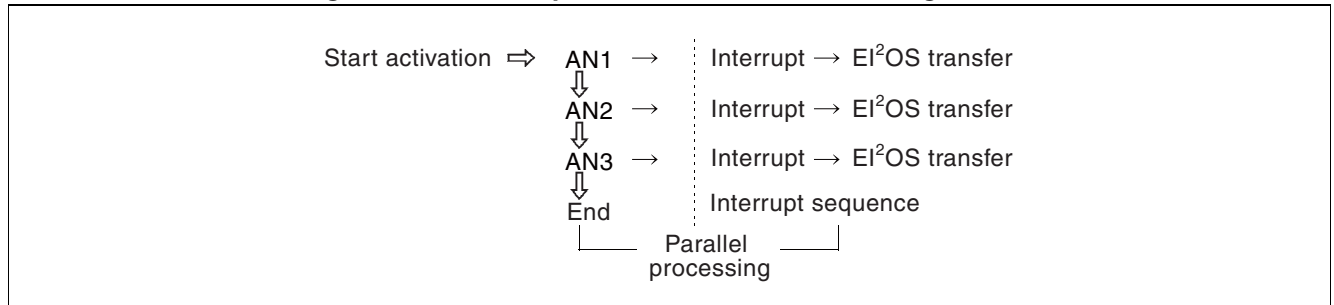
IOAL: Lower byte of the I/O address register

IOAH: Higher byte of the I/O address register

DCTL: Lower byte of the data counter

DCTH: Higher byte of the date counter

Figure 15.3-2 Example of EI²OS Activation in Single Mode



15.3.2 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 60CH sequentially
- Activate with external edge input
- Maximum interrupt level

■ Example of EI²OS Activation in Successive Mode

Table 15.3-2 Example of EI²OS Activation in Successive Mode

Items set	Example of programming	Description of operation
Setting of EI ² OS	MOV ICR0, #08H	Setting of maximum interrupt, activation of EI ² OS at interrupt, and setting of descriptor address.
	MOV BAPL, #00H	Destination address of conversion data.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Transfers word data and increments destination address after transmission. Transfer from I/O to memory. Terminates by a request from a resource.
	MOV IOAL, #3EH	Destination address
	MOV IOAH, #00H	
	MOV DCTL, #06H	Transfers data six times with EI ² OS. Transfers data of 3 channels × 2.
	MOV DCTH, #00H	
Setting of A/D converter	MOV ADCS0, #9DH	Single mode, start channel AN3, end channel AN5.
	MOV ADCS1, #A4H	Activation with external edge, start of A/D converter.
Other processing	-	-
EI ² OS termination interrupt sequence	MOV ADCS1, #80H	Recover from interrupt.
	RETI	

ICR3: 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

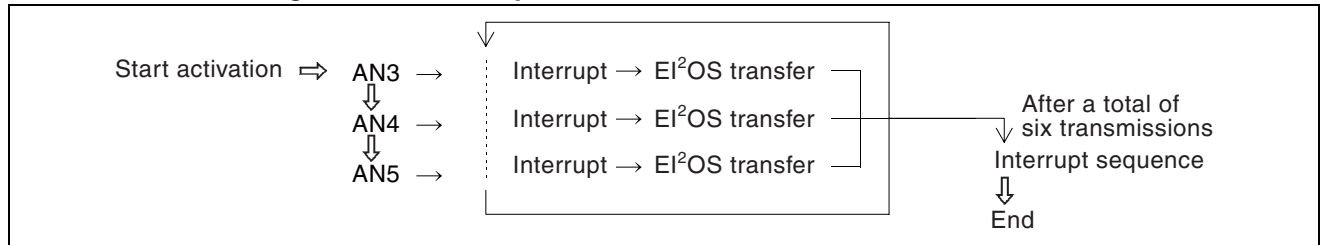
IOAL: Lower byte of the I/O address register

IOAH: Higher byte of the I/O address register

DCTL: Lower byte of the data counter

DCTH: Higher byte of the data counter

Figure 15.3-3 Example of EI²OS Activation in Successive Mode



15.3.3 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 600_H to 618_H sequentially
- Activate with external edge input
- Maximum interrupt level

■ Example of EI²OS Activation in Pause Mode

Table 15.3-3 Example of EI²OS Activation in Pause Mode

Items set	Example of programming	Description of operation
Setting of EI ² OS	MOV ICR0, #08H	Setting of maximum interrupt, activation of EI ² OS at interrupt, and setting of descriptor address.
	MOV BAPL, #00H	Destination address of conversion data.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #19H	Transfers word data and increments destination address after transmission. Transfer from I/O to memory. Terminates by a request from a resource.
	MOV IOAL, #3EH	Destination address
	MOV IOAH, #00H	
	MOV DCTL, #0CH	Transfers data 12 times with EI ² OS.
	MOV DCTH, #00H	
Setting of A/D converter	MOV ADCS0, #DBH	Single mode, start channel AN3, end channel AN3 (one-channel conversion).
	MOV ADCS1, #A4H	Activation with external edge, start of A/D converter.
Other processing	-	-
EI ² OS termination interrupt sequence	MOV ADCS1, #80H	Recover from interrupt.
	RETI	

ICR3: 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

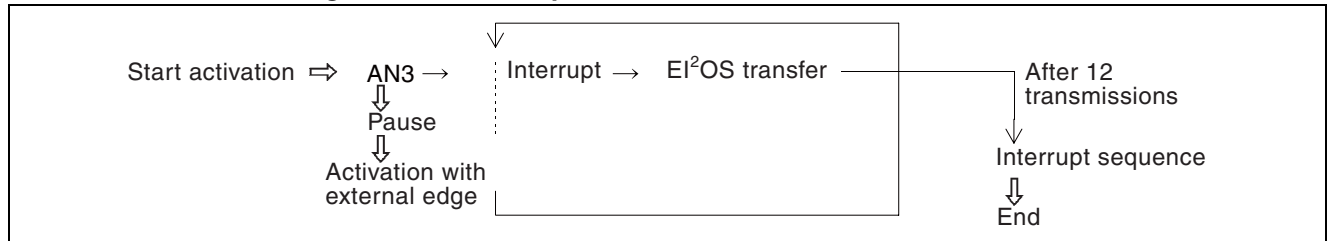
IOAL: Lower byte of the I/O address register

IOAH: Higher byte of the I/O address register

DCTL: Lower byte of the data counter

DCTH: Higher byte of the data counter

Figure 15.3-4 Example of EI²OS Activation in Pause Mode



15.4 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

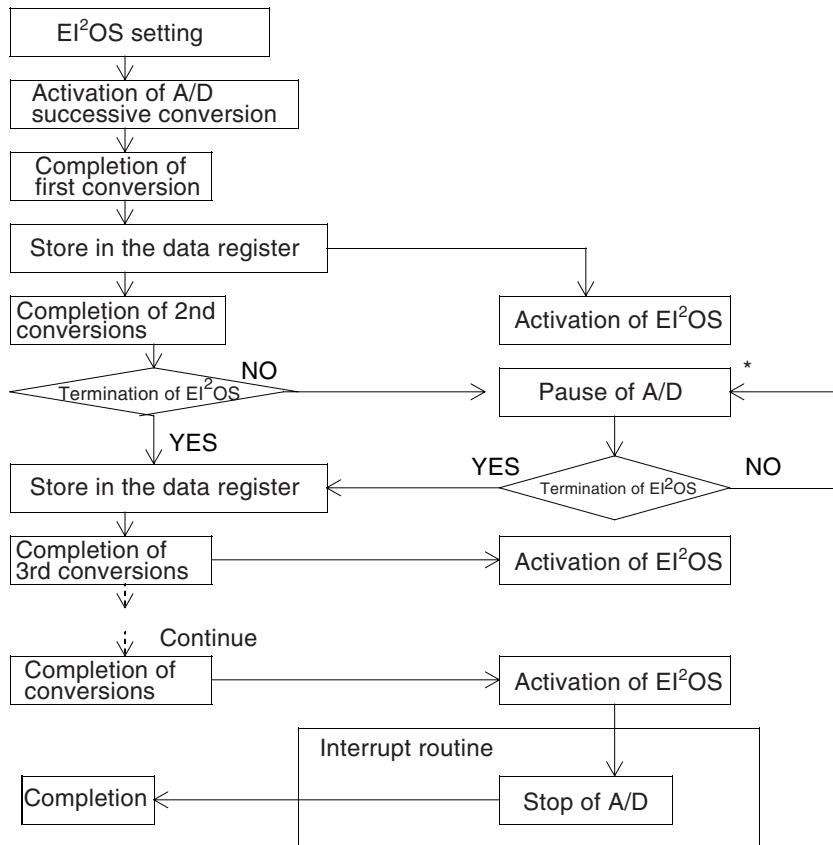
Because there is only one conversion data register, when successive A/D conversion is performed, the conversion data is stored upon completion of each conversion and the previous data is lost. To protect the previous data, this A/D converter pauses without storing new conversion data unless the previous 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.

Notes:

- This function is applied to the INT and INTE bits in the ADCS1 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 prohibited when the A/D is suspended during EI²OS operation, the A/D conversion starts and the new data may be written before the previous data is transferred. Also, the standby data is destroyed if the A/D is restarted during a suspension (pause).
-

Figure 15.4-1 Data Protection Function Flow (when EI²OS is Used)

* : If the converter is reactivated during pause, the conversion data in standby is destroyed.

(Note)

The flow of the A/D converter in pause is omitted.

CHAPTER 16 COMMUNICATION PRESCALER REGISTER

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

The output of the communication prescaler is used by the UART and I/O extended serial interface.

16.1 Overview of Communication Prescaler Register

16.2 Operation of Communication Prescaler Register

16.1 Overview of Communication Prescaler Register

The communication prescaler register controls the machine clock dividing ratio and is designed to assure a constant baud rate for various machine clocks.

The output of the communication prescaler is used by the UART and I/O extended serial interface.

■ Communication Prescaler Register (CDCR)

Figure 16.1-1 Communication Prescaler Register (CDCR)

Communication prescaler register	bit 15	14	13	12	11	10	9	8	
Address:000027 _H	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	CDCR
Read/write ⇒	(R/W)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(-)	(-)	(-)	(1)	(1)	(1)	(1)	

[bit15] MD (Machine clock divide mode select)

MD is an operation enable bit of the communication prescaler.

Table 16.1-1 Function of MD (Machine clock divide mode select) Bit

MD	Function
0	Communication prescaler stops. [Initial value]
1	Communication prescaler operates.

[bit11 to bit8] DIV3 to DIV0 (Divide 3 to 0)

DIV3 to DIV0 decide a machine clock dividing ratio.

Table 16.1-2 Function of DIV3 to DIV0 (Divide 3 to 0) 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

Notes:

- In actual use, set the above bits to something other than 1111.
 - When changing the dividing ratio, wait for two cycles as the clock stabilizing time before starting communication.
-

16.2 Operation of Communication Prescaler Register

Set the communication prescaler register as follows, depending on the machine clock ϕ used. For more information, see Section "17.4 UART Operations" and Section "18.3 Operation of I/O Extended Serial Interface".

■ Operation of Communication Prescaler Register

Table 16.2-1 Operation of Communication Prescaler Register

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 17 UART

This chapter describes the UART functions and operations.

17.1 Overview of UART

17.2 UART Block Diagram

17.3 UART Registers

17.4 UART Operations

17.5 Application of UART (During Operation in Mode 1)

17.1 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 (I/O extended serial interface) communication
- Support of multiprocessor
- Built-in dedicated baud rate generator (communication prescaler)

Table 17.1-1 Baud Rate

Operation	Baud rate *
Asynchronous	62500/31250/19230/9615/4808/2404/1202 bps
CLK-synchronous	2M/1M/500k/125k/62.5k bps

* : Values when internal machine clocks are 6, 8, 10, 12 and 16 MHz.

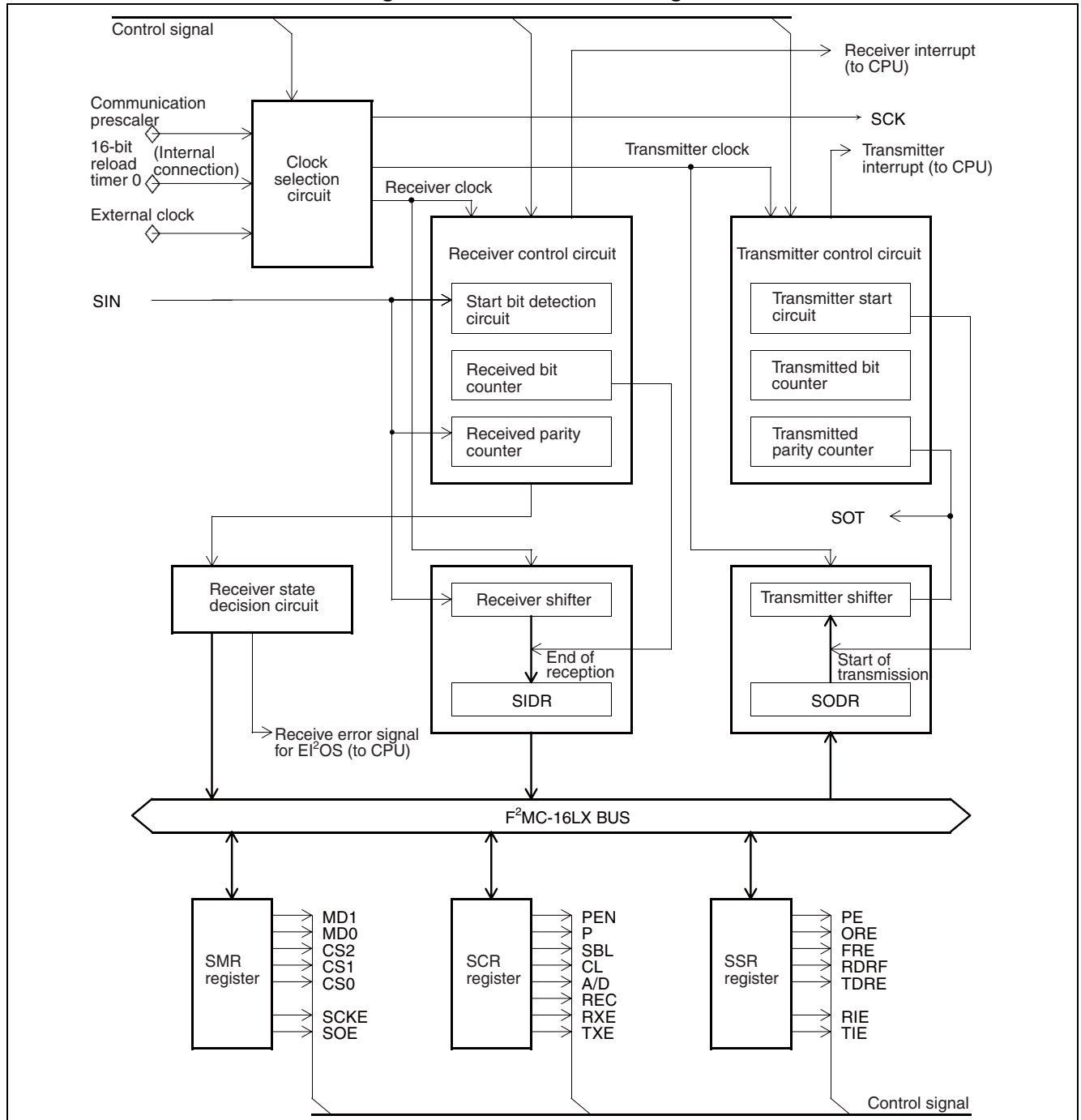
- Free baud rate setting by external clocks
- Error detection function (parity, framing, and overrun)
- Transfer signal of NRZ code
- Support of extended intelligent I/O services

17.2 UART Block Diagram

Figure 17.2-1 shows a UART block diagram.

■ UART Block Diagram

Figure 17.2-1 UART Block Diagram



17.3 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 17.3-1 UART Registers

Serial mode register								
	bit 7	6	5	4	3	2	1	0
Address:000020 _H	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)
Serial control register								
	bit 15	14	13	12	11	10	9	8
Address:000021 _H	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)
Serial input register/ serial output register								
	bit 7	6	5	4	3	2	1	0
Address:000022 _H	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)
Serial status register								
	bit 15	14	13	12	11	10	9	8
Address:000023 _H	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(R/W)	(R/W)
Initial value ⇒	(0)	(0)	(0)	(0)	(1)	(-)	(0)	(0)

17.3.1 Serial Mode Register (SMR)

The SMR 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 (SMR)

Figure 17.3-2 Configuration Of Serial Mode Register (SMR)

Serial mode register	bit 7	6	5	4	3	2	1	0	
Address:000020 _H	MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE	SMR
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)	

[bit7, bit6] MD1 and MD0 (Mode select)

MD1 and MD0 bits select a UART operating mode.

Table 17.3-1 MD1 and MD0 (Operating Mode Selecting Bits)

MD1	MD0	Mode	Operating mode
0	0	0	Asynchronous (start-stop synchronous) normal mode [Initial value]
0	1	1	Asynchronous (start-stop synchronous) multiprocessor mode
1	0	2	CLK-synchronous 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".

[bit5 to bit3] 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.

The bits are initialized to "000" when reset.

Table 17.3-2 CS0 to CS2 (Baud Rate Clock Source Selecting Bits)

CS2	CS1	CS0	Clock input
000 _B to 100 _B			Communication prescaler
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 MB90550A/B selects the output of 16-bit reload timer 0.

[bit2] Reserved

Bit2 is a reserved bit. When defining SMR settings, be sure to set this bit to "0".

[bit1] 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).

Set the bit to "0" in either CLK-synchronous mode or external clock mode.

Table 17.3-3 Function of SCKE (SCK Enable) Bit

SCKE	Function
0	Functions as a clock input terminal. [Initial value]
1	Functions as a clock output terminal.

Note:

An external clock source must be selected when using the bit as a clock input terminal.

[bit0] SOE (Serial output enable)

The external terminal (SOT) also serves as a general I/O port terminal and the SOE bit specifies whether to use it as a serial output terminal or an I/O port terminal.

Table 17.3-4 Function of SOE (Serial Output Enable) Bit

SOE	Function
0	Functions as a general I/O port terminal. [Initial value]
1	Functions as a serial data terminal (SOT).

17.3.2 Serial Control Register (SCR)

The serial control register (SCR) controls a transfer protocol for serial communication.

■ Serial Control Register (SCR)

Figure 17.3-3 Configuration of Serial Control Register (SCR)

Serial control register	bit15	14	13	12	11	10	9	8	
Address:000021 _H	PEN	P	SBL	CL	A/D	REC	RXE	TXE	SCR
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)	

[bit15] PEN (Parity enable)

The PEN bit specifies whether to add a parity bit when establishing serial data communication.

Table 17.3-5 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).

[bit14] P (Parity)

The P bit specifies even or odd parity when adding a parity bit for data communication.

Table 17.3-6 P (Event/Odd Parity Specification Bit)

P	Function
0	Even parity [Initial value]
1	Odd parity

[bit13] SBL (Stop bit length)

The SBL bit specifies the length of a stop bit, which is a frame end mark used in asynchronous (start-stop synchronous) communication.

Table 17.3-7 SBL (Stop Bit Length Specification Bit)

SBL	Function
0	1 stop bit [Initial value]
1	2 stop bits

[bit12] CL (Character length)

The CL bit specifies the data length of one frame to be transmitted or received.

Table 17.3-8 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 be processed only in the normal mode (mode 0) of asynchronous (start-stop synchronous) communication modes. In multiprocessor mode (mode 1) and CLK-synchronous communication mode (mode 2), set the bit to "1".

[bit11] A/D (Address/data)

The A/D bit specifies a data format of a frame to be transmitted or received in the multiprocessor mode (mode 1) of asynchronous (start-stop synchronous) communication modes.

Table 17.3-9 Function of A/D (Address/Data) Bit

A/D	Function
0	Data frame [Initial value]
1	Address frame

[bit10] REC (Receiver error clear)

Writing "0" to the REC bit clears the error flags (PE, ORE, and FRE) of the SSR register. Writing "1" is invalid. The read value is always "1".

[bit9] RXE (Receiver enable)

The RXE bit controls a UART receive operation.

Table 17.3-10 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.

[bit8] TXE (Transmitter enable)

The TXE bit controls a UART transmit operation.

Table 17.3-11 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.

17.3.3 Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)

The serial input data register (SIDR) and serial output data register (SODR) are receiver and transmitter data buffer registers.

■ Configuration of Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)

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 17.3-4 Configuration of Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)

Serial input register/ serial output register	bit 7	6	5	4	3	2	1	0	
Address:000022 _H	D7	D6	D5	D4	D3	D2	D1	D0	SIDR(read) SODR(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)	

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.

17.3.4 Serial Status Register (SSR)

The serial status register (SSR) is comprised of flags that represent the UART operating status.

■ Serial Status Register (SSR)

Figure 17.3-5 Configuration of Serial Status Register (SSR)

Serial status register	bit 15	14	13	12	11	10	9	8	
Address:000023 _H	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE	SSR
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(1)	(-)	(0)	(0)	

[bit15] 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 (bit10) of the SCR register.

If the bit is set, data in the SISR register becomes invalid.

Table 17.3-12 Function of PE (Parity Error) Bit

PE	Function
0	No parity error [Initial value]
1	Parity error occurs.

[bit14] 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 (bit10) of the SCR register.

If the bit is set, data in the SISR register becomes invalid.

Table 17.3-13 Function of ORE (Over Run Error)

ORE	Function
0	No overrun error [Initial value]
1	Overrun error occurs.

[bit13] 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 (bit10) of the SCR register. If the bit is set, data in the SISR register becomes invalid.

Table 17.3-14 Function of FRE (Framing Error)

FRE	Function
0	No framing error [Initial value]
1	Framing error occurs.

[bit12] 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 17.3-15 Function of RDRF (Receiver Data Register Full)

RDRF	Function
0	Register is not full with receiver data. [Initial value]
1	Register is full with receiver data.

[bit11] 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 17.3-16 Function of TDRE (Transmitter Data Register Empty)

TDRE	Function
0	Transmitter data cannot be written.
1	Transmitter data can be written. [Initial value]

[bit9] RIE (Receiver interrupt enable)

The RIE bit controls a receiver interrupt.

Table 17.3-17 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.

[bit8] TIE (Transmitter interrupt enable)

The TIE bit controls a transmitter interrupt.

Table 17.3-18 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.

17.4 UART Operations

The UART has operating modes as shown in Table 17.4-1 that can be switched by setting a value into the SMR and SCR registers.

■ UART Operations

Table 17.4-1 UART Operating Modes

Mode	Parity	Data length	Operating mode	Stop bit length
0	Yes/No	7 bits	Asynchronous (start-stop synchronous) normal mode	1 or 2 bits
	Yes/No	8 bits		
1	No	8bits+1bit	Asynchronous (start-stop synchronous) multiprocessor mode	
2	No	8 bits	CLK-synchronous mode	-

Notes:

- 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.
- The CLK-synchronous mode uses a clock control (I/O extended serial interface) method. In this mode, no start-stop bit is added to data.
- 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)".

17.4.1 UART Clock Selection

The following three kinds of UART clocks can be selected:

- Communication prescaler
- Internal timer
- External clock

■ Communication Prescaler

When the communication prescaler is selected, the following baud rates are available:

Table 17.4-2 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

Table 17.4-3 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 (For more information, see "CHAPTER 16 COMMUNICATION PRESCALER REGISTER".)

■ 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) 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

N: Timer count clock source

n: Timer reload value

Table 17.4-4 shows the relationship between a baud rate and a reload value when the machine clock is taken as 7.3728 MHz.

Table 17.4-4 Baud Rates and Reload Values

Baud rate		Reload value	
Asynchronous (bps)	CLK-synchronous (bps)	N = 2 ¹ (Machine clock divided by 2)	N = 2 ³ (Machine clock divided by 8)
38400	614400	2	-
19200	307200	5	-
9600	153600	11	2
4800	76800	23	5
2400	38400	47	11
1200	19200	95	23
600	9600	191	47
300	4800	383	95

If the internal timer (16-bit reload timer 0) is selected as a baud rate clock source, the output TOT0 of the 16-bit timer 0 has already been connected inside the controller. Therefore, there is no need to make an external connection from the external terminal TOT0 of the 16-bit timer 0 to the external clock input terminal SCK of the UART. Also, the output terminal of the timer 0 can 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.

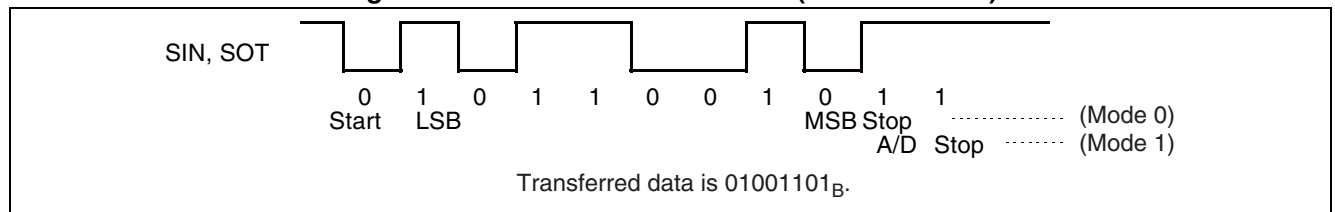
17.4.2 Asynchronous (Start-stop Synchronous) Mode

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).

■ Transfer Data Format

The UART handles only data of a NRZ (non return to zero) format. Figure 17.4-1 shows the transfer data format.

Figure 17.4-1 Transfer Data Format (Modes 0 and 1)



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

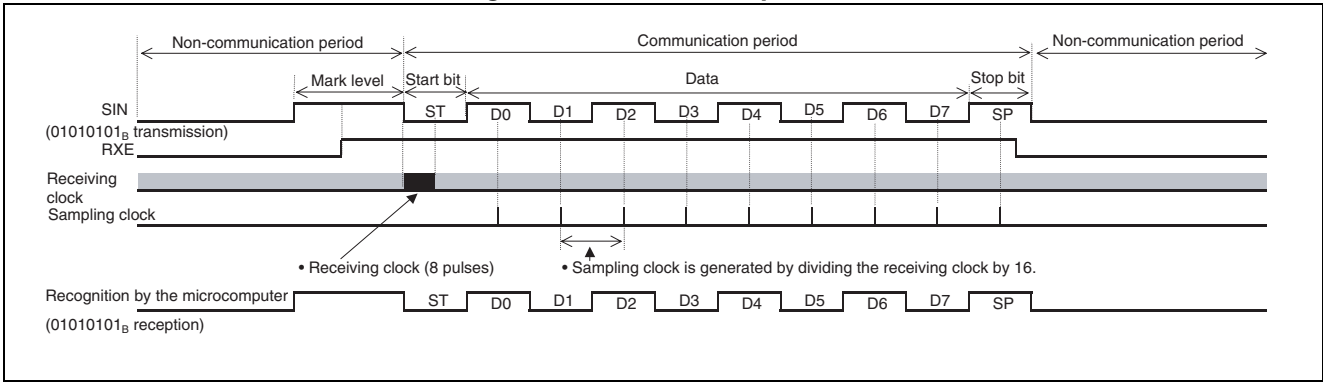
If the RXE bit of the SCR register is "1", a receiver operation is always performed. Once a start bit is detected, one-frame data is received according to the data format specified by that register. If an error occurs at the end of one-frame data reception, an error flag is set and the RDRF flag of the SSR register is then set. If the RIE bit 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.

○ Start bit detection method

Specify settings as follows for start bit detection:

- Immediately before the start of the communication period, be sure to set the communication line to "H" (mark level added).
- Set receive ready (RXE=H) while the communication line is at "H" (mark level).
- Do not set receive ready (RXE=H) during the non-communication period (mark level removed). Otherwise, data is not received correctly.
- When the stop bit is detected (the RDRF flag is set to "1"), set receive not ready (RXE=L) while the communication line is at "H" (mark level).

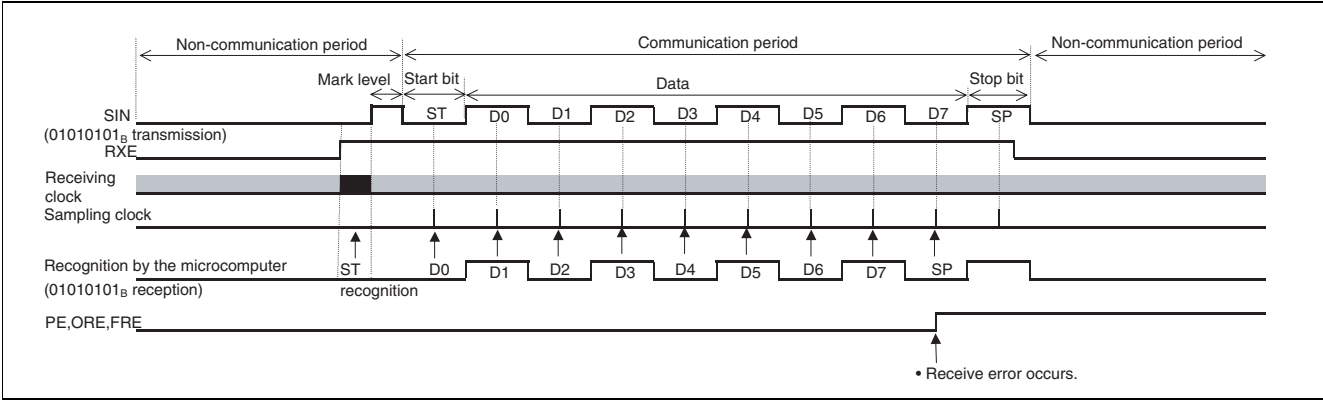
Figure 17.4-2 Normal Operation



Note that if receive ready is set at the timing represented in the following example, input data (SIN) is not recognized correctly by the microcomputer:

- Example of operation where receive ready (RXE=H) is set while the communication line is at L

Figure 17.4-3 Abnormal Operation



■ Transmitter Operation

When the TDRE flag of the SSR register is set to "1", transmitter data is written into the SODR register. If the TXE bit of the SCR register is "1" at this time, 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 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.

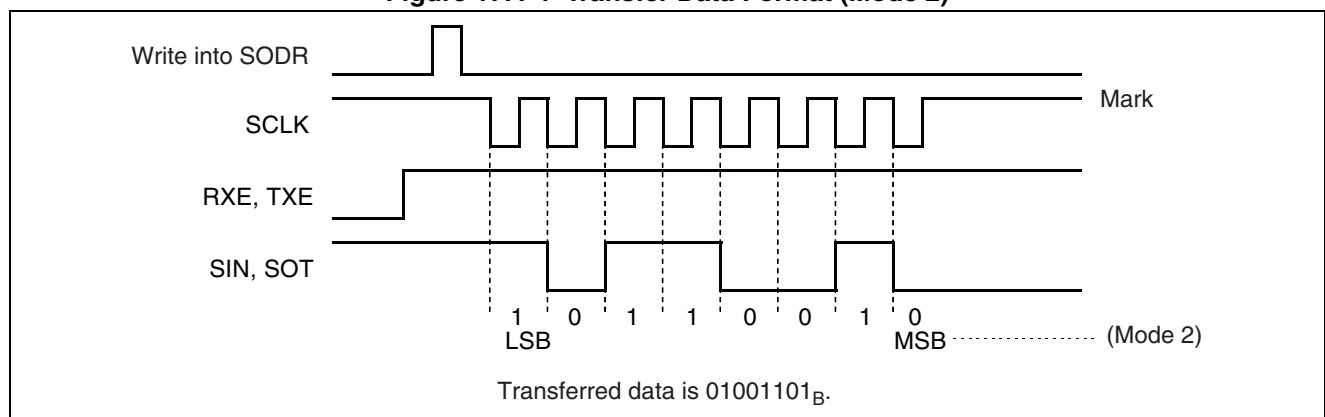
17.4.3 CLK-synchronous Mode

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.

■ Transfer Data Format

The UART handles only data in NRZ (non return to zero) format. Figure 17.4-2 shows the relationship between the transmitter or receiver clock and data.

Figure 17.4-4 Transfer Data Format (Mode 2)



If the internal clock (communication prescaler or internal timer) is selected, a synchronous clock for receiving data is generated automatically 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.

■ Initialization

The settings of the control registers used in the CLK-synchronous mode are shown below.

Table 17.4-5 Settings of Control Registers Used in CLK-synchronous Mode

Register name	Bit name	Setting
SMR register	MD1, MD0	10 _B
	CS2, CS1, CS0	Specifies clock input.
	SCKE	"1" for communication prescaler 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	These bits are not significant.
	CL	"1" (8-bit data)
	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.

17.4.4 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 (Overrun error)**

- **FRE (Framing error)**

These flags are set when a receiver error occurs, and cleared when "0" is written into the REC bit of the SCR register.

- **RDRF**

This flag is set when receiver data is loaded into the S IDR register, and cleared when data is read from the S IDR 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 S ODR register is empty and ready to write data, and cleared when data is written into the S ODR register.

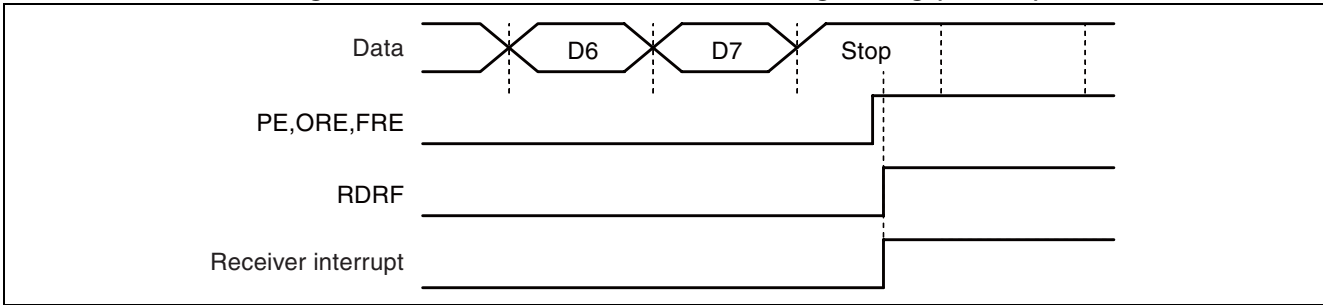
- The two interrupt sources are for both receiver and transmitter. During the receive operation, PE, ORE, FRE, or RDRF issues an interrupt request; during the transmit operation, TDRE issues an interrupt request.

■ 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 "H", data in the S IDR register becomes invalid.

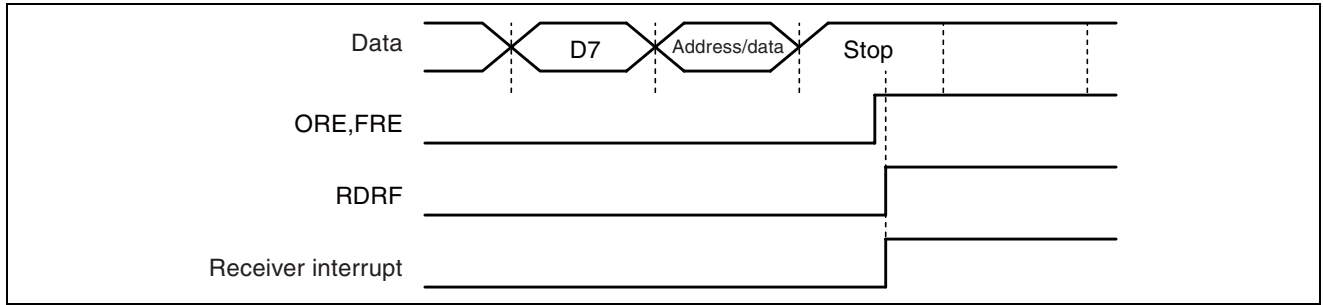
Figure 17.4-5 ORE, FRE, and RDRF Setting Timing (Mode 0)



○ 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 "H", data in the SIDR register becomes invalid.

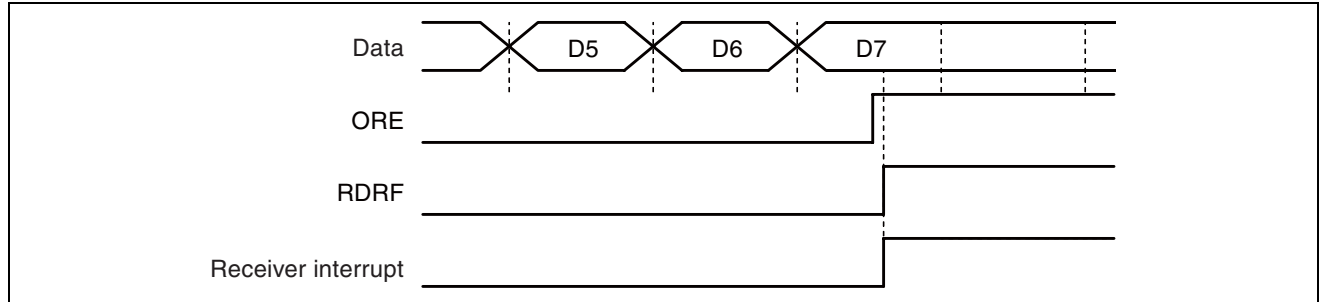
Figure 17.4-6 ORE, FRE, and RDRF Setting Timing (Mode 1)



○ During receiver operation in mode 2

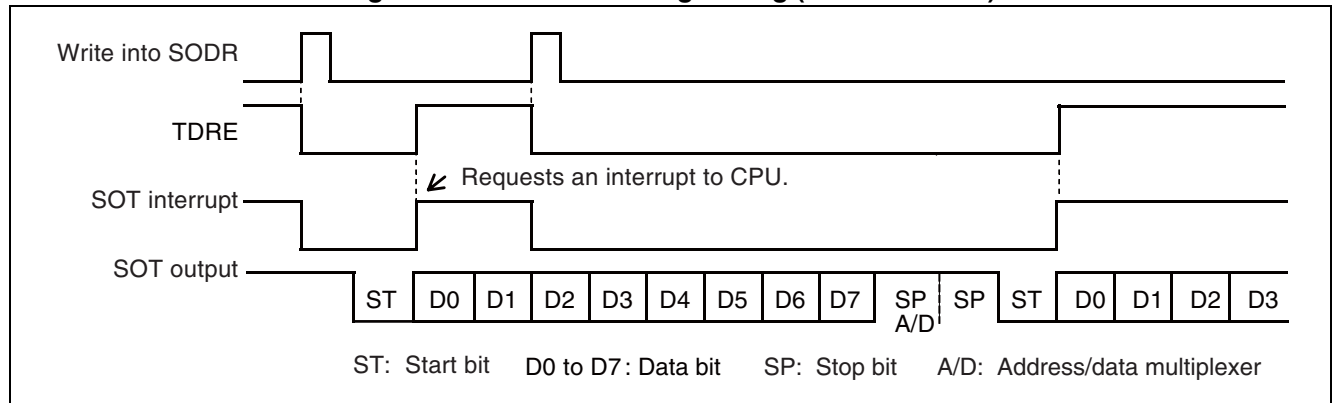
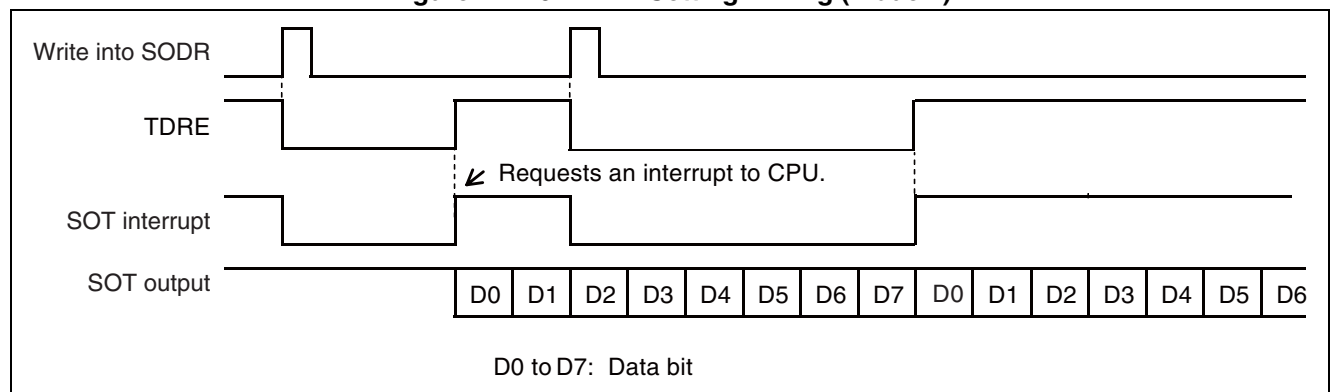
The ORE or RDRF is set when receive transfer is finished and the last data (D7) is detected, and an interrupt request to the CPU occurs. When the ORE is "H", data in the SIDR register becomes invalid.

Figure 17.4-7 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.

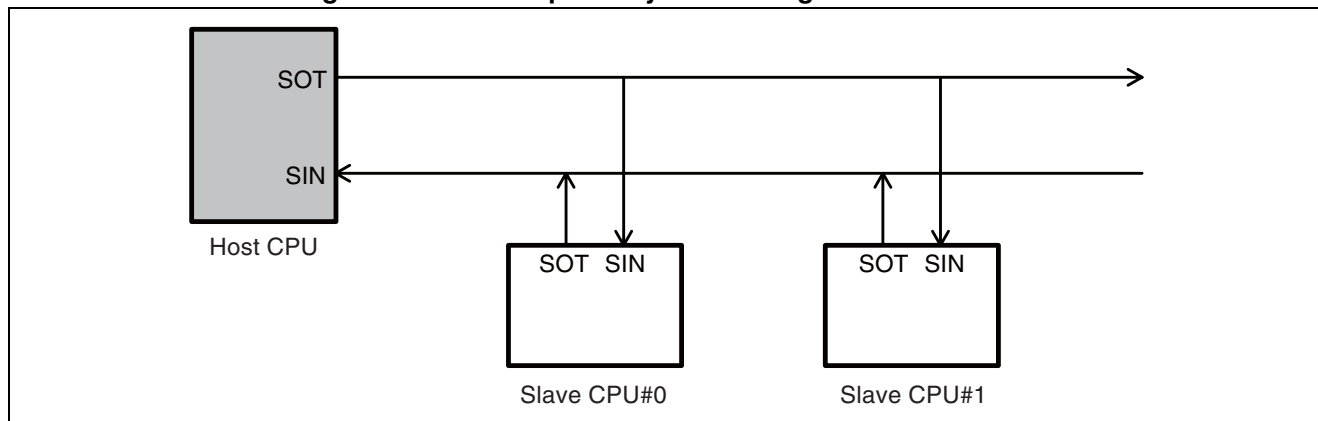
Figure 17.4-8 TDRE Setting Timing (Modes 0 and 1)**Figure 17.4-9 TDRE Setting Timing (mode 2)**

17.5 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 support the host communication interface (see Figure 17.5-1).

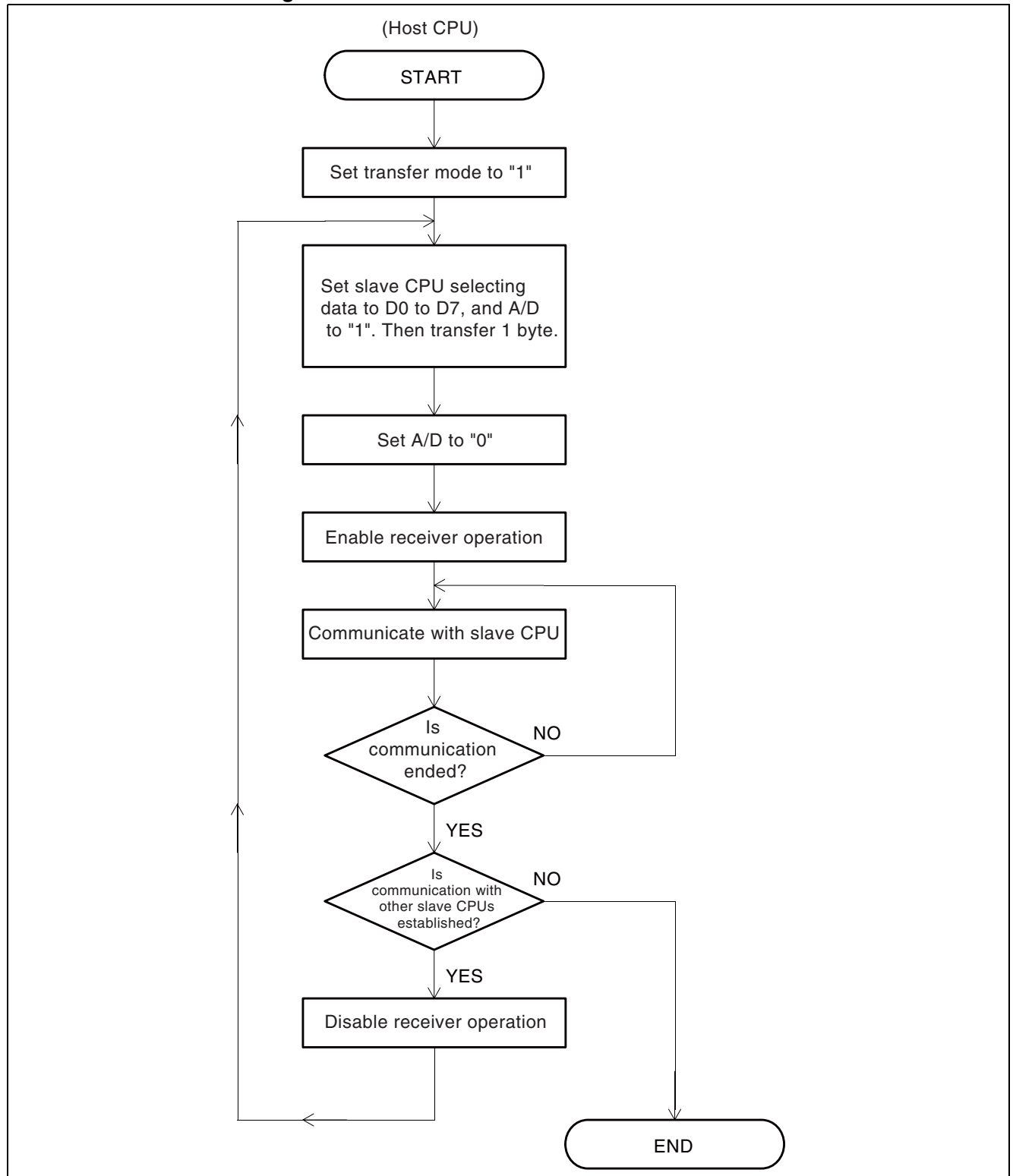
■ Application of UART (During Operation in Mode 1)

Figure 17.5-1 Example of System Configuration in Mode 1



As shown in Figure 17.5-2, 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 17.5-2 Communication Flowchart in Mode 1

CHAPTER 18 I/O EXTENDED SERIAL INTERFACE

This chapter describes the function and operation of the I/O extended serial interface.

- 18.1 Overview of the I/O Extended Serial Interface
- 18.2 Registers of the I/O Extended Serial Interface
- 18.3 Operation of I/O Extended Serial Interface

18.1 Overview of the I/O Extended Serial Interface

The I/O extended serial interface is a serial I/O interface having an eight-bit by one channel configuration, which can transfer data in synchronization with clocks. Also, the LSB first or MSB first mode can be selected for data transfer.

■ Overview of the I/O Extended Serial Interface

The I/O extended serial interface supports the following two operation modes:

- **Internal shift clock mode**

Transfers data in synchronization with internal clocks (communication prescaler).

- **External shift clock mode**

Transfers data in synchronization with clock input at the external pin (SCK). By manipulating the general-purpose port, which shares the external pin (SCK) in this mode, a transfer can also be executed by an instruction from the CPU.

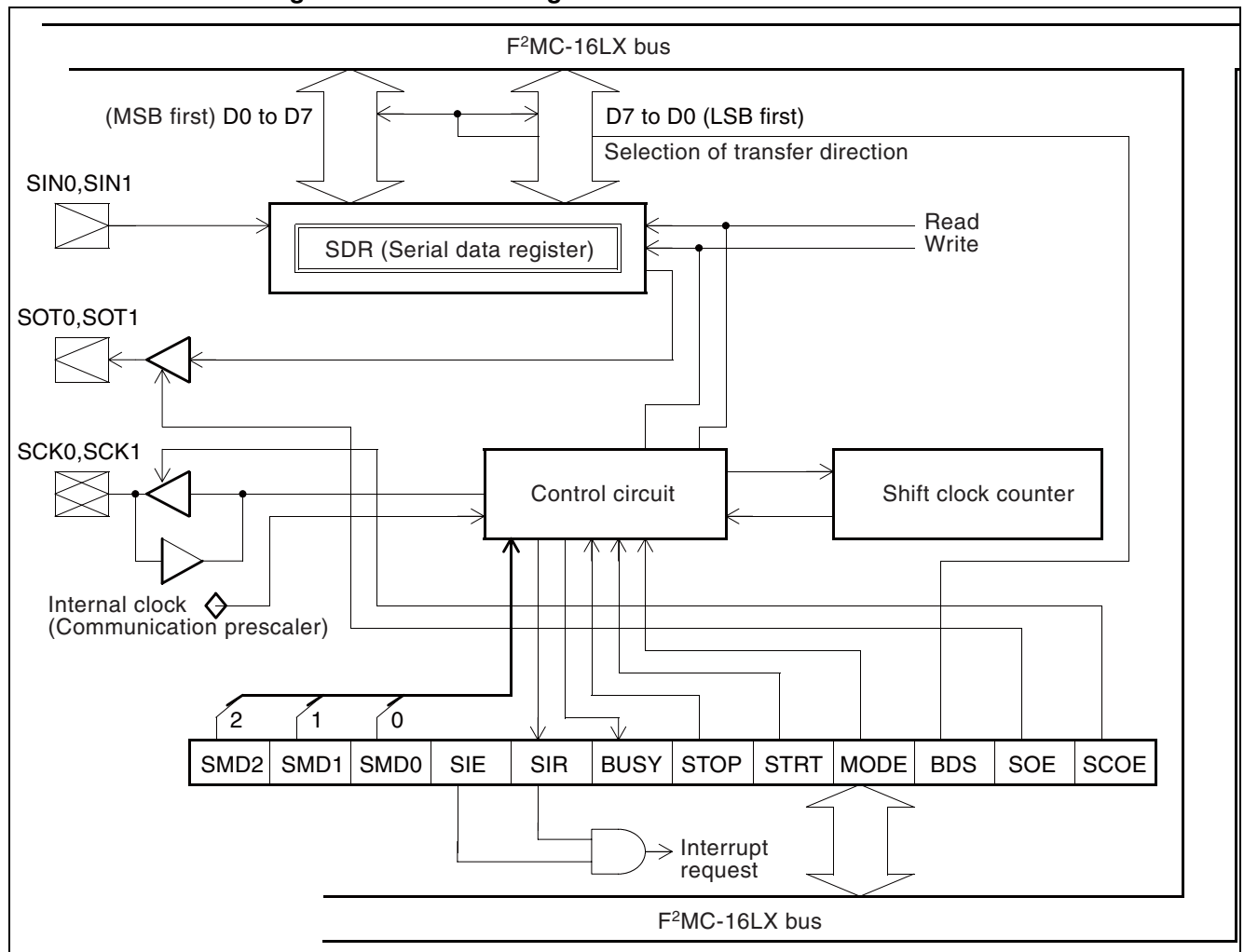
This series incorporates two channels of the I/O extended serial interface.

Note:

While channel 0 of the I/O extended serial interface is used, channel 0 of the I²C interface cannot be used because they share the same I/O port.

■ Block Diagram of the I/O Extended Serial Interface

Figure 18.1-1 Block Diagram of I/O Extended Serial Interface



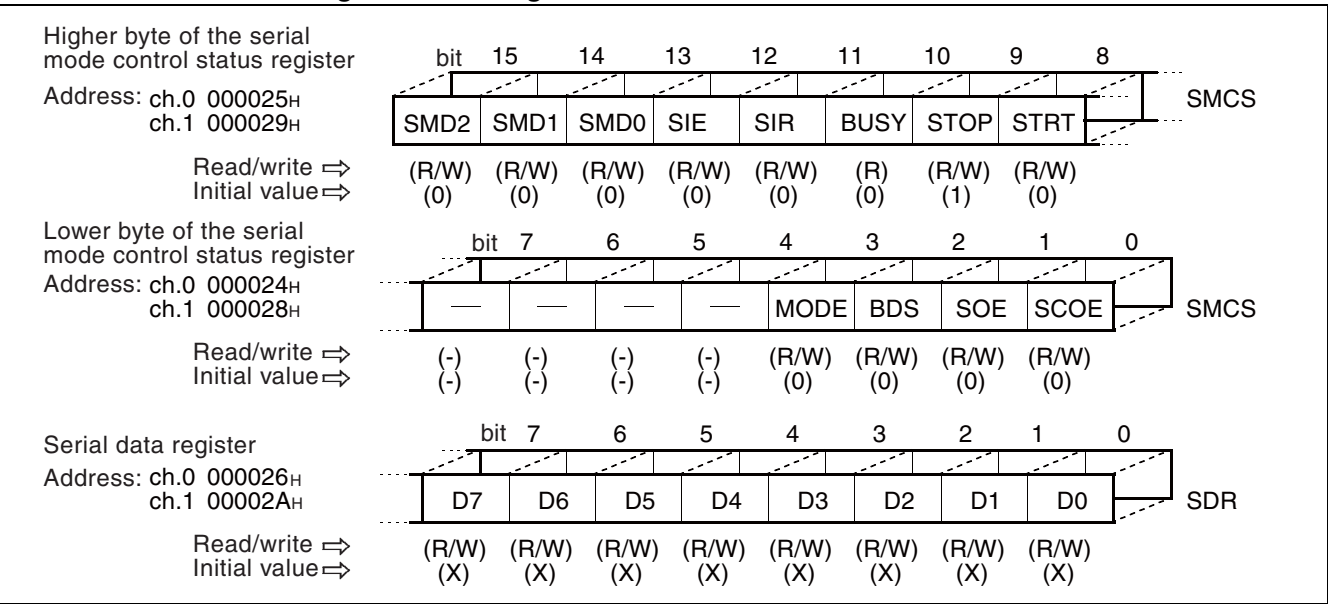
18.2 Registers of the I/O Extended Serial Interface

The I/O extended serial interface has the following three registers:

- High-order byte of the serial mode control status register
- Low-order byte of the serial mode control status register
- Serial data register

■ Registers of the I/O Extended Serial Interface

Figure 18.2-1 Registers of I/O Extended Serial Interface

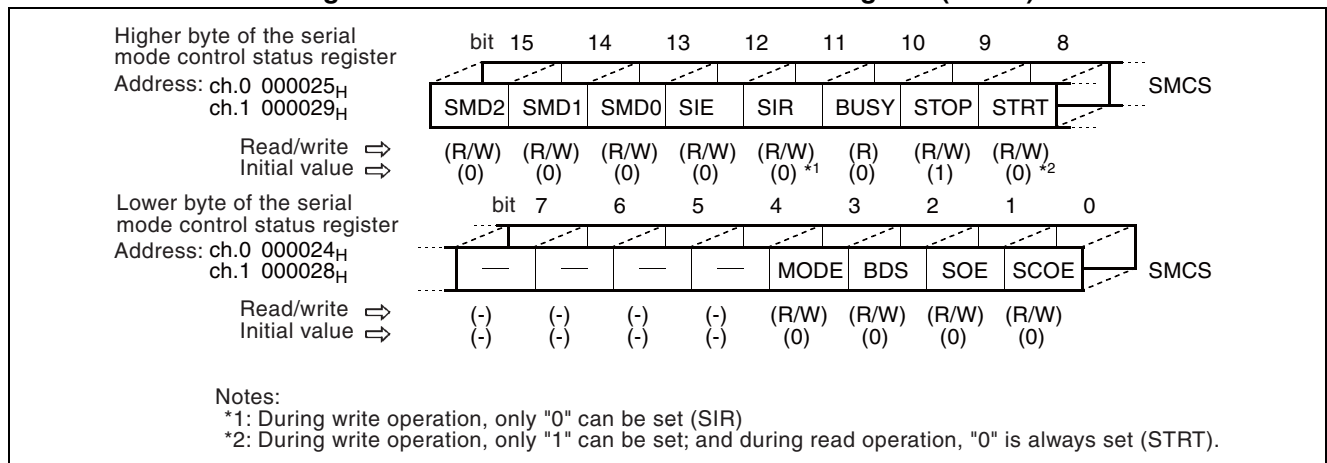


18.2.1 Serial Mode Control Status Register (SMCS)

The serial mode control status register (SMCS) is a register that controls the transfer operation mode of the serial I/O.

■ Serial Mode Control Status Register (SMCS)

Figure 18.2-2 Serial Mode Control Status Register (SMCS)



This section describes the function of the individual bits.

[bit15 to bit13] SMD2, SMD1, and SMD0 (Serial shift clock mode)

The SMD2, SMD1, and SMD0 bits select a serial shift clock mode as shown in Table 18.2-1.

These bits are initialized to 000 by a reset. During a transfer, rewriting these bits is prohibited.

A shift clock can be selected from five types of internal shift clocks and an external shift clock.

Do not set SMD2, SMD1, and SMD0 = 110 and 111, because they are reserved.

When SCOE = 0 is set for clock selection, manipulating the ports that share the SCK0 and SCK1 pins can also effect a shift operation in the unit of instruction.

Table 18.2-1 Function of SMD0 to SMD2 (Serial Shift Clock Mode Selection Bit)

SMD2	SMD1	SMD0	Serial shift clock mode
000 _B to 100 _B			Internal shift clock mode (Communication prescaler)
1	0	1	External shift clock mode
1	1	0	Do not set
1	1	1	Do not set

[bit12] SIE (Serial I/O interrupt enable)

The SIE bit controls an interrupt request from the serial I/O as shown in Table 18.2-2.

This bit is readable and writable.

Table 18.2-2 Function of SIE (Serial I/O Interrupt Request Control Bit)

SIE	Function
0	Prohibits the serial I/O interrupts. [Initial value]
1	Allows the serial I/O interrupts.

[bit11] SIR (Serial I/O interrupt request)

The SIR bit is set to "1" when a transfer of serial data terminated. When an interrupt is allowed (SIE = 1), if this bit becomes "1", an interrupt request to the CPU is generated. The clear conditions depend on the MODE bit. When the MODE bit is "0", the interrupt is cleared by writing "0" to the SIR bit; and when the MODE bit is "1", it is cleared by a read or write operation to the SDR register.

Also, it is cleared, regardless of the MODE bit, by a reset or writing 1 to the STOP bit.

It is ineffectual to write 1 to this bit. When it is read with a read-modify-write instruction, the read-out is always "1".

[bit10] BUSY

The BUSY bit indicates whether a serial transfer is in progress. This bit is read only.

Table 18.2-3 Function of BUSY Bit

BUSY	Function
0	Indicates the suspend or serial data register R/W wait state [Initial value]
1	Indicates the serial transfer state

[bit9] STOP

The STOP bit forcefully stops the serial transfer.

Setting "1" to this bit causes the stop state.

This bit is readable and writable.

Table 18.2-4 Function of STOP Bit

STOP	Function
0	Normal operation
1	Transfer stopped [Initial value]

[bit8] STRT

The STRT bit executes a serial transfer. Writing "1" to this bit under the suspend state starts a transfer. However, during a serial transfer operation or under the serial shift register R/W wait state, writing "1" is ignored. Writing "0" is ignored.

When this bit is read, the read-out is always "0".

[bit3] MODE

The MODE bit selects an execution condition under the suspend state. Rewriting it during operation, however, is prohibited.

This bit is readable and writable.

Set "1" when executing the extended intelligent I/O service.

Table 18.2-5 Function of MODE (Execution Condition Select Bit)

MODE	Function
0	Executes when STRT = 1 is set. [Initial value]
1	Executes by a read or write of the serial data register.

Note:

Even if MODE = 1 is selected, STRT = 1 must be set. In other words, when STRT = 1 has been set, writing data to the data register effects an output.

[bit2] BDS (Bit direction select)

Selects whether a transfer starts from the least significant bit (LSB first) or the most significant bit (MSB first) as shown in Table 18.2-6 when serial data is input or output.

This bit is readable and writable.

Table 18.2-6 Function of Bit Direction Select (BDS) Bit

BDS	Function
0	LSB first [Initial value]
1	MSB first

Note:

Set BDS to select the bit direction of transfer before writing data into the SDR register.

[bit1] SOE (Serial out enable)

The SOE bit controls an output at the output external pin for serial I/O (SOT0 and SOT1) as shown in Table 18.2-7.

This bit is readable and writable.

Table 18.2-7 Function of Serial Out Enable (SOE) Bit

SOE	Function
0	General-purpose port pin [Initial value]
1	Serial data output

[bit0] SCOE (SCLK output enable)

Controls an output at the input/output external pin for shift clock (SCK0 and SCK1) as shown in Table 18.2-8.

Set 0 when a transfer is executed in the unit of instruction in external shift clock mode. This bit is readable and writable.

Table 18.2-8 Function of SCLK Output Enable (SCOE) Bit

SCOE	Function
0	General-purpose port pin [Initial value]
1	Shift clock output pin

18.2.2 Serial Shift Data Register (SDR)

The SDR register is a register that holds the transfer data of the serial I/O. During a transfer, writing and reading the SDR register is prohibited.

■ Serial Shift Data Register (SDR)

Figure 18.2-3 Serial Shift Data Register (SDR)

Serial data register	bit	7	6	5	4	3	2	1	0	
Address: ch.0 000026 _H										
ch.1 00002A _H										
		D7	D6	D5	D4	D3	D2	D1	D0	SDR
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)	

18.3 Operation of I/O Extended Serial Interface

The I/O extended serial interface, which is used for input and output of eight-bit serial data, consists of the SMCS register and SDR register.

■ Operation of I/O Extended Serial Interface

For input and output of serial data, the contents of the shift register are output at the serial output pin (SOT0 and SOT1 pins) in bit series in synchronizing with the falling edge of a serial shift clock (an external clock or internal clock). And they are input to the SDR register at serial input pins (SIN0 and SIN1 pins) in bit series in synchronization with the rising edge. The direction of shift (transfer from the MSB or LSB) can be specified with the BDS bit of the SMCS register.

Upon completion of a transfer, the status changes to the suspend state or data register R/W wait state depending on the MODE bit of the SMCS register. These states are changed back to the transfer state in the following two ways.

- To recover from the suspend state, write "0" to the STOP bit and "1" to the STRT bit. (The STOP and STRT bits can be set at the same time.)
- To recover from the SDR register R/W wait state, read or write to the data register.

18.3.1 Shift Clock

The shift clock supports two types of modes, internal shift clock mode and external shift clock mode, that can be specified with the SMCS register. Change the modes while the serial I/O is in the suspend state. To ensure the suspend state, read the BUSY bit.

■ Internal Shift Clock Mode

By the output of the communication prescaler, a shift clock at duty cycle of 50% can be output at the SCK pin as a synchronization timing output. Data is transferred by one bit per clock.

The transfer rate is calculated by the following expression.

$$\text{Transfer rate (s)} = \frac{A}{\text{Machine cycle of the internal clock (Hz)}}$$

A is a division ratio, 2^1 , 2^2 , 2^4 , 2^5 , or 2^6 , specified with the SMD bit of the SMCS register.

Table 18.3-1 Set Value of SMD0 to SMD2 (Serial Shift Clock Mode Selection Bits) and Example of Setting Shift Clock

SMD2	SMD1	SMD0	$\phi/\text{div} = 4 \text{ MHz}$	$\phi/\text{div} = 2 \text{ MHz}$	$\phi/\text{div} = 1 \text{ MHz}$	Calculation formula
0	0	0	2 Mbps	1 Mbps	500 kbps	$(\phi/\text{div})/2^1$
0	0	1	1 Mbps	500 kbps	125 kbps	$(\phi/\text{div})/2^2$
0	1	0	250 kbps	125 kbps	62.5 kbps	$(\phi/\text{div})/2^4$
0	1	1	125 kbps	62.5 kbps	32.25 kbps	$(\phi/\text{div})/2^5$
1	0	0	62.5 kbps	31.25 kbps	15.625 kbps	$(\phi/\text{div})/2^6$

div expresses the set value of the communication prescaler. For detailed information, see "CHAPTER 16 COMMUNICATION PRESCALER REGISTER".

■ External Shift Clock Mode

In external shift clock mode, data is transferred by one bit per clock in synchronization with the external shift clock input at SCK0 and SCK1 pins. The transfer rate can vary from DC to 1/(8 machine cycles). For example, when 1 machine cycle = 62.5 ns, the transfer rate can be up to 2 MHz.

Data can also be transferred on a per-instruction basis by making the following setting.

1. Select external shift clock mode and set 0 to the SCOE bit of the SMCS register.
2. Write 1 to the direction register of the port that shares the SCK0 and SCK1 pins to set the port to output mode.

After setting as described above, when writing 1 and 0 to the data register of the port (PDR), the value of the port output at SCK0 and SCK1 pins is captured as an external clock to operate a transfer. Start the shift clock at H level.

Note:

Writing to the SMCS register and SDR register is prohibited during operation of the serial I/O.

18.3.2 Operation States of the Serial I/O

The serial I/O supports the following four operation states:

- STOP state
 - Suspend state
 - SDR register R/W wait state
 - Transfer state
-

■ STOP State

At a reset or when writing 1 to the STOP bit of SMCS, the shift counter is initialized and SIR = 0 is set. To recover from the stop state, set STOP = 0 and STRT = 1 (specifiable at the same time). The STOP bit has higher priority than the STRT bit. Therefore, while STOP = 1, STRT = 1 cannot start the transfer operation.

■ Suspend State

When the MODE bit is 0, terminating a transfer sets BUSY = 0 and SIR = 1 in the SMCS register, initializes the counter, and changes to the suspend state. To recover from the suspend state, set STRT = 1. Then the transfer operation restarts.

■ Serial Data Register R/W Wait State

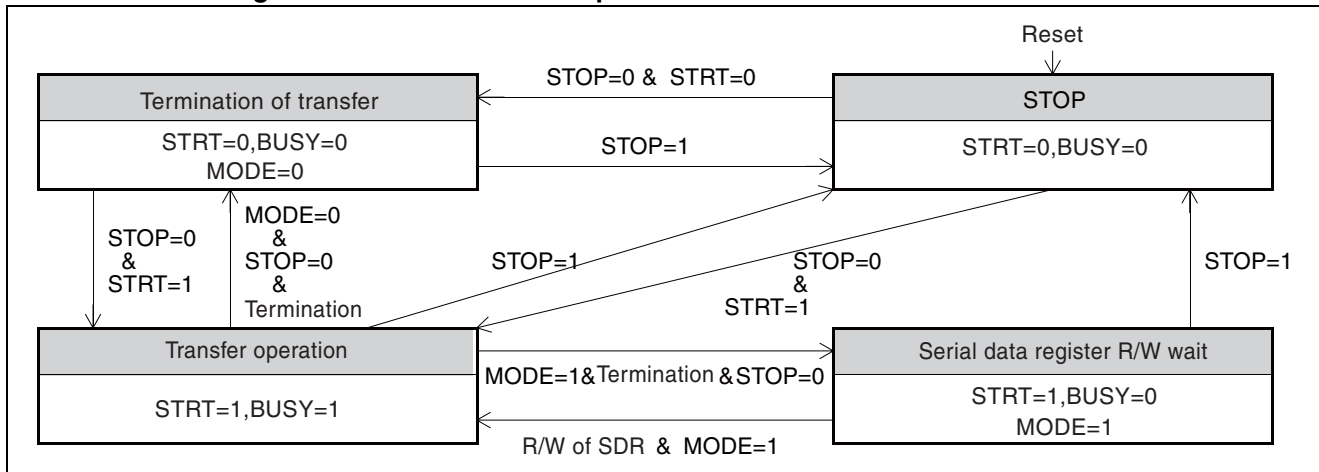
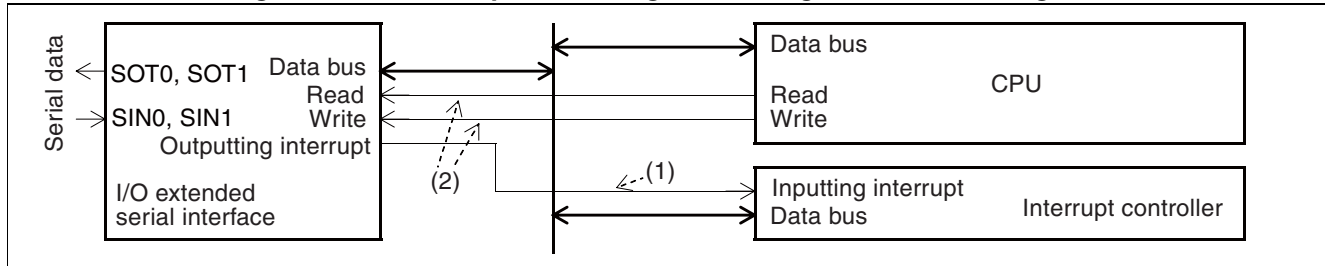
When the MODE bit of the SMCS register is "1", if a serial transfer terminates, BUSY = 0 and SIR = 1 are set and the serial I/O changes to the SDR register R/W wait state. If the interrupt enable register allows an interrupt, this block sends an interrupt signal.

To recover from the R/W wait state, read or write the SDR register to set BUSY = 1. The transfer operation then restarts.

■ Transfer State

In this state, BUSY = 1 and the serial transfer is being executed. This state transits to the suspend or R/W wait state depending on the MODE bit.

Figure 18.3-1 shows the transition of the operation states of the I/O extended serial interface. Figure 18.3-2 shows the concept of reading and writing to the serial data register.

Figure 18.3-1 Transition of Operation of I/O Extended Serial Interface**Figure 18.3-2 Concept of Reading and Writing to Serial Data Register**

(1) and (2) in Figure 18.3-2 are explained below.

- (1) When $MODE = 1$, if a transfer is terminated by the shift clock counter, $SIR = 1$ is set and the state transits to the read/write wait state. If the SIE bit is "1", an interrupt signal is generated. However, if a transfer is terminated by the inactive SIE bit or writing "1" to $STOP$, no interrupt signal is generated.
- (2) When the SDR register is read or written, the interrupt request is cleared to start a serial transfer.

18.3.3 Start/Stop Timing Of Shift Operation and Input/Output Timing

Start: Sets the STOP bit of the SMCS register to "0" and the STRT bit to "1".

Stop: Stopped by the termination of a transfer or by setting STOP = 1.

Stopped by STOP = 1 --> Regardless of the MODE bit, SIR = 0 is left set and stopped.

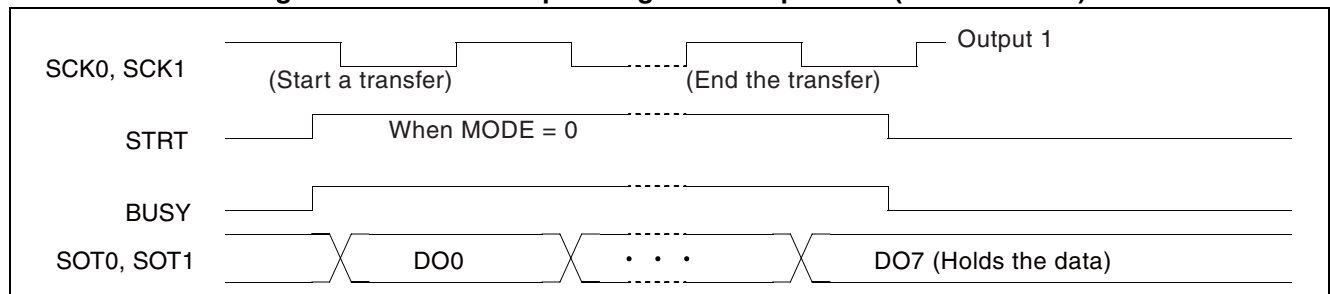
Stopped by termination of a transfer --> Regardless of the MODE bit, SIR = 1 is set and then stopped.

■ Start/Stop Timing of Shift Operation and Input/Output Timing

Regardless of the MODE bit, the BUSY bit is "1" under the serial transfer state; and "0" under the suspend or R/W wait state. To check the status of a transfer, read this bit.

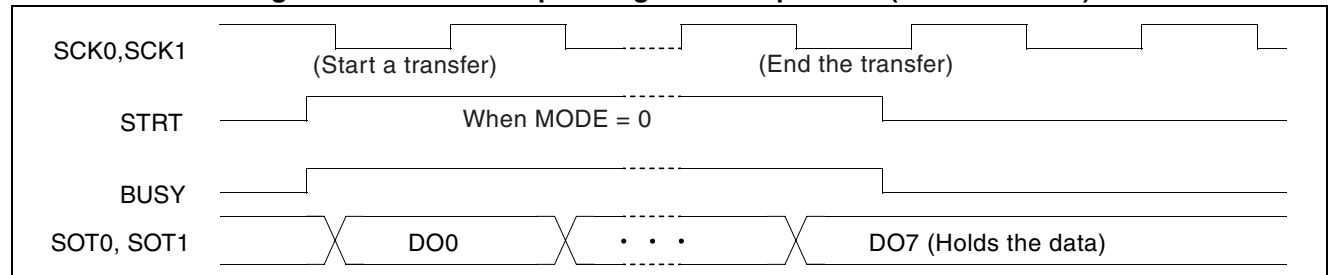
○ Internal shift clock mode (LSB first)

Figure 18.3-3 Start/Stop Timing of Shift Operation (Internal Clock)



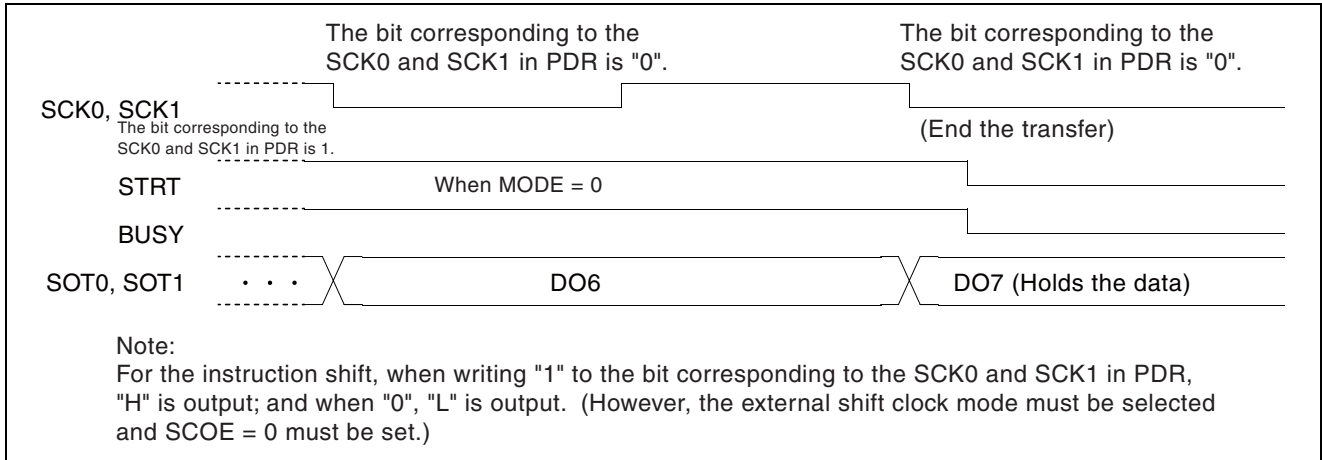
○ External shift clock mode (LSB first)

Figure 18.3-4 Start/Stop Timing of Shift Operation (External Clock)



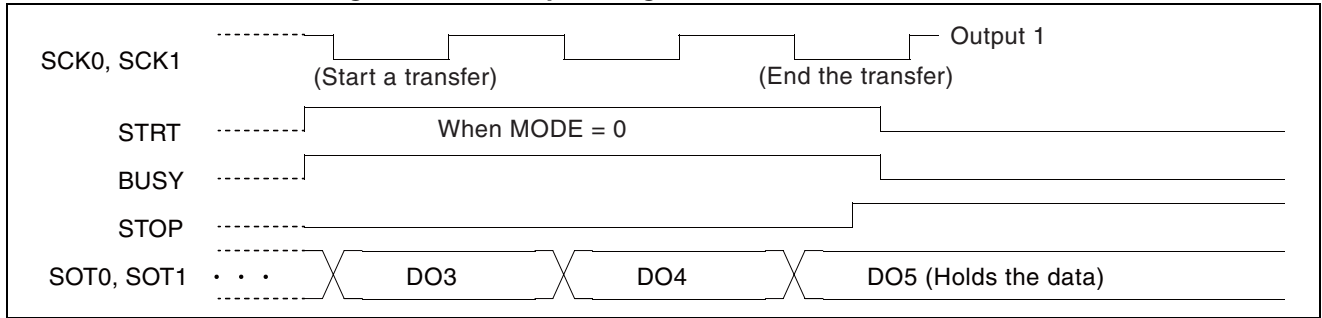
○ Instruction shift in external shift clock mode (LSB first)

Figure 18.3-5 Start/Stop Timing of Shift Operation (Shifted by Instruction in External Shift Clock Mode)



○ Stopped by STOP = 1 (LSB first, using internal clock)

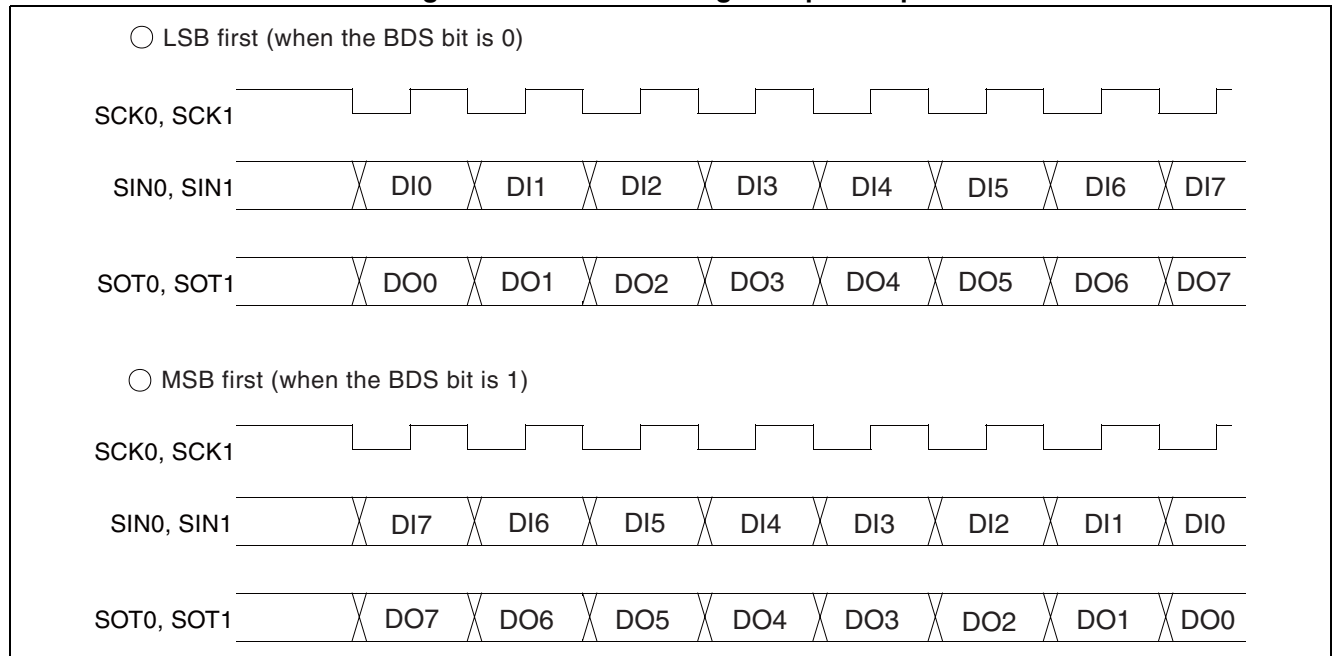
Figure 18.3-6 Stop Timing when STOP Bit Becomes 1



Note:

DO7 to DO0 indicate output data.

During a transfer of serial data, a data unit is output at the falling of the shift clock at the serial output pins (SOT0 and SOT1) and input a data unit at the rising at serial input pins (SIN0 and SIN1).

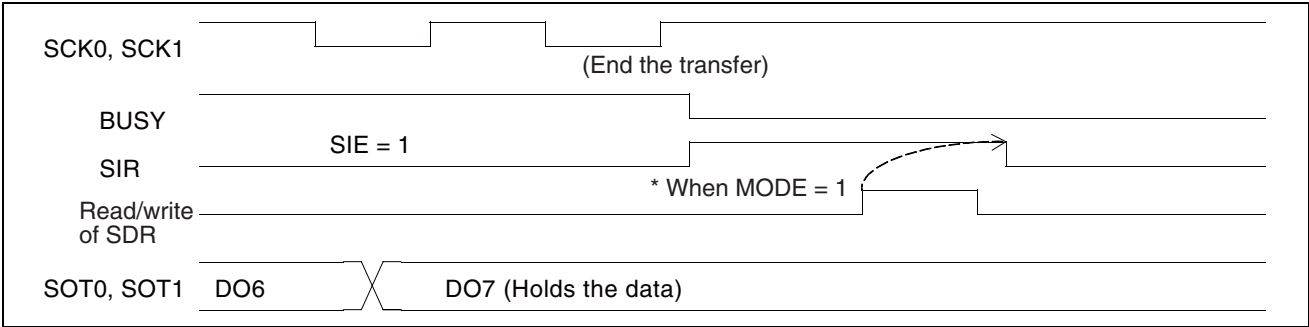
Figure 18.3-7 Shift Timing for Input/Output

18.3.4 Interrupt Function of the I/O Extended Serial Interface

The I/O extended serial interface can generate an interrupt request to the CPU. When data transfer has completed, the SIR bit will be set as an interrupt flag. If the SIE bit of the SMCS register is "1" to allow an interrupt, an interrupt request is output to the CPU.

■ Interrupt Function of the I/O Extended Serial Interface

Figure 18.3-8 Output Timing for Interrupt Signals



CHAPTER 19 I²C INTERFACE

This chapter provides an overview and describes the functions of the I²C interface.

19.1 Overview of the I²C Interface

19.2 Block Diagram and Structure of the I²C Interface

19.3 Registers of the I²C Interface

19.4 Operation of the I²C Interface

19.1 Overview of the I²C Interface

The I²C interface operates as a master or slave device on the I²C bus at the serial I/O port that supports an inter IC bus.

■ Features of the I²C Interface

The features of the I²C interface are follows:

- Transmission between the master and slave
- Arbitration function
- Clock synchronization function
- Function to detect the slave address and general call address
- Function to detect the transfer direction
- Function to repeat or detect the start condition
- Detection function of bus errors

The I²C interface of this series is comprised of two circuits with three channels.

Note:

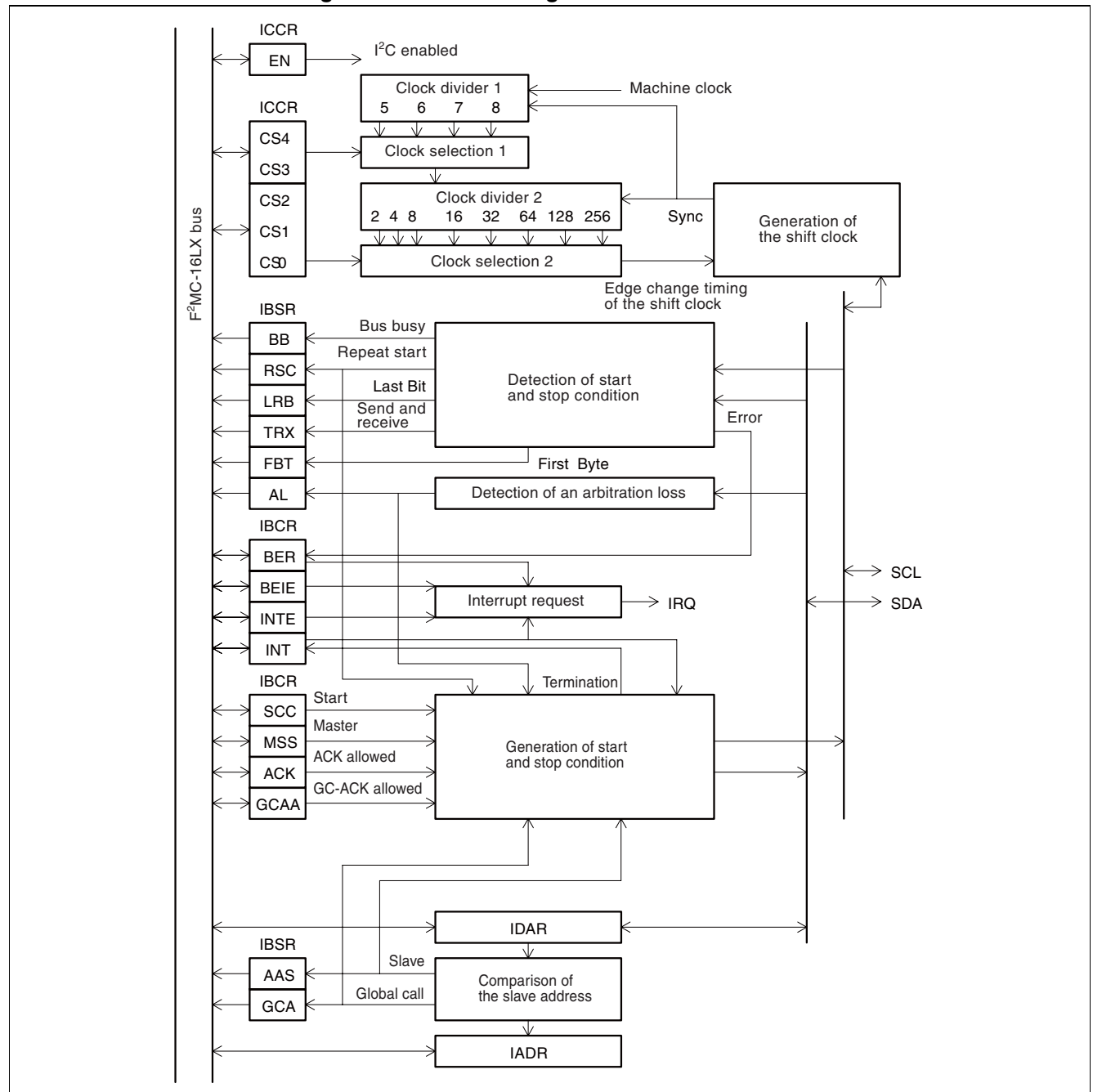
When channel 0 of the I²C interface is used, channel 0 of the I/O extended serial interface cannot be used because the I/O port is shared.

19.2 Block Diagram and Structure of the I²C Interface

Figure 19.2-1 shows a block diagram of the I²C interface, and Figure 19.2-2 shows the structure of the I²C interface.

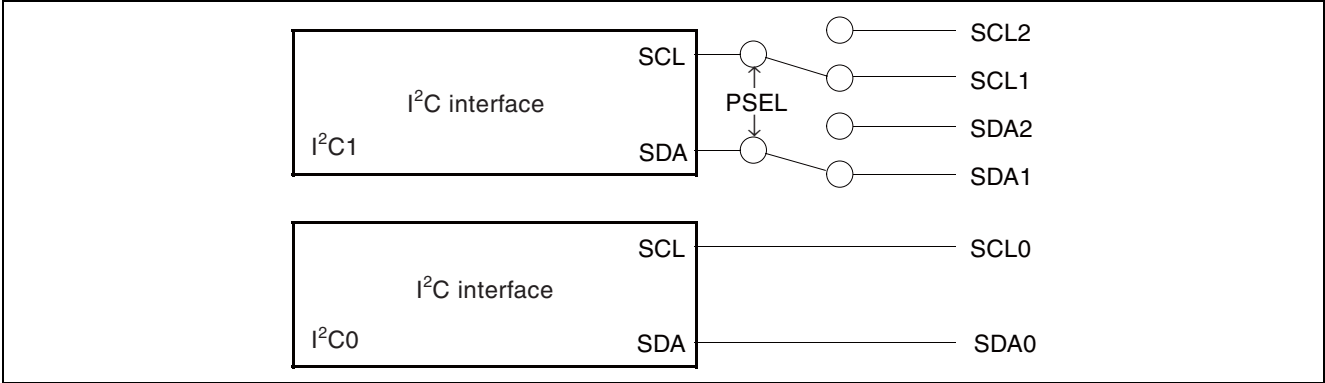
■ Block Diagram of the I²C Interface

Figure 19.2-1 Block Diagram of the I²C Interface



■ Structure of the I²C Interface

Figure 19.2-2 Structure of the I²C interface



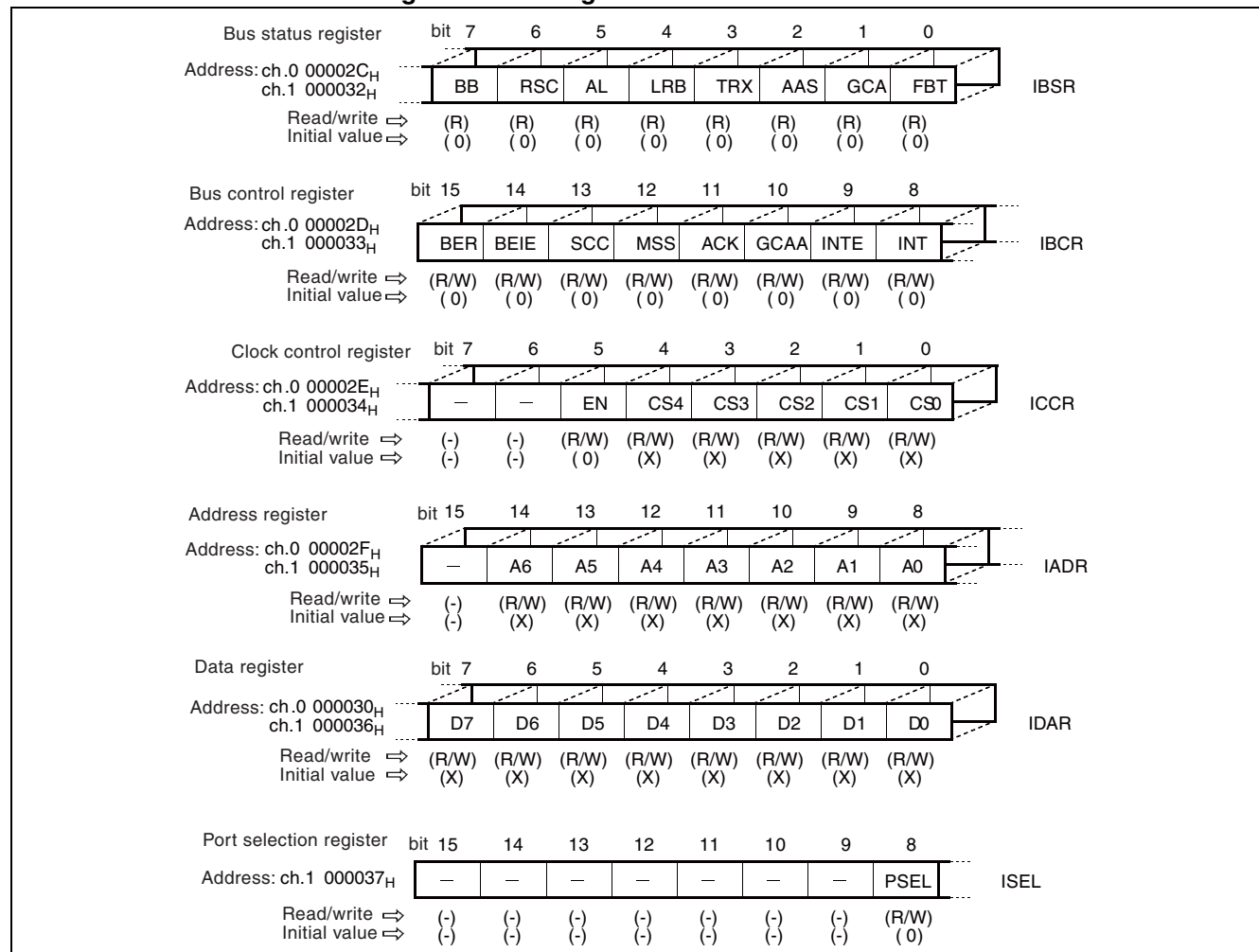
19.3 Registers of the I²C Interface

The I²C interface has the following six types of registers:

- Bus status register
- Bus control register
- Clock control register
- Address register
- Data register
- Port selection register

■ Registers of the I²C Interface

Figure 19.3-1 Registers of the I²C Interface

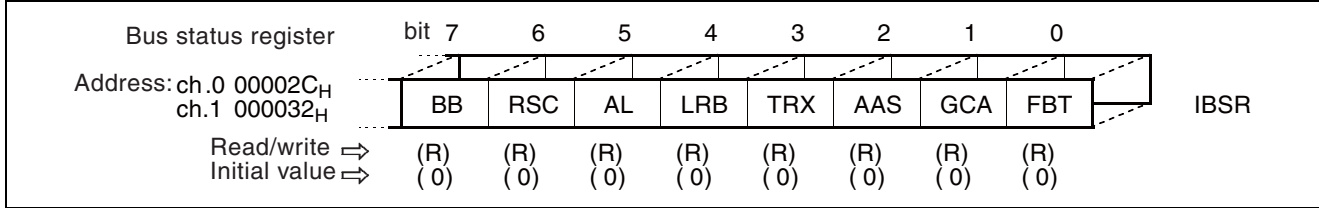


19.3.1 Bus Status Register (IBSR)

The bus status register (IBSR) shows the status of each function of the I²C interface.

■ Bus Satus Register (IBSR)

Figure 19.3-2 Bus status register (IBSR)



[bit7] BB (Bus busy)

The BB bit shows the status of the I²C bus.

Table 19.3-1 Functions of the BB (Bus Busy) Bit

BB	Status of the I ² C bus
0	Stop condition is detected. [Initial value]
1	Start condition is detected. (This means the bus is in use.)

[bit6] RSC (Repeated start condition)

The RSC bit detects the repeated start condition. This bit is cleared when (1) "0" is written to the INT bit, (2) no addressing was made in slave mode, (3) the start condition is detected while the bus stops, or (4) the stop condition is detected.

Table 19.3-2 Functions of the RSC (Repeated Start Condition) Bit

RSC	Status of the start condition detection
0	Repeated start condition is not detected. [Initial value]
1	Start condition is detected again while the bus is in use.

[bit5] AL (Arbitration lost)

The AL bit detects an arbitration loss.

This bit is cleared by writing "0" to the INT bit.

Table 19.3-3 Functions of the AL (Arbitration Lost) Bit

AL	Status of the arbitration loss detection
0	No arbitration loss is detected. [Initial value]
1	Either an arbitration loss has occurred during transmission by the master, or "1" has been written to the MSS bit while another system is using the bus.

[bit4] LRB (Last received bit)

The LRB is the acknowledgment store bit and stores acknowledgment from the receiving side.

Table 19.3-4 Functions of the LRB(Last Received Bit) Bit

LRB	Status of receiving acknowledgment
0	Receiving is acknowledged.
1	Receiving is not acknowledged.

This bit is cleared by the detection of a start or stop condition.

[bit3] TRX (Transfer/receive)

The TRX bit shows the direction of data transfer.

Table 19.3-5 Functions of the TRX (transfer/receive) Bit

TRX	Direction of data transfer
0	Receiving [Initial value]
1	Sending

[bit2] AAS (Addressed as slave)

The AAS bit detects addressing.

This bit is cleared by the detection of a start or stop condition.

Table 19.3-6 Functions of AAS (addressed as slave) Bit

AAS	Function
0	No addressing is made in slave mode. [Initial value]
1	Addressing is made in slave mode.

[bit1] GCA (General call address)

The GCA bit detects the general call address (00_H). This bit is cleared upon detection of a start or stop condition.

Table 19.3-7 Functions of the GCA (general call address) Bit

GCA	Function
0	The general call address is not received in slave mode. [Initial value]
1	The general call address is received in slave mode.

[bit0] FBT (First byte transfer)

The FBT bit detects the first byte. If this bit is set to "1" by the detection of a start condition, it is cleared when "0" is written to the INT bit or when no addressing is made in slave mode.

Table 19.3-8 Functions of the FBT (first byte transfer) Bit

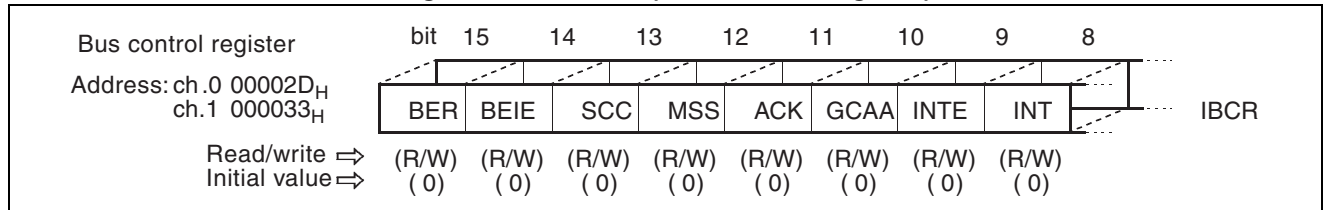
FBT	Function
0	The received data is not the first byte. [Initial value]
1	The received data is the first byte (address data).

19.3.2 Bus Control Register (IBCR)

The bus control register (IBCR) controls interrupts and functions of the I²C interface.

■ Bus Control Register (IBCR)

Figure 19.3-3 IBCR (Bus Control Register)



[bit15] BER (Bus error)

The BER bit is for the bus error interrupt request flag. When this bit is set, the EN bit of the CCR register is cleared, the I²C interface is suspended, and data transmission is aborted.

Table 19.3-9 Functions of the BER (Bus Error) Bit

BER		Function
When writing	0	Clears the bus error interrupt request flag. [Initial value]
	1	No meaning
When reading	0	No bus error is detected. [Initial value]
	1	An invalid start or stop condition is detected during data transmission.

[bit14] BEIE (bus error interrupt enable)

The BEIE bit is the interrupt permission bit for bus errors. When this bit is "1" and the BER bit is "1", an interrupt occurs.

Table 19.3-10 Functions of the BEIE (Bus Error Interrupt Enable) Bit

BEIE	Function
0	The bus error interrupt is disabled. [Initial value]
1	The bus error interrupt is enabled.

[bit13] SCC (Start condition continue)

The SCC bit generates a start condition. The read value of this bit is always "0".

Table 19.3-11 Functions of the SCC (Start Condition Continue) Bit when Writing

SCC	Function
0	No meaning [Initial value]
1	Generates a start condition again when the master is transmitting and starts transferring the address data.

[bit12] MSS (Master slave select)

The MSS bit is used for selection between the master and slave. This bit is cleared when an arbitration loss occurs during transmission by the master, enabling slave mode.

Table 19.3-12 Functions of the MSS (Master Slave Select) Bit

MSS	Function
0	Slave mode is turned on following the generation of a stop condition and termination of transmission. [Initial value]
1	Master mode is turned on and the start condition is generated, starting transmission.

[bit11] ACK (Acknowledge)

This bit enables generation of an acknowledgment upon receipt of data and is invalid when address data is received in slave mode.

Table 19.3-13 Functions of the ACK (Acknowledge) Bit

ACK	Function
0	No acknowledgment is generated. [Initial value]
1	An acknowledgment is generated.

[bit10] GCAA (General call address acknowledge)

This bit enables generation of an acknowledgment upon receipt of the general call address.

Table 19.3-14 Functions of the GCAA (General Call Address Acknowledge) Bit

GCAA	Function
0	No acknowledgment is generated. [Initial value]
1	An acknowledgment is generated.

[bit9] INTE (Interrupt enable)

The INTE bit is the interrupt permission bit. When this bit is "1" and the INT bit is "1", an interrupt occurs.

Table 19.3-15 Functions of the INTE (Interrupt Enable) Bit

INTE	Function
0	Interrupts are disabled. [Initial value]
1	Interrupts are enabled.

[bit8] INT (Interrupt)

The INT bit is for the interrupt request flag for transmission termination. When this bit is "1", the SCL line stays at L level. The next byte is transmitted after this bit is cleared by writing "0" and the SCL line is released. In master mode, generation of the start or stop condition resets this bit to "0".

Table 19.3-16 Functions of the INT (Interrupt) Bit

INT		Function
When writing	0	Clears the interrupt request flag for transmission termination. [Initial value]
	1	No meaning
When reading	0	The transmission is not terminated. [Initial value]
	1	This is set when transmission of one byte including the acknowledgment bit is terminated and any of the following conditions occur: <ul style="list-style-type: none"> • The device is the bus master. • The device is the addressed slave. • The general call address is received. • An arbitration loss has occurred. • An attempt was made to generate the start condition while another system is using the bus.

■ Competition Among the SCC, MSS, and INT Bits

When the SCC, MSS, and INT bits are written to at the same time, competition occurs among subsequent byte transmission and start and stop condition generation. The bits are prioritized as follows:

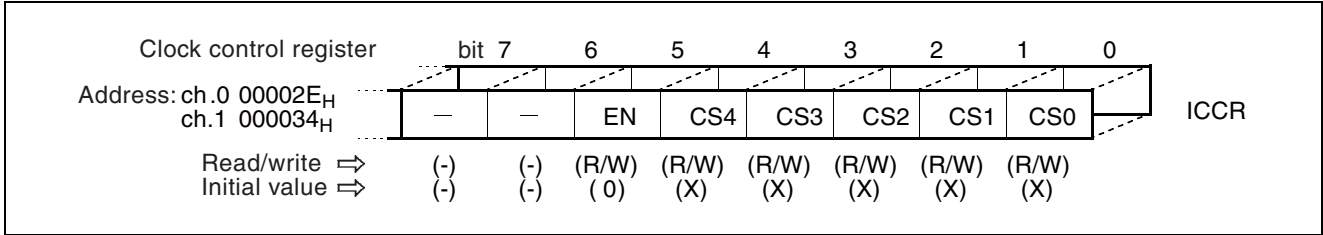
- 1) Transmission of the next byte and stop condition generation
When "0" is written to the INT bit and MSS bit, the MSS bit takes precedence and the stop condition is generated.
- 2) Transmission of the next byte and start condition generation
When "0" is written to the INT bit and "1" to the SCC bit, the SCC bit takes precedence and the start condition is generated.
- 3) Start condition generation and stop condition generation
It is not permitted to write "1" to the SCC bit and "0" to the MSS bit at the same time.

19.3.3 Clock Control Register (ICCR)

The clock control register (ICCR) controls the operation of the I²C interface and sets the frequency of the serial clock.

■ Clock Control Register (ICCR)

Figure 19.3-4 Clock Control Register (ICCR)



[bit5] EN (Enable)

The EN bit enables the I²C interface to operate. When this bit is "0", the bits of the BSR register and the BCR register (except for the BER and BEIE bits) are cleared. When the BER bit is set, this bit is cleared.

Table 19.3-17 Functions of the EN (Enable) Bit

EN	Operation status
0	Operation disabled [Initial value]
1	Operation enabled

[bit4 to bit0] CS4 to CS0 (Clock period select 4 to 0)

These bits are used to set the frequency of the serial clock.

fsck, the frequency of the shift clock, is set according to the following calculation:

$$fsck = \frac{\phi}{m \times n + 4}$$

ϕ : Machine clock

Table 19.3-18 Settings of the Serial Clock Frequency (CS4 and CS3)

m	CS4	CS3
5	0	0
6	0	1
7	1	0
8	1	1

Table 19.3-19 Settings of the Serial Clock Frequency (CS2 to CS0)

m	CS2	CS1	CS0
4	0	0	0
8	0	0	1
16	0	1	0
32	0	1	1
64	1	0	0
128	1	0	1
256	1	1	0
512	1	1	1

When ϕ is 16 MHz, for example, and 5 is set to m and 32 to n, the frequency of the serial clock is calculated to be 97.561 kHz.

Note:

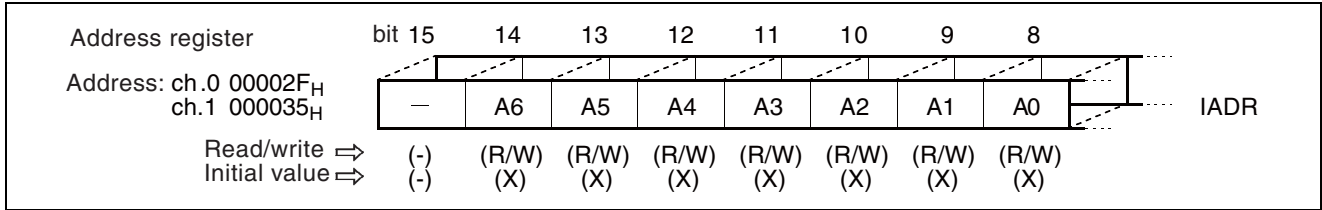
Four extra cycles are the minimum overhead for checking the changes of the output level of the SCL pin. When the SCL pin starts up with a long delay, or when the slave device suspends the clock, the value increases.

19.3.4 Address Register (IADR)

The address register (IADR) specifies the slave address.

■ Address Register (IADR)

Figure 19.3-5 IADR (Address Register)



[bit14 to bit8] A6 to A0 (Slave address bits)

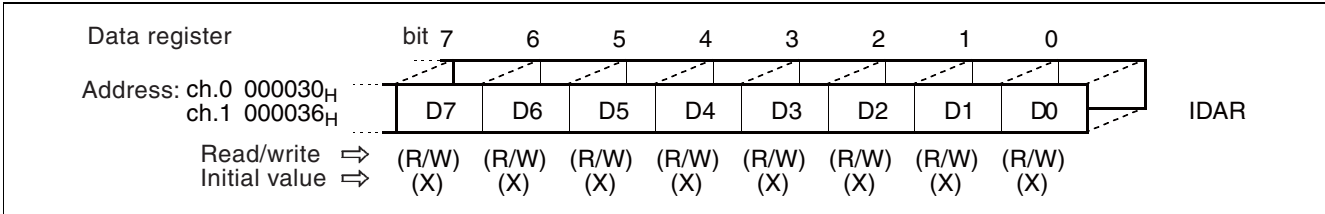
A6 to A0 are the registers used for specifying the slave address. In slave mode, the received address data is compared with the DAR register and an acknowledgment is sent to the master when a match occurs.

19.3.5 Data Register (IDAR)

The data register (IDAR) reads and writes the data used for serial transmission.

■ Data Register (IDAR)

Figure 19.3-6 Data Register (IDAR)



[bit7 to bit0] D7 to D0 (Data bits)

These bits are the data register used for serial transmission, which is transferred starting from the MSB. The data output value is 1 when data is being received (TRX = 0).

The writing side of this register is double-buffered. When the bus is in use (BB = 1), the written data is loaded in the register for serial transmission whenever one-byte is transmitted. The data is read directly from the register for serial transmission, which means the received data is available when the INT bit is set.

19.3.6 Port Selection Register (ISEL)

The port selection register (ISEL) selects the I/O port for the I²C interface.

■ Port Selection Register (ISEL)

Figure 19.3-7 Port Selection Register (ISEL)

Port selection register	bit 15	14	13	12	11	10	9	8	
Address: ch.1 000037 _H	—	—	—	—	—	—	—	PSEL	ISEL
Read/write ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

[bit8] PSEL

The PSEL bit selects the I/O port for the I²C interface. Confirm that the EN bit of the ICCR register is "0" before switching (writing) the I/O port.

The non-selected ports (P52 and P53, or P54 and P55) can be used as general purpose ports. This bit is available only for the I²C interface 1. The I²C interface 0 does not have this function.

Table 19.3-20 Functions of the PSEL Bit

PSEL	Function
0	Selects SCL1 and SDA1. [Initial value]
1	Selects SCL2 and SDA2.

19.4 Operation of the I²C Interface

The I²C bus is used for communication with two bi-directional bus lines; one is a serial data line (SDA) and the other is a serial clock line (SCL). The I²C interface has two corresponding open drain input-output pins (SDA and SCL) to support the wired logic.

■ Start Condition

If 1 is written to the MSS bit when the bus is free (BB = 0 and MSS = 0), the I²C interface is in master mode and the start condition is generated at the same time. In master mode, the start condition can be regenerated by writing "1" to the SCC bit even if the bus is in use (BB = 1).

The start condition is generated in the two following ways:

1. Write "1" to the MSS bit when the bus is not in use ($MSS = 0 \cdot BB = 0 \cdot INT = 0 \cdot AL = 0$).
2. Write "1" to the SCC bit in the interrupt state in bus master mode ($MSS = 1 \cdot BB = 1 \cdot INT = 1 \cdot AL = 0$).

If "1" is written to the MSS bit when another system (being idle) is using the bus, the AL bit is set to "1". In conditions other than the above 1) and 2), writing "1" to the MSS bit or the SCC bit is ignored.

■ Stop Condition

Writing "0" to the MSS bit in master mode ($MSS = 1$) generates the stop condition, and the mode switches to slave mode.

The condition necessary to generate the stop condition is as follows:

- Write "0" to the MSS bit in the interrupt state in bus master mode ($MSS = 1 \cdot BB = 1 \cdot INT = 1 \cdot AL = 0$).

Under other conditions, writing "0" to the MSS bit is ignored.

■ Addressing

In master mode, after the start condition is generated, the BB and TRX bits are set to "1" and the contents of the IDAR register are output starting from the MSB. When acknowledgment is received from the slave after the address data is sent, bit0 of the send data (bit0 of the sent IDAR register) is reversed and stored in the TRX bit.

In slave mode, after the start condition is generated, the BB bit is set to "1" and the TRX bit is set to "0", and the send data from the master is received by the IDAR register. After the address data is received, the IDAR register is compared with the IADR register. When a match occurs, the AAS bit is set to "1" and the acknowledgment is sent to the master. bit0 of the received data (bit0 of the received IDAR register) is then stored in the TRX bit.

■ Arbitration

If another master is sending data at the same time the master is sending data, arbitration occurs. When the send data is "1" and the data on the SDA line is at "L" level, the sender assumes that it lost arbitration and the AL bit is set to "1". As previously described, the AL bit is also set to "1" when the start condition is generated while the bus is in use.

When the AL bit is set to "1", the MSS bit and TRX bit are set to "0" and the mode switches to slave receive mode.

■ Acknowledgment

An acknowledgment is sent from the receiver to the sender. When data is received, whether to use acknowledgment can be selected by the setting of the ACK bit. When data is sent, acknowledgment from the receiver is stored in the LRB bit.

If the slave does not receive acknowledgment from the master when sending data, the TRX bit is set to "0" and the mode switches to slave receive mode, thereby enabling the master to generate a stop condition when the slave releases the SCL line.

■ Bus Error

If any of the following occur, a bus error is determined and the I²C interface is halted:

1. A basic specifications error is detected on the I²C bus during data transmission (including the ACK bit).
2. The stop condition is detected in master mode.

19.4.1 Flow of the I²C Interface Transmission

Figure 19.4-1 shows the flow of one-byte transmission from the master to slave.
Figure 19.4-2 shows the flow of one-byte transmission from the slave to master.

■ Flow of the I²C Interface Transmission

Figure 19.4-1 Flow of One-byte Transmission from the Master to Slave

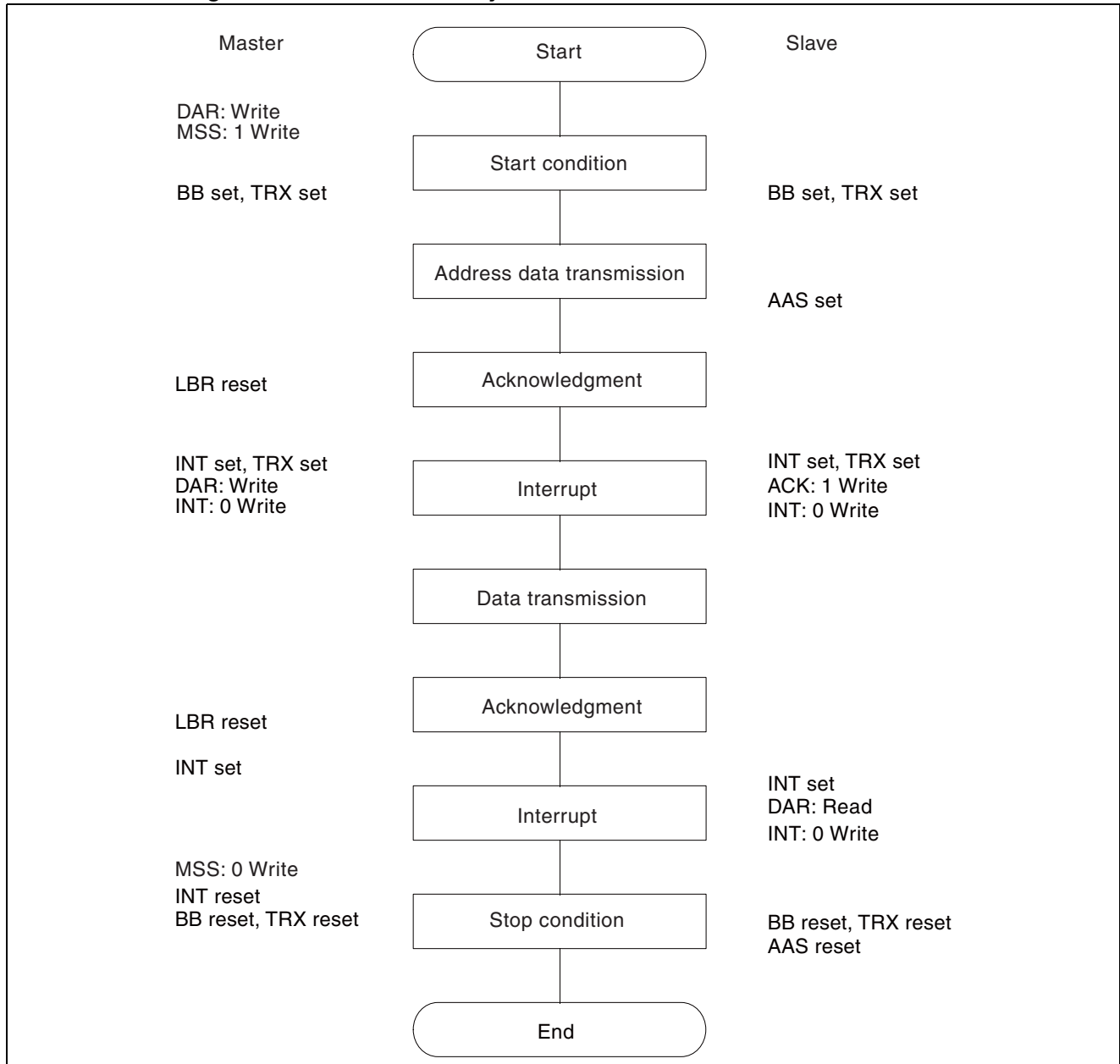
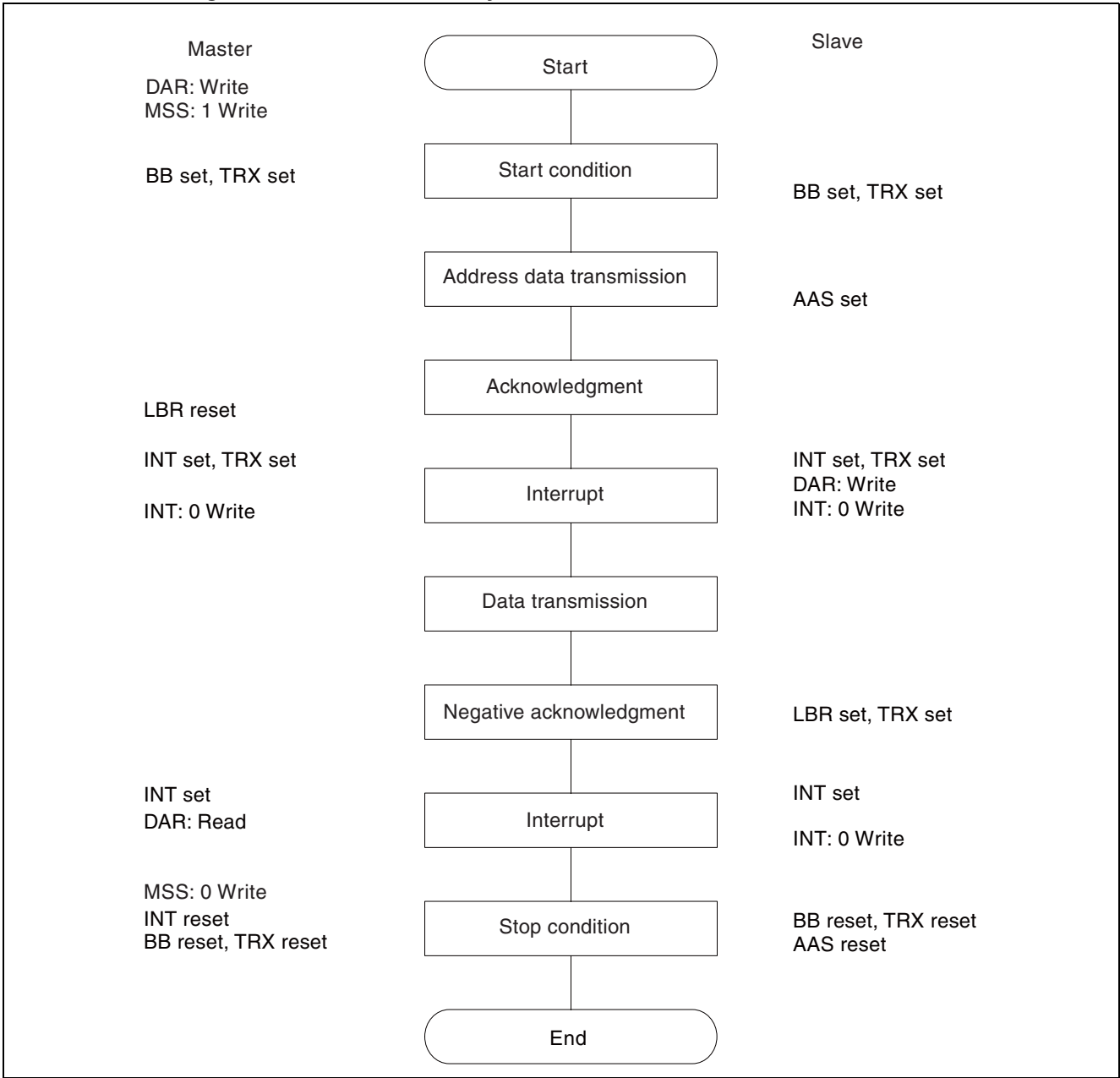


Figure 19.4-2 Flow of One-byte Transmission from the Slave to Master

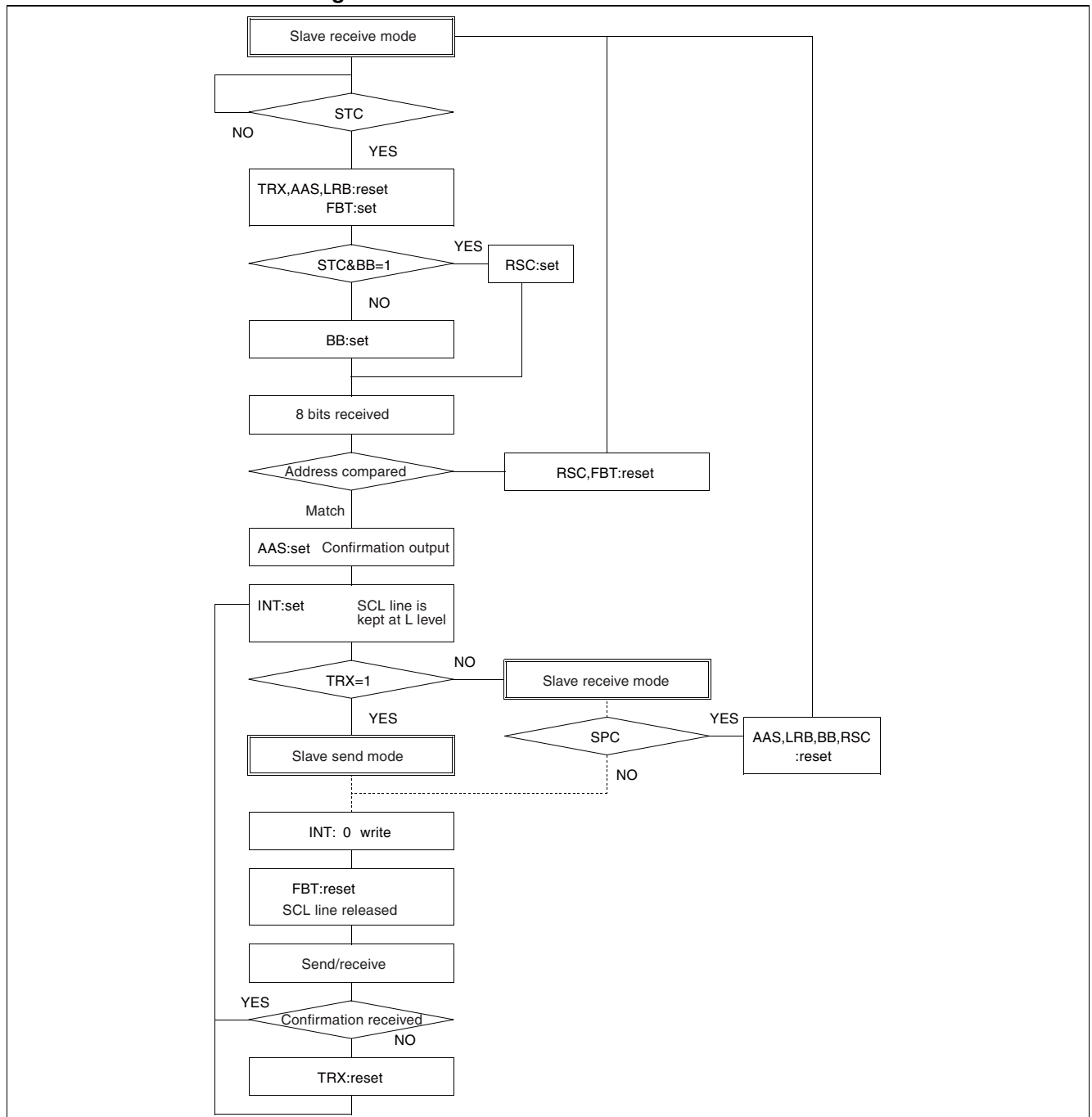


19.4.2 Flow of the I²C Interface Modes

Figure 19.4-3 shows the flow of the I²C interface modes.

■ Flow of the I²C Interface Modes

Figure 19.4-3 Flow of the I²C Interface Modes



CHAPTER 20 CLOCK MONITOR FUNCTION

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

20.1 Overview of the Clock Monitor Functions

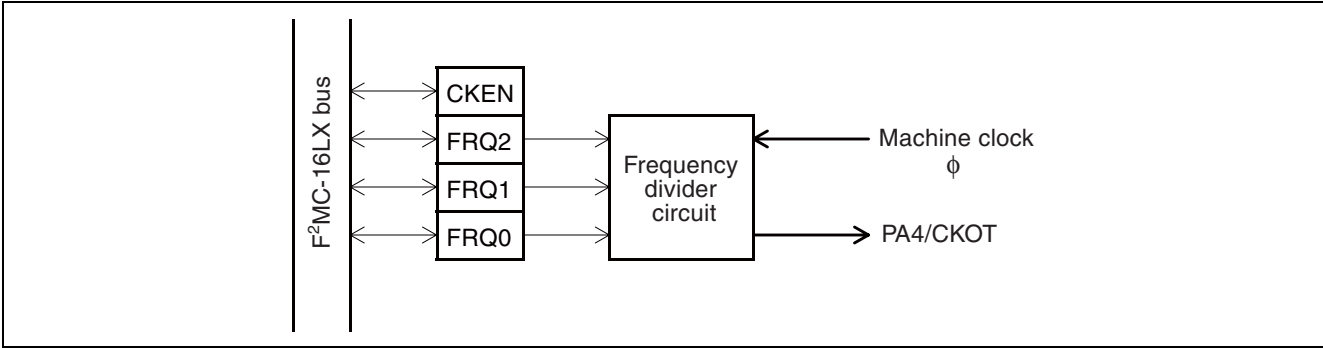
20.2 Clock Output Permission Register (CLKR)

20.1 Overview of the Clock Monitor Functions

The clock monitor function is to output a divided clock of the machine clock (clock for monitoring) from the CKOT pin.

■ Block Diagram of the Clock Monitor Functions

Figure 20.1-1 Block Diagram of the Clock Monitor Functions



20.2 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 20.2-1 Clock Output Permission Register (CLKR)

Clock output permission register	bit	7	6	5	4	3	2	1	0	
Address: 00058 _H		—	—	—	—	CKEN	FRQ2	FRQ1	FRQ0	CLKR
Read/write ⇒		(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒		(-)	(-)	(-)	(-)	(0)	(0)	(0)	(0)	

[bit3] CKEN

The CKEN bit is the CKOT output permission bit.

Table 20.2-1 Functions of CKEN Bit

CKEN	Function
0	Normal port
1	CKOT output

[bit2 to bit0] FRQ2, FRQ1, and FRQ0

The FRQ2, FRQ1, and FRQ0 bits are used to select the clock output frequency.

Table 20.2-2 Functions of the FRQ2, FRQ1, and FRQ0 Bits

FRQ2	FRQ1	FRQ0	Output clock	$\phi = 16 \text{ MHz}$	$\phi = 8 \text{ MHz}$	$\phi = 4 \text{ MHz}$
0	0	0	$\phi/2^1$	125	250	500
0	0	1	$\phi/2^2$	250	500	1
0	1	0	$\phi/2^3$	500	1	2
0	1	1	$\phi/2^4$	1	2	4
1	0	0	$\phi/2^5$	2	4	8
1	0	1	$\phi/2^6$	4	8	16
1	1	0	$\phi/2^7$	8	16	32
1	1	1	$\phi/2^8$	16	32	64

CHAPTER 21 ADDRESS MATCH DETECTION FUNCTION

This chapter describes the address match detection function and operation.

21.1 Overview of the Address Match Detection Function

21.2 Registers of the Address Match Detection Function

21.3 Operation of the Address Match Detection Function

21.4 Example of the Address Match Detection Function

21.5 Program Example of the Address Match Detection Function

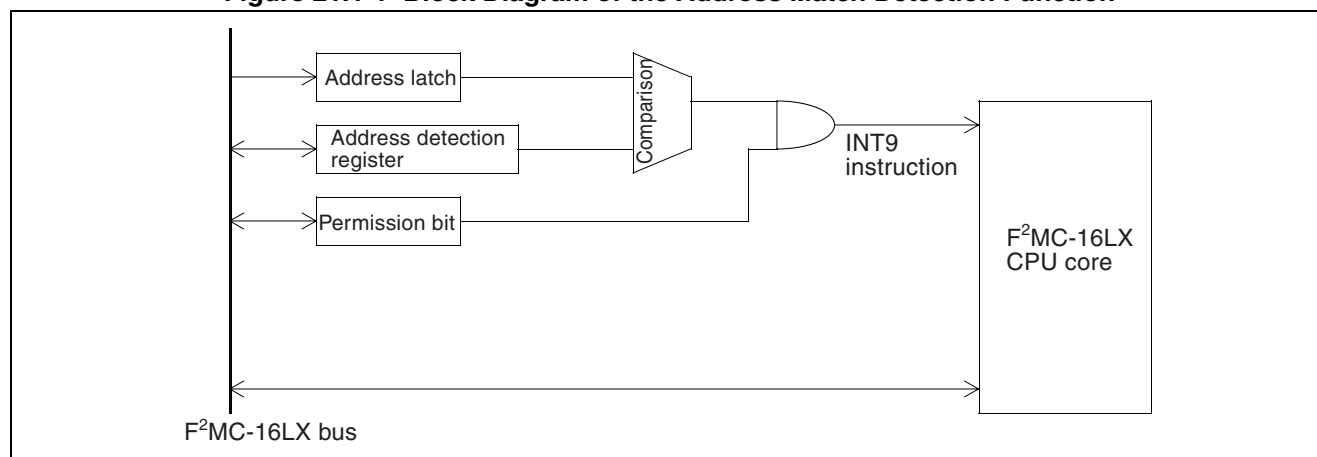
21.1 Overview of the Address Match Detection Function

When an address matches the value set in the address detection register, the instruction code to be read by the CPU is replaced with the INT9 instruction code (01_H). Consequently, the CPU executes the INT9 instruction when executing a specified instruction. Program patching is enabled by processing the INT #9 interrupt routine.

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 21.1-1 Block Diagram of the Address Match Detection Function



21.2 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 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 21.2-1 Program Address Detection Registers (PADR0 and PADR1)

Program address detection registers	byte	byte	byte	Access	Initial value
PADR0 1FF2 _H /1FF1 _H /1FF0 _H				R/W	Not defined
PADR1 1FF5 _H /1FF4 _H /1FF3 _H				R/W	Not defined

Table 21.2-1 lists the correspondence between the program address detection registers (PADR0 and PADR1) and PACSR.

Table 21.2-1 Correspondence between PADR0 and PADR1 Registers and PACSR

Address detection register	Interrupt permission bit
PADR0	AD0E
PADR1	AD1E

■ Program Address Detection Control Register (PACSR)

The program address detection control register (PACSR) controls the operation of the address detection function.

Figure 21.2-2 Program Address Detection Control Register (PACSR)

Program address detection control register	bit 7	6	5	4	3	2	1	0
PACSR 009E _H	Reserved	Reserved	Reserved	Reserved	AD1E	Reserved	AD0E	Reserved
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)

[bit7 to bit4] Reserved bits

bit7 to bit4 are reserved bits. When defining PACSR settings, be sure to set these bits to "0".

[bit3] AD1E (Address detect register 1 enable)

The AD1E bit is the operation permission bit of PADR1.

When this bit is "1", the address is compared with the PADR1 register. If they match, the INT9 instruction is issued.

[bit2] Reserved bit

bit2 is a reserved bit. When defining PACSR settings, be sure to set this bit to "0".

[bit1] AD0E (Address detect register 0 enable)

The AD0E bit is the operation permission bit of PADR0.

When this bit is "1", the address is compared with the PADR0 register. If they match, the INT9 instruction is issued.

[bit0] Reserved bit

bit0 is a reserved bit. When defining PACSR settings, be sure to set this bit to "0".

21.3 Operation of the Address Match Detection Function

When an address matches the value set in the address detection register, the instruction code to be read by the CPU is replaced with the INT9 instruction code (01_H). Consequently, the CPU executes the INT9 instruction when it executes specified instruction. Program patching is enabled by processing the INT #9 interrupt routine.

■ Operation of the Address Match Detection Function

There are two registers for address detection, each with an interrupt permission bit. When an address matches the value set in the address detection register and its interrupt permission bit is set to "1", the instruction code interrupted by the CPU is forcibly replaced with the INT9 instruction code.

■ Note About The Operation of the Address Match Detection Function

The address match detection function is effective only for internal ROM. If an address of external memory area is set, the INT9 instruction will not be issued.

If an address other than the address of the first byte of the instruction is set to the address detection registers, this function will not work normally. Because the set address data is changed to 01_H, a wrong instruction is executed or wrong address is accessed.

Confirm that the interrupt permission bit is "0" before altering the settings of the address detection registers. If the interrupt permission bit is "1", unwanted address detection may occur during the writing, which may result in abnormal operation.

21.4 Example of the Address Match Detection Function

Figure 21.4-1 shows a system configuration example of the address match detection function. Table 21.4-1 lists the E²PROM memory map.

■ System Configuration Example of the Address Match Detection Function

Figure 21.4-1 System Configuration Example of the Address Match Detection Function

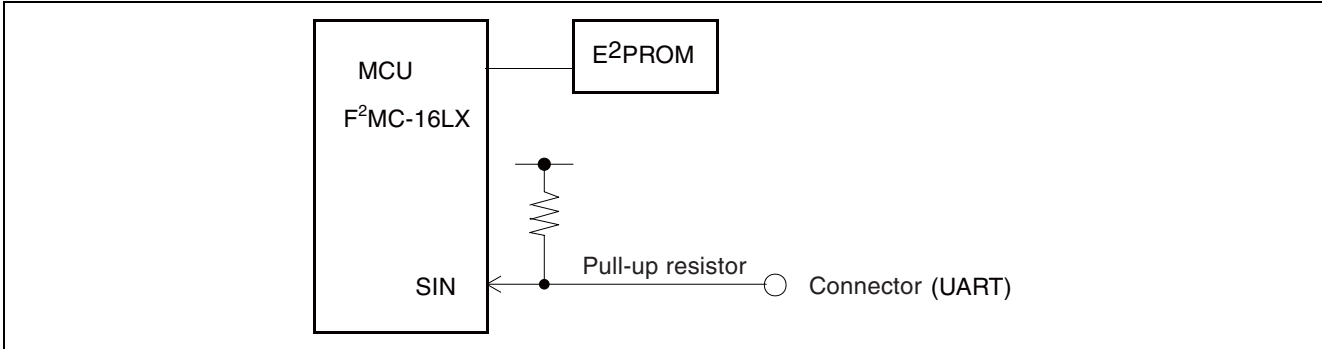


Table 21.4-1 E²PROM Memory Map

Address	Description
0000 _H	Number of bytes of patch program No.0 (If 0, no program error exists.)
0001 _H	Program address No.0 bit7 to bit0
0002 _H	Program address No.0 bit15 to bit8
0003 _H	Program address No.0 bit24 to bit16
0004 _H	Number of bytes of patch program No.1 (If 0, no program error exists.)
0005 _H	Program address No.1 bit7 to bit0
0006 _H	Program address No.1 bit15 to bit8
0007 _H	Program address No.1 bit24 to bit16
Up to 0010 _H plus the number of bytes of patch program No. 0	Main body of patch program No. 0

○ **Initial status**

E²PROM 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 E²PROM.

- **Reset sequence**

The MCU reads the data of E²PROM after reset. If the number of bytes of the patch program is not "0", the MCU reads the main body of the patch program and writes it to RAM. The program address is set to either PADR0 or ADR1, and the operation is allowed. The first address of the program written in RAM is held in the specified RAM of each address detection register.

- **INT9 interrupt**

During the interrupt routine, the address detection that caused the interrupt based on the program counter (PC) value saved in the stack is determined, and it then makes a branch to the block to which an interrupt request is output.

If a branch is made to a program, information that is stacked because of the interrupt becomes invalid.

21.5 Program Example of the Address Match Detection Function

Figure 21.5-1 shows an example of program patch processing, and Figure 21.5-2 shows the flow of program patch processing

■ Program Example of the Address Match Detection Function

Figure 21.5-1 Example of Program Patch Processing

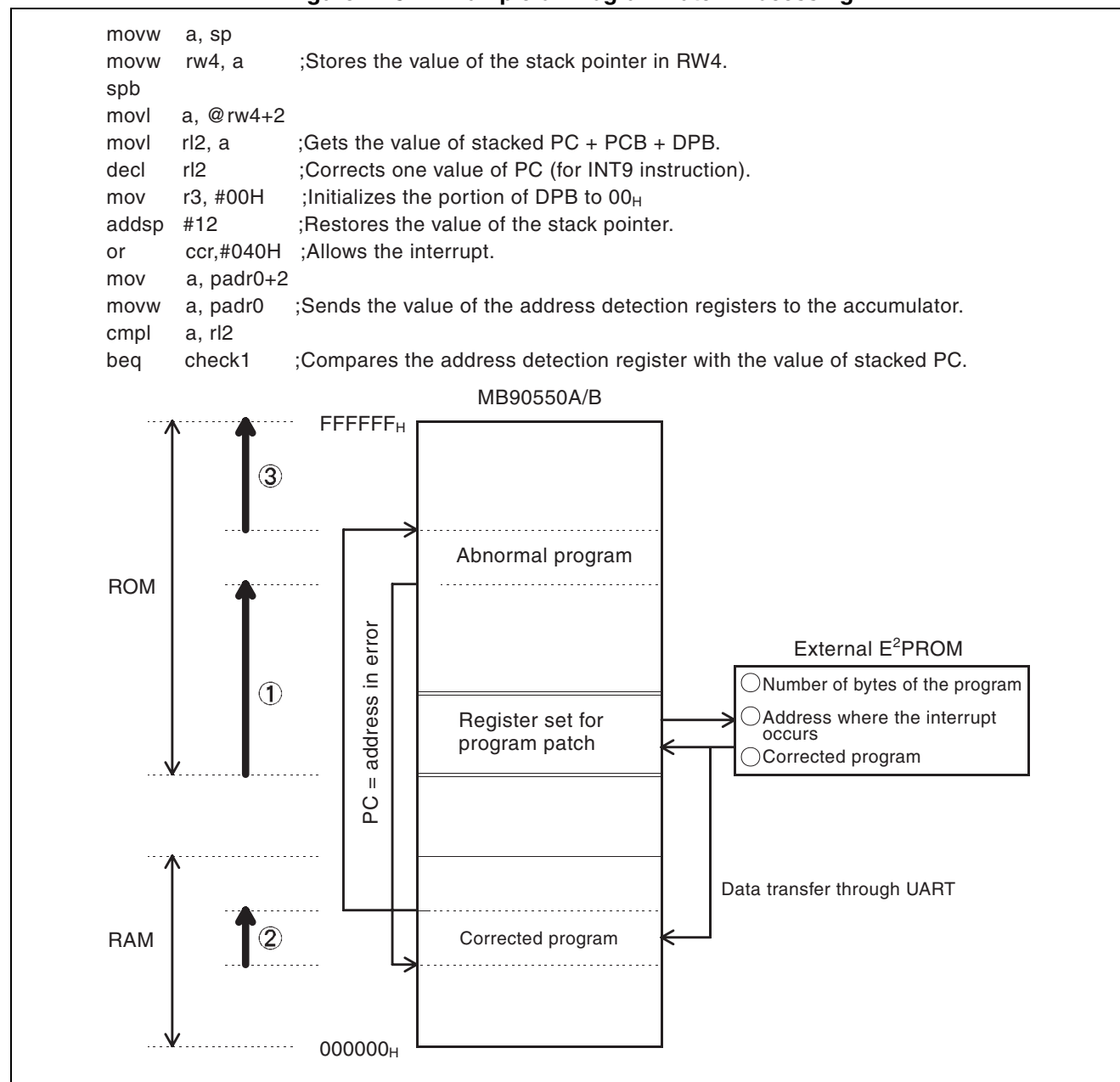
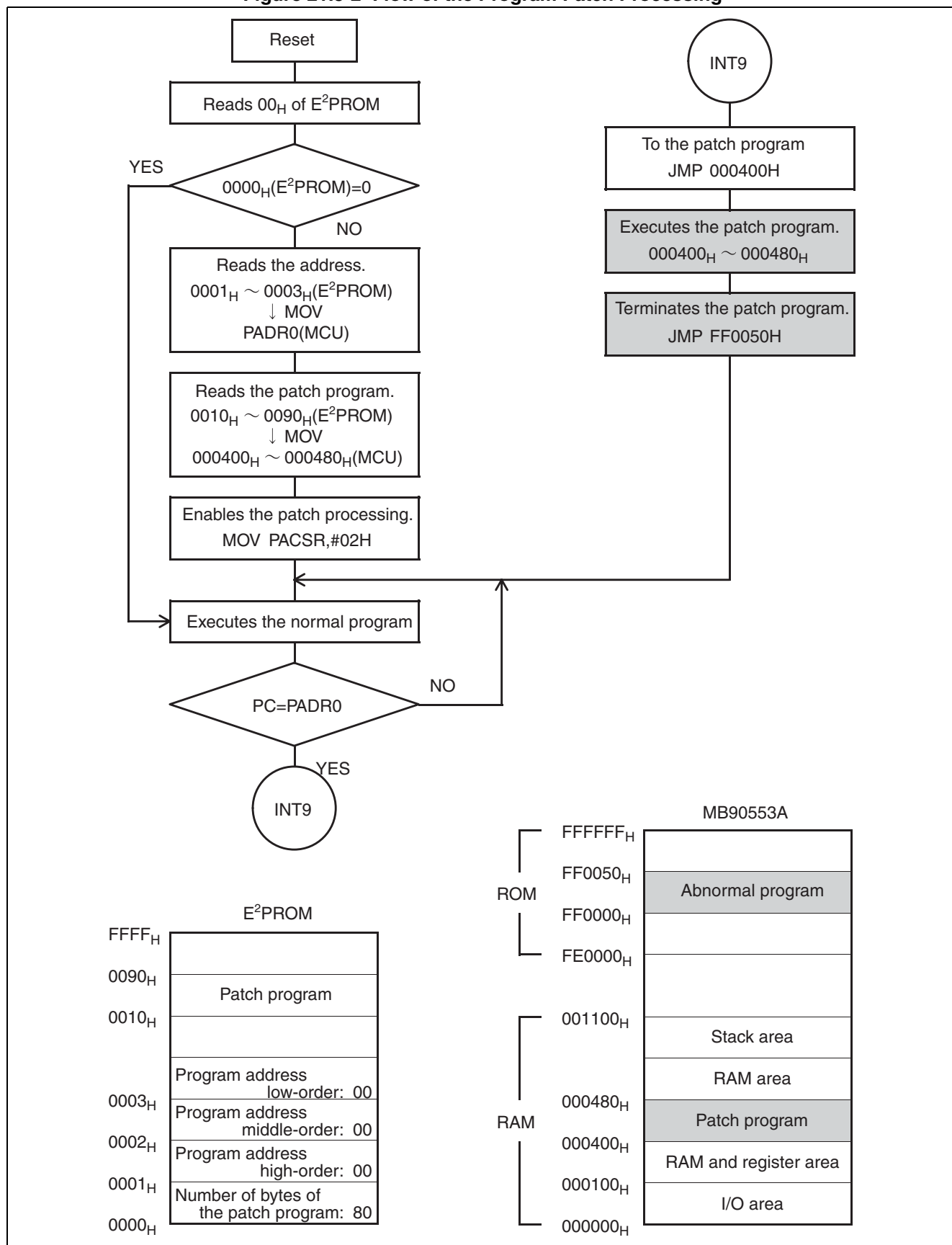


Figure 21.5-2 Flow of the Program Patch Processing



CHAPTER 22 ROM MIRROR FUNCTION SELECTION MODULE

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

22.1 Overview of the ROM Mirror Function Selection Module

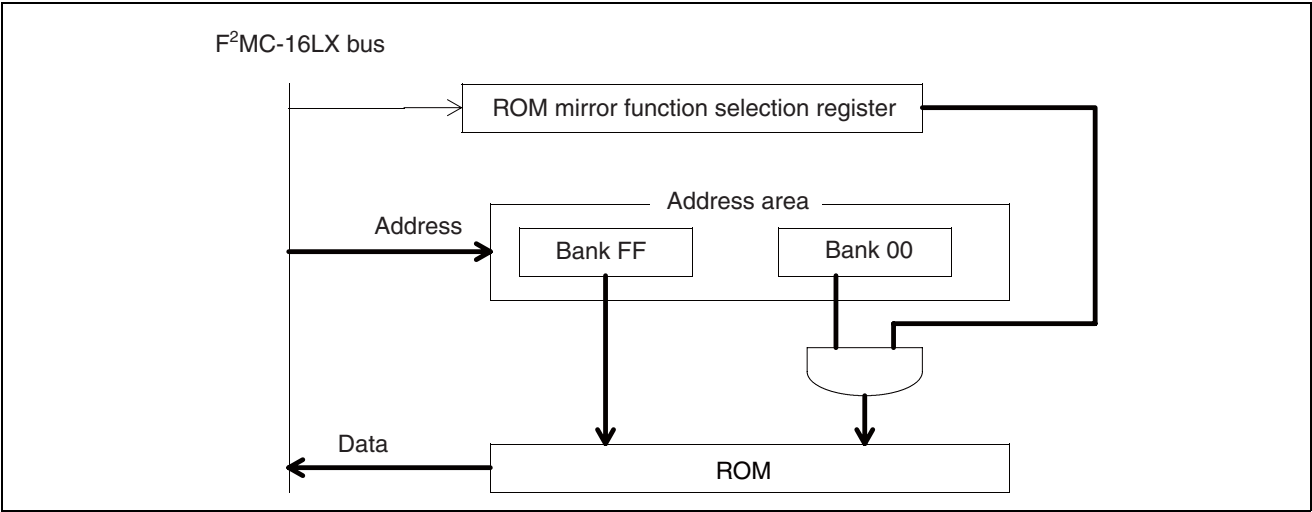
22.2 ROM Mirror Function Selection Register (ROMM)

22.1 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 22.1-1 Block Diagram of the ROM Mirror Function Selection Module



22.2 ROM Mirror Function Selection Register (ROMM)

When "1" is written in the MI bit of the ROM mirror function selection register (ROMM), the ROM data of bank FF can be read in bank 00. When "0" is written, this function is invalid.

■ ROM Mirror Function Selection Register (ROMM)

Figure 22.2-1 ROM Mirror Function Selection Register (ROMM)

ROM mirror function selection module	bit 15	14	13	12	11	10	9	8	
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.

[bit8] MI

When "1" is written in the MI bit, ROM data in bank FF can be read in bank 00. When "0" is written in the MI bit, this function does not work in bank 00. This bit is writable only.

Figure 22.2-2 shows the memory space in the single-chip mode. Figure 22.2-3 shows the memory space in the internal ROM external bus mode.

Note:

Since only addresses FF4000_H to FFFFFFF_H are mirrored to addresses 004000_H to 00FFFF_H in bank 00 when the ROM mirror function is activated, address FF3FFF_H and preceding addresses in ROM are not mirrored in the 00 bank even if the ROM mirror function is set with their values.

Table 22.2-1 Memory Space Address

	MB90552A/B	MB90553A/B	MB90P553A	MB90V550A
Address 1	FF0000 _H	FE0000 _H	FE0000 _H	FE0000 _H
Address 2	000900 _H	001100 _H	001100 _H	001900 _H

Figure 22.2-2 Memory Space in Single-chip Mode

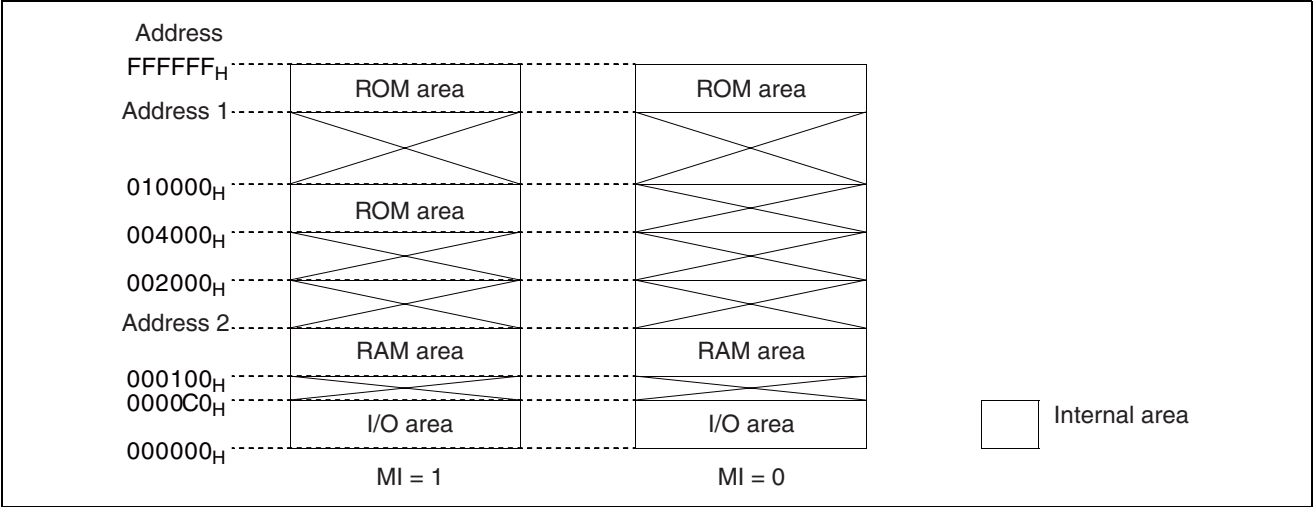
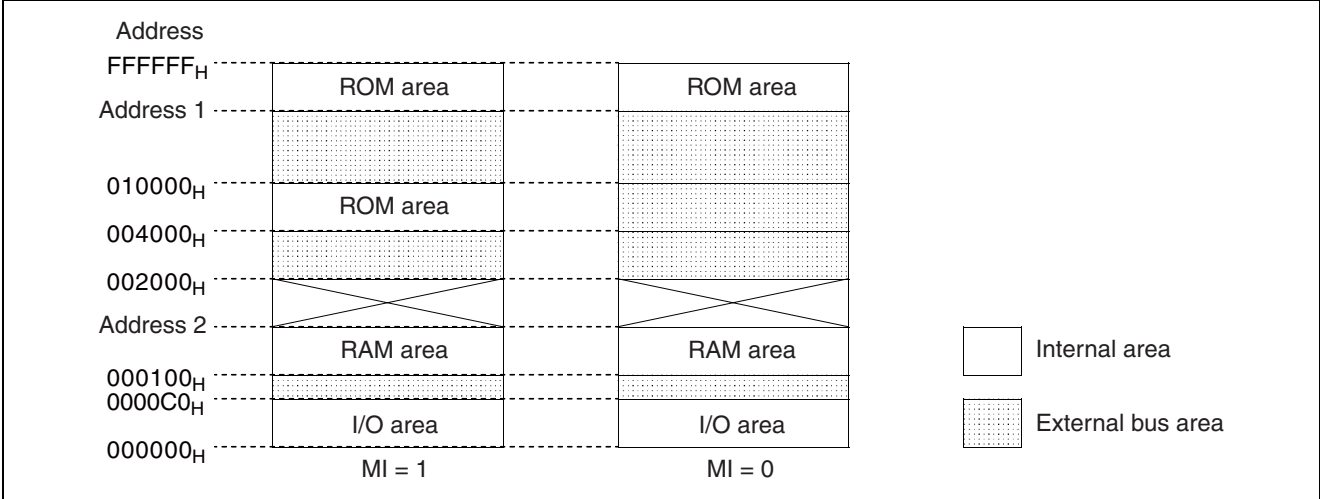


Figure 22.2-3 Memory Space in the Internal ROM External Bus Mode



CHAPTER 23 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:

- 1. Parallel writer**
- 2. Serial-only writer**
- 3. Writing or deletion by program execution**

This chapter describes the above method 3, "Writing or deletion by program execution".

- 23.1 Overview of the 1M-Bit Flash Memory
- 23.2 Block Diagram and Sector Configuration of the Entire Flash Memory
- 23.3 Writing and Deletion Modes
- 23.4 Flash Memory Control Status Register (FMCS)
- 23.5 Activating the Automatic Algorithm of the Flash Memory
- 23.6 Confirming the Automatic Algorithm Execution Status
- 23.7 Detailed Explanations of Flash Memory Writing and Deletion
- 23.8 Example of the 1M-Bit Flash Memory Program

23.1 Overview of the 1M-Bit Flash Memory

The 1M-bit flash memory is allocated in banks FE to FF 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 128K words × 8K or 64K words × 16 bits (16K + 512 × 2 + 7K + 8K + 32K + 64K) sector
- Automatic program algorithm (Embedded Algorithm: Same as for MBM29F400TA)
- 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

Embedded Algorithm™ is a trademark of Advanced Micro Devices, 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)

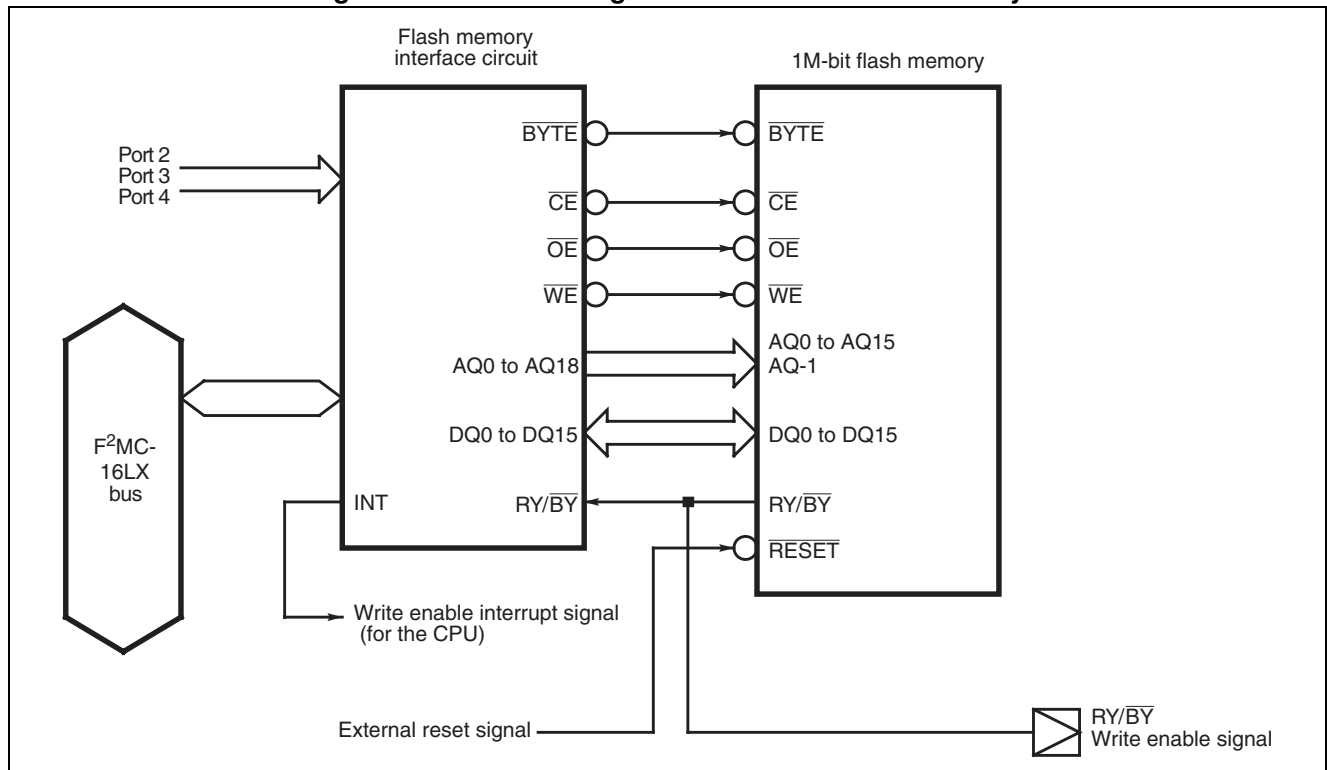
	bit	7	6	5	4	3	2	1	0
Address: 0000AE _H		INTE	RDYINT	WE	RDY	Reserved	—	—	LPM
Read/write		(R/W)	(R/W)	(R/W)	(R)	(W)	(W)	(W)	(R/W)
Initial value		(0)	(0)	(0)	(1)	(0)	(-)	(-)	(0)

23.2 Block Diagram and Sector Configuration of the Entire Flash Memory

Figure 23.2-1 shows the block diagram of the entire flash memory having the flash memory interface circuit. Figure 23.2-2 shows the sector configuration of the flash memory.

■ Block Diagram of the Entire Flash Memory

Figure 23.2-1 Block Diagram of the Entire Flash Memory



■ Sector Configuration of the 1M-bit Flash Memory

Figure 23.2-2 shows the sector configuration of the 1M-bit flash memory and the high-order and low-order address of each sector.

For access from the CPU, SA0 is stored in the FE bank register and SA1 to SA6 are stored in the FF bank register.

Figure 23.2-2 Sector Configuration of the 1M-bit Flash Memory

Flash memory	CPU address	Writer address *
SA6 (16K bytes)	FFFFFF _H	7FFFF _H
	FFC000 _H	7C000 _H
SA5 (512 bytes)	FFBFFF _H	7BFFF _H
	FFBE00 _H	7BE00 _H
SA4 (512 bytes)	FFBDFF _H	7BDFF _H
	FFBC00 _H	7BC00 _H
SA3 (7K bytes)	FFBBFF _H	7BBFF _H
	FFA000 _H	7A000 _H
SA2 (8K bytes)	FF9FFF _H	79FFF _H
	FF8000 _H	78000 _H
SA1 (32K bytes)	FF7FFF _H	77FFF _H
	FF0000 _H	70000 _H
SA0 (64K bytes)	FEFFFF _H	6FFFF _H
	FE0000 _H	60000 _H

*: A writer address is associated with a CPU address when data is written in the flash memory by the parallel writer. This address is used to perform writing or deletion using the general-purpose writer.

23.3 Writing and Deletion Modes

To access the flash memory, the following two methods are used: flash memory mode and other modes. In the flash memory mode, writing and deletion are possible directly from external pins. In other modes, writing and deletion are possible from the CPU via the internal bus. A mode is selected by the mode external pin.

■ Flash Memory Mode

If the generated reset signal sets the mode pin to "111_B", the CPU stops.

Because the flash memory interface circuit is connected directly to ports 0, 2, 3, and 4; it can be controlled directly from the external pins. In this mode, the MCU performs the same operation as that of the standard flash memory in external pins, and writing and deletion are possible using the flash memory programmer.

In this mode, all operations supported by the automatic algorithm of the flash memory can be used.

■ Other Modes

The flash memory is allocated in banks FC to FF of the CPU memory space. As with the ordinary mask ROM, it can be subjected to a read access and program access from the CPU through the flash memory interface circuit.

Because writing and deletion for the flash memory are executed by instructions from the CPU through the flash memory interface circuit, other modes enable data to be written again even if the MCU is soldered to the target board.

In these modes, the sector protect operation cannot be executed.

■ Flash Memory Control Signals

Table 23.3-1 shows the flash memory control signals in the flash memory mode.

There is almost one-to-one correspondence between flash memory control signals and external pins of MBM29F400TA. The V_{ID} (12V) pin required for the sector protect operation are MD0, MD1, and MD2, instead of A9, \overline{RESET} , and \overline{OE} of MBM29F400TA.

Because the memory size of MB90F553A is 1/4 of MBM29F400TA, pins AQ18 and AQ17 associated with address signals A17 and A16 of MBM29F400TA are redundant and must always be set to "1".

In the flash memory mode, the width of the external data bus is limited to eight bits to allow only a one-byte access. DQ15 to DQ8 are not supported. The pin \overline{BYTE} must always be set to "0".

Table 23.3-1 Flash Control Signals

MB90F553A			MBM29F400TA
Pin No.	Ordinary function	Flash memory mode	
1 to 8	P20 to P27	AQ0 to AQ7	A-1, A0 to A6
9	P30	AQ16	A15
10	P31	\overline{CE}	\overline{CE}
12	P32	\overline{OE}	\overline{OE}
13	P33	\overline{WE}	\overline{WE}
14, 15	P34, P35	AQ17, AQ18	A16, A17
16	P36	\overline{BYTE}	\overline{BYTE}
17	P37	RY/ \overline{B}	RY/ \overline{B}
18 to 22	P40 to P44	AQ8 to AQ12	A7 to A11
24 to 26	P45 to P47	AQ13 to AQ15	A12 to A14
49	MD0	MDO	A9 (V_{ID})
50	MD1	MD1	\overline{RESET} (V_{ID})
51	MD2	MD2	\overline{OE} (V_{ID})
85 to 92	P00 to P07	DQ0 to DQ7	DQ0 to DQ7
77	\overline{RST}	\overline{RESET}	\overline{RESET}
Unusable			DQ8 to DQ15

23.4 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	7	6	5	4	3	2	1	0
Address: 0000AE _H		INTE	RDYINT	WE	RDY	Reserved	—	—	LPM
Read/write		(R/W)	(R/W)	(R/W)	(R)	(W)	(W)	(W)	(R/W)
Initial value		(0)	(0)	(0)	(1)	(0)	(-)	(-)	(0)

○ Bits

[bit7] 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.

[bit6] 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 "23.5 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)

[bit5] 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 "23.5 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.

[bit4] 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).

[bit3] Reserved bit

Reserved for testing. Set this bit to "0" for normal use.

[bit2, bit1] Free bits

Set these bits to "0" for normal use.

[bit0] LPM (Low power mode)

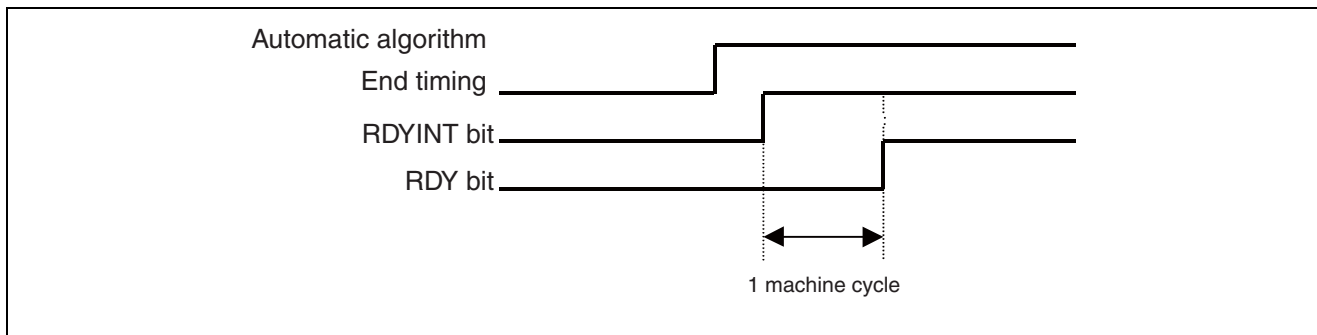
When this bit is set to "1", the select signal to the flash memory at flash memory access is minimized and power consumption of the flash memory is suppressed. However, because access time is greater than that at LPM = 0, memory access is impossible during high-speed operation of the CPU. To use this mode, operate the CPU with a frequency of 4 MHz or less.

0: Ordinary power consumption mode

1: Low-power consumption mode (operation with internal operation frequency 4 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.



23.5 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 23.5-1 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 23.5-1 Command Sequence

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).											

Notes:

- 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 "■Sector Configuration of the 1M-bit 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.

23.6 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 4-bit outputs DQ7, DQ6, DQ5, and DQ3, which have functions of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit excess flag (DQ5), and sector deletion timer flag (DQ3), 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 23.5-1 in Section "23.5 Activating the Automatic Algorithm of the Flash Memory") is set. Table 23.6-1 shows the bit allocation for the hardware sequence flags.

Table 23.6-1 Bit Allocation for the Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

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 status 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 23.6-2 lists the hardware sequence flag functions.

Table 23.6-2 Hardware Sequence Flag Functions

Status		DQ7	DQ6	DQ5	DQ3
Status change at normal operation	Writing --> writing completion (At write address specification)	$\overline{\text{DQ7}}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3
	Chip or sector deletion --> deletion completion	0 --> 1	Toggle --> Stop	0 --> 1	1
	Sector deletion wait --> deletion start	0	Toggle	0	0 --> 1
	Deletion --> temporary stop of sector deletion (Sector being deleted)	0 --> 1	Toggle --> 1	0	1 --> 0
	Temporary stop of sector deletion --> deletion restart (Sector being deleted)	1 --> 0	1 --> Toggle	0	0 --> 1
	Temporary stop of sector deletion ongoing (Sector not being deleted)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Writing	$\overline{\text{DQ7}}$	Toggle	1	0
	Chip or sector deletion operation	0	Toggle	1	1

23.6.1 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 23.6-3 and Table 23.6-4 list the transitions of data polling flag statuses.

Table 23.6-3 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 23.6-4 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 bit7 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 bit7 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 bit7 (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.

23.6.2 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 23.6-5 and Table 23.6-6 show status transitions of the toggle bit flag.

Table 23.6-5 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 23.6-6 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 bit6 and outputs bit6 (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 bit6 (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 2μs is performed.

At deletion, if all selected sectors are protected against rewriting, the toggle bits involve the toggle operation of about 100μs, and then the read or reset status is returned without rewriting data.

23.6.3 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 23.6-7 and Table 23.6-8 show status transitions of the timing limit excess flag.

Table 23.6-7 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 23.6-8 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.

23.6.4 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 23.6-9 and Table 23.6-10 show status transitions of the sector deletion timer flag.

Table 23.6-9 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 23.6-10 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 temporary stop of 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 bit3 (DATA:3) of the read value of the indicated address.

23.7 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 23.5-1 in Section "23.5 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

23.7.1 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 23.5-1 in Section "23.5 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.

23.7.2 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 23.5-1 in Section "23.5 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

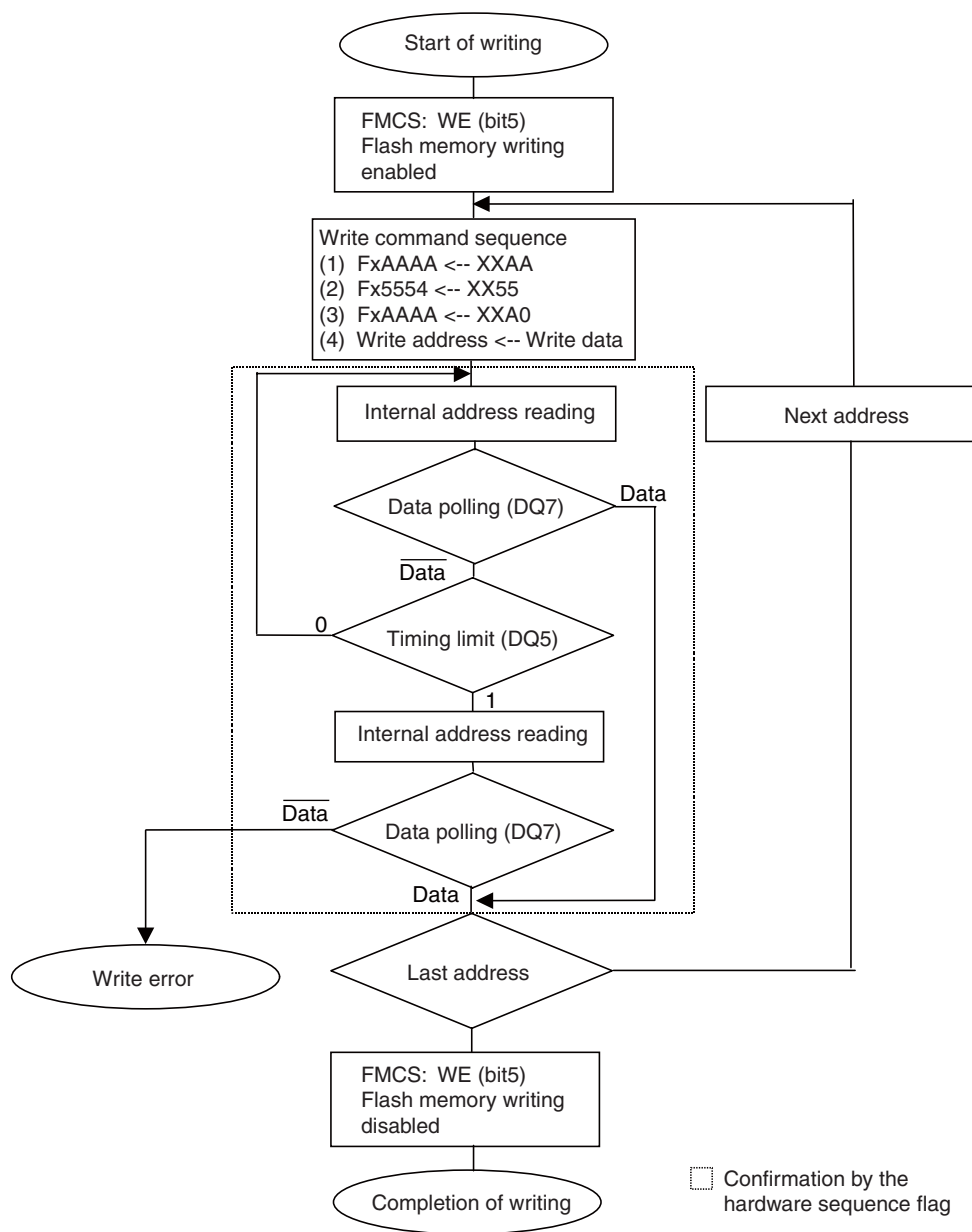
Figure 23.7-1 shows an example of the flash memory writing. The status of the automatic algorithm in the flash memory can be determined using the hardware sequence flag (see Section "23.6 Confirming the Automatic Algorithm Execution Status"). The data polling flag (DQ7) is used to confirm the end of 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 23.7-1 Example of the Procedure for Flash Memory Writing



23.7.3 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 23.5-1 in Section "23.5 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.

23.7.4 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 23.5-1 in Section "23.5 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 (30_H) into any even address that can be accessed within the target sector. To delete two or more sectors, write the deletion code (30_H) 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.

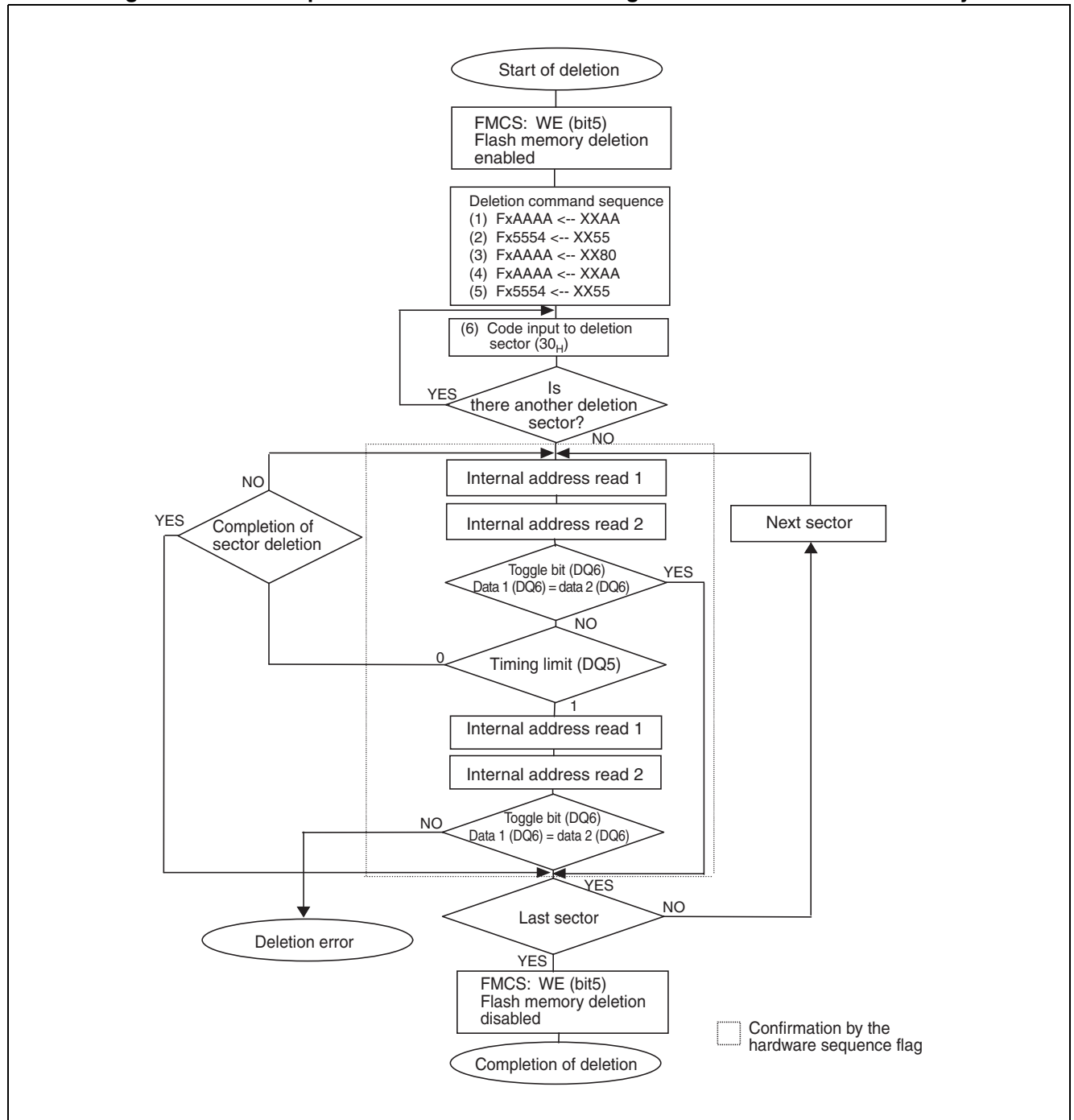
■ Procedure for Deleting a Sector from the Flash Memory

The status of the automatic algorithm within the flash memory can be determined using the hardware sequence flag (see Section "23.6 Confirming the Automatic Algorithm Execution Status"). Figure 23.7-2 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 23.7-2 Example of the Procedure for Deleting a Sector from the Flash Memory

23.7.5 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 sending sector deletion temporary stop commands, listed in the command sequence table (see Table 23.5-1 in Section "23.5 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 (B0_H). 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.

23.7.6 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 sending sector deletion restart commands, listed in the command sequence table (see Table 23.5-1 in Section "23.5 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 (30_H). However, any address in the flash memory area must be indicated.

The issuance of the sector deletion restart command is ignored during sector deletion.

23.8 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
                                   ;segment

          ORG      0000H
PDR0      RB       1
PDR1      RB       1
PDR2      RB       1
PDR3      RB       1
          ORG      0010H
DDR0      RB       1
DDR1      RB       1
DDR2      RB       1
DDR3      RB       1
          ORG      00A1H
CKSCR     RB       1
          ORG      00AEH
FMCS      RB       1
          ORG      006FH
ROMM      RB       1
RESOUS    ENDS
;

```

```

SSTA      SSEG
          RW      0127H
STA_T     RW      1
SSTA      ENDS
;
DATA      DSEG    ABS=0FFH          ;FLASH command address
          ORG      5554H
COMADR2   RW      1
          ORG      0AAAAH
COMADR1   RW      1
DATA      ENDS
;////////////////////////////////////
;Main program (FFA000H)
;////////////////////////////////////
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
;
;      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
;////////////////////////////////////
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,
MOVW    A,@RW2+00           ;AL for each reading,
;from DQ6 during
;writing.
XORW    A                   ;XOR of AH and AL (1 writing
; (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)
BZ      ELS                ;Is sector deletion ended?
MOV     FMCS,#00H          ;Release of the FLASH
;deletion mode
RETP                      ;Return to the main program

```

```

;////////////////////////////////////////
;Error
;////////////////////////////////////////
ERROR    MOV      FMCS,#00H          ;Release of the FLASH mode
          MOV      PDR0,#0FFH        ;Confirmation of the error
                                           ;handling
          MOV      ADB:COMADR1,#0F0H ;Reset command (reading
                                           ;possible)
          RETP                          ;Return to the main program
RAMPRG    ENDS
;////////////////////////////////////////
VECT      CSEG      ABS=0FFH
          ORG        0FFDCH
          DSL        START
          DB          00H
VECT      ENDS
;
          END        START

```

CHAPTER 24 EXAMPLE OF MB90F553A SERIAL PROGRAMMING CONNECTION

This chapter provides examples of serial programming connections using the AF220/AF210/AF120/AF110 flash microcontroller programmer manufactured by Yokogawa Digital Computer Co., Ltd.

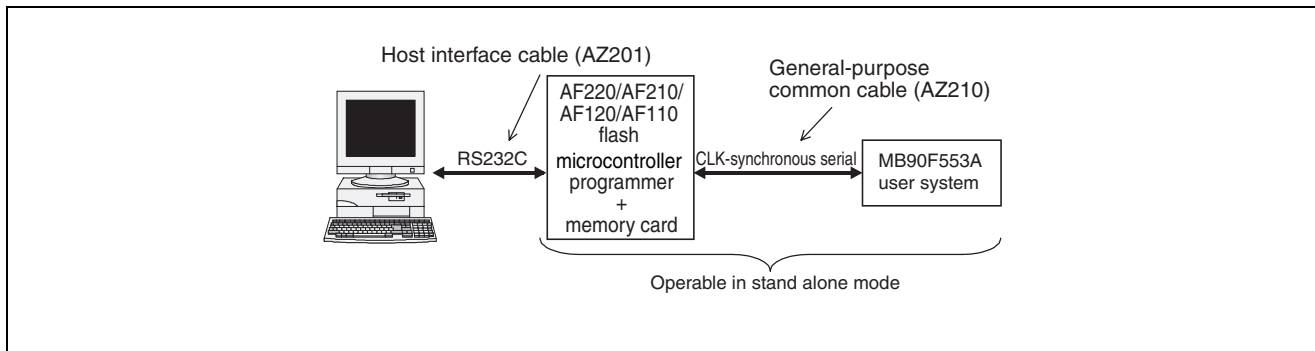
- 24.1 Basic Configuration of MB90F553A Serial Programming Connection
- 24.2 Example of Serial Programming Connection (When User Power Supply Is Used)
- 24.3 Example of Serial Programming Connection (When Power Is Supplied from a Writer)
- 24.4 Example of Minimal Connection with the Flash Microcontroller Programmer (When User Power Supply Is Used)
- 24.5 Example of Minimal Connection with the Flash Microcontroller Programmer (When Power Is Supplied from a Writer)

24.1 Basic Configuration of MB90F553A Serial Programming Connection

The MB90F553A supports flash ROM serial onboard programming (Fujitsu standard). This section describes the specifications.

■ Basic Configuration of MB90F553A Serial Programming Connection

The AF220/AF210/AF120/AF110 flash microcontroller programmer manufactured by Yokogawa Digital Computer Co., Ltd. is used for Fujitsu standard serial onboard programming.



Note:

For the function and operation method of the AF220/AF210/AF120/AF110 flash microcontroller programmer and the general-purpose common cable (AZ210) and connector, contact Yokogawa Digital Computer Co., Ltd.

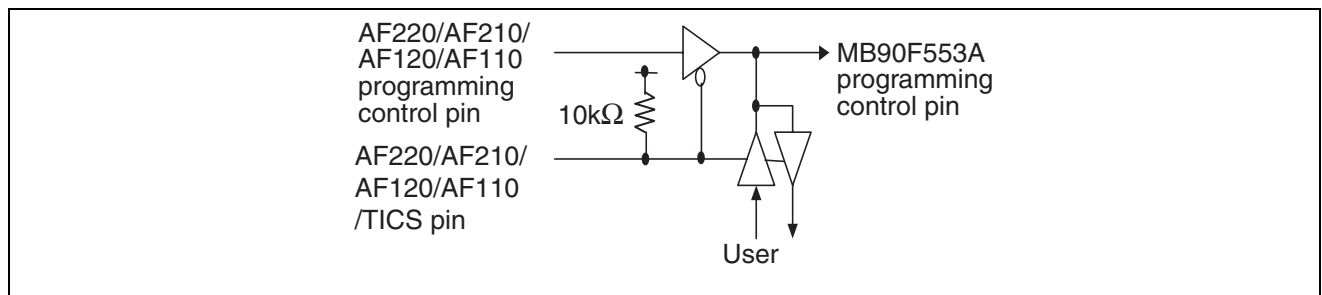
Table 24.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming (1/2)

Pin	Function	Description
MD2, MD1 MD0	Mode pins	Programming mode is controlled from the flash microcontroller 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 1 to 16 MHz.
P00, P01	Programming program activation pin	-
$\overline{\text{RST}}$	Reset pin	-
SIN	Serial data input pin	UART is used as CLK synchronization mode.
SOT	Serial data output pin	
SCK	Serial clock pulse input pin	
C	C pin	Capacitor pin for power supply stabilization. Connect a ceramic capacitor of about 0.1 μ F externally.

Table 24.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming (2/2)

Pin	Function	Description
VCC	Power supply voltage supply pin	Connection with the flash microcontroller 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.
VSS	GND pin	Common to the GND of the flash microcontroller programmer.
$\overline{\text{HST}}$	Hardware standby pin	Input the H level in serial programming mode.

If the P00, SIN, SOT, and SCK 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 microcontroller programmer during serial programming.



Sections 24.2 to 24.5 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 writer)
- Example of minimal connection with the flash microcontroller programmer (when user power supply is used)
- Example of minimal connection with the flash microcontroller programmer (when power is supplied from a writer)

**Table 24.1-2 System Configuration of the Flash Microcontroller Program
(Yokogawa Digital Computer Co., Ltd.)**

Type		Function
Main unit	AF200/AC4P	Model with built-in Ethernet interface/100V to 220V power adapter
	AF210/AC4P	Standard model/100V to 220V power adapter
	AF120/AC4P	Single-key model with built-in Ethernet interface/100V to 220V power adapter
	AF110/AC4P	Single-key model/100V to 220V power adapter
AZ221		Writer-dedicated RS232C cable for PC/AT
AZ210		Standard target probe (a) Length: 1 m
FF201		Fujitsu control module for F ² MC-16LX flash microcontroller
AZ290		Remote controller
/P2		2MB PC Card (option) Flash memory capacity: up to 128 KB supported
/P4		4MB PC Card (option) Flash memory capacity: up to 512 KB supported

Contact address: Yokogawa Digital Computer Co., Ltd.

Phone: (81)-42-333-6224

Note:

AF200 flash microcontroller programmer is a product that is no longer being manufactured, but it can be used with control module FF201. Section "24.2 Example of Serial Programming Connection (When User Power Supply Is Used)" shows an example of a serial programming connection.

■ **Oscillating Clock Frequency and Serial Clock Input Frequency**

A formula to calculate the serial clock frequency that can be input for MB90F553A is shown below. Set the serial clock input frequency to the flash microcontroller programmer, adjusting it to the oscillating clock frequency.

Serial clock frequency that can be input = $0.125 \times$ oscillating clock frequency

Table 24.1-3 Examples of Serial Clock Frequency That can be Input

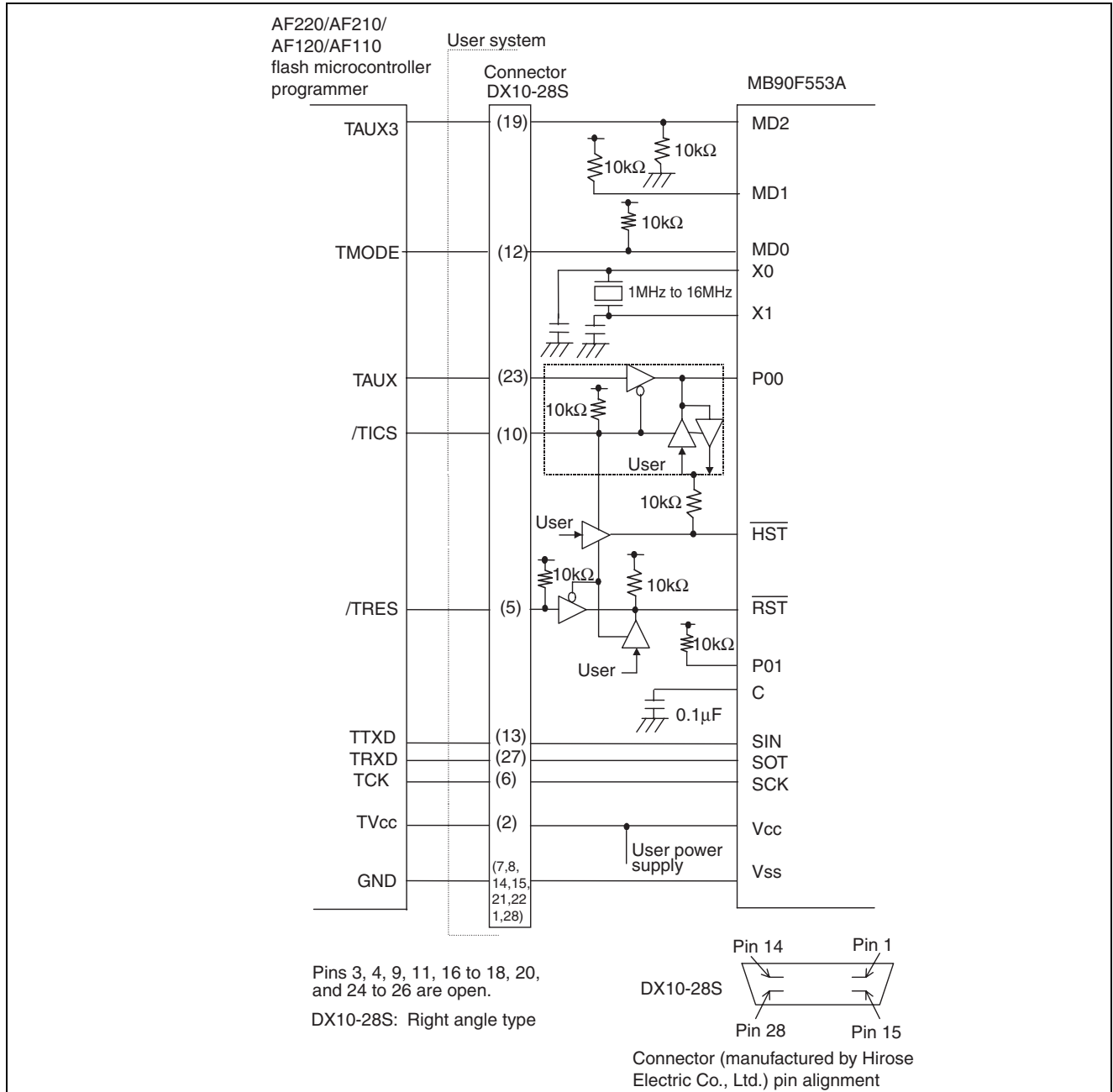
Oscillating clock frequency	Maximum microcontroller serial clock frequency that can be input	Maximum serial clock frequency that can be set in AF220/AF210/AF120/AF110	Maximum serial clock frequency that can be set in AF200
4 MHz	500 kHz	500 kHz	500 kHz
8 MHz	1 MHz	850 kHz	500 kHz
16 MHz	2 MHz	1.25 MHz	500 kHz

24.2 Example of Serial Programming Connection (When User Power Supply Is Used)

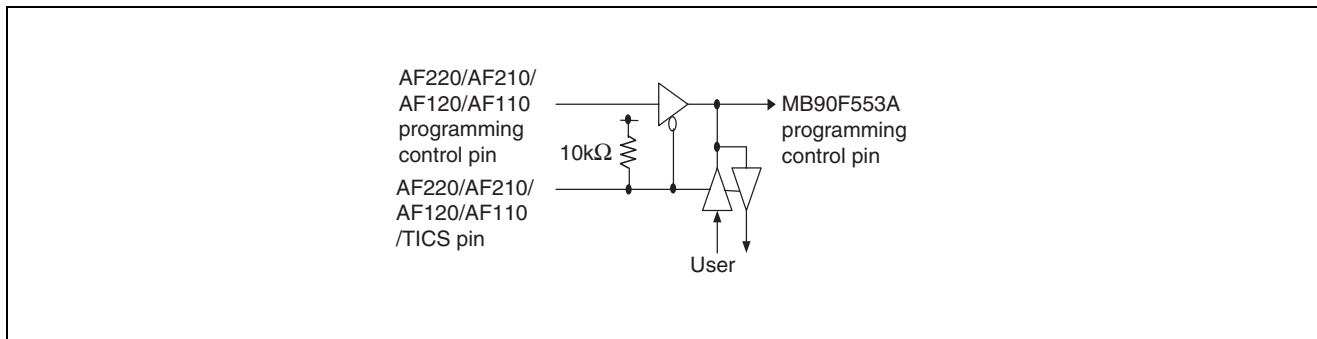
Figure 24.2-1 shows an example of serial programming connection when the power supply voltage of the microcontroller is supplied from the user power supply. Mode pins MD2, MD1, and MD0 are set to "011" to set internal vector modes (single-chip mode, internal ROM external bus mode).

■ Example of Serial Programming Connection (when User Power Supply is Used)

Figure 24.2-1 Example of MB90F553A 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 SIN, SOT, and SCK pins are also used by the user system. (The user circuit can be disconnected by the /TICS signal of the flash microcontroller programmer during serial programming.)



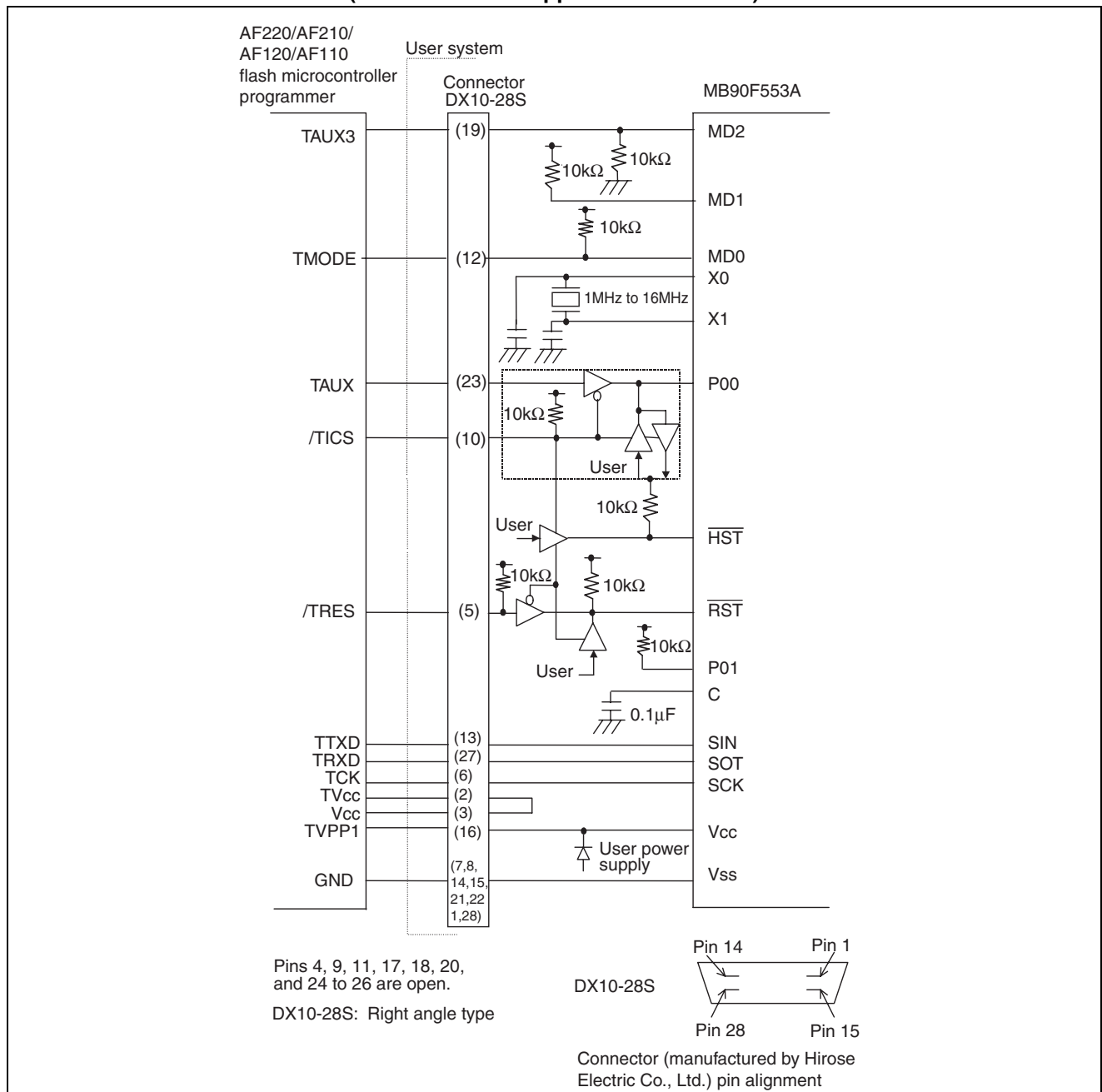
- Connect the AF220/AF210/AF120/AF110, turning the user power supply off.

24.3 Example of Serial Programming Connection (When Power Is Supplied from a Writer)

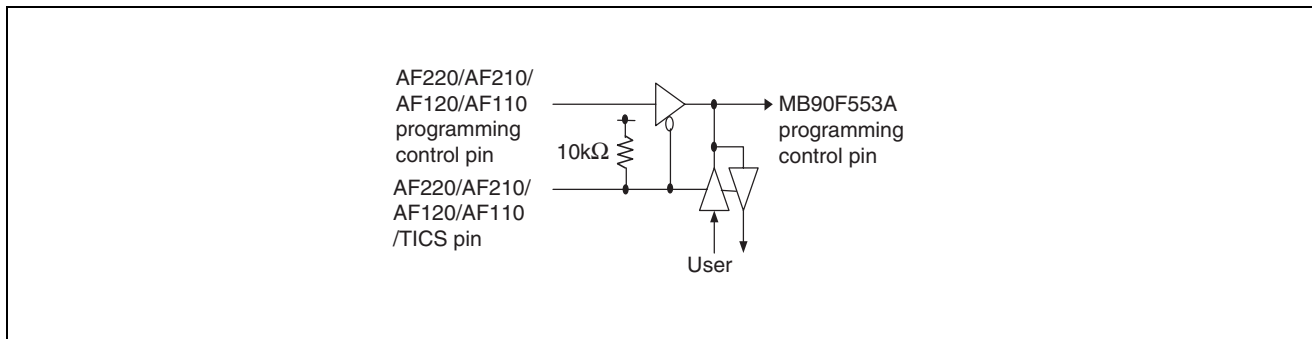
Figure 24.3-1 shows an example of serial programming connection when the power supply voltage of the microcontroller is supplied from writer power supply. Mode pins MD2, MD1, and MD0 are set to "011" to set internal vector modes (single-chip mode, internal ROM external bus mode).

■ Example of Serial Programming Connection (when Power is Supplied from a Writer)

**Figure 24.3-1 Example of MB90F553A Serial Programming Connection
(when Power is Supplied from a Writer)**



- As with the case in which the P00 pin is also used, the following control circuit is required when the SIN, SOT, and SCK pins are also used by the user system. (The user circuit can be disconnected by the /TICS signal of the flash microcontroller 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.

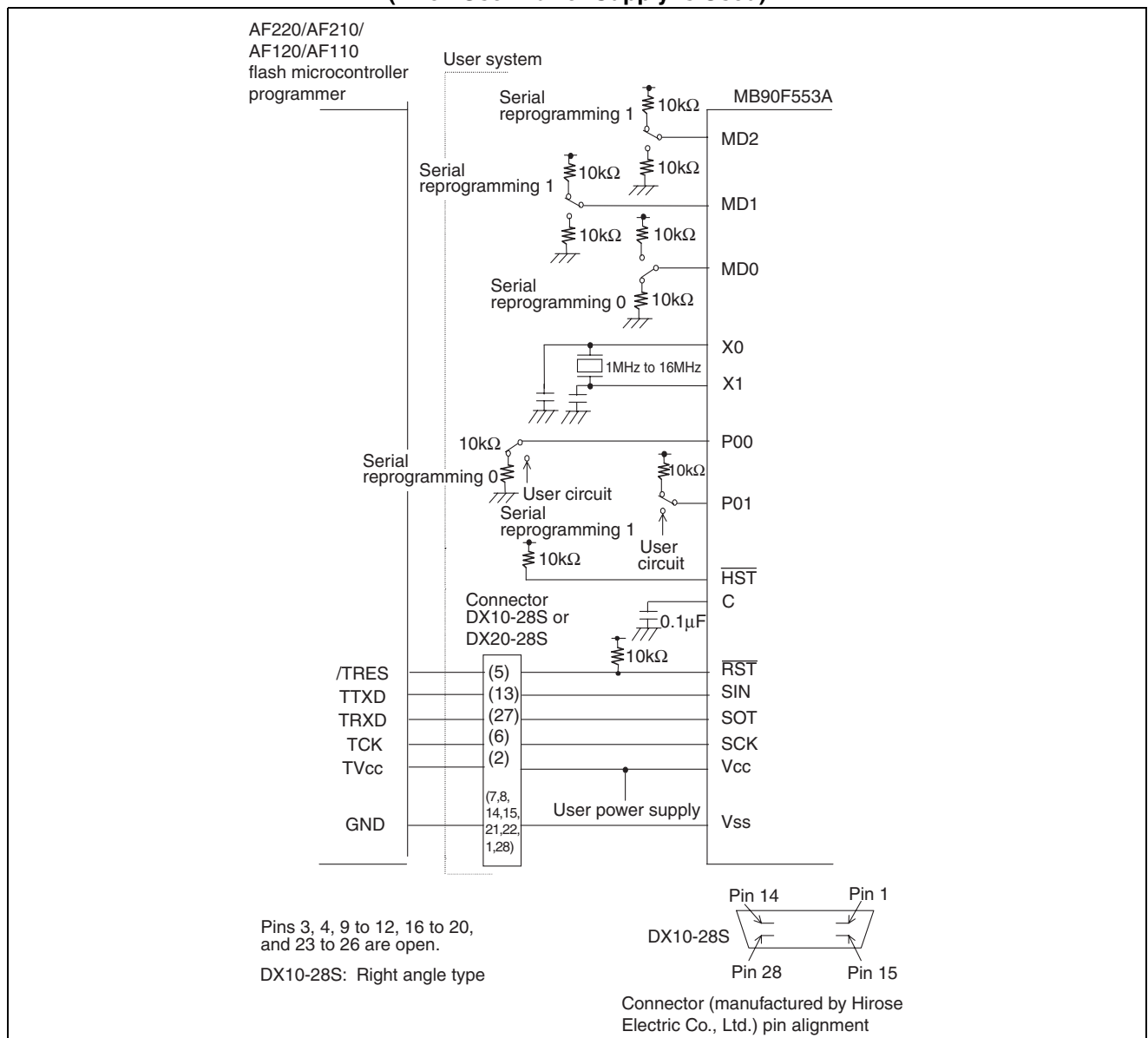
24.4 Example of Minimal Connection with the Flash Microcontroller Programmer (When User Power Supply Is Used)

Figure 24.4-1 shows an example of minimal connection with the flash microcontroller programmer when the power supply voltage of the microcontroller is supplied from the user power supply.

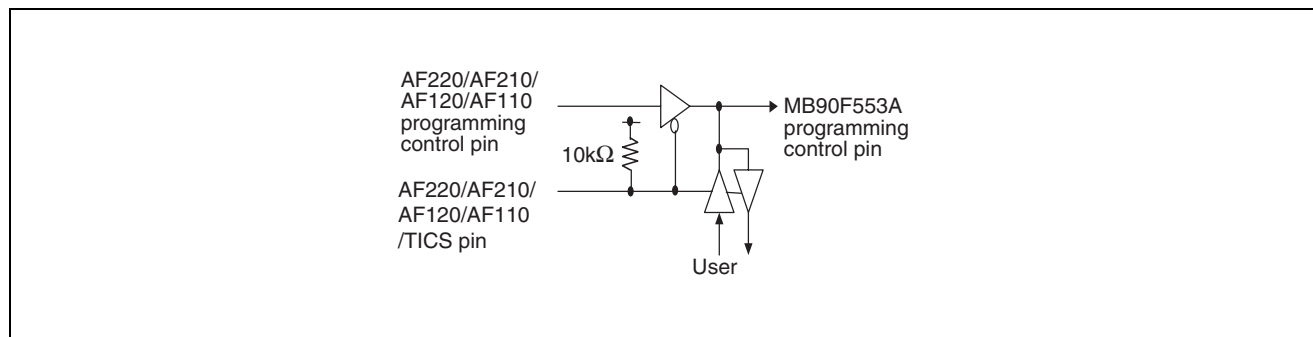
■ Example of Minimal Connection with the Flash Microcontroller Programmer (when User Power Supply is Used)

The MD2, MD1, MD0, and P00 pins need not be connected to the flash microcontroller programmer if each pin is connected, as shown below, for flash memory programming.

Figure 24.4-1 Example of Minimal Connection with Flash Microcontroller Programmer (when User Power Supply is Used)



- When the SIN, SOT, and SCK 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 microcontroller programmer during serial programming.)



- Connect the AF220/AF210/AF120/AF110, turning the user power supply off.

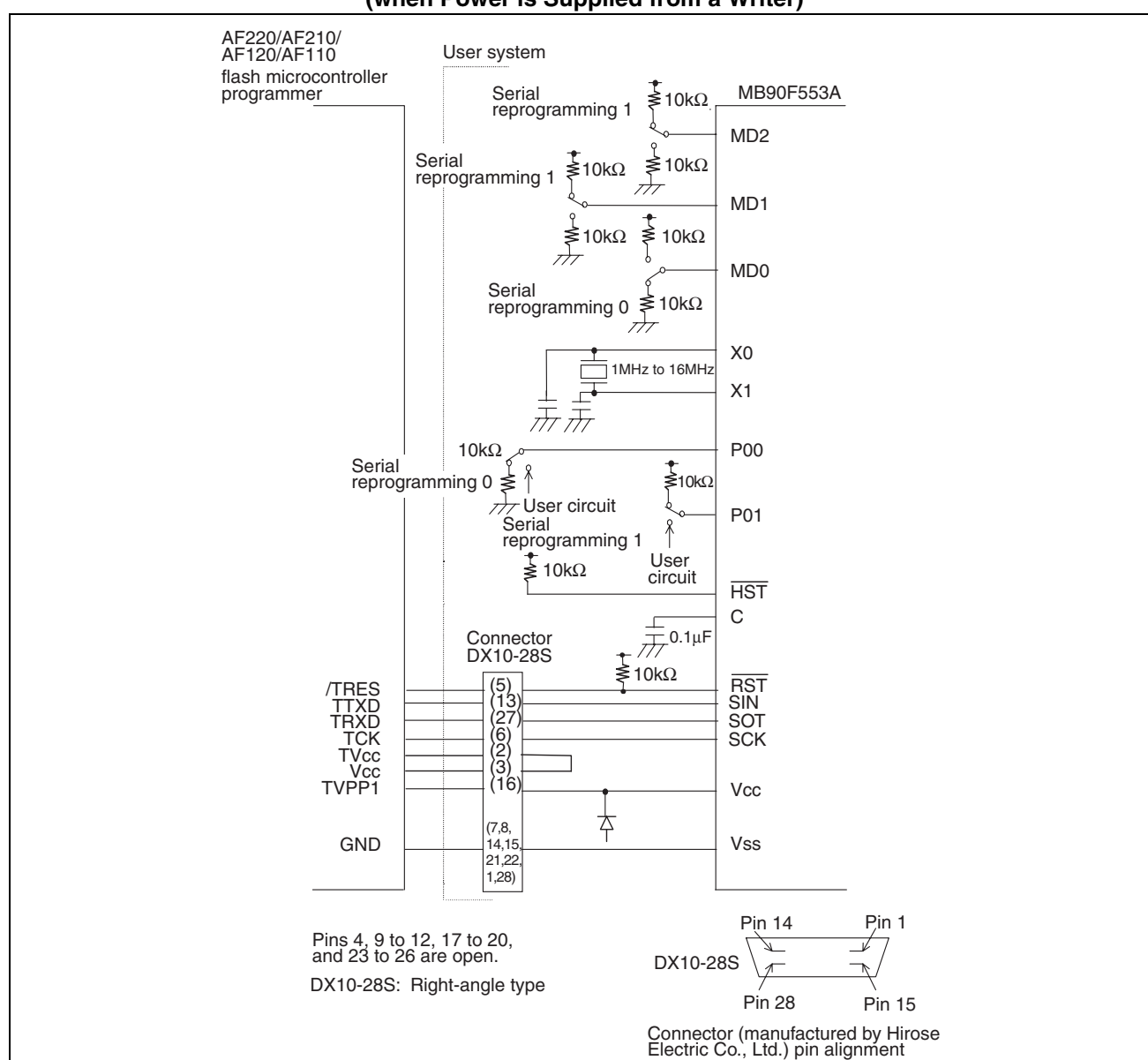
24.5 Example of Minimal Connection with the Flash Microcontroller Programmer (When Power Is Supplied from a Writer)

Figure 24.5-1 shows an example of minimal connection with the flash microcontroller programmer when the power supply voltage of the microcontroller is supplied from a writer.

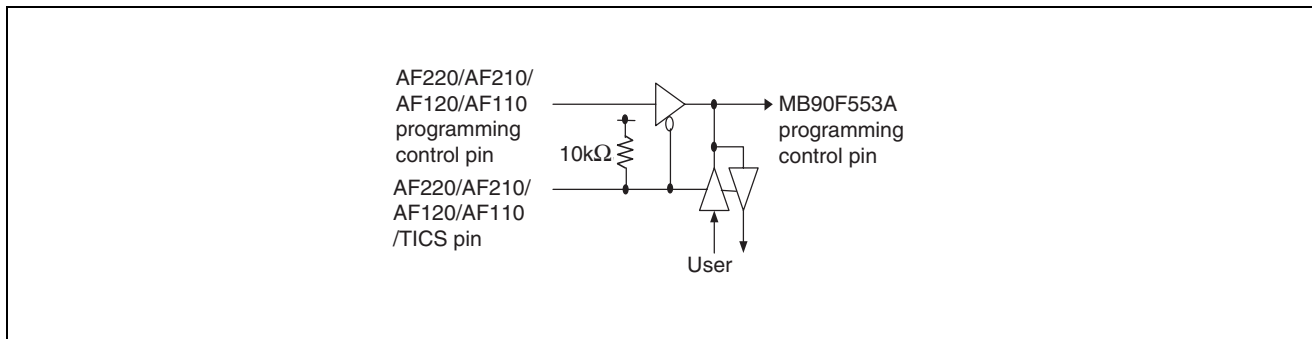
■ Example of Minimal Connection with the Flash Microcontroller Programmer (when Power is Supplied from a Writer)

The MD2, MD1, MD0, and P00 pins need not be connected to the flash microcontroller programmer if each pin is connected, as shown below, for flash memory programming.

**Figure 24.5-1 Example of Minimal Connection with Flash Microcontroller Programmer
(when Power is Supplied from a Writer)**



- When the SIN, SOT, and SCK 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 microcontroller 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, instruction, and OTPROM programming.

APPENDIX A I/O MAP

APPENDIX B Instructions

APPENDIX C Programming the OTPROM

APPENDIX A I/O MAP

Addresses are allocated to registers of each peripheral circuit of the microcontroller.

■ I/O Map

Table A-1 I/O Map (1/8)

Address	Register	Abbreviation	Access	Peripheral	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	--11111 _B
06 _H	Port 6 data register	PDR6	R/W	Port 6	XXXXXXXX _B
07 _H	Port 7 data register	PDR7	R/W	Port 7	XXXXXXXX _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	---XXXX _B
0B _H to 0F _H	Not available				
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	Not available				
16 _H	Port 6 data direction register	DDR6	R/W	Port 6	00000000 _B
17 _H	Port 7 data direction register	DDR7	R/W	Port 7	00000000 _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	---00000 _B
1B _H	Port 4 output pin register	ODR4	R/W	Port 4	00000000 _B

Table A-1 I/O Map (2/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1C _H	Port 0 resistor register	RDR0	R/W	Port 0	00000000 _B
1D _H	Port 1 resistor register	RDR1	R/W	Port 1	00000000 _B
1E _H	Not available				
1F _H	Analog input enable register	ADER	R/W	Port 6, A/D	11111111 _B
20 _H	Serial mode register	SMR	R/W	UART	00000000 _B
21 _H	Serial control register	SCR	R/W		00000100 _B
22 _H	Serial input register/serial output register	SIDR/SODR	R/W		XXXXXXXX _B
23 _H	Serial status register	SSR	R/W		00001-00 _B
24 _H	Serial mode control status register 0	SMCS0	R/W	I/O extended serial interface 0	----0000 _B
25 _H	Serial mode control status register 0		R/W!		0000010 _B
26 _H	Serial data register 0	SDR0	R/W		XXXXXXXX _B
27 _H	Clock divide control register	CDCR	R/W	Communi- cation prescaler	0---1111 _B
28 _H	Serial mode control register 1	SMCS1	R/W	I/O extended serial interface 1	----0000 _B
29 _H	Serial mode control register 1		R/W!		00000010 _B
2A _H	Serial data register 1	SDR1	R/W		XXXXXXXX _B
2B _H	Not available				
2C _H	I ² C bus status register 0	IBSR0	R	I ² C interface 0	00000000 _B
2D _H	I ² C bus control register 0	IBCR0	R/W		00000000 _B
2E _H	I ² C bus clock selection register 0	ICCR0	R/W		--0XXXXX _B
2F _H	I ² C bus address register 0	IADR0	R/W		-XXXXXXX _B
30 _H	I ² C bus data register 0	IDAR0	R/W		XXXXXXXX _B
31 _H	Not available				
32 _H	I ² C bus status register 1	IBSR1	R/W	I ² C interface 1	0000000 _B
33 _H	I ² C bus control register 1	IBCR1	R/W		00000000 _B
34 _H	I ² C bus clock selection register 1	ICCR1	R/W		--0XXXXX _B
35 _H	I ² C bus address register 1	IADR1	R/W		-XXXXXX _B
36 _H	I ² C bus data register 1	IDAR1	R/W		XXXXXXXX _B
37 _H	I ² C bus port selection register	ISEL	R/W		-----0 _B

Table A-1 I/O Map (3/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
38 _H	Interrupt/DTP enable register	ENIR	R/W	DTP/ external interrupt circuit	00000000 _B
39 _H	Interrupt/DTP source register	EIRR	R/W		XXXXXXXX _B
3A _H	Request level setting register	ELVR	R/W		00000000 _B
3B _H					00000000 _B
3C _H	Control status register	ADCS0	R/W	A/D converter	00000000 _B
3D _H		ADCS1	R/W!		00000000 _B
3E _H	Data register	ADCR0	R/W!		XXXXXXXX _B
3F _H		ADCR1	R/W!		00001-XX _B
40 _H	Reload register L (ch.0)	PRL0	R/W	8/16 bit PPG0/1	XXXXXXXX _B
41 _H	Reload register H (ch.0)	PRLH0	R/W		XXXXXXXX _B
42 _H	Reload register L (ch.1)	PRL1	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		0-000--1 _B
45 _H	PPG1 operating mode control register	PPGC1	R/W		0-000001 _B
46 _H	PPG0, PPG1 output control register	PPGE1	R/W		00000000 _B
47 _H	Not available				
48 _H	Reload register L (ch.2)	PRL2	R/W	8/16 bit PPG2/3	XXXXXXXX _B
49 _H	Reload register H (ch.2)	PRLH2	R/W		XXXXXXXX _B
4A _H	Reload register L (ch.3)	PRL3	R/W		XXXXXXXX _B
4B _H	Reload register H (ch.3)	PRLH3	R/W		XXXXXXXX _B
4C _H	PPG2 operating mode control register	PPGC2	R/W		0-000--1 _B
4D _H	PPG3 operating mode control register	PPGC3	R/W		0-000001 _B
4E _H	PPG2, PPG3 output control register	PPGE2	R/W		00000000 _B
4F _H	Not available				

Table A-1 I/O Map (4/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
50 _H	Reload register L (ch.4)	PRL4	R/W	8/16 bit PPG4/5	XXXXXXXX _B
51 _H	Reload register H (ch.4)	PRLH4	R/W		XXXXXXXX _B
52 _H	Reload register L (ch.5)	PRL5	R/W		XXXXXXXX _B
53 _H	Reload register H (ch.5)	PRLH5	R/W		XXXXXXXX _B
54 _H	PPG4 operating mode control register	PPGC4	R/W		0-000--1 _B
55 _H	PPG5 operating mode control register	PPGC5	R/W		0-000001 _B
56 _H	PPG4, PPG5 output control register	PPGE3	R/W		00000000 _B
57 _H	Not available				
58 _H	Clock pulse output enable register	CLKR	R/W	Clock monitor function	----0000 _B
59 _H	Not available				
5A _H	Control status register 0	TMCSR0	R/W	16-bit reload timer 0	00000000 _B
5B _H					----0000 _B
5C _H	16-bit timer register 0/ 16-bit reload register 0	TMR0/ TMRLR0	R/W		XXXXXXXX _B
5D _H					XXXXXXXX _B
5E _H	Control status register 1	TMCSR1	R/W	16-bit reload timer 1	00000000 _B
5F _H					----0000 _B
60 _H	16-bit timer register 1	TMR1	R/W		XXXXXXXX _B
61 _H	16-bit reload register 1	TMRLR1			XXXXXXXX _B

Table A-1 I/O Map (5/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
62 _H	Input capture register (low-order byte of ch.0)	IPCP0	R	16-bit I/O timer input capture (ch.0 to ch.3)	XXXXXXXX _B
63 _H	Input capture register (high-order byte of ch.0)				XXXXXXXX _B
64 _H	Input capture register (low-order byte of ch.1)	IPCP1	R		XXXXXXXX _B
65 _H	Input capture register (high-order byte of ch.1)				XXXXXXXX _B
66 _H	Input capture register (low-order byte of ch.2)	IPCP2	R		XXXXXXXX _B
67 _H	Input capture register (high-order byte of ch.2)				XXXXXXXX _B
68 _H	Input capture register (low-order byte of ch.3)	IPCP3	R		XXXXXXXX _B
69 _H	Input capture register (high-order byte of ch.3)				XXXXXXXX _B
6A _H	Input capture control status register	ICS01	R/W		00000000 _B
6B _H	Input capture control status register	ICS23	R/W		00000000 _B
6C _H	Timer data register (low-order byte)	TCDT	R/W	16-bit I/O timer free-run timer	00000000 _B
6D _H	Timer data register (high-order byte)		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

Table A-1 I/O Map (6/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
70 _H	Compare register ch.0 (low-order byte)	OCCP0	R/W	16-bit I/O timer output compare (ch.0 to ch.3)	XXXXXXXX _B
71 _H	Compare register ch.0 (high-order byte)				XXXXXXXX _B
72 _H	Compare register ch.1 (low-order byte)	OCCP1	R/W		XXXXXXXX _B
73 _H	Compare register ch.1 (high-order byte)				XXXXXXXX _B
74 _H	Compare register ch.2 (low-order byte)	OCCP2	R/W		XXXXXXXX _B
75 _H	Compare register ch.2 (high-order byte)				XXXXXXXX _B
76 _H	Compare register ch.3 (low-order byte)	OCCP3	R/W		XXXXXXXX _B
77 _H	Compare register ch.3 (high-order byte)				XXXXXXXX _B
78 _H	Compare control status register ch.0	OCS0	R/W		0000--00 _B
79 _H	Compare control status register ch.1	OCS1	R/W		---00000 _B
7A _H	Compare control status register ch.2	OCS2	R/W		0000--00 _B
7B _H	Compare control status register ch.3	OCS3	R/W		---00000 _B
7C _H to 9D _H	Not available				
9E _H	Program address detection control register	PACSR	R/W	Address match detection function	00000000 _B
9F _H	Delay interrupt source occurrence/release register	DIRR	R/W	Delay interrupt	-----0 _B
A0 _H	Low-power consumption mode register	LPMCR	R/W!	Low-power consumption control circuit	00011000 _B
A1 _H	Clock selection register	CKSCR	R/W!		11111100 _B
A2 _H to A4 _H	Not available				
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

Table A-1 I/O Map (7/8)

Address	Register	Abbreviation	Access	Peripheral	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 to AD _H	Not available				
AE _H	Flash memory control status register	FMCS	R/W	Flash memory interface circuit	00000--0 _B
AF _H	Not available				
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 (8/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1FF0 _H	Program address detection register 0	PADR0	R/W	Program patch processing	XXXXXXXX _B
1FF1 _H	Program address detection register 1		R/W		XXXXXXXX _B
1FF2 _H	Program address detection register 2		R/W		XXXXXXXX _B
1FF3 _H	Program address detection register 3	PADR1	R/W		XXXXXXXX _B
1FF4 _H	Program address detection register 4		R/W		XXXXXXXX _B
1FF5 _H	Program address detection register 5		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 00FF_H 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. (For the detail, see Section "4.5 Registers not Initialized by Reset Input".)

R/W! in the access column indicates that the register contains only-read or only-write bits.

If a register for which R/W!, R/W*, or W is indicated in the access column is accessed by a read modify write instruction (such as a bit set instruction), the bit designated by the instruction indicates the predetermined value. However, if the other bits include write-only bits, a malfunction occurs. Therefore, do not access the register by an instruction of this type.

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></tr><tr><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></tr><tr><td>1</td><td>2</td><td>1</td><td>2</td></tr></table> (Some instructions transfer AL to AH.)	4	4	5	5	1	2	1	2
4	4	5	5							
1	2	1	2							

- Register direct addressing

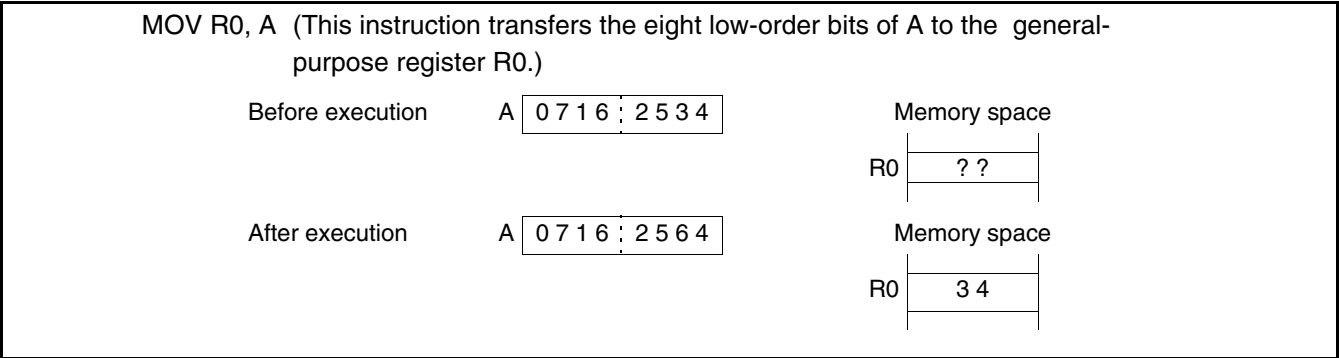
Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

Table B.3-1 Direct Addressing Registers

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

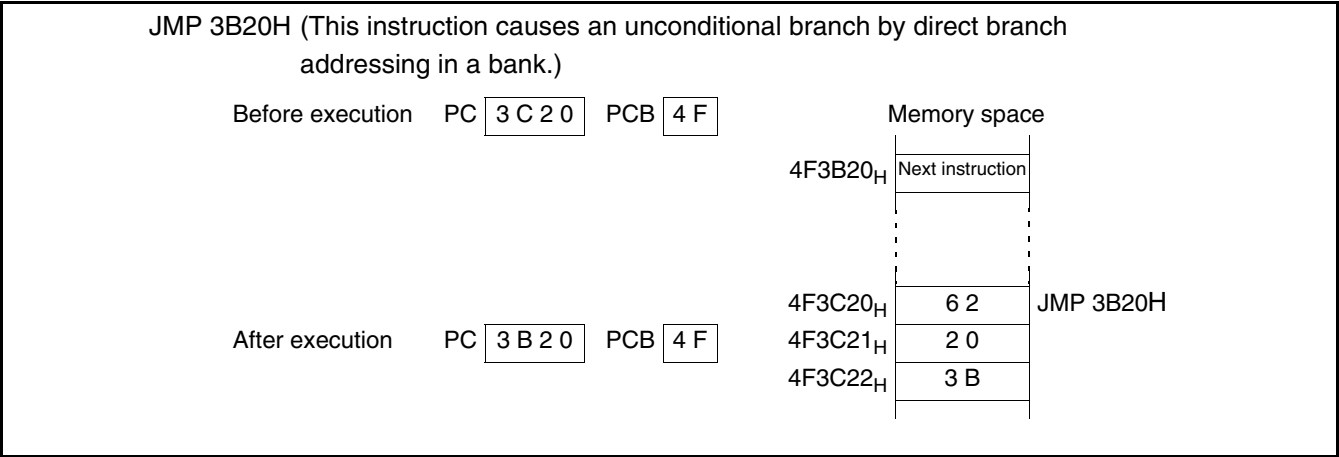
Figure B.3-2 Example of Register Direct Addressing



● Direct branch addressing (addr16)

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bit23 to bit16 of the address are specified by the program counter bank register (PCB).

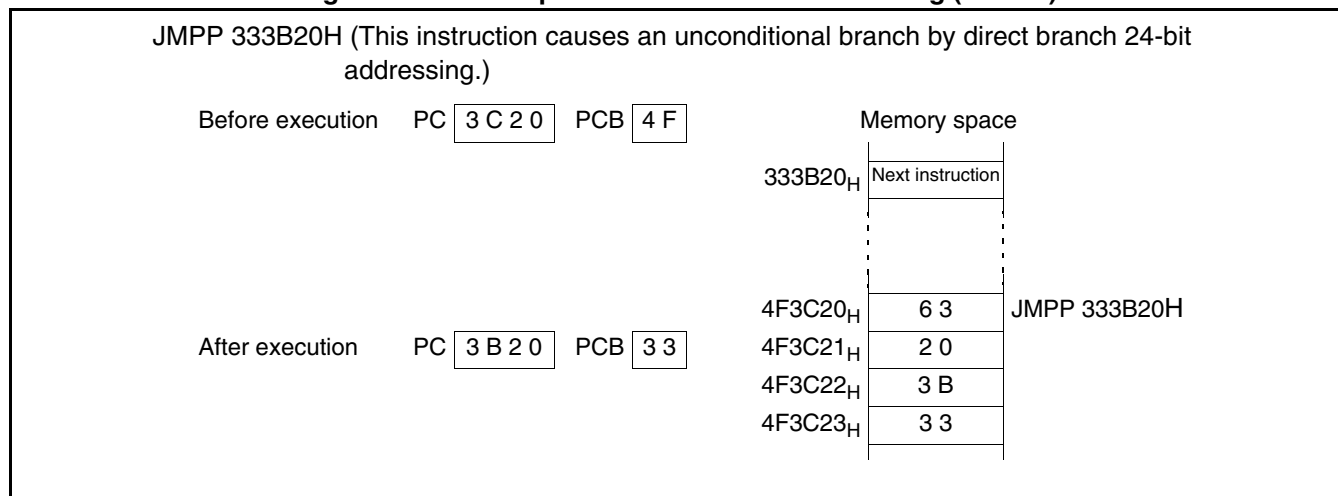
Figure B.3-3 Example of Direct Branch Addressing (addr16)



- Physical direct branch addressing (addr24)

Specify an offset explicitly for the branch destination address. The size of the offset is 24 bits. Physical direct branch addressing is used for unconditional branch, subroutine call, or software interrupt instruction.

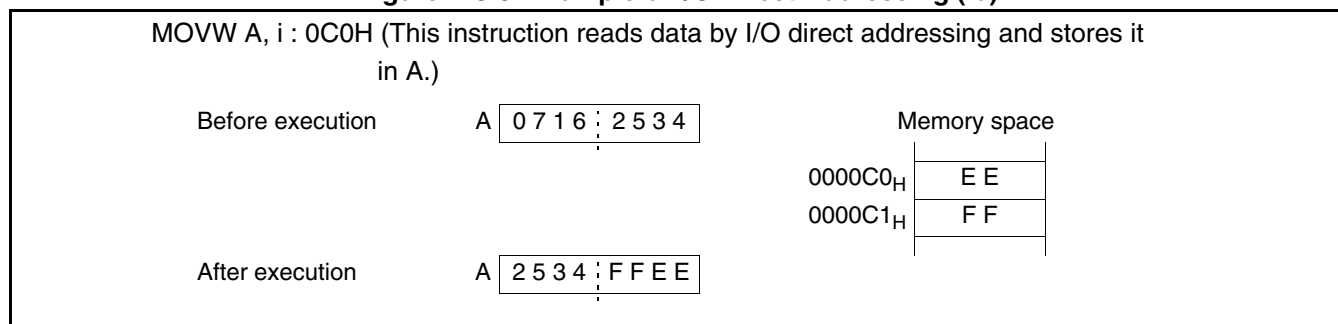
Figure B.3-4 Example of Direct Branch Addressing (addr24)



- I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000_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.

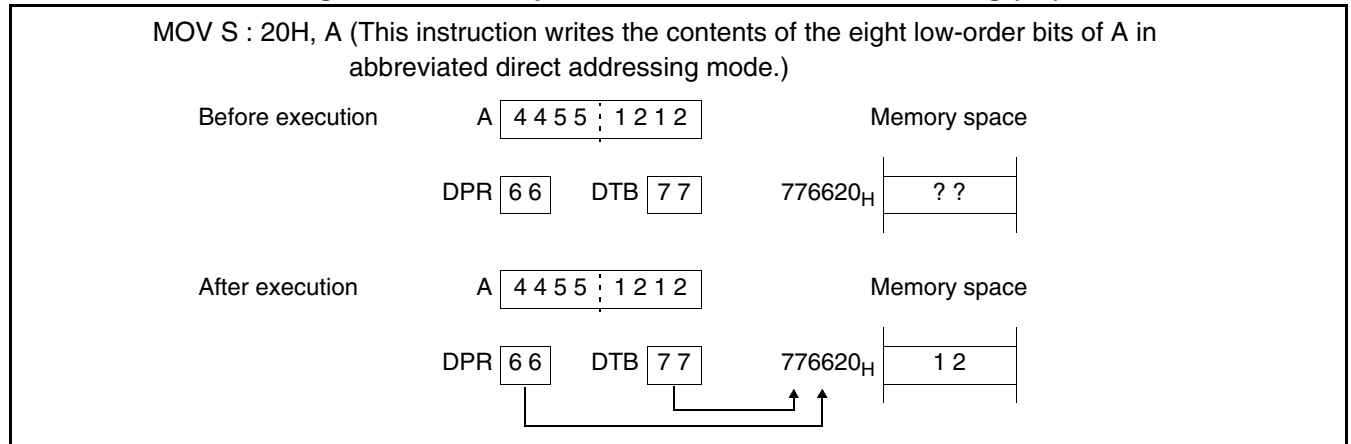
Figure B.3-5 Example of I/O Direct Addressing (io)



- 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).

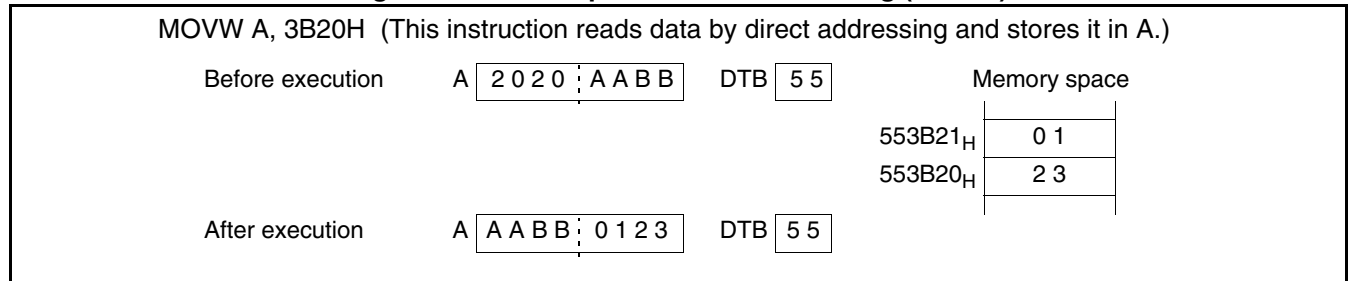
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)



- 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.

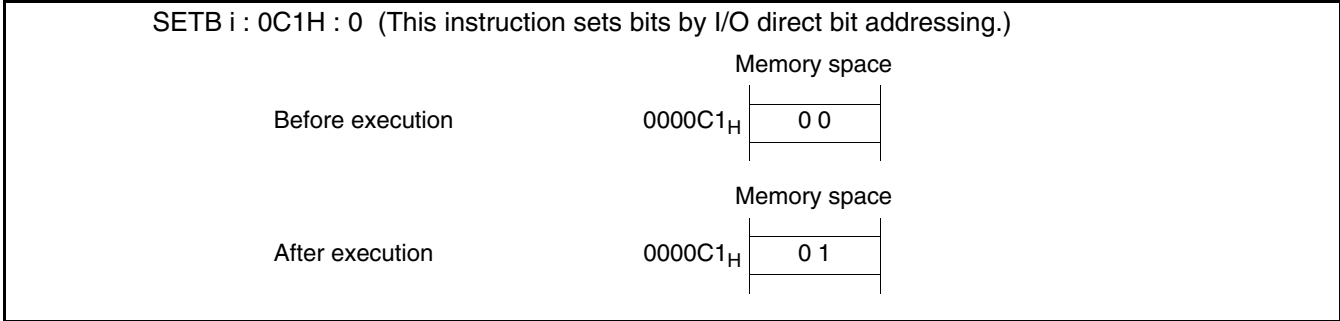
Figure B.3-7 Example of Direct Addressing (addr16)



● 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).

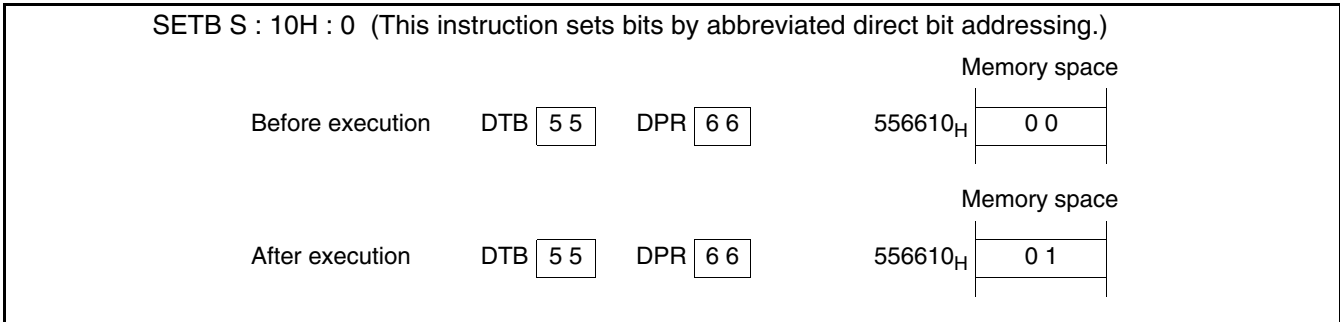
Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)



● 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).

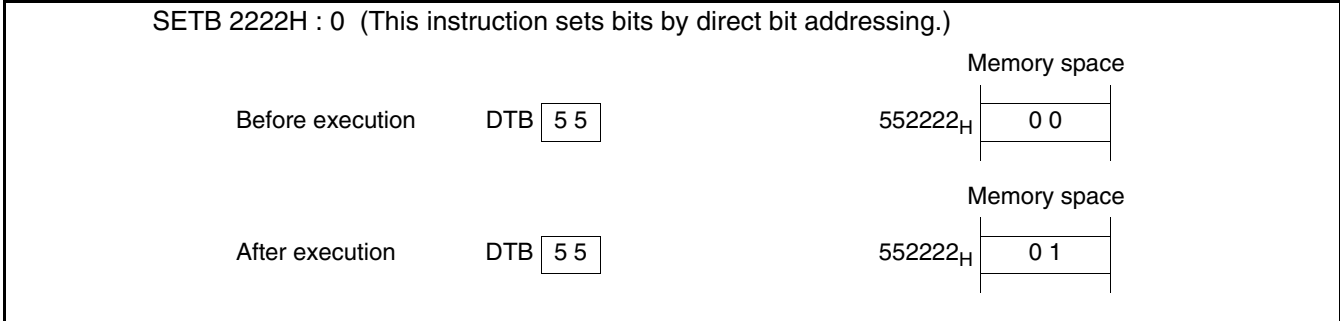
Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)



● 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)

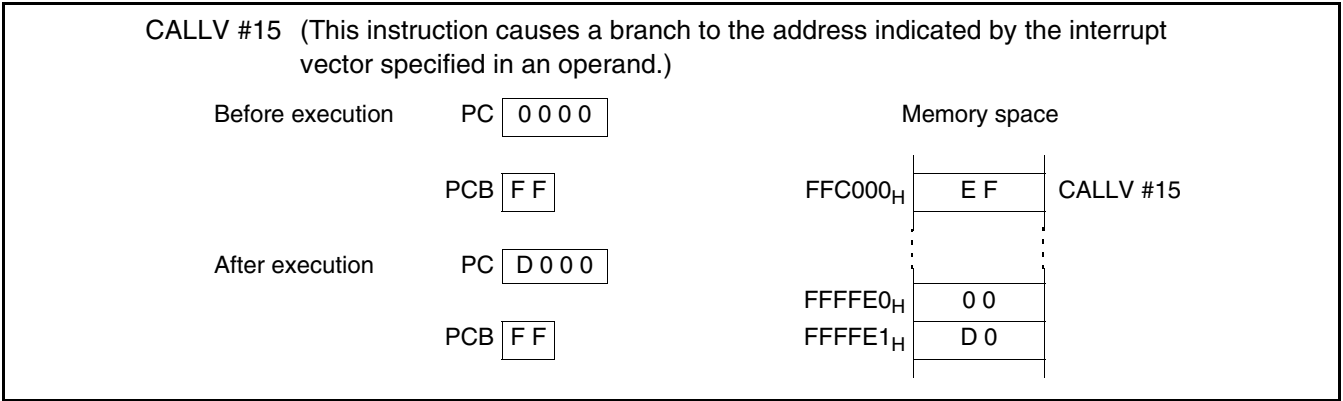


Table B.3-2 CALLV Vector List

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	XXFFEF _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

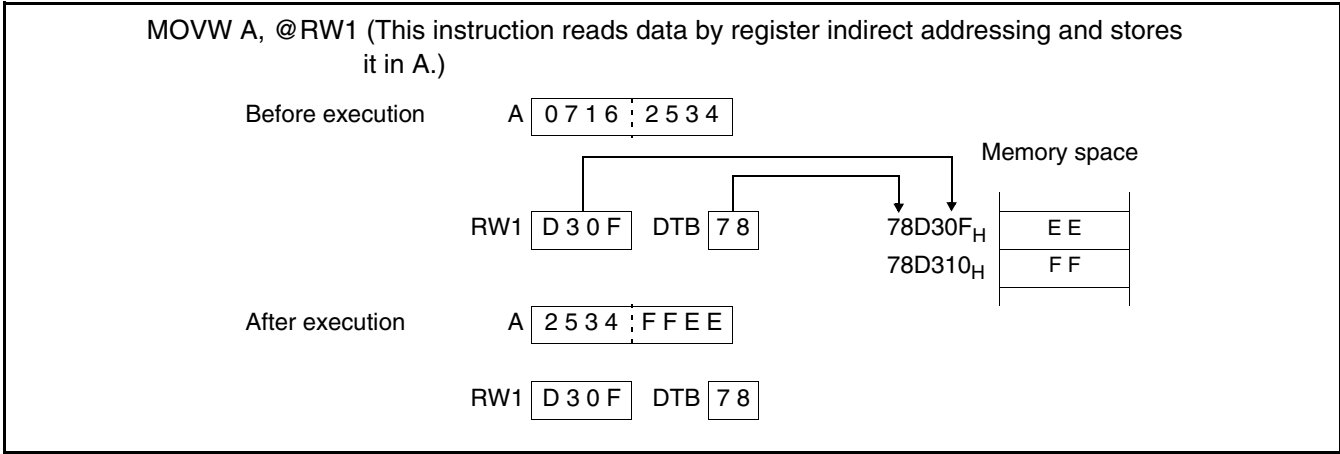
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ 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.

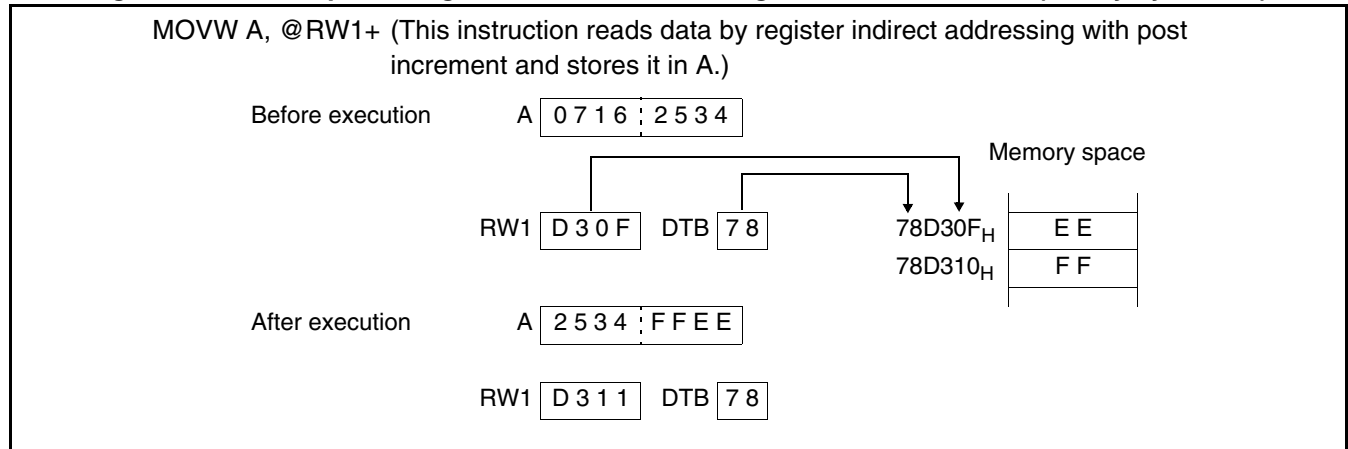
Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)



- Register indirect addressing with post increment (@RWj+ j = 0 to 3)

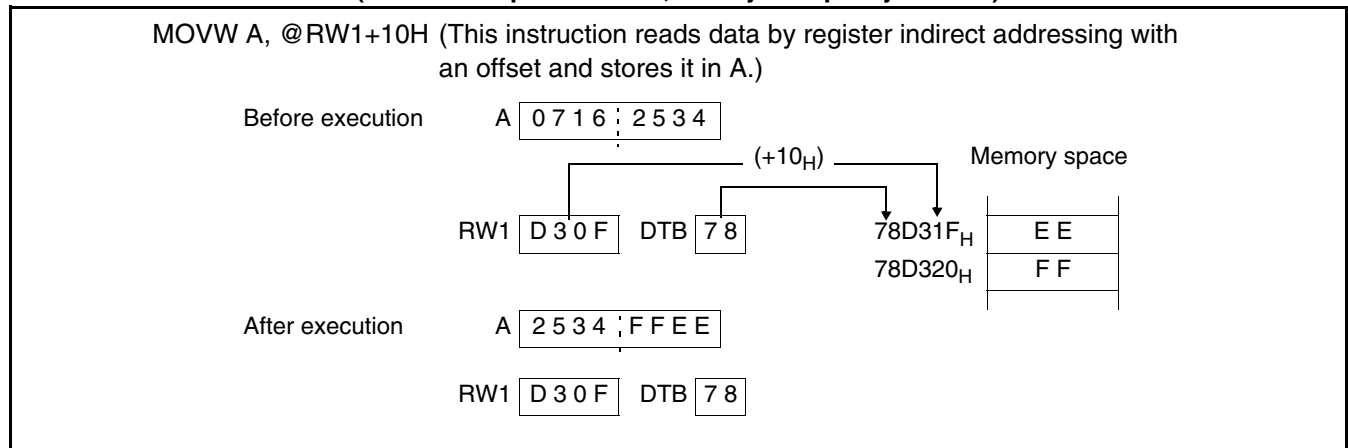
Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system 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)

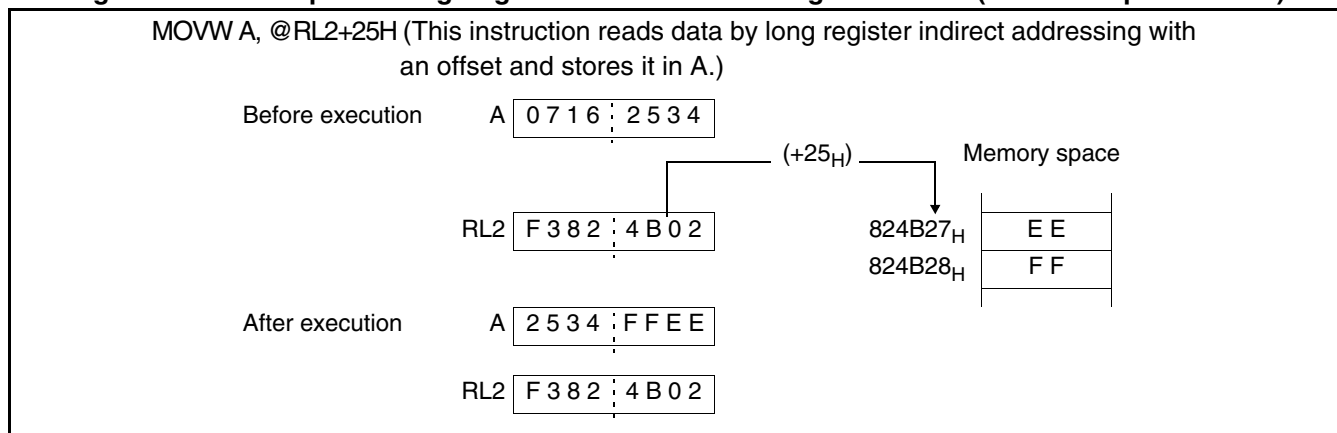
Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or 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)

- 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)

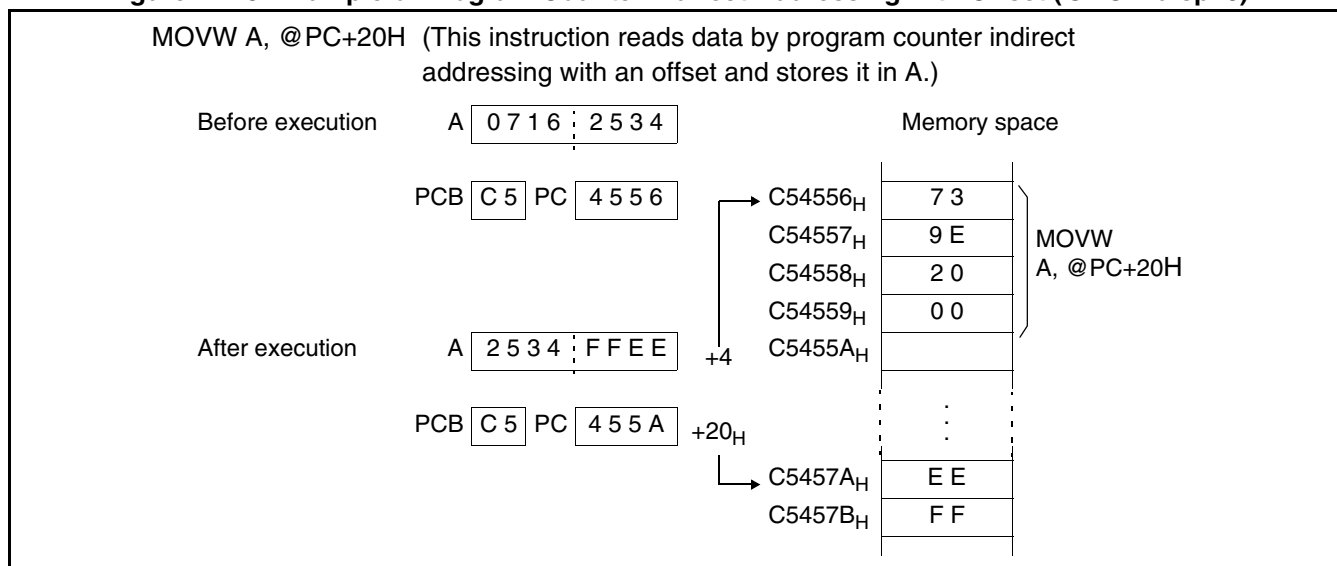


- Program counter indirect addressing with offset (@PC + disp16)

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

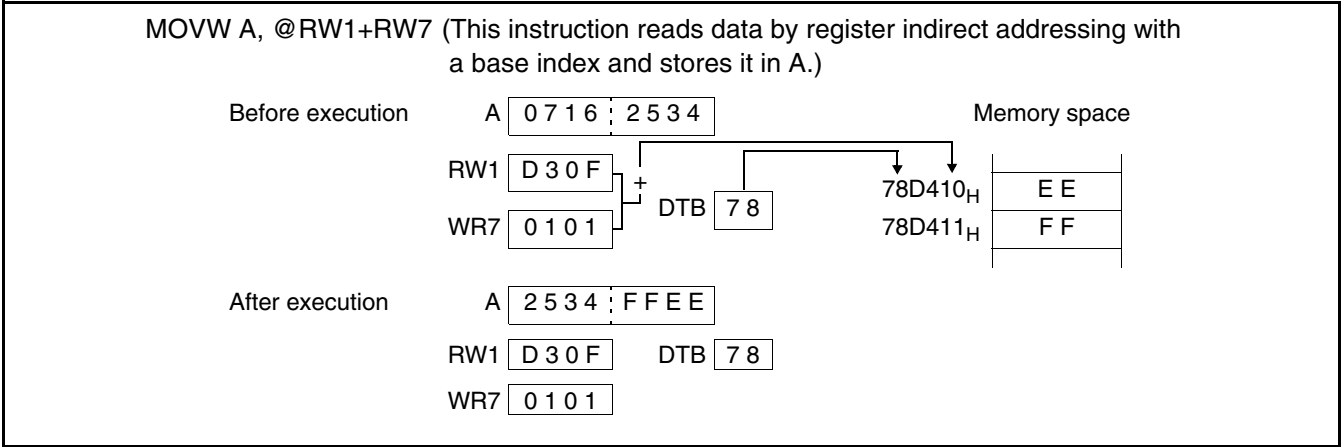
Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16)



● 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).

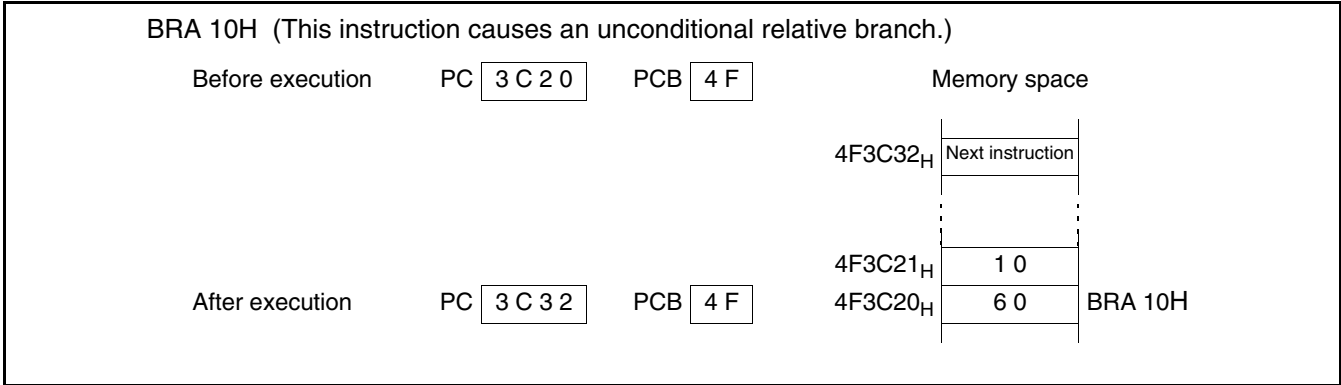
Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)



● Program counter relative branch addressing (rel)

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-kilobyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program counter bank register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



● Register list (rlst)

Specify a register to be pushed onto or popped from a stack.

Figure B.4-8 Configuration of the Register List

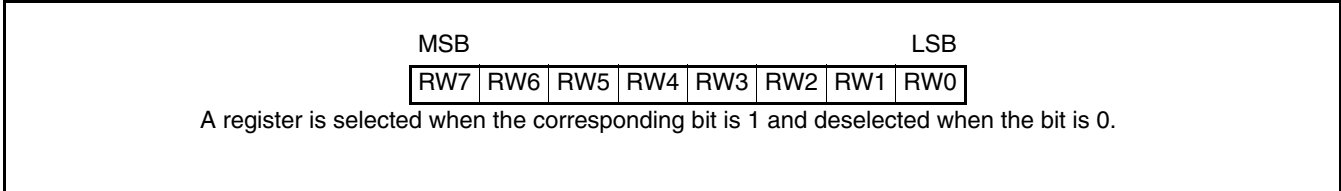
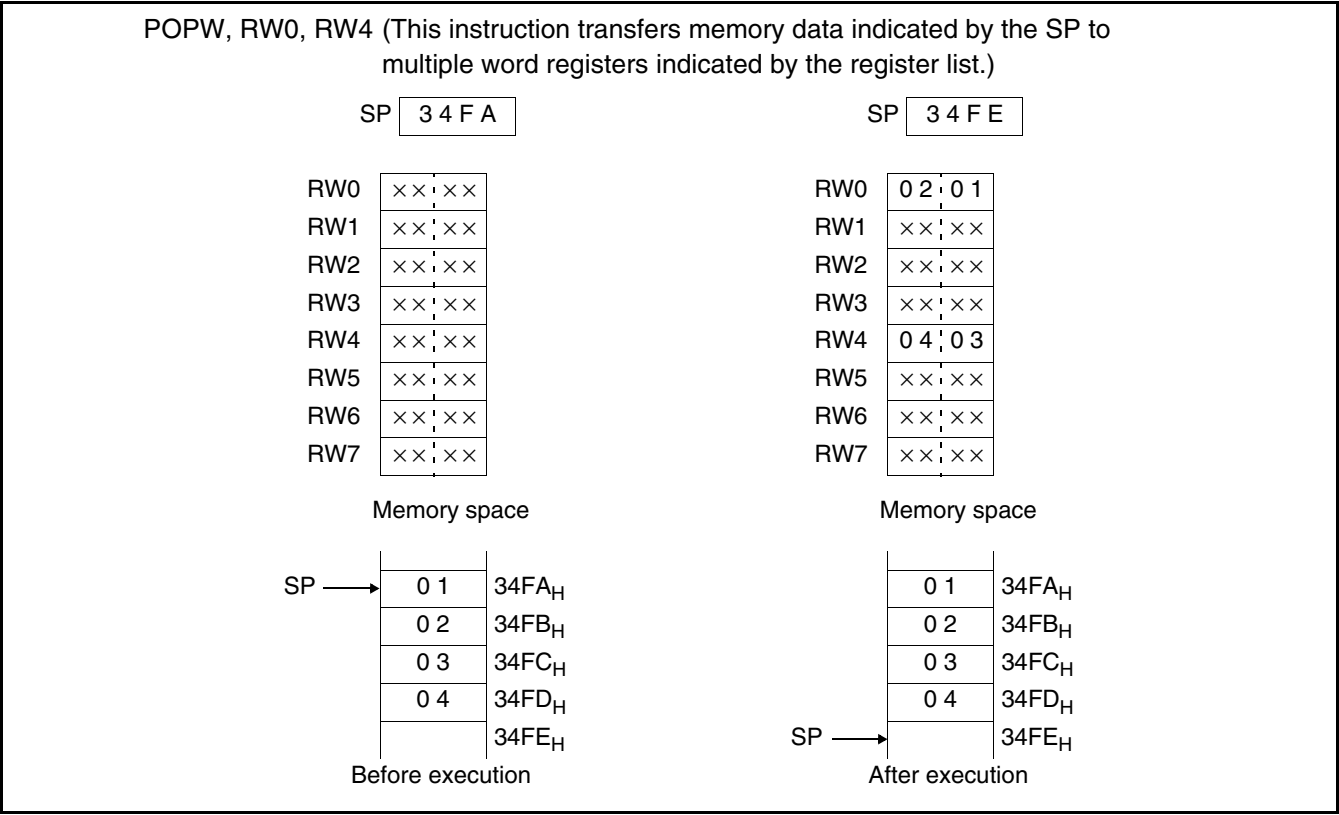


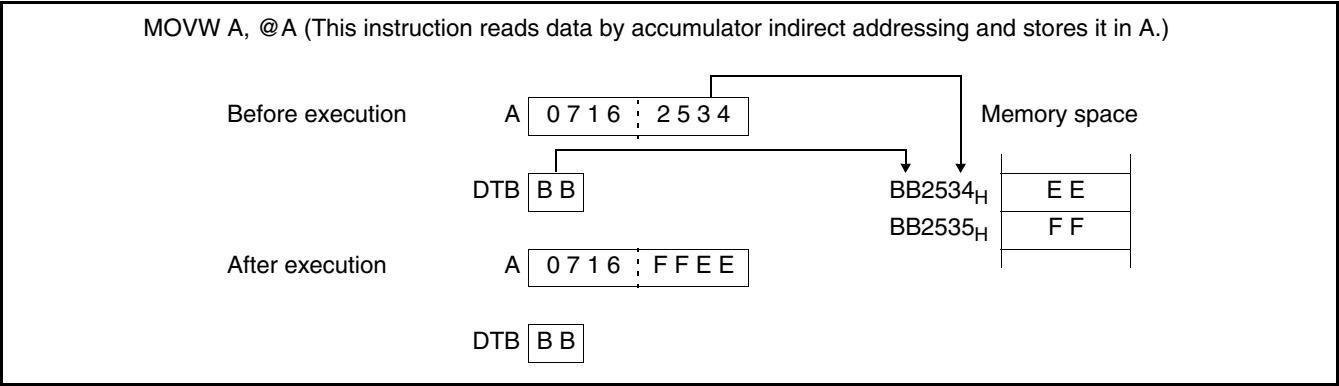
Figure B.4-9 Example of Register List (rlist)



● Accumulator indirect addressing (@A)

Memory is accessed using the address indicated by the contents of the low-order bytes (16 bits) of the accumulator (AL). Address bits 16 to 23 are specified by a mnemonic in the data bank register (DTB).

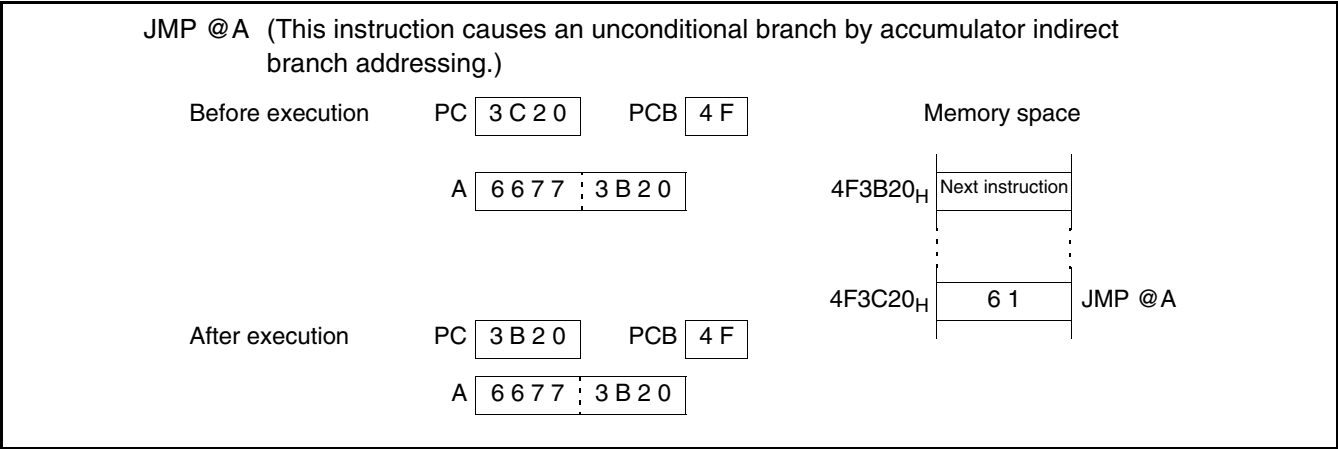
Figure B.4-10 Example of Accumulator Indirect Addressing (@A)



● Accumulator indirect branch addressing (@A)

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program counter bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

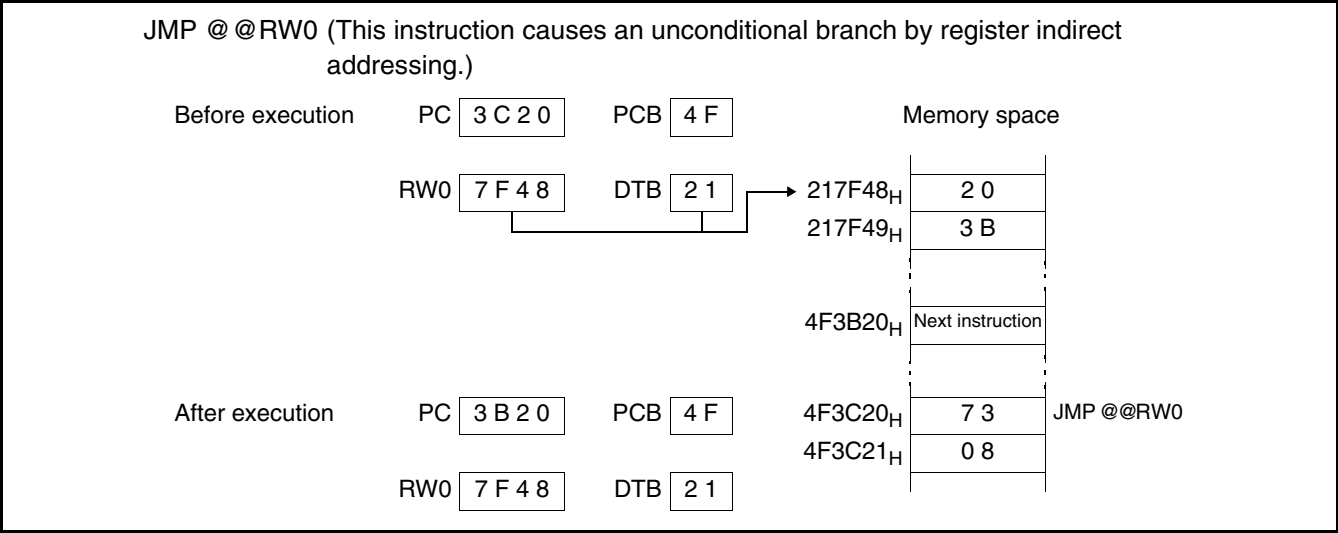
Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)



● Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

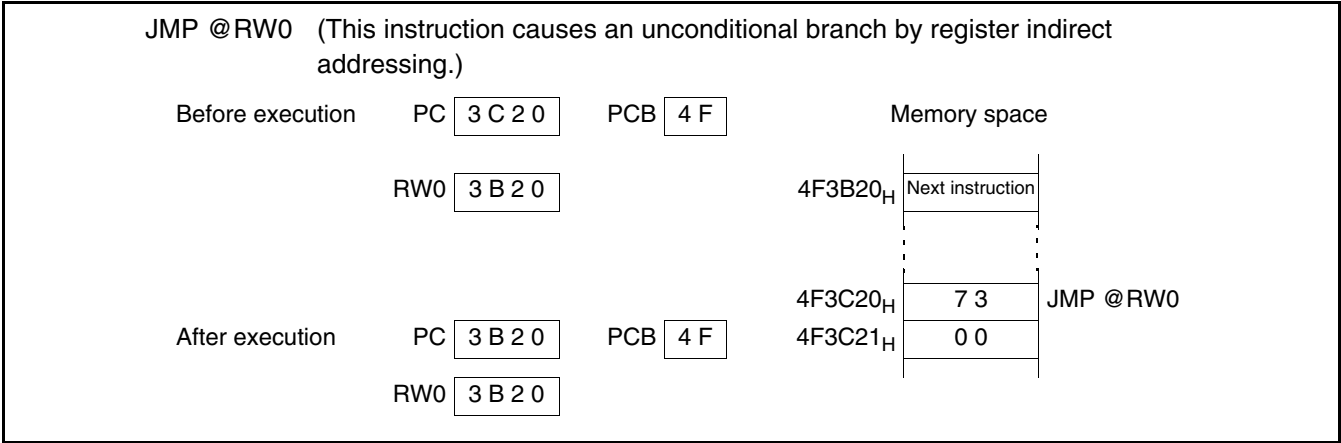
Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)



● Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



B.5 Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.

■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

■ Calculating the Execution Cycle Count

Table B.5-1 lists execution cycle counts and Table B.5-2 and Table B.5-3 summarize correction value data.

Table B.5-1 Execution Cycle Counts in Each Addressing Mode

Code	Operand	(a) *	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00 07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08 0B	@RWj	2	1
0C 0F	@RWj+	4	2
10 17	@RWi+disp8	2	1
18 1B	@RWi+disp16	2	1
1C 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</p> <p> -: Not Read Modify Write instruction</p> <p>Note:</p> <p>Cannot be used for an address that has different meanings between read and write operations.</p>

Table B.7-2 Explanation on Symbols in the Instruction List (1/2)

Symbol	Explanation
A	<p>The bit length used varies depending on the 32-bit accumulator instruction.</p> <p>Byte: Low-order 8 bits of byte AL</p> <p>Word: 16 bits of word AL</p> <p>Long word: 32 bits of AL and AH</p>
AH	16 high-order bits of A
AL	16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	program counter bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Abbreviated direct addressing
addr16	Direct addressing
addr24	Physical direct addressing
ad24 0-15	Bit0 to bit15 of addr24

Table B.7-2 Explanation on Symbols in the Instruction List (1/2)

Symbol	Explanation
ad24 16-23	Bit16 to bit23 of addr24
io	I/O area (000000 _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.

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.

Table B.8-4 12 Increment/decrement Instructions (Byte, Word, Long Word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	$2 \times (b)$	byte (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	$2 \times (b)$	byte (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	$2 \times (c)$	word (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	$2 \times (c)$	word (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	$2 \times (d)$	long (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	$2 \times (d)$	long (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-5 11 Compare Instructions (Byte, Word, Long Word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-6 11 Unsigned Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU A	1	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW A	1	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0 7: Overflow 15: Normal

*2: 4: Division by 0 8: Overflow 16: Normal

*3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal

*4: 4: Division by 0 7: Overflow 22: Normal

*5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal

*6: (b): Division by 0 or overflow 2 × (b): Normal

*7: (c): Division by 0 or overflow 2 × (c): Normal

*8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.

*9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.

*10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.

*11: 3: Word (AH) is 0. 11: Word (AH) is not 0.

*12: 4: Word (ear) is 0. 12: Word (ear) is not 0.

*13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-7 11 Signed Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV A	2	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL A	2	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW A	2	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal

*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal

*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal

*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal

When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal

*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal

When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal

*6: (b): Division by 0 or overflow, 2 × (b): Normal

*7: (c): Division by 0 or overflow, 2 × (c): Normal

*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative

*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative

*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative

*11: 3: Word (AH) is 0, 16: result is positive, 19: result is negative

*12: 4: Word (ear) is 0, 17: result is positive, 20: result is negative

*13: 5+(a): Word (eam) is 0, 18+(a): result is positive, 21+(a): result is negative

Notes:

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-8 39 Logic 1 Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A,#imm8	2	2	0	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	3	1	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	2	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	3	1	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	2	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	3	1	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	2	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	3	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOT eam	2+	5+(a)	0	2 × (b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	3	1	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	2	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	3	1	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	2	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	3	1	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	2	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	3	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW eam	2+	5+(a)	0	2 × (c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-9 6 Logic 2 Instructions (Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	6	2	0	$\text{long}(A) \leftarrow (A) \text{ and } (\text{ear})$	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	$\text{long}(A) \leftarrow (A) \text{ and } (\text{eam})$	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	$\text{long}(A) \leftarrow (A) \text{ or } (\text{ear})$	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	$\text{long}(A) \leftarrow (A) \text{ or } (\text{eam})$	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	$\text{long}(A) \leftarrow (A) \text{ xor } (\text{ear})$	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	$\text{long}(A) \leftarrow (A) \text{ xor } (\text{eam})$	-	-	-	-	-	*	*	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	$\text{byte}(A) \leftarrow 0 - (A)$	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	$\text{byte}(\text{ear}) \leftarrow 0 - (\text{ear})$	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 × (b)	$\text{byte}(\text{eam}) \leftarrow 0 - (\text{eam})$	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	$\text{word}(A) \leftarrow 0 - (A)$	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	$\text{word}(\text{ear}) \leftarrow 0 - (\text{ear})$	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 × (c)	$\text{word}(\text{eam}) \leftarrow 0 - (\text{eam})$	-	-	-	-	-	*	*	*	*	*

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	$\text{long}(A) \leftarrow \text{Shift left to the position where '1' is set for the first time.}$ $\text{byte}(R0) \leftarrow \text{Shift count at that time}$	-	-	-	-	-	-	*	-	-	-

*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

Table B.8-12 18 Shift Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	3	2	0	byte (ear) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	*
ASR A,R0	2	*1	1	0	byte (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSR A,R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSL A,R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW A/SHRW A	1	2	0	0	word (A) ← Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW A/SHLW A	1	2	0	0	word (A) ← Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW A,R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW A,R0	2	*1	1	0	word (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL A,R0	2	*2	1	0	long (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL A,R0	2	*2	1	0	long (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL A,R0	2	*2	1	0	long (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

*1: 6 when R0 is 0; otherwise, 5 + (R0)

*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-13 31 Branch 1 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/ BNE	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/ BHS	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT rel	2	*1	0	0	Branch on (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE rel	2	*1	0	0	Branch on (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP @A	1	2	0	0	word (PC) ← (A)	-	-	-	-	-	-	-	-	-	-
JMP addr16	3	3	0	0	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
JMP @ear	2	3	1	0	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
JMP @eam	2+	4+(a)	0	(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
JMPP @ear *3	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP @eam *3	2+	6+(a)	0	(d)	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP addr24	4	4	0	0	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL @ear *4	2	6	1	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
CALL @eam *4	2+	7+(a)	0	2 × (c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
CALL addr16 *5	3	6	0	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
CALLV #vct4 *5	1	7	0	2 × (c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP @ear *6	2	10	2	2 × (c)	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
CALLP @eam *6	2+	11+(a)	0	*2	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
CALLP addr24 *7	4	10	0	2 × (c)	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-

*1: 4 when a branch is made; otherwise, 3

*2: 3 × (c) + (b)

*3: Read (word) of branch destination address

*4: W: Save to stack (word) R: Read (word) of branch destination address

*5: Save to stack (word)

*6: W: Save to stack (long word), R: Read (long word) of branch destination address

*7: Save to stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-14 19 Branch 2 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE eam,#imm16,rel *9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ ear,rel	3	*5	2	0	byte (ear) \leftarrow (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ eam,rel	3+	*6	2	$2 \times (b)$	byte (eam) \leftarrow (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*5	2	0	word (ear) \leftarrow (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ eam,rel	3+	*6	2	$2 \times (c)$	word (eam) \leftarrow (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	20	0	$8 \times (c)$	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	16	0	$6 \times (c)$	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	17	0	$6 \times (c)$	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	20	0	$8 \times (c)$	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI	1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK #imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK	1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET *10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP *11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

*1: 5 when a branch is made; otherwise, 4

*2: 13 when a branch is made; otherwise, 12

*3: 7+(a) when a branch is made; otherwise, 6+(a)

*4: 8 when a branch is made; otherwise, 7

*5: 7 when a branch is made; otherwise, 6

*6: 8+(a) when a branch is made; otherwise, 7+(a)

*7: $3 \times (b) + 2 \times (c)$ when jumping to the next interruption request; $6 \times (c)$ when returning from the current interruption

*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption

*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.

*10: Return from stack (word)

*11: Return from stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-15 28 Other Control Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*5	*4	(SP) \leftarrow (SP) - 2n, ((SP)) \leftarrow (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	0	(c)	word (A) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	0	(c)	word (AH) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	4	0	(c)	word (PS) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*5	*4	(rlst) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2n	-	-	-	-	-	-	-	-	-	-
JCTX @A	1	14	0	6 \times (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	0	byte (CCR) \leftarrow (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	0	byte (CCR) \leftarrow (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	0	byte (RP) \leftarrow imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	0	byte (ILM) \leftarrow imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	1	0	word (RWi) \leftarrow ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	1	0	word (RWi) \leftarrow eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	1	0	0	word (A) \leftarrow ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	0	word (A) \leftarrow eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	0	word (SP) \leftarrow (SP) + ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	0	word (SP) \leftarrow (SP) + imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	0	byte (A) \leftarrow (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	0	byte (brg2) \leftarrow (A)	-	-	-	-	-	*	*	-	-	-
NOP	1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB	1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB	1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB	1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB	1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC	1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR	1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2

*2: $7 + 3 \times (\text{POP count}) + 2 \times (\text{POP last register number})$, 7 when RLST = 0 (no transfer register)*3: $29 + 3 \times (\text{PUSH count}) - 3 \times (\text{PUSH last register number})$, 8 when RLST = 0 (no transfer register)*4: $(\text{POP count}) \times (c)$ or $(\text{PUSH count}) \times (c)$ *5: (POP count) or (PUSH count) **Note:**

See Table B.5-1 and Table B.5-2 for information on (a) and (c) in the table.

Table B.8-16 21 Bit Operand Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A,dir:bp	3	5	0	(b)	byte (A) \leftarrow (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,addr16:bp	4	5	0	(b)	byte (A) \leftarrow (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,io:bp	3	4	0	(b)	byte (A) \leftarrow (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB dir:bp,A	3	7	0	2 \times (b)	bit (dir:bp)b \leftarrow (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	7	0	2 \times (b)	bit (addr16:bp)b \leftarrow (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	6	0	2 \times (b)	bit (io:bp)b \leftarrow (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	7	0	2 \times (b)	bit (dir:bp)b \leftarrow 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	7	0	2 \times (b)	bit (addr16:bp)b \leftarrow 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	7	0	2 \times (b)	bit (io:bp)b \leftarrow 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	7	0	2 \times (b)	bit (dir:bp)b \leftarrow 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	7	0	2 \times (b)	bit (addr16:bp)b \leftarrow 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	7	0	2 \times (b)	bit (io:bp)b \leftarrow 0	-	-	-	-	-	-	-	-	-	*
BBC dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*3	0	2 \times (b)	Branch on (addr16:bp) b = 1, bit (addr16:bp) b \leftarrow 1	-	-	-	-	-	-	*	-	-	*
WBTS io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1: 8 when a branch is made; otherwise, 7

*2: 7 when a branch is made; otherwise, 6

*3: 10 when the condition is met; otherwise, 9

*4: Undefined count

*5: Until the condition is met

Note:

See Table B.5-1 and Table B.5-2 for information on (b) in the table.

Table B.8-17 6 Accumulator Operation Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 \leftrightarrow (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW	1	2	0	0	word (AH) \leftrightarrow (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extension	-	Z	-	-	-	R	*	-	-	-

Table B.8-18 10 String Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*5	*3	byte transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*5	*3	byte transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*8	*4	byte search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*8	*4	byte search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*8	*3	byte fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*5	*6	word transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*5	*6	word transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*8	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*8	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*8	*6	word fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, $4 + 7 \times (RW0)$ when the counter expires, or $7n + 5$ when a match occurs

*2: 5 when RW0 is 0; otherwise, $4 + 8 \times (RW0)$

*3: $(b) \times (RW0) + (b) \times (RW0)$ When the source and destination access different areas, calculate the (b) item individually.

*4: $(b) \times n$

*5: $2 \times (b) \times (RW0)$

*6: $(c) \times (RW0) + (c) \times (RW0)$ When the source and destination access different areas, calculate the (c) item individually.

*7: $(c) \times n$

*8: $(b) \times (RW0)$

Note:

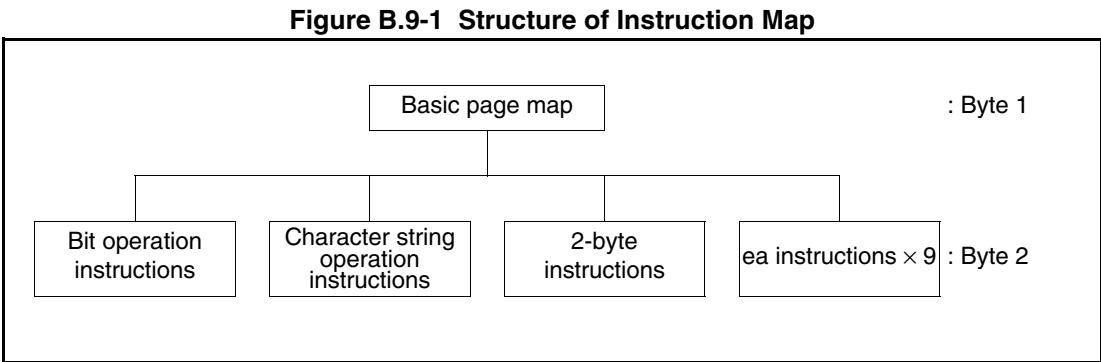
m: RW0 value (counter value), n: Loop count

See Table B.5-1 and Table B.5-2 for information on (b) and (c) in the table.

B.9 Instruction Map

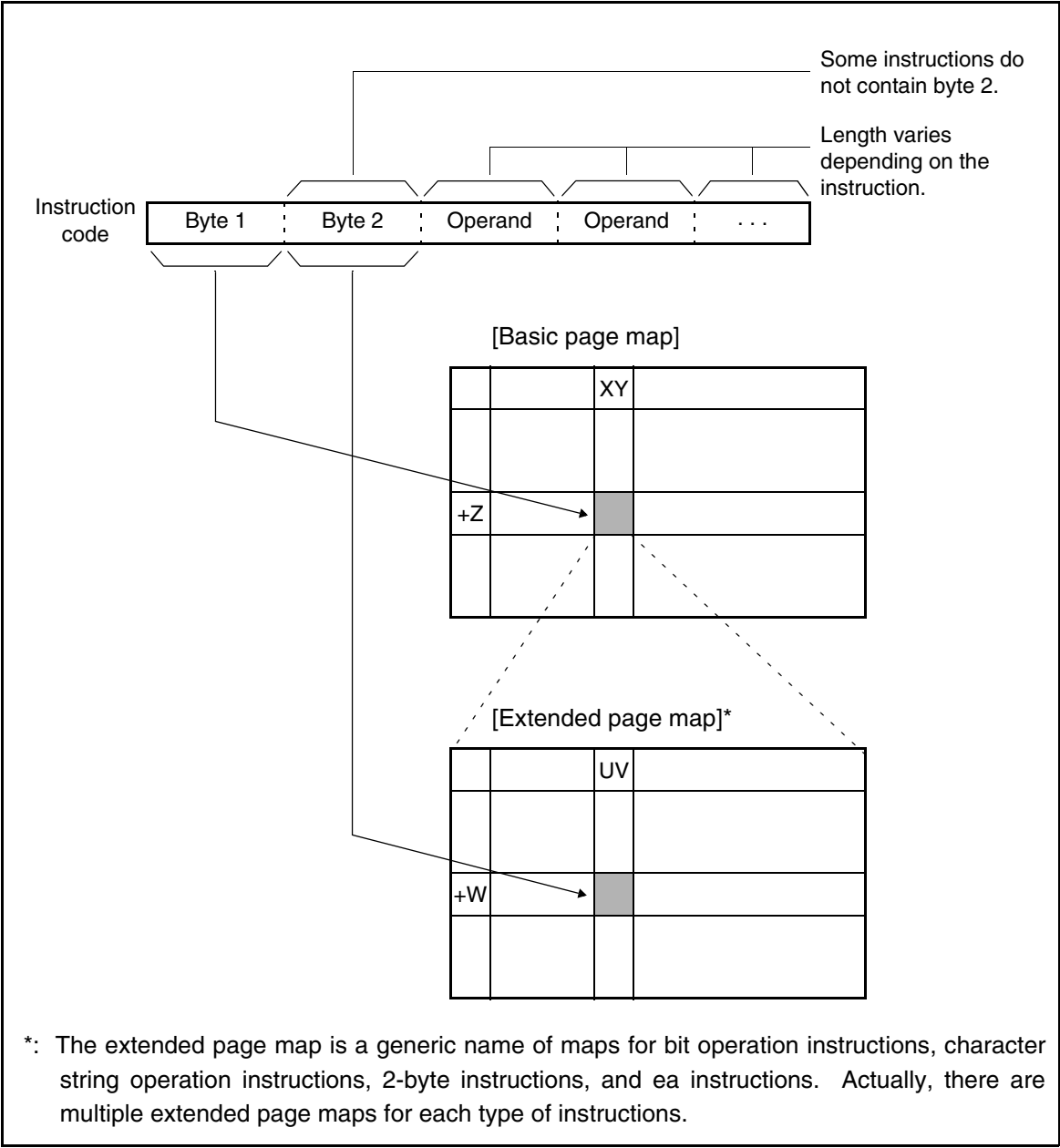
Each F²MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 to Table B.9-21 summarize the F²MC-16LX instruction map.

■ Structure of Instruction Map



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	MOVW A, @RWI+d8	MOVW @RWI+d8, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT	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				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, rel
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table B.9-4 Character String Operation Instruction Map (First Byte = 6E_H)

	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								PCB DTB	PCB DTB	PCB DTB	PCB DTB	PCB DTB			
+2	PCB, ADB								PCB DTB	PCB DTB	PCB DTB	PCB DTB	PCB DTB			
+3	PCB, SPB								PCB DTB	PCB DTB	PCB DTB	PCB DTB	PCB DTB			
+4	DTB, PCB								ADB SPB	ADB SPB	ADB SPB	ADB SPB	ADB SPB			
+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
+1	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+2	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+3	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+4	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+5	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+6	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+7	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+8	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+9	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+A	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+B	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+C	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+D	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+E	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH
+F	ROLc	ROLc	RORc	RORc	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX	XCH	XCH

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, A, @RW0+d8	MOVW RW0, A, @RW0+d8	MOVW RW0, #16, @RW0+d8, #16	MOVW RW0, #16, @RW0+d8, #16	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, A, @RW1+d8	MOVW RW1, A, @RW1+d8	MOVW RW1, #16, @RW1+d8, #16	MOVW RW1, #16, @RW1+d8, #16	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, A, @RW2+d8	MOVW RW2, A, @RW2+d8	MOVW RW2, #16, @RW2+d8, #16	MOVW RW2, #16, @RW2+d8, #16	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, A, @RW3+d8	MOVW RW3, A, @RW3+d8	MOVW RW3, #16, @RW3+d8, #16	MOVW RW3, #16, @RW3+d8, #16	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, A, @RW4+d8	MOVW RW4, A, @RW4+d8	MOVW RW4, #16, @RW4+d8, #16	MOVW RW4, #16, @RW4+d8, #16	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, A, @RW5+d8	MOVW RW5, A, @RW5+d8	MOVW RW5, #16, @RW5+d8, #16	MOVW RW5, #16, @RW5+d8, #16	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, A, @RW6+d8	MOVW RW6, A, @RW6+d8	MOVW RW6, #16, @RW6+d8, #16	MOVW RW6, #16, @RW6+d8, #16	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, A, @RW7+d8	MOVW RW7, A, @RW7+d8	MOVW RW7, #16, @RW7+d8, #16	MOVW RW7, #16, @RW7+d8, #16	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, #16, @RW0+d16, #16	MOVW @RW0, #16, @RW0+d16, #16	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, #16, @RW1+d16, #16	MOVW @RW1, #16, @RW1+d16, #16	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, #16, @RW2+d16, #16	MOVW @RW2, #16, @RW2+d16, #16	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, #16, @RW3+d16, #16	MOVW @RW3, #16, @RW3+d16, #16	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+, #16, @RW0+RW7, #16	MOVW @RW0+, #16, @RW0+RW7, #16	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+, #16, @RW1+RW7, #16	MOVW @RW1+, #16, @RW1+RW7, #16	XCHW A, RW1+, @RW1+RW7	XCHW A, RW1+, @RW1+RW7
+E	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+, #16, @RW2+d16, #16	MOVW @RW2+, #16, @RW2+d16, #16	XCHW A, RW2+, @RW2+d16	XCHW A, RW2+, @RW2+d16
+F	JMP @RW3+, @RW3+addr16	JMP @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	INCW @RW3+, @RW3+addr16	INCW @RW3+, @RW3+addr16	DECW @RW3+, @RW3+addr16	DECW @RW3+, @RW3+addr16	MOVW A, RW3+, @RW3+addr16	MOVW A, RW3+, @RW3+addr16	MOVW @RW3+, A, @RW3+addr16	MOVW @RW3+, A, @RW3+addr16	MOVW @RW3+, #16, @RW3+addr16, #16	MOVW @RW3+, #16, @RW3+addr16, #16	XCHW A, RW3+, @RW3+addr16	XCHW A, RW3+, @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+RW7	SUB A, @RW0+, @RW0+RW7	SUB A, @RW0+, @RW0+RW7	SUB A, @RW0+, @RW0+RW7	ADDC A, @RW0+, @RW0+RW7	ADDC A, @RW0+, @RW0+RW7	CMP A, @RW0+, @RW0+RW7	CMP A, @RW0+, @RW0+RW7	AND A, @RW0+, @RW0+RW7	AND A, @RW0+, @RW0+RW7	OR A, @RW0+, @RW0+RW7	OR A, @RW0+, @RW0+RW7	XOR A, @RW0+, @RW0+RW7	XOR A, @RW0+, @RW0+RW7	DBNZ @RW0+, r, W0+RW7, r	DBNZ @RW0+, r, W0+RW7, r
+D	ADD A, @RW1+, @RW1+RW7	SUB A, @RW1+, @RW1+RW7	SUB A, @RW1+, @RW1+RW7	SUB A, @RW1+, @RW1+RW7	ADDC A, @RW1+, @RW1+RW7	ADDC A, @RW1+, @RW1+RW7	CMP A, @RW1+, @RW1+RW7	CMP A, @RW1+, @RW1+RW7	AND A, @RW1+, @RW1+RW7	AND A, @RW1+, @RW1+RW7	OR A, @RW1+, @RW1+RW7	OR A, @RW1+, @RW1+RW7	XOR A, @RW1+, @RW1+RW7	XOR A, @RW1+, @RW1+RW7	DBNZ @RW1+, r, W1+RW7, r	DBNZ @RW1+, r, W1+RW7, r
+E	ADD A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	ADDC A, @RW2+, @PC+d16	ADDC A, @RW2+, @PC+d16	CMP A, @RW2+, @PC+d16	CMP A, @RW2+, @PC+d16	AND A, @RW2+, @PC+d16	AND A, @RW2+, @PC+d16	OR A, @RW2+, @PC+d16	OR A, @RW2+, @PC+d16	XOR A, @RW2+, @PC+d16	XOR A, @RW2+, @PC+d16	DBNZ @RW2+, r, PC+d16, r	DBNZ @RW2+, r, PC+d16, r
+F	ADD A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	ADDC A, @RW3+, A, addr16	ADDC A, @RW3+, A, addr16	CMP A, @RW3+, A, addr16	CMP A, @RW3+, A, addr16	AND A, @RW3+, A, addr16	AND A, @RW3+, A, addr16	OR A, @RW3+, A, addr16	OR A, @RW3+, A, addr16	XOR A, @RW3+, A, addr16	XOR A, @RW3+, A, addr16	DBNZ @RW3+, r, addr16, r	DBNZ @RW3+, r, addr16, r

Table B.9-11 ea Instruction 6 (First Byte = 75_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	SUBC 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 R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	NOT R0, @RW0+d8, A	NOT R0, @RW0+d8, A
+1	ADD R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUBC 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 R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	NOT R1, @RW1+d8, A	NOT R1, @RW1+d8, A
+2	ADD R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUBC 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 R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	NOT R2, @RW2+d8, A	NOT R2, @RW2+d8, A
+3	ADD R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUBC 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 R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	NOT R3, @RW3+d8, A	NOT R3, @RW3+d8, A
+4	ADD R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUBC 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 R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	NOT R4, @RW4+d8, A	NOT R4, @RW4+d8, A
+5	ADD R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUBC 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 R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	NOT R5, @RW5+d8, A	NOT R5, @RW5+d8, A
+6	ADD R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUBC 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 R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	NOT R6, @RW6+d8, A	NOT R6, @RW6+d8, A
+7	ADD R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUBC 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 R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	NOT R7, @RW7+d8, A	NOT R7, @RW7+d8, A
+8	ADD @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUBC 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 @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	NOT @RW0, @RW0+d16, A	NOT @RW0, @RW0+d16, A
+9	ADD @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUBC 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 @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	NOT @RW1, @RW1+d16, A	NOT @RW1, @RW1+d16, A
+A	ADD @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUBC 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 @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	NOT @RW2, @RW2+d16, A	NOT @RW2, @RW2+d16, A
+B	ADD @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUBC 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 @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	NOT @RW3, @RW3+d16, A	NOT @RW3, @RW3+d16, A
+C	ADD @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUBC 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 @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	NOT @RW0+, @RW0+RW7, A	NOT @RW0+, @RW0+RW7, A
+D	ADD @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUBC 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 @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	NOT @RW1+, @RW1+RW7, A	NOT @RW1+, @RW1+RW7, A
+E	ADD @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUBC 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 @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	NOT @RW2+, @PC+d16, A	NOT @RW2+, @PC+d16, A
+F	ADD @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUBC 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 @RW3+, A, addr16, A	OR @RW3+, A, addr16, A	OR @RW3+, A, addr16, A	XOR @RW3+, A, addr16, A	XOR @RW3+, A, addr16, A	NOT @RW3+, addr16, A	NOT @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, r1 @RW0+d8,r	DWBZ RW0, r1 @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, r1 @RW1+d8,r	DWBZ RW1, r1 @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, r1 @RW2+d8,r	DWBZ RW2, r1 @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, r1 @RW3+d8,r	DWBZ RW3, r1 @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, r1 @RW4+d8,r	DWBZ RW4, r1 @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, r1 @RW5+d8,r	DWBZ RW5, r1 @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, r1 @RW6+d8,r	DWBZ RW6, r1 @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, r1 @RW7+d8,r	DWBZ RW7, r1 @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, r1 @RW0+d16,r	DWBZ RW0, r1 @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, r1 @RW1+d16,r	DWBZ RW1, r1 @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, r1 @RW2+d16,r	DWBZ RW2, r1 @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, r1 @RW3+d16,r	DWBZ RW3, r1 @RW3+d16,r
+C	ADDW A, RW0+, @RW0+RW7	SUBW A, RW0+, @RW0+RW7	SUBW A, RW0+, @RW0+RW7	SUBW A, RW0+, @RW0+RW7	ADDCW A, RW0+, @RW0+RW7	ADDCW A, RW0+, @RW0+RW7	CMPW A, RW0+, @RW0+RW7	CMPW A, RW0+, @RW0+RW7	ANDW A, RW0+, @RW0+RW7	ANDW A, RW0+, @RW0+RW7	ORW A, RW0+, @RW0+RW7	ORW A, RW0+, @RW0+RW7	XORW A, RW0+, @RW0+RW7	XORW A, RW0+, @RW0+RW7	DWBZ RW0+, r1 @RW0+RW7,r	DWBZ RW0+, r1 @RW0+RW7,r
+D	ADDW A, RW1+, @RW1+RW7	SUBW A, RW1+, @RW1+RW7	SUBW A, RW1+, @RW1+RW7	SUBW A, RW1+, @RW1+RW7	ADDCW A, RW1+, @RW1+RW7	ADDCW A, RW1+, @RW1+RW7	CMPW A, RW1+, @RW1+RW7	CMPW A, RW1+, @RW1+RW7	ANDW A, RW1+, @RW1+RW7	ANDW A, RW1+, @RW1+RW7	ORW A, RW1+, @RW1+RW7	ORW A, RW1+, @RW1+RW7	XORW A, RW1+, @RW1+RW7	XORW A, RW1+, @RW1+RW7	DWBZ RW1+, r1 @RW1+RW7,r	DWBZ RW1+, r1 @RW1+RW7,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+, r1 @PC+d16,r	DWBZ RW2+, r1 @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+, r1 addr 16,r	DWBZ RW3+, r1 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, A, RW0' @RW0+d8	MULUW A, A, RW0' @RW0+d8	MUL A, R0' @RW0+d8	MUL A, R0' @RW0+d8	MULW A, A, RW0' @RW0+d8	MULW A, A, RW0' @RW0+d8	DIVU A, R0' @RW0+d8	DIVU A, R0' @RW0+d8	DIVUW A, A, RW0' @RW0+d8	DIVUW A, A, RW0' @RW0+d8	DIV A, R0' @RW0+d8	DIV A, R0' @RW0+d8	DIVW A, A, RW0' @RW0+d8	DIVW A, A, RW0' @RW0+d8
+1	MULU A, R1' @RW1+d8	MULU A, R1' @RW1+d8	MULUW A, A, RW1' @RW1+d8	MULUW A, A, RW1' @RW1+d8	MUL A, R1' @RW1+d8	MUL A, R1' @RW1+d8	MULW A, A, RW1' @RW1+d8	MULW A, A, RW1' @RW1+d8	DIVU A, R1' @RW1+d8	DIVU A, R1' @RW1+d8	DIVUW A, A, RW1' @RW1+d8	DIVUW A, A, RW1' @RW1+d8	DIV A, R1' @RW1+d8	DIV A, R1' @RW1+d8	DIVW A, A, RW1' @RW1+d8	DIVW A, A, RW1' @RW1+d8
+2	MULU A, R2' @RW2+d8	MULU A, R2' @RW2+d8	MULUW A, A, RW2' @RW2+d8	MULUW A, A, RW2' @RW2+d8	MUL A, R2' @RW2+d8	MUL A, R2' @RW2+d8	MULW A, A, RW2' @RW2+d8	MULW A, A, RW2' @RW2+d8	DIVU A, R2' @RW2+d8	DIVU A, R2' @RW2+d8	DIVUW A, A, RW2' @RW2+d8	DIVUW A, A, RW2' @RW2+d8	DIV A, R2' @RW2+d8	DIV A, R2' @RW2+d8	DIVW A, A, RW2' @RW2+d8	DIVW A, A, RW2' @RW2+d8
+3	MULU A, R3' @RW3+d8	MULU A, R3' @RW3+d8	MULUW A, A, RW3' @RW3+d8	MULUW A, A, RW3' @RW3+d8	MUL A, R3' @RW3+d8	MUL A, R3' @RW3+d8	MULW A, A, RW3' @RW3+d8	MULW A, A, RW3' @RW3+d8	DIVU A, R3' @RW3+d8	DIVU A, R3' @RW3+d8	DIVUW A, A, RW3' @RW3+d8	DIVUW A, A, RW3' @RW3+d8	DIV A, R3' @RW3+d8	DIV A, R3' @RW3+d8	DIVW A, A, RW3' @RW3+d8	DIVW A, A, RW3' @RW3+d8
+4	MULU A, R4' @RW4+d8	MULU A, R4' @RW4+d8	MULUW A, A, RW4' @RW4+d8	MULUW A, A, RW4' @RW4+d8	MUL A, R4' @RW4+d8	MUL A, R4' @RW4+d8	MULW A, A, RW4' @RW4+d8	MULW A, A, RW4' @RW4+d8	DIVU A, R4' @RW4+d8	DIVU A, R4' @RW4+d8	DIVUW A, A, RW4' @RW4+d8	DIVUW A, A, RW4' @RW4+d8	DIV A, R4' @RW4+d8	DIV A, R4' @RW4+d8	DIVW A, A, RW4' @RW4+d8	DIVW A, A, RW4' @RW4+d8
+5	MULU A, R5' @RW5+d8	MULU A, R5' @RW5+d8	MULUW A, A, RW5' @RW5+d8	MULUW A, A, RW5' @RW5+d8	MUL A, R5' @RW5+d8	MUL A, R5' @RW5+d8	MULW A, A, RW5' @RW5+d8	MULW A, A, RW5' @RW5+d8	DIVU A, R5' @RW5+d8	DIVU A, R5' @RW5+d8	DIVUW A, A, RW5' @RW5+d8	DIVUW A, A, RW5' @RW5+d8	DIV A, R5' @RW5+d8	DIV A, R5' @RW5+d8	DIVW A, A, RW5' @RW5+d8	DIVW A, A, RW5' @RW5+d8
+6	MULU A, R6' @RW6+d8	MULU A, R6' @RW6+d8	MULUW A, A, RW6' @RW6+d8	MULUW A, A, RW6' @RW6+d8	MUL A, R6' @RW6+d8	MUL A, R6' @RW6+d8	MULW A, A, RW6' @RW6+d8	MULW A, A, RW6' @RW6+d8	DIVU A, R6' @RW6+d8	DIVU A, R6' @RW6+d8	DIVUW A, A, RW6' @RW6+d8	DIVUW A, A, RW6' @RW6+d8	DIV A, R6' @RW6+d8	DIV A, R6' @RW6+d8	DIVW A, A, RW6' @RW6+d8	DIVW A, A, RW6' @RW6+d8
+7	MULU A, R7' @RW7+d8	MULU A, R7' @RW7+d8	MULUW A, A, RW7' @RW7+d8	MULUW A, A, RW7' @RW7+d8	MUL A, R7' @RW7+d8	MUL A, R7' @RW7+d8	MULW A, A, RW7' @RW7+d8	MULW A, A, RW7' @RW7+d8	DIVU A, R7' @RW7+d8	DIVU A, R7' @RW7+d8	DIVUW A, A, RW7' @RW7+d8	DIVUW A, A, RW7' @RW7+d8	DIV A, R7' @RW7+d8	DIV A, R7' @RW7+d8	DIVW A, A, RW7' @RW7+d8	DIVW A, A, RW7' @RW7+d8
+8	MULU A, @RW0' @RW0+d16	MULU A, @RW0' @RW0+d16	MULUW A, A, @RW0' @RW0+d16	MULUW A, A, @RW0' @RW0+d16	MUL A, @RW0' @RW0+d16	MUL A, @RW0' @RW0+d16	MULW A, A, @RW0' @RW0+d16	MULW A, A, @RW0' @RW0+d16	DIVU A, @RW0' @RW0+d16	DIVU A, @RW0' @RW0+d16	DIVUW A, A, @RW0' @RW0+d16	DIVUW A, A, @RW0' @RW0+d16	DIV A, @RW0' @RW0+d16	DIV A, @RW0' @RW0+d16	DIVW A, A, @RW0' @RW0+d16	DIVW A, A, @RW0' @RW0+d16
+9	MULU A, @RW1' @RW1+d16	MULU A, @RW1' @RW1+d16	MULUW A, A, @RW1' @RW1+d16	MULUW A, A, @RW1' @RW1+d16	MUL A, @RW1' @RW1+d16	MUL A, @RW1' @RW1+d16	MULW A, A, @RW1' @RW1+d16	MULW A, A, @RW1' @RW1+d16	DIVU A, @RW1' @RW1+d16	DIVU A, @RW1' @RW1+d16	DIVUW A, A, @RW1' @RW1+d16	DIVUW A, A, @RW1' @RW1+d16	DIV A, @RW1' @RW1+d16	DIV A, @RW1' @RW1+d16	DIVW A, A, @RW1' @RW1+d16	DIVW A, A, @RW1' @RW1+d16
+A	MULU A, @RW2' @RW2+d16	MULU A, @RW2' @RW2+d16	MULUW A, A, @RW2' @RW2+d16	MULUW A, A, @RW2' @RW2+d16	MUL A, @RW2' @RW2+d16	MUL A, @RW2' @RW2+d16	MULW A, A, @RW2' @RW2+d16	MULW A, A, @RW2' @RW2+d16	DIVU A, @RW2' @RW2+d16	DIVU A, @RW2' @RW2+d16	DIVUW A, A, @RW2' @RW2+d16	DIVUW A, A, @RW2' @RW2+d16	DIV A, @RW2' @RW2+d16	DIV A, @RW2' @RW2+d16	DIVW A, A, @RW2' @RW2+d16	DIVW A, A, @RW2' @RW2+d16
+B	MULU A, @RW3' @RW3+d16	MULU A, @RW3' @RW3+d16	MULUW A, A, @RW3' @RW3+d16	MULUW A, A, @RW3' @RW3+d16	MUL A, @RW3' @RW3+d16	MUL A, @RW3' @RW3+d16	MULW A, A, @RW3' @RW3+d16	MULW A, A, @RW3' @RW3+d16	DIVU A, @RW3' @RW3+d16	DIVU A, @RW3' @RW3+d16	DIVUW A, A, @RW3' @RW3+d16	DIVUW A, A, @RW3' @RW3+d16	DIV A, @RW3' @RW3+d16	DIV A, @RW3' @RW3+d16	DIVW A, A, @RW3' @RW3+d16	DIVW A, A, @RW3' @RW3+d16
+C	MULU A, @RW0+ @RW0-RW7	MULU A, @RW0+ @RW0-RW7	MULUW A, A, @RW0+ @RW0-RW7	MULUW A, A, @RW0+ @RW0-RW7	MUL A, @RW0+ @RW0-RW7	MUL A, @RW0+ @RW0-RW7	MULW A, A, @RW0+ @RW0-RW7	MULW A, A, @RW0+ @RW0-RW7	DIVU A, @RW0+ @RW0-RW7	DIVU A, @RW0+ @RW0-RW7	DIVUW A, A, @RW0+ @RW0-RW7	DIVUW A, A, @RW0+ @RW0-RW7	DIV A, @RW0+ @RW0-RW7	DIV A, @RW0+ @RW0-RW7	DIVW A, A, @RW0+ @RW0-RW7	DIVW A, A, @RW0+ @RW0-RW7
+D	MULU A, @RW1+ @RW1-RW7	MULU A, @RW1+ @RW1-RW7	MULUW A, A, @RW1+ @RW1-RW7	MULUW A, A, @RW1+ @RW1-RW7	MUL A, @RW1+ @RW1-RW7	MUL A, @RW1+ @RW1-RW7	MULW A, A, @RW1+ @RW1-RW7	MULW A, A, @RW1+ @RW1-RW7	DIVU A, @RW1+ @RW1-RW7	DIVU A, @RW1+ @RW1-RW7	DIVUW A, A, @RW1+ @RW1-RW7	DIVUW A, A, @RW1+ @RW1-RW7	DIV A, @RW1+ @RW1-RW7	DIV A, @RW1+ @RW1-RW7	DIVW A, A, @RW1+ @RW1-RW7	DIVW A, A, @RW1+ @RW1-RW7
+E	MULU A, @RW2+ @PC+d16	MULU A, @RW2+ @PC+d16	MULUW A, A, @RW2+ @PC+d16	MULUW A, A, @RW2+ @PC+d16	MUL A, @RW2+ @PC+d16	MUL A, @RW2+ @PC+d16	MULW A, A, @RW2+ @PC+d16	MULW A, A, @RW2+ @PC+d16	DIVU A, @RW2+ @PC+d16	DIVU A, @RW2+ @PC+d16	DIVUW A, A, @RW2+ @PC+d16	DIVUW A, A, @RW2+ @PC+d16	DIV A, @RW2+ @PC+d16	DIV A, @RW2+ @PC+d16	DIVW A, A, @RW2+ @PC+d16	DIVW A, A, @RW2+ @PC+d16
+F	MULU A, @RW3+ addr16	MULU A, @RW3+ addr16	MULUW A, A, @RW3+ addr16	MULUW A, A, @RW3+ addr16	MUL A, @RW3+ addr16	MUL A, @RW3+ addr16	MULW A, A, @RW3+ addr16	MULW A, A, @RW3+ addr16	DIVU A, @RW3+ addr16	DIVU A, @RW3+ addr16	DIVUW A, A, @RW3+ addr16	DIVUW A, A, @RW3+ addr16	DIV A, @RW3+ addr16	DIV A, @RW3+ addr16	DIVW A, A, @RW3+ addr16	DIVW A, A, @RW3+ addr16

Table B.9-15 MOVEA RWi, ea Instruction (First Byte = 79_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVEA RW0,RW0,@RW0-d8	MOVEA RW0,RW0	MOVEA RW1,RW0,@RW0-d8	MOVEA RW1,RW0	MOVEA RW2,RW0	MOVEA RW2,RW0	MOVEA RW3,RW0	MOVEA RW3,RW0	MOVEA RW4,RW0	MOVEA RW4,RW0	MOVEA RW5,RW0	MOVEA RW5,RW0	MOVEA RW6,RW0	MOVEA RW6,RW0	MOVEA RW7,RW0	MOVEA RW7,RW0
+1	MOVEA RW0,RW1,@RW1-d8	MOVEA RW0,RW1	MOVEA RW1,RW1	MOVEA RW1,RW1	MOVEA RW2,RW1	MOVEA RW2,RW1	MOVEA RW3,RW1	MOVEA RW3,RW1	MOVEA RW4,RW1	MOVEA RW4,RW1	MOVEA RW5,RW1	MOVEA RW5,RW1	MOVEA RW6,RW1	MOVEA RW6,RW1	MOVEA RW7,RW1	MOVEA RW7,RW1
+2	MOVEA RW0,RW2,@RW2-d8	MOVEA RW0,RW2	MOVEA RW1,RW2	MOVEA RW1,RW2	MOVEA RW2,RW2	MOVEA RW2,RW2	MOVEA RW3,RW2	MOVEA RW3,RW2	MOVEA RW4,RW2	MOVEA RW4,RW2	MOVEA RW5,RW2	MOVEA RW5,RW2	MOVEA RW6,RW2	MOVEA RW6,RW2	MOVEA RW7,RW2	MOVEA RW7,RW2
+3	MOVEA RW0,RW3,@RW3-d8	MOVEA RW0,RW3	MOVEA RW1,RW3	MOVEA RW1,RW3	MOVEA RW2,RW3	MOVEA RW2,RW3	MOVEA RW3,RW3	MOVEA RW3,RW3	MOVEA RW4,RW3	MOVEA RW4,RW3	MOVEA RW5,RW3	MOVEA RW5,RW3	MOVEA RW6,RW3	MOVEA RW6,RW3	MOVEA RW7,RW3	MOVEA RW7,RW3
+4	MOVEA RW0,RW4,@RW4-d8	MOVEA RW0,RW4	MOVEA RW1,RW4	MOVEA RW1,RW4	MOVEA RW2,RW4	MOVEA RW2,RW4	MOVEA RW3,RW4	MOVEA RW3,RW4	MOVEA RW4,RW4	MOVEA RW4,RW4	MOVEA RW5,RW4	MOVEA RW5,RW4	MOVEA RW6,RW4	MOVEA RW6,RW4	MOVEA RW7,RW4	MOVEA RW7,RW4
+5	MOVEA RW0,RW5,@RW5-d8	MOVEA RW0,RW5	MOVEA RW1,RW5	MOVEA RW1,RW5	MOVEA RW2,RW5	MOVEA RW2,RW5	MOVEA RW3,RW5	MOVEA RW3,RW5	MOVEA RW4,RW5	MOVEA RW4,RW5	MOVEA RW5,RW5	MOVEA RW5,RW5	MOVEA RW6,RW5	MOVEA RW6,RW5	MOVEA RW7,RW5	MOVEA RW7,RW5
+6	MOVEA RW0,RW6,@RW6-d8	MOVEA RW0,RW6	MOVEA RW1,RW6	MOVEA RW1,RW6	MOVEA RW2,RW6	MOVEA RW2,RW6	MOVEA RW3,RW6	MOVEA RW3,RW6	MOVEA RW4,RW6	MOVEA RW4,RW6	MOVEA RW5,RW6	MOVEA RW5,RW6	MOVEA RW6,RW6	MOVEA RW6,RW6	MOVEA RW7,RW6	MOVEA RW7,RW6
+7	MOVEA RW0,RW7,@RW7-d8	MOVEA RW0,RW7	MOVEA RW1,RW7	MOVEA RW1,RW7	MOVEA RW2,RW7	MOVEA RW2,RW7	MOVEA RW3,RW7	MOVEA RW3,RW7	MOVEA RW4,RW7	MOVEA RW4,RW7	MOVEA RW5,RW7	MOVEA RW5,RW7	MOVEA RW6,RW7	MOVEA RW6,RW7	MOVEA RW7,RW7	MOVEA RW7,RW7
+8	MOVEA RW0,@RW0	MOVEA RW0	MOVEA RW1,@RW0	MOVEA RW1	MOVEA RW2,@RW0	MOVEA RW2	MOVEA RW3	MOVEA RW3	MOVEA RW4	MOVEA RW4	MOVEA RW5	MOVEA RW5	MOVEA RW6	MOVEA RW6	MOVEA RW7	MOVEA RW7
+9	MOVEA RW0,RW1	MOVEA RW0	MOVEA RW1,RW1	MOVEA RW1	MOVEA RW2,RW1	MOVEA RW2	MOVEA RW3,RW1	MOVEA RW3	MOVEA RW4,RW1	MOVEA RW4	MOVEA RW5,RW1	MOVEA RW5	MOVEA RW6,RW1	MOVEA RW6	MOVEA RW7,RW1	MOVEA RW7
+A	MOVEA RW0,RW2	MOVEA RW0	MOVEA RW1,RW2	MOVEA RW1	MOVEA RW2,RW2	MOVEA RW2	MOVEA RW3,RW2	MOVEA RW3	MOVEA RW4,RW2	MOVEA RW4	MOVEA RW5,RW2	MOVEA RW5	MOVEA RW6,RW2	MOVEA RW6	MOVEA RW7,RW2	MOVEA RW7
+B	MOVEA RW0,RW3	MOVEA RW0	MOVEA RW1,RW3	MOVEA RW1	MOVEA RW2,RW3	MOVEA RW2	MOVEA RW3,RW3	MOVEA RW3	MOVEA RW4,RW3	MOVEA RW4	MOVEA RW5,RW3	MOVEA RW5	MOVEA RW6,RW3	MOVEA RW6	MOVEA RW7,RW3	MOVEA RW7
+C	MOVEA RW0,RW4	MOVEA RW0	MOVEA RW1,RW4	MOVEA RW1	MOVEA RW2,RW4	MOVEA RW2	MOVEA RW3,RW4	MOVEA RW3	MOVEA RW4,RW4	MOVEA RW4	MOVEA RW5,RW4	MOVEA RW5	MOVEA RW6,RW4	MOVEA RW6	MOVEA RW7,RW4	MOVEA RW7
+D	MOVEA RW0,RW5	MOVEA RW0	MOVEA RW1,RW5	MOVEA RW1	MOVEA RW2,RW5	MOVEA RW2	MOVEA RW3,RW5	MOVEA RW3	MOVEA RW4,RW5	MOVEA RW4	MOVEA RW5,RW5	MOVEA RW5	MOVEA RW6,RW5	MOVEA RW6	MOVEA RW7,RW5	MOVEA RW7
+E	MOVEA RW0,RW6	MOVEA RW0	MOVEA RW1,RW6	MOVEA RW1	MOVEA RW2,RW6	MOVEA RW2	MOVEA RW3,RW6	MOVEA RW3	MOVEA RW4,RW6	MOVEA RW4	MOVEA RW5,RW6	MOVEA RW5	MOVEA RW6,RW6	MOVEA RW6	MOVEA RW7,RW6	MOVEA RW7
+F	MOVEA RW0,RW7	MOVEA RW0	MOVEA RW1,RW7	MOVEA RW1	MOVEA RW2,RW7	MOVEA RW2	MOVEA RW3,RW7	MOVEA RW3	MOVEA RW4,RW7	MOVEA RW4	MOVEA RW5,RW7	MOVEA RW5	MOVEA RW6,RW7	MOVEA RW6	MOVEA RW7,RW7	MOVEA RW7

Table B.9-16 MOV Ri, ea Instruction (First Byte = 7A_H)

[illegible]

Table B.9-17 MOVW RWi, ea Instruction (First Byte = 7B_H)

[illegible]

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)

	00	10	20	30	40	50	60	70	80	90	A	B0	C0	D0	E0	F0
+0	XCH R0, R0' @RW0+d8	XCH R0, R0' @RW0+d8	XCH R1, R1' @RW0+d8	XCH R2, R2' @RW0+d8	XCH R3, R3' @RW0+d8	XCH R4, R4' @RW0+d8	XCH R5, R5' @RW0+d8	XCH R6, R6' @RW0+d8	XCH R7, R7' @RW0+d8	XCH R8, R8' @RW0+d8	XCH R9, R9' @RW0+d8	XCH R10, R10' @RW0+d8	XCH R11, R11' @RW0+d8	XCH R12, R12' @RW0+d8	XCH R13, R13' @RW0+d8	XCH R14, R14' @RW0+d8
+1	XCH R0, R1' @RW1+d8	XCH R0, R1' @RW1+d8	XCH R1, R2' @RW2+d8	XCH R2, R3' @RW3+d8	XCH R3, R4' @RW4+d8	XCH R4, R5' @RW5+d8	XCH R5, R6' @RW6+d8	XCH R6, R7' @RW7+d8	XCH R7, R8' @RW8+d8	XCH R8, R9' @RW9+d8	XCH R9, R10' @RW10+d8	XCH R10, R11' @RW11+d8	XCH R11, R12' @RW12+d8	XCH R12, R13' @RW13+d8	XCH R13, R14' @RW14+d8	XCH R14, R15' @RW15+d8
+2	XCH R0, R2' @RW2+d8	XCH R0, R2' @RW2+d8	XCH R1, R3' @RW3+d8	XCH R2, R4' @RW4+d8	XCH R3, R5' @RW5+d8	XCH R4, R6' @RW6+d8	XCH R5, R7' @RW7+d8	XCH R6, R8' @RW8+d8	XCH R7, R9' @RW9+d8	XCH R8, R10' @RW10+d8	XCH R9, R11' @RW11+d8	XCH R10, R12' @RW12+d8	XCH R11, R13' @RW13+d8	XCH R12, R14' @RW14+d8	XCH R13, R15' @RW15+d8	XCH R14, R16' @RW16+d8
+3	XCH R0, R3' @RW3+d8	XCH R0, R3' @RW3+d8	XCH R1, R4' @RW4+d8	XCH R2, R5' @RW5+d8	XCH R3, R6' @RW6+d8	XCH R4, R7' @RW7+d8	XCH R5, R8' @RW8+d8	XCH R6, R9' @RW9+d8	XCH R7, R10' @RW10+d8	XCH R8, R11' @RW11+d8	XCH R9, R12' @RW12+d8	XCH R10, R13' @RW13+d8	XCH R11, R14' @RW14+d8	XCH R12, R15' @RW15+d8	XCH R13, R16' @RW16+d8	XCH R14, R17' @RW17+d8
+4	XCH R0, R4' @RW4+d8	XCH R0, R4' @RW4+d8	XCH R1, R5' @RW5+d8	XCH R2, R6' @RW6+d8	XCH R3, R7' @RW7+d8	XCH R4, R8' @RW8+d8	XCH R5, R9' @RW9+d8	XCH R6, R10' @RW10+d8	XCH R7, R11' @RW11+d8	XCH R8, R12' @RW12+d8	XCH R9, R13' @RW13+d8	XCH R10, R14' @RW14+d8	XCH R11, R15' @RW15+d8	XCH R12, R16' @RW16+d8	XCH R13, R17' @RW17+d8	XCH R14, R18' @RW18+d8
+5	XCH R0, R5' @RW5+d8	XCH R0, R5' @RW5+d8	XCH R1, R6' @RW6+d8	XCH R2, R7' @RW7+d8	XCH R3, R8' @RW8+d8	XCH R4, R9' @RW9+d8	XCH R5, R10' @RW10+d8	XCH R6, R11' @RW11+d8	XCH R7, R12' @RW12+d8	XCH R8, R13' @RW13+d8	XCH R9, R14' @RW14+d8	XCH R10, R15' @RW15+d8	XCH R11, R16' @RW16+d8	XCH R12, R17' @RW17+d8	XCH R13, R18' @RW18+d8	XCH R14, R19' @RW19+d8
+6	XCH R0, R6' @RW6+d8	XCH R0, R6' @RW6+d8	XCH R1, R7' @RW7+d8	XCH R2, R8' @RW8+d8	XCH R3, R9' @RW9+d8	XCH R4, R10' @RW10+d8	XCH R5, R11' @RW11+d8	XCH R6, R12' @RW12+d8	XCH R7, R13' @RW13+d8	XCH R8, R14' @RW14+d8	XCH R9, R15' @RW15+d8	XCH R10, R16' @RW16+d8	XCH R11, R17' @RW17+d8	XCH R12, R18' @RW18+d8	XCH R13, R19' @RW19+d8	XCH R14, R20' @RW20+d8
+7	XCH R0, R7' @RW7+d8	XCH R0, R7' @RW7+d8	XCH R1, R8' @RW8+d8	XCH R2, R9' @RW9+d8	XCH R3, R10' @RW10+d8	XCH R4, R11' @RW11+d8	XCH R5, R12' @RW12+d8	XCH R6, R13' @RW13+d8	XCH R7, R14' @RW14+d8	XCH R8, R15' @RW15+d8	XCH R9, R16' @RW16+d8	XCH R10, R17' @RW17+d8	XCH R11, R18' @RW18+d8	XCH R12, R19' @RW19+d8	XCH R13, R20' @RW20+d8	XCH R14, R21' @RW21+d8
+8	XCH R0, @RW0' @RW0+d16	XCH R0, @RW0' @RW0+d16	XCH R1, @RW1' @RW1+d16	XCH R2, @RW2' @RW2+d16	XCH R3, @RW3' @RW3+d16	XCH R4, @RW4' @RW4+d16	XCH R5, @RW5' @RW5+d16	XCH R6, @RW6' @RW6+d16	XCH R7, @RW7' @RW7+d16	XCH R8, @RW8' @RW8+d16	XCH R9, @RW9' @RW9+d16	XCH R10, @RW10' @RW10+d16	XCH R11, @RW11' @RW11+d16	XCH R12, @RW12' @RW12+d16	XCH R13, @RW13' @RW13+d16	XCH R14, @RW14' @RW14+d16
+9	XCH R0, @RW1' @RW1+d16	XCH R0, @RW1' @RW1+d16	XCH R1, @RW2' @RW2+d16	XCH R2, @RW3' @RW3+d16	XCH R3, @RW4' @RW4+d16	XCH R4, @RW5' @RW5+d16	XCH R5, @RW6' @RW6+d16	XCH R6, @RW7' @RW7+d16	XCH R7, @RW8' @RW8+d16	XCH R8, @RW9' @RW9+d16	XCH R9, @RW10' @RW10+d16	XCH R10, @RW11' @RW11+d16	XCH R11, @RW12' @RW12+d16	XCH R12, @RW13' @RW13+d16	XCH R13, @RW14' @RW14+d16	XCH R14, @RW15' @RW15+d16
+A	XCH R0, @RW2' @RW2+d16, A	XCH R0, @RW2' @RW2+d16, A	XCH R1, @RW3' @RW3+d16	XCH R2, @RW4' @RW4+d16	XCH R3, @RW5' @RW5+d16	XCH R4, @RW6' @RW6+d16	XCH R5, @RW7' @RW7+d16	XCH R6, @RW8' @RW8+d16	XCH R7, @RW9' @RW9+d16	XCH R8, @RW10' @RW10+d16	XCH R9, @RW11' @RW11+d16	XCH R10, @RW12' @RW12+d16	XCH R11, @RW13' @RW13+d16	XCH R12, @RW14' @RW14+d16	XCH R13, @RW15' @RW15+d16	XCH R14, @RW16' @RW16+d16
+B	XCH R0, @RW3' @RW3+d16	XCH R0, @RW3' @RW3+d16	XCH R1, @RW4' @RW4+d16	XCH R2, @RW5' @RW5+d16	XCH R3, @RW6' @RW6+d16	XCH R4, @RW7' @RW7+d16	XCH R5, @RW8' @RW8+d16	XCH R6, @RW9' @RW9+d16	XCH R7, @RW10' @RW10+d16	XCH R8, @RW11' @RW11+d16	XCH R9, @RW12' @RW12+d16	XCH R10, @RW13' @RW13+d16	XCH R11, @RW14' @RW14+d16	XCH R12, @RW15' @RW15+d16	XCH R13, @RW16' @RW16+d16	XCH R14, @RW17' @RW17+d16
+C	XCH R0, @RW0+ @RW0+RW7	XCH R0, @RW0+ @RW0+RW7	XCH R1, @RW1+ @RW1+RW7	XCH R2, @RW2+ @RW2+RW7	XCH R3, @RW3+ @RW3+RW7	XCH R4, @RW4+ @RW4+RW7	XCH R5, @RW5+ @RW5+RW7	XCH R6, @RW6+ @RW6+RW7	XCH R7, @RW7+ @RW7+RW7	XCH R8, @RW8+ @RW8+RW7	XCH R9, @RW9+ @RW9+RW7	XCH R10, @RW10+ @RW10+RW7	XCH R11, @RW11+ @RW11+RW7	XCH R12, @RW12+ @RW12+RW7	XCH R13, @RW13+ @RW13+RW7	XCH R14, @RW14+ @RW14+RW7
+D	XCH R0, @RW1+ @RW1+RW7	XCH R0, @RW1+ @RW1+RW7	XCH R1, @RW2+ @RW2+RW7	XCH R2, @RW3+ @RW3+RW7	XCH R3, @RW4+ @RW4+RW7	XCH R4, @RW5+ @RW5+RW7	XCH R5, @RW6+ @RW6+RW7	XCH R6, @RW7+ @RW7+RW7	XCH R7, @RW8+ @RW8+RW7	XCH R8, @RW9+ @RW9+RW7	XCH R9, @RW10+ @RW10+RW7	XCH R10, @RW11+ @RW11+RW7	XCH R11, @RW12+ @RW12+RW7	XCH R12, @RW13+ @RW13+RW7	XCH R13, @RW14+ @RW14+RW7	XCH R14, @RW15+ @RW15+RW7
+E	XCH R0, @RW2+ @PC+d16	XCH R0, @RW2+ @PC+d16	XCH R1, @RW3+ @PC+d16	XCH R2, @RW4+ @PC+d16	XCH R3, @RW5+ @PC+d16	XCH R4, @RW6+ @PC+d16	XCH R5, @RW7+ @PC+d16	XCH R6, @RW8+ @PC+d16	XCH R7, @RW9+ @PC+d16	XCH R8, @RW10+ @PC+d16	XCH R9, @RW11+ @PC+d16	XCH R10, @RW12+ @PC+d16	XCH R11, @RW13+ @PC+d16	XCH R12, @RW14+ @PC+d16	XCH R13, @RW15+ @PC+d16	XCH R14, @RW16+ @PC+d16
+F	XCH R0, @RW3+ @PC+d16	XCH R0, @RW3+ @PC+d16	XCH R1, @RW4+ @PC+d16	XCH R2, @RW5+ @PC+d16	XCH R3, @RW6+ @PC+d16	XCH R4, @RW7+ @PC+d16	XCH R5, @RW8+ @PC+d16	XCH R6, @RW9+ @PC+d16	XCH R7, @RW10+ @PC+d16	XCH R8, @RW11+ @PC+d16	XCH R9, @RW12+ @PC+d16	XCH R10, @RW13+ @PC+d16	XCH R11, @RW14+ @PC+d16	XCH R12, @RW15+ @PC+d16	XCH R13, @RW16+ @PC+d16	XCH R14, @RW17+ @PC+d16

Table B.9-21 XCHW RWi, ea Instruction (First Byte = 7F_H)

[illegible]

APPENDIX C Programming the OTPROM

The OTPROM for MB90P553A supports the functions equivalent to MBM27C1000A in EPROM mode.

A dedicated adapter socket allows the OTPROM to be programmed using the general-purpose EPROM programmer. Electronic signature (device identification code) mode, however, is not supported.

■ Dedicated Adapter Socket

Table C-1 Adapter Socket Used Only for Programming the OTPROM

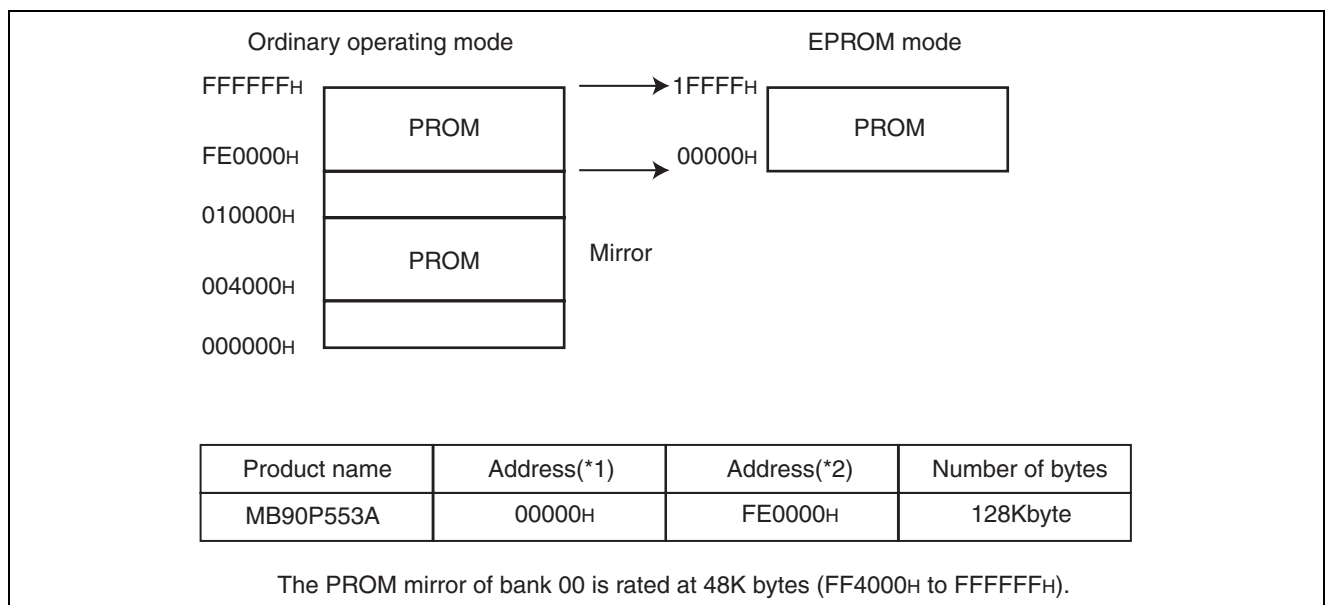
Package name	Suitable adapter model	Manufacturer
QFP-100	ROM-100QF-32DP-16L	Sun Hayato Inc.

■ Procedure for Programming the OTPROM

- (1) Set the EPROM programmer for MBM27C1000A.
- (2) Load address(*1) 1FFFF_H of the EPROM programmer with program data. (ROM address(*2) FFFFF_H in operating mode corresponds to address(*1) 1FFFF_H in EPROM programming mode.)

Figure C-1 illustrates the memory space in EPROM mode.

Figure C-1 Memory Space in EPROM Mode



- (3) Set MB90P553A in the adapter socket. Mount the adapter socket on the EPROM programmer. In this case, note the device and adapter socket directions.
- (4) Program the OTPROM.

Notes:

- The mask ROM products (MB90553A/B and MB90552A/B) do not support EPROM mode and, therefore, cannot be read by the EPROM programmer.
 - When purchasing the EPROM programmer, contact the appropriate marketing department.
-

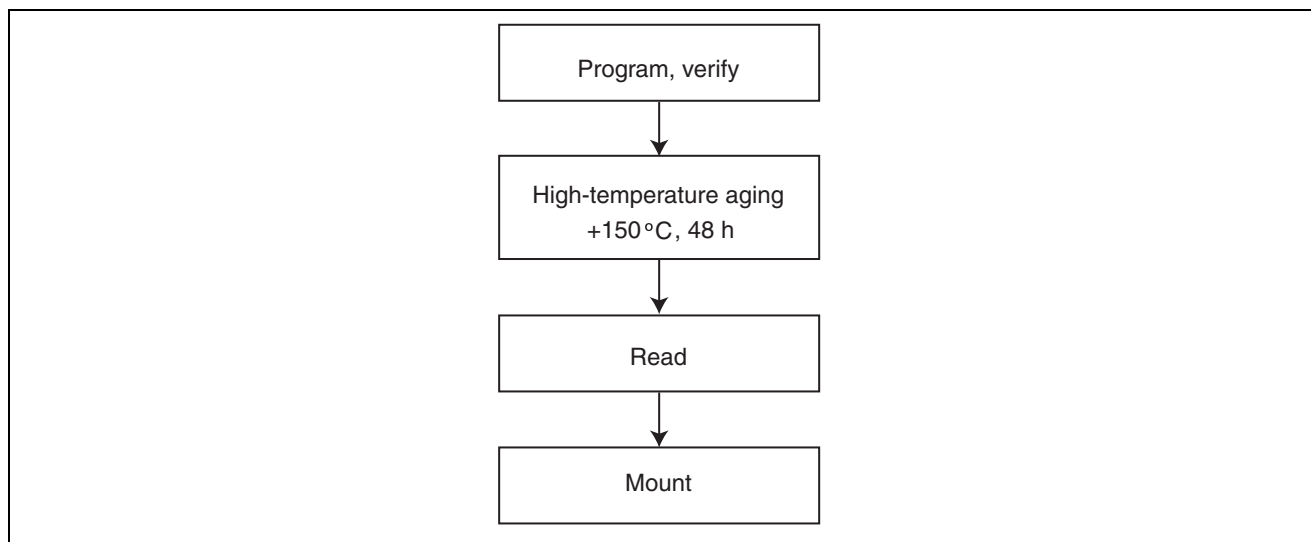
■ Programming Mode

The default status of all bits of MB90P553A is "1". To set information, selectively program a desired bit with "0". A value of 1 cannot be written electronically.

■ Screening Conditions Recommended for the OTPROM

For a product not provided with the OTPROM microcontroller program, high-temperature aging is recommended as the screening method before installation.

Figure C-2 Screening Conditions Recommended for the OTPROM



■ OTPROM Programming Yield

For a program not provided with the OTPROM microcontroller program, a programming test on all bits cannot be carried out. Therefore, the programming yield may not always be 100 percent.

INDEX

**The index follows on the next page.
This is listed in alphabetic order.**

Index

Numerics

16-bit Data Bus	
Status of Each Pin in the External Bus 16-bit Data Bus Mode	106
16-bit Free-run Timer	
16-bit Free-run Timer (x 1)	160
16-bit Free-run Timer Count Timing	175
16-bit Free-run Timer Operations.....	174
16-bit I/O Timer	
16-bit I/O Timer Block Diagram.....	162
16-bit I/O Timer Registers.....	163
16-bit Input Capture	
16-bit Input Capture Operations.....	179
16-bit Output Compare	
16-bit Output Compare Operations.....	176
16-bit Output Compare Timing.....	177
16-bit Reload Register	
16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)	187
16-bit Reload Timer	
Block Diagram of the 16-bit Reload Timer (with the Event Count Function).....	182
Overview of the 16-bit Reload Timer (with the Event Count Function)	182
Registers of the 16-bit Reload Timer (with the Event Count Function)	183
16-bit Timer Register	
16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)	187
1M-bit Flash Memory	
Example of the 1M-bit Flash Memory Program	366
Features of the 1M-bit Flash Memory.....	342
Sector Configuration of the 1M-bit Flash Memory	344
8/16-bit PPG	
8/16-bit PPG Interrupt	206
8/16-bit PPG Operation.....	206
8/16-bit PPG Operation Modes	208
Overview of the 8/16-bit PPG.....	194
Registers in the 8/16-bit PPG.....	197
8-bit Data Bus	
Status of Each Pin in the External Bus 8-bit Data Bus Mode	107
8-bit PPG	
Block Diagrams of the 8-bit PPG	195

A

A	
Accumulator (A).....	33
A/D Converter	
Block Diagram of A/D Converter	234
Cautions on Using the A/D Converter	233
Overview of the A/D Converter.....	232
Registers of the A/D Converter	235
Accumulator	
Accumulator (A).....	33
Acknowledgment	
Acknowledgment.....	318
ADB	
Additional Bank Register (ADB).....	41
ADCR	
Data Registers (ADCR1 and ADCR0)	240
ADCS	
Control Status Registers (ADCS0 and ADCS1)	236
Additional Bank Register	
Additional Bank Register (ADB).....	41
Address Match Detection Function	
Block Diagram of the Address Match Detection Function.....	328
Note About The Operation of the Address Match Detection Function.....	331
Operation of the Address Match Detection Function	331
Program Example of the Address Match Detection Function.....	334
System Configuration Example of the Address Match Detection Function.....	332
Address Register	
Address Register (IADR).....	314
Addressing	
Addressing	317, 394
Addressing by the Bank Method.....	27
Direct Addressing	396
Indirect Addressing	402
Linear Addressing Methods	27
ADER	
Analog Input Enable Register (ADER)	145
Analog Input Enable Register	
Analog Input Enable Register (ADER)	145
Arbitration	
Arbitration	318
ARSR	
Automatic Ready Function Selection Register (ARSR).....	122

Automatic Ready Function Selection Register	
Automatic Ready Function Selection Register (ARSR).....	122
Available Models	
Available Models	5
B	
Bank Method	
Addressing by the Bank Method.....	27
Bank Registers	
Settings and Data Access of Bank Registers	41
Bank Selection Prefixes	
Bank Selection Prefixes	44
BAP	
Buffer Address Pointer (BAP)	76
Basic Configuration	
Basic Configuration of MB90F553A Serial Programming Connection	372
Block Diagram	
16-bit I/O Timer Block Diagram.....	162
Block Diagram of A/D Converter	234
Block Diagram of External Memory Access (External Bus Pin Control Circuit)	120
Block Diagram of the 16-bit Reload Timer (with the Event Count Function).....	182
Block Diagram of the Address Match Detection Function.....	328
Block Diagram of the Clock Monitor Functions	324
Block Diagram of the Delayed Interrupt Generating Module	228
Block Diagram of the DTP/External Interrupt Circuit	217
Block Diagram of the Entire Flash Memory.....	343
Block Diagram of the I/O Extended Serial Interface	285
Block Diagram of the I2C Interface	303
Block Diagram of the Low-power Consumption Control Circuit.....	92
Block Diagram of the ROM Mirror Function Selection Module	338
Block Diagram of the System Architecture	6
Block Diagrams of the 8-bit PPG.....	195
I/O Port Block Diagram	135
Time-based Timer Block Diagram	148
UART Block Diagram.....	259
Watchdog Timer Block Diagram	154
Buffer Address Pointer	
Buffer Address Pointer (BAP)	76
Bus Control Register	
Bus Control Register (IBCR)	309
Bus Control Signal Selection Register	
Bus Control Signal Selection Register (ECSR)	125
Bus Error	
Bus Error	318
Bus Mode	
Memory Space for Each Bus Mode	117
Recommended Setting Sample of Memory Space for Each Bus Mode	118
Bus Status Register	
Bus Status Register (IBSR)	306
C	
Calculating	
Calculating the Execution Cycle Count.....	411
CCR	
Condition Code Register (CCR)	36
CDCR	
Communication Prescaler Register (CDCR)	254
Chip Deletion	
Deleting the Data From the Flash Memory (Chip Deletion).....	361
Circuit	
Block Diagram of External Memory Access (External Bus Pin Control Circuit)	120
Block Diagram of the DTP/External Interrupt Circuit	217
Block Diagram of the Low-power Consumption Control Circuit	92
External Interrupt/DTP Circuit Operation Procedure	224
External Memory Access (External Bus Pin Control Circuit)	120
Operation of the Low-power Consumption Control Circuit	98
Overview of the Low-power Consumption Control Circuit	90
Registers for External Memory Access (External Bus Pin Control Circuit)	121
Registers in the DTP/External Interrupt Circuit	216
CKSCR	
Clock Selection Register (CKSCR)	95
CLKR	
Clock Output Permission Register (CLKR)	325
Clock	
External Clock	272
External Shift Clock Mode	293
Input Pin Function (for the Internal Clock Mode)	190
Internal Clock Operations.....	188
Internal Shift Clock Mode	293
Oscillating Clock Frequency and Serial Clock Input Frequency	374
Clock Control Register	
Clock Control Register (ICCR)	312
Clock Generator	
Cautions as to the Clock Generator.....	82

INDEX

Clock Monitor	
Block Diagram of the Clock Monitor Functions	
.....	324
Clock Output Permission Register	
Clock Output Permission Register (CLKR).....	325
Clock Selection Register	
Clock Selection Register (CKSCR).....	95
Clock Supply Map	
Clock Supply Map.....	83
CMR	
Common Register Bank Prefix (CMR).....	45
Command Sequence Table	
Command Sequence Table.....	349
Common Register Bank Prefix	
Common Register Bank Prefix (CMR).....	45
Communication	
End of Communication.....	276
Start of Communication.....	276
Communication Prescaler	
Communication Prescaler.....	271
Communication Prescaler Register	
Communication Prescaler Register (CDCR).....	254
Operation of Communication Prescaler Register	
.....	256
Compare Register	
Compare Register (OCCP0 and OCCP1).....	168
Condition Code Register	
Condition Code Register (CCR).....	36
Consecutive Prefix Codes	
In the Case of Consecutive Prefix Codes.....	46
Control Status Register	
Control Status Register (FMCS).....	347
Control Status Register (ICCS).....	166
Control Status Register (OCS0 to OCS2).....	169
Control Status Registers (ADCS0 and ADCS1)	
.....	236
Control Status Registers (ICS23 and ICS01).....	172
Conversion	
Conversion with the EI ² OS.....	243
Conversion Data Protection Function	
Conversion Data Protection Function.....	250
Count Clock	
Notes on Selecting a Count Clock.....	211
Counter	
Counter Operation Statuses.....	192
CPU	
Intermittent CPU Operation Function.....	108
D	
Data Bank Register	
Data Bank Register (DTB).....	41
Data Counter	
Data Counter (DCT).....	73
Data Format	
Transfer Data Format.....	273, 275
Data Polling Flag	
Data Polling Flag (DQ7).....	352
Data Register	
Data Register.....	165
Data Register (IDAR).....	315
Data Registers (ADCR1 and ADCR0).....	240
DCT	
Data Counter (DCT).....	73
DDR _x	
Port Data Direction Register (DDR _x).....	142
Dedicated Adapter	
Dedicated Adapter Socket.....	453
Dedicated Registers	
Dedicated Registers.....	31
Delayed Interrupt Generating Module	
Block Diagram of the Delayed Interrupt Generating	
Module.....	228
Register in the Delayed Interrupt Generating Module	
.....	228
Delayed Interrupt Request Latch	
Note on Use of the Delayed Interrupt Request Latch	
.....	229
Deletion	
Deleting the Data From the Flash Memory (Chip	
Deletion).....	361
Detailed Explanation of Flash Memory Writing and	
Deletion.....	357
Flash Memory from which any Data Item is Deleted	
(Sector Deletion).....	362
Restarting the Flash Memory Sector Deletion	
.....	365
Temporarily Stopping the Sector Deletion from the	
Flash Memory.....	364
Description	
Description of Instruction Presentation Items and	
Symbols.....	414
Devices	
Cautions on Handling Devices.....	21
Conditions of Peripheral Devices Connected	
Externally when DTP is Used.....	224
Dimensions	
External Dimensions of the FPT-100P-M05 Package	
.....	8
External Dimensions of the FPT-100P-M06 Package	
.....	7
Direct Addressing	
Direct Addressing.....	396
Direct Page Register	
Direct Page Register (DPR).....	40
DIV A,Ri	
Using the "DIV A,Ri" and "DIVW A,RWi"	
Instructions.....	47

DIVW A,RWi	
Using the "DIV A,Ri" and "DIVW A,RWi"	
Instructions.....	47
DPR	
Direct Page Register (DPR)	40
DQ3	
Sector Deletion Timer Flag (DQ3).....	356
DQ5	
Timing Limit Excess Flag (DQ5).....	355
DQ6	
Toggle Bit Flag (DQ6)	354
DQ7	
Data Polling Flag (DQ7).....	352
DTB	
Data Bank Register (DTB).....	41
DTP	
Conditions of Peripheral Devices Connected	
Externally when DTP is Used.....	224
DTP Operation	221
Switching between an External Interrupt Request and	
a DTP Request.....	222
DTP/External Interrupt Circuit	
Block Diagram of the DTP/External Interrupt Circuit	
.....	217
Registers in the DTP/External Interrupt Circuit	
.....	216
E	
ECSR	
Bus Control Signal Selection Register (ECSR)	
.....	125
Effective Address Field	
Effective Address Field	395, 413
EI ² OS	
Configuration of the Expanded Intelligent I/O Service	
(EI ² OS).....	69
Conversion with the EI ² OS	243
Example of EI ² OS Activation in Pause Mode	
.....	248
Example of EI ² OS Activation in Single Mode	
.....	244
Example of EI ² OS Activation in Successive Mode	
.....	246
Execution Time of the Expanded Intelligent I/O	
Service (EI ² OS)	79
Extended Intelligent I/O Service (EI ² OS) Function	
and Interrupts	189
Extended Intelligent I/O Services (EI ² OS).....	270
Notes in Connection with Using the EI ² OS Feature by	
Extended I/O Serial 2	55
Operational Flow of the Expanded Intelligent I/O	
Service (EI ² OS)	77
Overview of the Expanded Intelligent I/O Service	
(EI ² OS)	68
EI ² OS Status Register	
EI ² OS Status Register (ISCS)	74
EIRR	
Interrupt/DTP Source Register (EIRR)	218
ELVR	
Request Level Setting Register (ELVR).....	219
ENIR	
Interrupt/DTP Enable Register (ENIR)	218
Entire Flash Memory	
Block Diagram of the Entire Flash Memory	343
Error	
Bus Error	318
Event Count Function	
Block Diagram of the 16-bit Reload Timer (with the	
Event Count Function)	182
Overview of the 16-bit Reload Timer (with the Event	
Count Function).....	182
Registers of the 16-bit Reload Timer (with the Event	
Count Function).....	183
Exception	
Occurrence of Exceptions because of Executing	
Undefined Instructions	80
Execution Cycle Count	
Calculating the Execution Cycle Count.....	411
Execution Cycle Count.....	410
Expanded Intelligent I/O Service	
Configuration of the Expanded Intelligent I/O Service	
(EI ² OS)	69
Execution Time of the Expanded Intelligent I/O	
Service (EI ² OS).....	79
Operational Flow of the Expanded Intelligent I/O	
Service (EI ² OS).....	77
Overview of the Expanded Intelligent I/O Service	
(EI ² OS)	68
Expanded Intelligent I/O Service Descriptor	
Expanded Intelligent I/O Service Descriptor (ISD)	
.....	73
Extended I/O	
Notes in Connection with Using the EI ² OS Feature by	
Extended I/O Serial 2	55
Extended Intelligent I/O Service	
Extended Intelligent I/O Service (EI ² OS) Function	
and Interrupts	189
Extended Intelligent I/O Services (EI ² OS)	270
External Address Output Control Register	
External Address Output Control Register (HACR)	
.....	124
External Bus	
Status of Each Pin in the External Bus 16-bit Data Bus	
Mode.....	106

INDEX

Status of Each Pin in the External Bus 8-bit Data Bus Mode	107
External Bus 16-bit Data Bus Mode	
Status of Each Pin in the External Bus 16-bit Data Bus Mode	106
External Bus 8-bit Data Bus Mode	
Status of Each Pin in the External Bus 8-bit Data Bus Mode	107
External Bus Pin Control Circuit	
Block Diagram of External Memory Access (External Bus Pin Control Circuit).....	120
External Memory Access (External Bus Pin Control Circuit).....	120
Registers for External Memory Access (External Bus Pin Control Circuit)	121
External Clock	
External Clock	272
External Dimensions	
External Dimensions of the FPT-100P-M05 Package	8
External Dimensions of the FPT-100P-M06 Package	7
External Event Count	
External Event Count.....	188
External Interrupt	
External Interrupt Operation.....	221
External Interrupt Request Level.....	224
Switching between an External Interrupt Request and a DTP Request	222
External Interrupt/DTP Circuit	
External Interrupt/DTP Circuit Operation Procedure	224
External Memory Access	
Block Diagram of External Memory Access (External Bus Pin Control Circuit).....	120
External Memory Access (External Bus Pin Control Circuit).....	120
External Memory Access Control Signal	128
Registers for External Memory Access (External Bus Pin Control Circuit)	121
External Shift Clock Mode	
External Shift Clock Mode	293
F	
F2MC-16LX Instruction List	
F2MC-16LX Instruction List.....	417
Flag	
Data Polling Flag (DQ7)	352
Hardware Sequence Flag.....	350
Interrupt Flag Setting Timing in Operating Modes	277
Sector Deletion Timer Flag (DQ3)	356
Timing Limit Excess Flag (DQ5)	355
Toggle Bit Flag (DQ6)	354
Flag Change Suppression Prefix	
Flag Change Suppression Prefix (NCC)	45
Flags	
Five Flags (PE,ORE,FRE,RDRF,and TDRE) and Two Interrupt Sources.....	277
Flash	
Block Diagram of the Entire Flash Memory	343
Flash Memory	
Deleting the Data From the Flash Memory (Chip Deletion).....	361
Detailed Explanation of Flash Memory Writing and Deletion	357
Example of the 1M-bit Flash Memory Program	366
Features of the 1M-bit Flash Memory	342
Flash Memory Control Signals	345
Flash Memory from which any Data Item is Deleted (Sector Deletion).....	362
Flash Memory Mode	345
Flash Memory Register	342
Procedure for Deleting a Sector from the Flash Memory	362
Procedure for Writing Data in the Flash Memory	359
Restarting the Flash Memory Sector Deletion	365
Sector Configuration of the 1M-bit Flash Memory	344
Setting the Flash Memory to the Read or Reset Status	358
Temporarily Stopping the Sector Deletion from the Flash Memory	364
Writing and Deleting Data for the Flash Memory	342
Writing Data in the Flash Memory.....	359
Flash Memory Mode	
Flash Memory Mode	345
Flash Microcontroller Programmer	
Example of Minimal Connection with the Flash Microcontroller Programmer (when Power is Supplied from a Writer).....	381
Example of Minimal Connection with the Flash Microcontroller Programmer (when User Power Supply is Used)	379
FMCS	
Control Status Register (FMCS)	347
Format	
Transfer Data Format	273, 275
FPT-100P-M05	
External Dimensions of the FPT-100P-M05 Package	8

FPT-100P-M06	
External Dimensions of the FPT-100P-M06 Package	7
FRE	
Five Flags (PE,ORE,FRE,RDRF,and TDRE) and Two Interrupt Sources	277
Free-run Timer	
16-bit Free-run Timer (x 1)	160
16-bit Free-run Timer Count Timing	175
Frequency	
Oscillating Clock Frequency and Serial Clock Input Frequency	374
FTP-100P-M05	
Pin Arrangement of the FTP-100P-M05	10
FTP-100P-M06	
Pin Arrangement of the FTP-100P-M06	9
G	
General-purpose Registers	
General-purpose Registers	42
H	
HACR	
External Address Output Control Register (HACR)	124
Hardware Components	
Initial Values In Hardware Components	207
Hardware Interrupt	
Example of Procedure for Using Hardware Interrupts	65
Hardware Interrupt Request during Writing to the Internal Resource Area	59
Notes on the Use of Hardware Interrupts	60
Operating Flow for Hardware Interrupts	64
Operations of Hardware Interrupts	61
Overview of Hardware Interrupts	58
Processing Time for a Hardware Interrupt	63
Structure of Hardware Interrupts	58
Hardware Sequence Flag	
Hardware Sequence Flag	350
Hardware Standby Mode	
Releasing the Hardware Standby Mode	104
Transition to the Hardware Standby Mode	104
Hold Function	
Hold Function	132
I	
I/O	
I/O Map	384
I/O Circuit Types	
I/O Circuit Types	17
I/O Extended Serial Interface	
Block Diagram of the I/O Extended Serial Interface	285
Interrupt Function of the I/O Extended Serial Interface	300
Operation of I/O Extended Serial Interface	292
Overview of the I/O Extended Serial Interface	284
Registers of the I/O Extended Serial Interface	286
I/O Port	
I/O Port Block Diagram	135
I/O Port Overview	134
I/O Port Registers	138
I/O Register Address Pointer	
I/O Register Address Pointer (IOA)	74
I/O Timer	
16-bit I/O Timer Block Diagram	162
I2C Interface	
Block Diagram of the I2C Interface	303
Features of the I2C Interface	302
Flow of the I2C Interface Modes	321
Flow of the I2C Interface Transmission	319
Registers of the I2C Interface	305
Structure of the I2C Interface	304
IADR	
Address Register (IADR)	314
IBCR	
Bus Control Register (IBCR)	309
IBSR	
Bus Status Register (IBSR)	306
ICCR	
Clock Control Register (ICCR)	312
ICCS	
Control Status Register (ICCS)	166
ICR	
Interrupt Control Register (ICR)	70
ICS	
Control Status Registers (ICS23 and ICS01)	172
IDAR	
Data Register (IDAR)	315
ILM	
Interrupt Level Mask Register (ILM)	37
Indirect Addressing	
Indirect Addressing	402
Initialization	
Initialization	276
Input Capture	
16-bit Input Capture Operations	179
Input Capture (x 4)	161
Input Capture Input Timing	180

INDEX

Input Capture Data Register	
Input Capture Data Register (IPCO0 to IPCO3)	
.....	172
Input Resistor Register	
Input Resistor Registers (RDR0 and RDR1).....	144
Input/Output Timing	
Start/Stop Timing of Shift Operation and Input/Output Timing	297
Instruction	
Description of Instruction Presentation Items and Symbols	414
F2MC-16LX Instruction List.....	417
Instruction Types.....	393
Interrupt Stop Instruction	59
Occurrence of Exceptions because of Executing Undefined Instructions	80
Structure of Instruction Map.....	431
Using the "DIV A,Ri" and "DIVW A,RWi" Instructions	47
Instruction Presentation Items and Symbols	
Description of Instruction Presentation Items and Symbols	414
INT	
Competition Among the SCC,MSS,and INT Bits	311
Intermittent CPU Operation	
Intermittent CPU Operation Function	108
Internal Clock	
Internal Clock Operations	188
Internal Clock Mode	
Input Pin Function (for the Internal Clock Mode)	190
Internal Resource	
Hardware Interrupt Request during Writing to the Internal Resource Area	59
Internal Shift Clock Mode	
Internal Shift Clock Mode	293
Internal timer	
Internal timer	272
Interrupt	
8/16-bit PPG Interrupt	206
Example of Procedure for Using Hardware Interrupts	65
Extended Intelligent I/O Service (EI ² OS) Function and Interrupts	189
Five Flags (PE,ORE,FRE,RDRF,and TDRE) and Two Interrupt Sources	277
Hardware Interrupt Request during Writing to the Internal Resource Area	59
Interrupt Causes	53
Interrupt Flag Setting Timing in Operating Modes	277
Interrupt Function of the I/O Extended Serial Interface	300
Interrupt Vectors.....	56
Multiple Interrupts	59
Notes on Software Interrupts.....	67
Notes on the Use of Hardware Interrupts	60
Operating Flow for Hardware Interrupts	64
Operation of Software Interrupts	66
Operations of Hardware Interrupts	61
Overview of Hardware Interrupts	58
Overview of Interrupts	52
Overview of Software Interrupts.....	66
Processing Time for a Hardware Interrupt	63
Saving a Register to the Stack at an Interrupt	59
Structure of Hardware Interrupts	58
Structure of Software Interrupts.....	66
Interrupt Causes	
Interrupt Causes	53
Interrupt Control Register	
Interrupt Control Register (ICR)	70
Interrupt Generating Module	
Operation of the Delayed Interrupt Generating Module	229
Interrupt Level Mask Register	
Interrupt Level Mask Register (ILM)	37
Interrupt Stop Instruction	
Interrupt Stop Instruction	59
Interrupt Suppression	
Restrictions on Interrupt Suppression and Prefix Instructions	46
Interrupt Suppression Instructions	
Interrupt Suppression Instructions	46
Interrupt Vectors	
Interrupt Vectors.....	56
Interrupt/DTP Enable Register	
Interrupt/DTP Enable Register (ENIR).....	218
Interrupt/DTP Source Register	
Interrupt/DTP Source Register (EIRR).....	218
Interval Interrupt	
Interval Interrupt Function	151
IOA	
I/O Register Address Pointer (IOA)	74
IPCO	
Input Capture Data Register (IPCO0 to IPCO3)	
.....	172
ISCS	
EI ² OS Status Register (ISCS).....	74
ISD	
Expanded Intelligent I/O Service Descriptor (ISD)	
.....	73
ISEL	
Port Selection Register (ISEL)	316

L

Latch

Note on Use of the Delayed Interrupt Request Latch	229
--	-----

Level

External Interrupt Request Level	224
--	-----

Linear Addressing Methods

Linear Addressing Methods	27
---------------------------------	----

Low-power Consumption Control Circuit

Block Diagram of the Low-power Consumption Control Circuit.....	92
Operation of the Low-power Consumption Control Circuit.....	98
Overview of the Low-power Consumption Control Circuit.....	90

Low-power Consumption Mode Control Register

Low-power Consumption Mode Control Register (LPMCR)	93
---	----

LPMCR

Low-power Consumption Mode Control Register (LPMCR)	93
---	----

M

Machine Clock

Machine Clock Initialization	110
Switching the Machine Clock.....	110

MB90F553A

Basic Configuration of MB90F553A Serial Programming Connection	372
--	-----

Memory Access Mode

Memory Access Mode Overview.....	114
----------------------------------	-----

Memory Space

Allocating Multiple-byte Data in a Memory Space	30
Memory Space.....	26
Memory Space for Each Bus Mode.....	117
Recommended Setting Sample of Memory Space for Each Bus Mode.....	118

Minimal Connection

Example of Minimal Connection with the Flash Microcontroller Programmer (when Power is Supplied from a Writer)	381
Example of Minimal Connection with the Flash Microcontroller Programmer (when User Power Supply is Used)	379

Mode

Application of UART (During Operation in Mode 1)	280
Example of EI ² OS Activation in Pause Mode	248
Example of EI ² OS Activation in Single Mode	244

Example of EI²OS Activation in Successive Mode

.....	246
External Shift Clock Mode	293
Flash Memory Mode.....	345
Flow of the I2C Interface Modes	321
Input Pin Function (for the Internal Clock Mode)	190
Internal Shift Clock Mode	293
Memory Access Mode Overview	114
Memory Space for Each Bus Mode	117
Other Modes	345
Pause Mode	243
Programming Mode	454
Recommended Setting Sample of Memory Space for Each Bus Mode	118
Releasing the Hardware Standby Mode.....	104
Releasing the Sleep Mode	100
Releasing the Stop Mode.....	103
Releasing the Watch Mode	101
Single Mode.....	242
Status of Each Pin in the External Bus 16-bit Data Bus Mode.....	106
Status of Each Pin in the External Bus 8-bit Data Bus Mode.....	107
Status of Each Pin in the Single Chip Mode	105
Successive Mode	242
Transition to the Hardware Standby Mode	104
Transition to the Sleep Mode	100
Transition to the Stop Mode.....	103
Transition to the Watch Mode.....	101

Mode Data

Mode Data	116
-----------------	-----

Mode Pins

Mode Pins.....	115
----------------	-----

MSS

Competition Among the SCC,MSS,and INT Bits	311
--	-----

Multiple Interrupt

Multiple Interrupts	59
---------------------------	----

Multiple-byte Data

Access of Multiple-byte Data.....	30
Allocating Multiple-byte Data in a Memory Space	30

N

NCC

Flag Change Suppression Prefix (NCC)	45
--	----

O

OCCP

Compare Register (OCCP0 and OCCP1)	168
--	-----

OCS

Control Status Register (OCS0 to OCS2)	169
--	-----

INDEX

ODR	
Output Pin Register (ODR4).....	143
ORE	
Five Flags (PE,ORE,FRE,RDRF,and TDRE) and Two Interrupt Sources	277
Oscillating Clock Frequency	
Oscillating Clock Frequency and Serial Clock Input Frequency.....	374
Oscillation Stabilization Time	
Setting the Oscillation Stabilization Time	109
OTPROM	
OTPROM Programming Yield	454
Procedure for Programming the OTPROM	453
Screening Conditions Recommended for the OTPROM.....	454
Output Compare	
16-bit Output Compare Operations.....	176
16-bit Output Compare Timing.....	177
Output Compare (x 4)	160
Output Pin Register	
Output Pin Register (ODR4).....	143
P	
Package	
External Dimensions of the FPT-100P-M05 Package	8
External Dimensions of the FPT-100P-M06 Package	7
PACSR	
Program Address Detection Control Register (PACSR)	329
PADR	
Program Address Detection Registers (PADR0 and PADR1)	329
Pause Mode	
Example of EI ² OS Activation in Pause Mode	248
Pause Mode	243
PC	
Program Counter (PC)	39
PCB	
Program Bank Register (PCB)	41
PDRx	
Port Data Register (PDRx)	140
PE	
Five Flags (PE,ORE,FRE,RDRF,and TDRE) and Two Interrupt Sources	277
Peripheral Devices	
Conditions of Peripheral Devices Connected Externally when DTP is Used.....	224
Pin Arrangement	
Pin Arrangement of the FTP-100P-M05	10
Pin Arrangement of the FTP-100P-M06	9
Pin Functions	
Description of the Pin Functions.....	11
Port Data Direction Register	
Port Data Direction Register (DDRx).....	142
Port Data Register	
Port Data Register (PDRx).....	140
Port Selection Register	
Port Selection Register (ISEL)	316
PPG	
8/16-bit PPG Interrupt	206
8/16-bit PPG Operation	206
8/16-bit PPG Operation Modes.....	208
Block Diagrams of the 8-bit PPG.....	195
Overview of the 8/16-bit PPG	194
PPG Output Operation.....	209
Registers in the 8/16-bit PPG	197
PPG0 Operation Mode Control Register	
PPG0 Operation Mode Control Register (PPGC0)	198
PPG0/1 Output Pin Control Register	
PPG0/1 Output Pin Control Register (PPGE)	203
PPG1 Operation Mode Control Register	
PPG1 Operation Mode Control Register (PPGC1)	200
PPGC	
PPG0 Operation Mode Control Register (PPGC0)	198
PPG1 Operation Mode Control Register (PPGC1)	200
PPGE	
PPG0/1 Output Pin Control Register (PPGE)	203
Prefix	
Common Register Bank Prefix (CMR).....	45
Flag Change Suppression Prefix (NCC)	45
Prefix Codes	
In the Case of Consecutive Prefix Codes	46
Prefix Instructions	
Restrictions on Interrupt Suppression and Prefix Instructions	46
Prescaler	
Communication Prescaler	271
PRL	
Reload Registers (PRL/PRLH).....	205
Processor Status	
Processor Status (PS)	36
Program Address Detection Control Register	
Program Address Detection Control Register (PACSR).....	329
Program Address Detection Register	
Program Address Detection Registers (PADR0 and PADR1)	329

Program Bank Register		Control Status Registers (ICS23 and ICS01)	172
Program Bank Register (PCB).....	41	Data Bank Register (DTB)	41
Program Counter		Data Register	165
Program Counter (PC)	39	Data Register (IDAR)	315
Programming		Data Registers (ADCR1 and ADCR0)	240
OTPROM Programming Yield	454	Direct Page Register (DPR)	40
Procedure for Programming the OTPROM	453	EI ² OS Status Register (ISCS)	74
Programming Mode	454	External Address Output Control Register (HACR)	
Programming Mode		124
Programming Mode	454	Input Capture Data Register (IPCO0 to IPCO3)	
PS		172
Processor Status (PS).....	36	Input Resistor Registers (RDR0 and RDR1)	144
Pulse Output		Interrupt Control Register (ICR)	70
Controlling Pulse Output on Pins	212	Interrupt Level Mask Register (ILM)	37
Pulse Width		Interrupt/DTP Enable Register (ENIR)	218
Relationship between the Reloaded Value and Pulse		Interrupt/DTP Source Register (EIRR)	218
Width.....	210	Low-power Consumption Mode Control Register	
		(LPMCR).....	93
R		Operation of Communication Prescaler Register	
RDR		256
Input Resistor Registers (RDR0 and RDR1)	144	Output Pin Register (ODR4).....	143
RDRF		Port Data Direction Register (DDRx)	142
Five Flags (PE,ORE,FRE,RDRF,and TDRE) and Two		Port Data Register (PDRx)	140
Interrupt Sources.....	277	Port Selection Register (ISEL)	316
Ready Function		PPG0 Operation Mode Control Register (PPGC0)	
Ready Function.....	130	198
Receiver		PPG0/1 Output Pin Control Register (PPGE)	203
Receiver Operation	273	PPG1 Operation Mode Control Register (PPGC1)	
Register		200
16-bit Timer Register (TMR) and 16-bit Reload		Program Address Detection Control Register	
Register (TMRLR).....	187	(PACSR)	329
Additional Bank Register (ADB)	41	Program Address Detection Registers (PADR0 and	
Address Register (IADR).....	314	PADR1).....	329
Analog Input Enable Register (ADER).....	145	Program Bank Register (PCB)	41
Automatic Ready Function Selection Register		Reload Registers (PRLL/PRLH)	205
(ARSR).....	122	Request Level Setting Register (ELVR)	219
Bus Control Register (IBCR)	309	ROM Mirror Function Selection Register (ROMM)	
Bus Control Signal Selection Register (ECSR)		339
.....	125	Serial Control Register (SCR).....	263
Clock Control Register (ICCR)	312	Serial Mode Control Status Register (SMCS)	
Clock Output Permission Register (CLKR)	325	287
Clock Selection Register (CKSCR).....	95	Serial Shift Data Register (SDR)	291
Communication Prescaler Register (CDCR)	254	Serial Status Register (SSR)	267
Compare Register (OCCP0 and OCCP1).....	168	Time-based Timer Control Register (TBTC)	149
Configuration of Serial Input Data Register (SIDR)		Timer Control Status Register (TMCSR)	184
and Serial Output Data Register (SODR)		User Stack Bank Register (USB) and System Stack	
.....	266	Bank Register (SSB)	41
Control Status Register (FMCS)	347	Watchdog Timer Control Register (WDTC)	155
Control Status Register (ICCS).....	166	Register Bank Pointer	
Control Status Register (OCS0 to OCS2)	169	Register Bank Pointer (RP).....	37
Control Status Registers (ADCS0 and ADCS1)		Register Banks	
.....	236	Register Banks	43

INDEX

Reload Register	
Reload Registers (PRL/PRH)	205
Reload Registers	
Write Timing for the Reload Registers	213
Reload Timer	
Block Diagram of the 16-bit Reload Timer (with the Event Count Function)	182
Overview of the 16-bit Reload Timer (with the Event Count Function)	182
Registers of the 16-bit Reload Timer (with the Event Count Function)	183
Reloaded Value	
Relationship between the Reloaded Value and Pulse Width	210
Request Level Setting Register	
Request Level Setting Register (ELVR)	219
Reset	
Operation after a Reset is Released	86
Preventing a Watchdog Timer Reset	157
Registers not Initialized by Reset Input	87
Reset Causes	84
Setting the Flash Memory to the Read or Reset Status	358
Reset Causes	
Reset Causes	84
Register	
Bus Status Register (IBSR)	306
ROM Mirror Function Selection Module	
Block Diagram of the ROM Mirror Function Selection Module	338
ROM Mirror Function Selection Register	
ROM Mirror Function Selection Register (ROMM)	339
ROMM	
ROM Mirror Function Selection Register (ROMM)	339
RP	
Register Bank Pointer (RP)	37
S	
SCC	
Competition Among the SCC, MSS, and INT Bits	311
SCR	
Serial Control Register (SCR)	263
Screening Conditions	
Screening Conditions Recommended for the OTPROM	454
SDR	
Serial Shift Data Register (SDR)	291
Sector	
Procedure for Deleting a Sector from the Flash Memory	362
Sector Deletion	
Flash Memory from which any Data Item is Deleted (Sector Deletion)	362
Restarting the Flash Memory Sector Deletion	365
Temporarily Stopping the Sector Deletion from the Flash Memory	364
Sector Deletion Timer Flag	
Sector Deletion Timer Flag (DQ3)	356
Serial Clock Input Frequency	
Oscillating Clock Frequency and Serial Clock Input Frequency	374
Serial Control Register	
Serial Control Register (SCR)	263
Serial Data Register R/W Wait State	
Serial Data Register R/W Wait State	295
Serial Input Data Register	
Configuration of Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)	266
Serial Mode Control Status Register	
Serial Mode Control Status Register (SMCS)	287
Serial Mode Register	
Serial Mode Register (SMR)	261
Serial Output Data Register	
Configuration of Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)	266
Serial Programming Connection	
Basic Configuration of MB90F553A Serial Programming Connection	372
Example of Serial Programming Connection (when Power is Supplied from a Writer)	377
Example of Serial Programming Connection (when User Power Supply is Used)	375
Serial Shift Data Register	
Serial Shift Data Register (SDR)	291
Serial Status Register	
Serial Status Register (SSR)	267
Shift Operation	
Start/Stop Timing of Shift Operation and Input/Output Timing	297
SIDR	
Configuration of Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)	266
Single Chip Mode	
Status of Each Pin in the Single Chip Mode	105

Single Mode	
Example of EI ² OS Activation in Single Mode	244
Single Mode	242
Sleep Mode	
Releasing the Sleep Mode	100
Transition to the Sleep Mode	100
SMCS	
Serial Mode Control Status Register (SMCS)	287
SMR	
Serial Mode Register (SMR)	261
Socket	
Dedicated Adapter Socket	453
SODR	
Configuration of Serial Input Data Register (SIDR) and Serial Output Data Register (SODR)	266
Software Interrupt	
Notes on Software Interrupts	67
Operation of Software Interrupts	66
Overview of Software Interrupts	66
Structure of Software Interrupts	66
SSB	
User Stack Bank Register (USB) and System Stack Bank Register (SSB)	41
SSP	
User Stack Pointer (USP) and System Stack Pointer (SSP)	35
SSR	
Serial Status Register (SSR)	267
Stack	
Saving a Register to the Stack at an Interrupt	59
Standby State	
Return from the Standby State	224
Start/Stop Timing	
Start/Stop Timing of Shift Operation and Input/Output Timing	297
State	
Serial Data Register R/W Wait State	295
STOP State	295
Suspend State	295
Transfer State	295
Stop Mode	
Releasing the Stop Mode	103
Transition to the Stop Mode	103
STOP State	
STOP State	295
Structure	
Structure of Instruction Map	431
Successive Mode	
Example of EI ² OS Activation in Successive Mode	246
Successive Mode	242
Suspend State	
Suspend State	295
System Architecture	
Block Diagram of the System Architecture	6
System Stack Bank Register	
User Stack Bank Register (USB) and System Stack Bank Register (SSB)	41
System Stack Pointer	
User Stack Pointer (USP) and System Stack Pointer (SSP)	35
T	
TBTC	
Time-based Timer Control Register (TBTC)	149
TDRE	
Five Flags (PE, ORE, FRE, RDRF, and TDRE) and Two Interrupt Sources	277
Time-based Timer	
Time-based Timer Block Diagram	148
Time-based Timer Operations	151
Time-based Timer Register	148
Time-based Timer Control Register	
Time-based Timer Control Register (TBTC)	149
Timer Control Status Register	
Timer Control Status Register (TMCSR)	184
Timing Limit Excess Flag	
Timing Limit Excess Flag (DQ5)	355
TMCSR	
Timer Control Status Register (TMCSR)	184
TMR	
16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)	187
TMRLR	
16-bit Timer Register (TMR) and 16-bit Reload Register (TMRLR)	187
Toggle Bit Flag	
Toggle Bit Flag (DQ6)	354
Transfer	
Transfer Data Format	273, 275
Transfer State	
Transfer State	295
Transmission	
Flow of the I2C Interface Transmission	319
Transmitter	
Transmitter Operation	274

INDEX

U	
UART	
Application of UART (During Operation in Mode 1)	280
Features of UART	258
UART Block Diagram	259
UART Operations	270
UART Registers	260
Undefined Instruction	
Occurrence of Exceptions because of Executing Undefined Instructions	80
Underflow Operation	
Underflow Operation	189
USB	
User Stack Bank Register (USB) and System Stack Bank Register (SSB)	41
User Power Supply	
Example of Minimal Connection with the Flash Microcontroller Programmer (when User Power Supply is Used)	379
Example of Serial Programming Connection (when User Power Supply is Used)	375
User Stack Bank Register	
User Stack Bank Register (USB) and System Stack Bank Register (SSB)	41
User Stack Pointer	
User Stack Pointer (USP) and System Stack Pointer (SSP)	35
USP	
User Stack Pointer (USP) and System Stack Pointer (SSP)	35
W	
Watch Mode	
Releasing the Watch Mode	101
Transition to the Watch Mode	101
Watchdog	
Stopping the Watchdog	157
Watchdog Timer	
Activating the Watchdog Timer	157
Clearing the Watchdog Timer	157
Preventing a Watchdog Timer Reset	157
Watchdog Timer Block Diagram	154
Watchdog Timer Register	154
Watchdog Timer Control Register	
Watchdog Timer Control Register (WDTC)	155
WDTC	
Watchdog Timer Control Register (WDTC)	155
Writer	
Example of Minimal Connection with the Flash Microcontroller Programmer (when Power is Supplied from a Writer)	381
Example of Serial Programming Connection (when Power is Supplied from a Writer)	377
Writing	
Detailed Explanation of Flash Memory Writing and Deletion	357
Procedure for Writing Data in the Flash Memory	359
Writing Data in the Flash Memory	359

CM44-10103-6E

FUJITSU MICROELECTRONICS • CONTROLLER MANUAL

F²MC-16LX

16-BIT MICROCONTROLLER

MB90550A/B Series

HARDWARE MANUAL

July 2008 the sixth edition

Published **FUJITSU MICROELECTRONICS LIMITED**

Edited Business & Media Promotion Dept.
