

F²MC-16LX
16-BIT MICROCONTROLLER
MB90590 Series
HARDWARE MANUAL

F²MC-16LX
16-BIT MICROCONTROLLER
MB90590 Series
HARDWARE MANUAL

PREFACE

■ Objectives and Intended Reader

Thank you very much for your continued patronage of Fujitsu semiconductor products.

The MB90590 series has been developed as a general-purpose version of the F²MC-16LX series, which is an original 16-bit single-chip microcontroller compatible with the Application Specific IC (ASIC).

This manual explains the functions and operation of the MB90590 series for designers who actually use the MB90590 series to design products. Read this manual first.

■ Trademark

F²MC is the abbreviation of FUJITSU Flexible Microcontroller.

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

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

■ Structure of This Manual

CHAPTER 1 "OVERVIEW"

The MB90590 Series is a family member of the F²MC-16LX microcontrollers.

CHAPTER 2 "CPU"

This chapter explains the CPU.

CHAPTER 3 "INTERRUPTS"

This chapter explains the interrupt functions and operations.

CHAPTER 4 "DELAYED INTERRUPTS"

This chapter explains the functions and operations of the delayed interrupt.

CHAPTER 5 "CLOCK AND RESET"

This chapter explains the functions and operations of clocks and resets.

CHAPTER 6 "LOW-POWER CONTROL CIRCUIT"

This chapter explains the functions and operations of the low-power control circuits.

CHAPTER 7 "MEMORY ACCESS MODES"

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

CHAPTER 8 "I/O PORTS"

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

CHAPTER 9 "TIMEBASE TIMER"

This chapter explains the functions and operations of the timebase timer.

CHAPTER 10 "WATCH-DOG TIMER"

This chapter explains the functions and operations of the watch-dog timer.

CHAPTER 11 "16-BIT I/O TIMER"

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

CHAPTER 12 "16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)"

This chapter explains the functions and operations of the 16-bit reload timer (with the event count function).

CHAPTER 13 "WATCH TIMER"

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

CHAPTER 14 "8/16-BIT PPG"

This chapter explains the 8/16-bit PPG and explains its functions.

CHAPTER 15 "DTP/EXTERNAL INTERRUPTS"

This chapter explains the functions and operations of the DTP/external interrupts.

CHAPTER 16 "A/D CONVERTER"

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

CHAPTER 17 "UART0"

This chapter explains the UART0 functions and operations.

CHAPTER 18 "SERIAL I/O"

This chapter explains the functions and operations of the serial I/O.

CHAPTER 19 "CAN CONTROLLER"

This chapter explains the functions and operations of the CAN controller.

CHAPTER 20 "STEPPING MOTOR CONTROLLER"

This chapter explains the functions and operations of the stepping motor controller.

CHAPTER 21 "SOUND GENERATOR"

This chapter explains the functions and operations of the sound generator.

CHAPTER 22 "ADDRESS MATCH DETECTION FUNCTION"

This chapter explains the address match detection function and operation.

CHAPTER 23 "ROM MIRRORING MODULE"

This chapter explains the ROM mirroring module.

CHAPTER 24 "2M/3M-BIT FLASH MEMORY"

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

CHAPTER 25 "EXAMPLES OF MB90F594A/MB90F594G/MB90F591A/MB90F591G SERIAL PROGRAMMING CONNECTION"

This chapter provides examples of F²MC-16LX MB90F594A/MB90F594G/MB90F591A/MB90F591G serial programming connection.

APPENDIX

The appendixes provide I/O maps, instructions, and other information.

- The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.
- The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.
- 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 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.
- If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the prior authorization by Japanese government will be required for export of those products from Japan.

CONTENTS

CHAPTER 1 OVERVIEW	1
1.1 Product Overview	2
1.2 Features	3
1.3 Block Diagram	5
1.4 Pin Assignment	6
1.5 Package Dimensions	7
1.6 Pin Functions	8
1.7 Input-Output Circuits	12
1.8 Handling Device	14
CHAPTER 2 CPU	19
2.1 Outline of CPU	20
2.2 Memory Space	21
2.3 Memory Space Map	22
2.4 Linear Addressing	23
2.5 Bank Addressing Types	24
2.6 Multi-byte Data in Memory Space	26
2.7 Registers	27
2.7.1 Accumulator (A)	30
2.7.2 User Stack Pointer (USP) and System Stack Pointer (SSP)	31
2.7.3 Processor Status (PS)	32
2.7.4 Program Counter (PC)	35
2.8 Register Bank	36
2.9 Prefix Codes	38
2.10 Interrupt Disable Instructions	40
2.11 Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions	41
CHAPTER 3 INTERRUPTS	43
3.1 Outline of Interrupts	44
3.2 Interrupt Vector	47
3.3 Interrupt Control Registers (ICR)	48
3.4 Interrupt Flow	51
3.5 Hardware Interrupts	53
3.5.1 Hardware Interrupt Operation	54
3.5.2 Occurrence and Release of Hardware Interrupt	55
3.5.3 Multiple interrupts	57
3.6 Software Interrupts	58
3.7 Extended Intelligent I/O Service (EI ² OS)	60
3.7.1 Extended Intelligent I/O Service Descriptor (ISD)	62
3.7.2 EI ² OS Status Register (ISCS)	64
3.8 Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI ² OS)	65
3.9 Exceptions	68

CHAPTER 4	DELAYED INTERRUPT	69
4.1	Outline of Delayed Interrupt Module	70
4.2	Delayed Interrupt Register	71
4.3	Delayed Interrupt Operation	72
CHAPTER 5	CLOCK AND RESET	73
5.1	Clock Generator	74
5.2	Reset Cause Occurrence	75
5.3	Reset Causes	78
CHAPTER 6	LOW-POWER CONTROL CIRCUIT	81
6.1	Outline of Low-Power Control Circuit	82
6.2	Registers	84
6.2.1	Low-Power Mode Control Register (LPMCR)	85
6.2.2	Clock Selection Register (CKSCR)	87
6.3	Low-Power Mode Operation	89
6.3.1	Sleep Mode	91
6.3.2	Watch Mode	92
6.3.3	Stop Mode	93
6.3.4	Hardware Standby Mode	95
6.4	Intermittent CPU Operation	96
6.5	Switching Machine Clocks	97
6.6	Status Transition of Clock Selection	98
CHAPTER 7	MEMORY ACCESS MODES	101
7.1	Outline of Memory Access Modes	102
7.2	Mode Pins	103
7.3	Mode Data	104
CHAPTER 8	I/O PORTS	107
8.1	I/O Ports	108
8.2	I/O Port Registers	109
8.2.1	Port Data Register	110
8.2.2	Port Direction Register	111
8.2.3	Analog Input Enable Register	112
CHAPTER 9	TIMEBASE TIMER	113
9.1	Outline of Timebase Timer	114
9.2	Timebase Timer Control Register	115
9.3	Operations of Timebase Timer	117
CHAPTER 10	WATCH-DOG TIMER	119
10.1	Outline of Watch-Dog Timer	120
10.2	Watch-dog Timer Operation	123
CHAPTER 11	16-BIT I/O TIMER	125
11.1	Outline of 16-Bit I/O Timer	126
11.2	16-Bit I/O Timer Registers	128

11.3	16-bit Free-running Timer	129
11.3.1	Data Register	130
11.3.2	Control Status Register	131
11.3.3	16-bit Free-running Timer Operation	134
11.4	Output Compare	136
11.4.1	Output Compare Register	137
11.4.2	Control Status Register of Output Compare	138
11.4.3	16-bit Output Compare Operation	141
11.5	Input Capture	144
11.5.1	Input Capture Register Details	146
11.5.2	16-bit Input Capture Operation	148
CHAPTER 12 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)		151
12.1	Outline of 16-Bit Reload Timer (with Event Count Function)	152
12.2	16-Bit Reload Timer (with Event Count Function)	154
12.2.1	Timer Control Status Register (TMCSR)	155
12.2.2	Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)	158
12.3	Internal Clock and External Clock Operations of 16-bit Reload Timer	159
12.4	Underflow Operation of 16-bit Reload Timer	161
12.5	Output Pin Functions of 16-bit Reload Timer	162
12.6	Counter Operation State	163
CHAPTER 13 WATCH TIMER		165
13.1	Outline of Watch Timer	166
13.2	Watch Timer Registers	167
13.2.1	Timer Control Register	168
13.2.2	Sub-second Registers	170
13.2.3	Second/Minute/Hour Registers	171
CHAPTER 14 8/16-BIT PPG		173
14.1	Outline of 8/16-bit PPG	174
14.2	Block Diagram of 8/16-bit PPG	175
14.3	8/16-bit PPG Registers	177
14.3.1	PPG0 Operation Mode Control Register (PPGC0)	178
14.3.2	PPG1 Operation Mode Control Register (PPGC1)	180
14.3.3	PPG0, 1 Output Control Register (PPG01)	182
14.3.4	Reload Register (PRLH/PRLH)	184
14.4	Operations of 8/16-bit PPG	185
14.5	Selecting a Count Clock for 8/16-bit PPG	187
14.6	Controlling Pin Output of 8/16-bit PPG Pulses	188
14.7	8/16-bit PPG Interrupts	189
14.8	Initial Values of 8/16-bit PPG Hardware	190
CHAPTER 15 DTP/EXTERNAL INTERRUPTS		193
15.1	Outline of DTP/External Interrupts	194
15.2	DTP/External Interrupt Registers	196
15.3	Operations of DTP/External Interrupts	198
15.4	Switching between External Interrupt and DTP Requests	200

15.5	Notes on Using DTP/External Interrupts	201
CHAPTER 16	A/D Converter	203
16.1	Features of A/D Converter	204
16.2	Block Diagram of A/D Converter	206
16.3	A/D Converter Registers	207
16.3.1	Control Status Registers (ADCS0)	208
16.3.2	Control Status Register (ADCS1)	211
16.3.3	Data Registers (ADCR1 and ADCR0)	214
16.4	Operations of A/D Converter	216
16.5	Conversion Using EI ² OS	218
16.5.1	Starting EI ² OS in Single Mode	219
16.5.2	Starting EI ² OS in Continuous Mode	221
16.5.3	Starting EI ² OS in Stop Mode	223
16.6	Conversion Data Protection	225
CHAPTER 17	UART0	227
17.1	Feature of UART0	228
17.2	UART Block Diagram	229
17.3	UART Registers	230
17.3.1	Serial Mode Control Register (UMC)	231
17.3.2	Status Register (USR)	233
17.3.3	Input Data Register (UIDR) and Output Data Register (UODR)	235
17.3.4	Rate and Data Register (URD)	236
17.4	UART0 Operation	238
17.5	Baud Rate	239
17.6	Internal and External Clock	242
17.7	Transfer Data Format	243
17.8	Parity Bit	244
17.9	Interrupt Generation and Flag Set Timings	245
17.9.1	Flag Set Timings for a Receive Operation (in Mode 0, 1, or 3)	246
17.9.2	Flag Set Timings for a Receive Operation (in Mode 2)	247
17.9.3	Flag Set Timings for a Transmit Operation	248
17.9.4	Status Flag During Transmit and Receive Operation	249
17.10	UART0 Application Example	250
CHAPTER 18	SERIAL I/O	253
18.1	Outline of Serial I/O	254
18.2	Serial I/O Registers	255
18.2.1	Serial Mode Control Status Register (SMCS)	256
18.2.2	Serial Shift Data Register (SDR)	260
18.3	Serial I/O Prescaler (CDCR)	261
18.4	Serial I/O Operation	262
18.4.1	Shift Clock	263
18.4.2	Serial I/O Operation	264
18.4.3	Shift Operation Start/Stop Timing	266
18.4.4	Interrupt Function of the Extended Serial I/O Interface	269
18.5	Negative Clock Operation	270

CHAPTER 19 CAN CONTROLLER	271
19.1 Features of CAN Controller	272
19.2 Block Diagram of CAN Controller	273
19.3 List of Overall Control Registers	274
19.4 List of Message Buffers (ID Registers)	276
19.5 List of Message Buffers (DLC Registers and Data Registers)	279
19.6 Classifying the CAN Controller Registers	283
19.6.1 Control Status Register (CSR)	284
19.6.2 Bus Operation Stop Bit (HALT = 1)	287
19.6.3 Last Event Indicator Register (LEIR)	288
19.6.4 Receive and Transmit Error Counters (RTEC)	290
19.6.5 Bit Timing Register (BTR)	291
19.6.6 Message Buffer Valid Register (BVALR)	294
19.6.7 IDE register (IDER)	295
19.6.8 Transmission Request Register (TREQR)	296
19.6.9 Transmission RTR Register (TRTRR)	297
19.6.10 Remote Frame Receiving Wait Register (RFWTR)	298
19.6.11 Transmission Cancel Register (TCANR)	299
19.6.12 Transmission Complete Register (TCR)	300
19.6.13 Transmission Interrupt Enable Register (TIER)	301
19.6.14 Reception Complete Register (RCR)	302
19.6.15 Remote Request Receiving Register (RRTRR)	303
19.6.16 Receive Overrun Register (ROVRR)	304
19.6.17 Reception Interrupt Enable Register (RIER)	305
19.6.18 Acceptance Mask Select Register (AMSR)	306
19.6.19 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)	308
19.6.20 Message Buffers	310
19.6.21 ID Register x (x = 0 to 15) (IDRx)	311
19.6.22 DLC Register x (x = 0 to 15) (DLCRx)	313
19.6.23 Data Register x (x = 0 to 15) (DTRx)	314
19.7 Transmission of CAN Controller	316
19.8 Reception of CAN Controller	319
19.9 Reception Flowchart of CAN Controller	322
19.10 How to Use the CAN Controller	323
19.11 Procedure for Transmission by Message Buffer (x)	325
19.12 Procedure for Reception by Message Buffer (x)	327
19.13 Setting Configuration of Multi-level Message Buffer	329
19.14 Precautions when Using CAN Controller	331
CHAPTER 20 STEPPING MOTOR CONTROLLER	333
20.1 Outline of Stepping Motor Controller	334
20.2 Stepping Motor Controller Registers	335
20.2.1 PWM Control 0 register	336
20.2.2 PWM1&2 Compare Registers	337
20.2.3 PWM1&2 Select Registers	338
CHAPTER 21 SOUND GENERATOR	341
21.1 Outline of Sound Generator	342

21.2	Sound Generator Registers	343
21.2.1	Sound Control Register	344
21.2.2	Frequency Data register	346
21.2.3	Amplitude Data Register	347
21.2.4	Decrement Grade Register	348
21.2.5	Tone Count Register	349
CHAPTER 22	ADDRESS MATCH DETECTION FUNCTION	351
22.1	Outline of the Address Match Detection Function	352
22.2	Registers of the Address Match Detection Function	353
22.3	Operation of the Address Match Detection Function	355
22.4	Example of the Address Match Detection Function	356
CHAPTER 23	ROM MIRRORING MODULE	359
23.1	Outline of ROM Mirroring Module	360
23.2	ROM Mirroring Register (ROMM)	361
CHAPTER 24	2M/3M-BIT FLASH MEMORY	363
24.1	Overview of 2M/3M-bit Flash Memory	364
24.2	Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory	366
24.3	Write/Erase Modes	368
24.4	Flash Memory Control Status Register (FMCS)	370
24.5	Starting the Flash Memory Automatic Algorithm	372
24.6	Confirming the Automatic Algorithm Execution State	374
24.6.1	Data Polling Flag (DQ7)	376
24.6.2	Toggle Bit Flag (DQ6)	378
24.6.3	Timing Limit Exceeded Flag (DQ5)	379
24.6.4	Sector Erase Timer Flag (DQ3)	380
24.6.5	Toggle Bit-2 Flag (DQ2)	382
24.7	Detailed Explanation of Writing to and Erasing Flash Memory	384
24.7.1	Setting The Read/Reset State	385
24.7.2	Writing Data	386
24.7.3	Erasing All Data (Erasing Chips)	388
24.7.4	Erasing Optional Data (Erasing Sectors)	389
24.7.5	Suspending Sector Erase	391
24.7.6	Restarting Sector Erase	392
24.8	Notes on using 2M-bit Flash Memory	393
24.9	Reset Vector Address in Flash Memory	395
24.10	Example of Programming 2M/3M-bit Flash Memory	396
CHAPTER 25	EXAMPLES OF MB90F594A/MB90F594G/MB90F591A/MB90F591G SERIAL PROGRAMMING CONNECTION	401
25.1	Basic Configuration of MB90F594A/MB90F594G/MB90F591A/MB90F591G Serial Programming Connection	402
25.2	Example of Serial Programming Connection (User Power Supply Used)	406
25.3	Example of Serial Programming Connection (Power Supplied from the Programmer)	408
25.4	Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)	410

25.5 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer).....	412
--	-----

APPENDIX 415

APPENDIX A I/O Maps	416
APPENDIX B Instructions	426
B.1 Instruction Types	427
B.2 Addressing	428
B.3 Direct Addressing	430
B.4 Indirect Addressing	435
B.5 Execution Cycle Count	441
B.6 Effective Address Field	444
B.7 How to Read the Instruction List	445
B.8 F ² MC-16LX Instruction List	448
B.9 Instruction Map	462
APPENDIX C Timing Diagrams in Flash Memory Mode	484
APPENDIX D List of MB90590 Interrupt Vectors	489

INDEX 493

CHAPTER 1 OVERVIEW

The MB90590 Series is a family member of the F²MC-16LX microcontrollers.

- 1.1 "Product Overview"
- 1.2 "Features"
- 1.3 "Block Diagram"
- 1.4 "Pin Assignment"
- 1.5 "Package Dimensions"
- 1.6 "Pin Functions"
- 1.7 "Input-Output Circuits"
- 1.8 "Handling Device"

1.1 Product Overview

Table 1.1-1 "Product Overview" provides a quick outlook of the MB90590 Series.

■ Product Overview

Table 1.1-1 Product Overview

Features	MB90V590G	MB90F594A/F594G/F591A/ F591G	MB90594G/591/591G
Product type	Evaluation sample	Flash version	Mask ROM version
CPU	F ² MC-16LX CPU		
System clock	On-chip PLL clock multiplier (x1, x2, x3, x4, 1/2 when PLL stop) Minimum instruction execution time: 62.5 ns (4 MHz osc. PLL x4)		
ROM/Flash memory	External	Boot-block Flash memory 256K/384K bytes with Hard-wired reset vector	Mask ROM 256K/384K bytes
RAM	8Kbytes	6K/8K bytes	
Package	PGA-256	QFP100	
Emulator- specific power supply (*1)	None	-	

*1: It is setting of DIP switch S2 when Emulation pod (MB2145-507) is used.
Please refer to the MB2145-507 hardware manual (2.7 Emulator-specific Power Pin) about details.

Note:

With the product with G-suffix at the end of part numbers, functionality the CAN controller is enhanced.
Please refer to the description of the Bit Timing Register in the CAN chapter.

1.2 Features

Table 1.2-1 "MB90590 Features" lists the features of the MB90590 series.

■ Features

Table 1.2-1 MB90590 Features

Function	Feature
UART (3 channels)	Full duplex double buffer Supports asynchronous/synchronous (with start/stop bit) transfer Baud rate: 4808/5208/9615/10417/19230/38460/62500/500000 bps (asynchronous) 500k/1M/2M bps (synchronous) at System clock = 16 MHz
Serial I/O	Transfer can be started from MSB or LSB Supports internal clock synchronized transfer and external clock synchronized transfer Supports positive-edge and negative-edge clock synchronization Baud rate: 31.25k/62.5k/125k/500k/1M/2M bps at System clock = 16 MHz
A/D Converter	10 or 8-bit resolution 8 input channels Conversion time: 26.3 μ s (per one channel)
16-bit Reload Timer (2 channels)	Operation clock frequency: $f_{sys}/2^1$, $f_{sys}/2^3$, $f_{sys}/2^5$ (f_{sys} = System clock frequency) Supports External Event Count function
Watch Timer	Directly operates with the oscillation clock Facility to correct oscillation deviation Read/Write accessible Second/Minute/Hour registers Signals interrupts
16-bit I/O Timer	Signals an interrupt when overflow Supports Timer Clear when a match with Output Compare (Channel 0) Operation clock frequency: $f_{sys}/2^2$, $f_{sys}/2^4$, $f_{sys}/2^6$, $f_{sys}/2^8$ (f_{sys} = System clock frequency)
16-bit Output Compare (6 channels)	Signals an interrupt when a match with 16-bit I/O Timer Six 16-bit compare registers A pair of compare registers can be used to generate an output signal
16-bit Input Capture (6 channels)	Rising edge, falling edge or rising & falling edge sensitive Six 16-bit Capture registers Signals an interrupt upon external event

CHAPTER 1 OVERVIEW

Table 1.2-1 MB90590 Features (Continued)

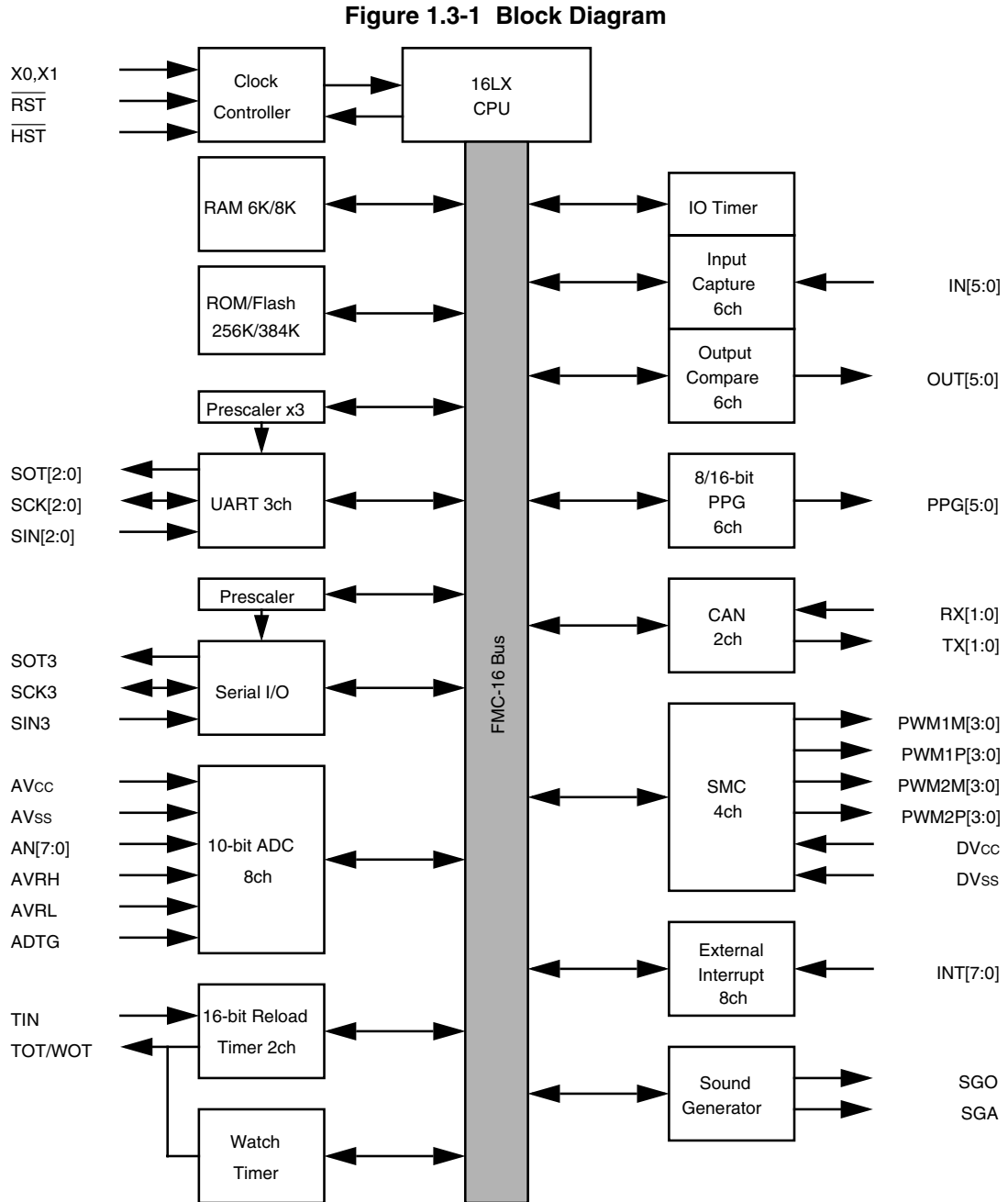
Function	Feature
8/16-bit Programmable Pulse Generator (6 channels)	<p>Supports 8-bit and 16-bit operation modes</p> <p>Twelve 8-bit reload counters</p> <p>Twelve 8-bit reload registers for L pulse width</p> <p>Twelve 8-bit reload registers for H pulse width</p> <p>A pair of 8-bit reload counters can be configured as one 16-bit reload counter or as 8-bit prescaler plus 8-bit reload counter</p> <p>6 output pins</p> <p>Operation clock frequency: f_{sys}, $f_{sys}/2^1$, $f_{sys}/2^2$, $f_{sys}/2^3$, $f_{sys}/2^4$ or $128 \mu s @ f_{osc} = 4 \text{ MHz}$</p> <p>($f_{sys}$ = System clock frequency, f_{osc} = Oscillation clock frequency)</p>
CAN Interface (2 channels)	<p>Conforms to CAN Specification Version 2.0 Part A and B</p> <p>Automatic re-transmission in case of error</p> <p>Automatic transmission responding to Remote Frame</p> <p>Prioritized 16 message buffers for data and ID's</p> <p>Supports multiple messages</p> <p>Flexible acceptance filter</p> <p>Full bit compare / Full bit mask / Two partial bit masks</p> <p>Supports up to 1M bps</p>
Stepping Motor Controller (4 channels)	<p>Four high current outputs for each channel</p> <p>Synchronized two 8-bit PWM's for each channel</p> <p>Succeeds to MB89940 design resource</p>
External Interrupt (8 channels)	<p>Either edge detection or level detection can be specified.</p> <p>The MB90V590G supports only four of the eight input channels.</p>
Sound Generator	<p>8-bit PWM signal is mixed with tone frequency from 8-bit reload counter</p> <p>PWM frequency: 62.5k, 31.2k, 15.6k, 7.8kHz at System clock = 16 MHz</p> <p>Tone frequency: $\text{PWM frequency} / 2 / (\text{reload value} + 1)$</p>
I/O Ports	<p>Virtually all external pins can be used as general purpose I/O</p> <p>All push-pull outputs and schmitt trigger inputs</p> <p>Bit-wise programmable as input/output or peripheral signal</p>
Flash Memory	<p>Supports automatic programming, Embedded Algorithm™ (*1)</p> <p>Write/Erase/Erase-Suspend/Resume commands</p> <p>A flag indicating completion of the algorithm</p> <p>Hard-wired reset vector available in order to point to a fixed boot sector in Flash Memory</p> <p>Boot block configuration</p> <p>Erase can be performed on each block</p>

*1: Embedded Algorithm is a trade mark of Advanced Micro Devices Inc.

1.3 Block Diagram

Figure 1.3-1 "Block Diagram" shows a block diagram of the MB90590 series.

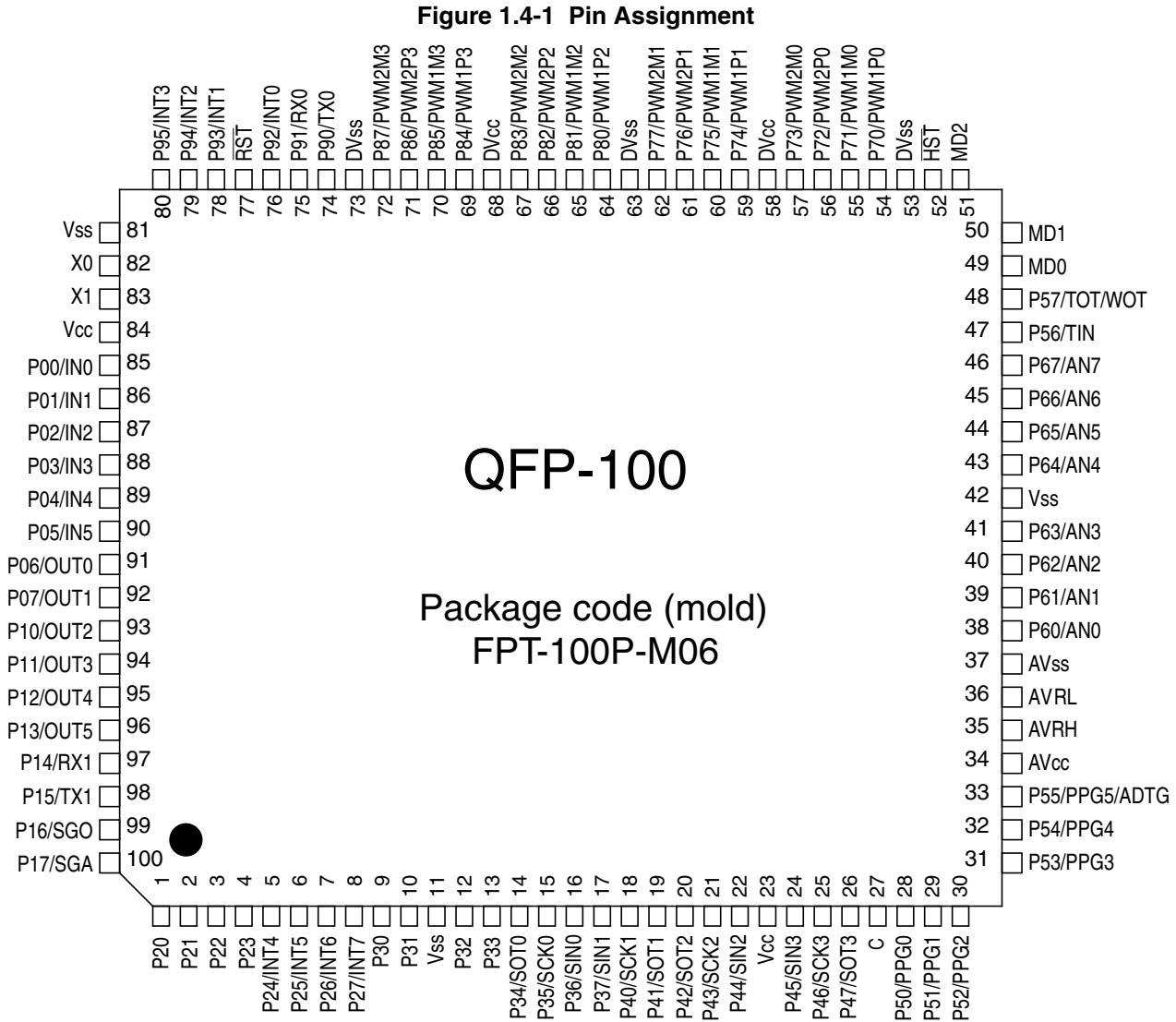
■ Block Diagram



1.4 Pin Assignment

Figure 1.4-1 "Pin Assignment" shows the pin assignments for the MB90590 series.

■ Pin Assignment



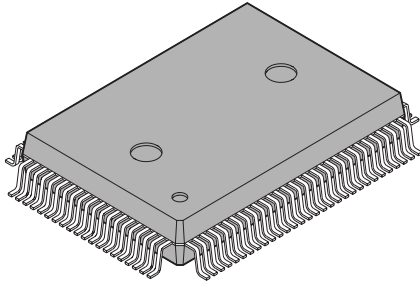
1.5 Package Dimensions

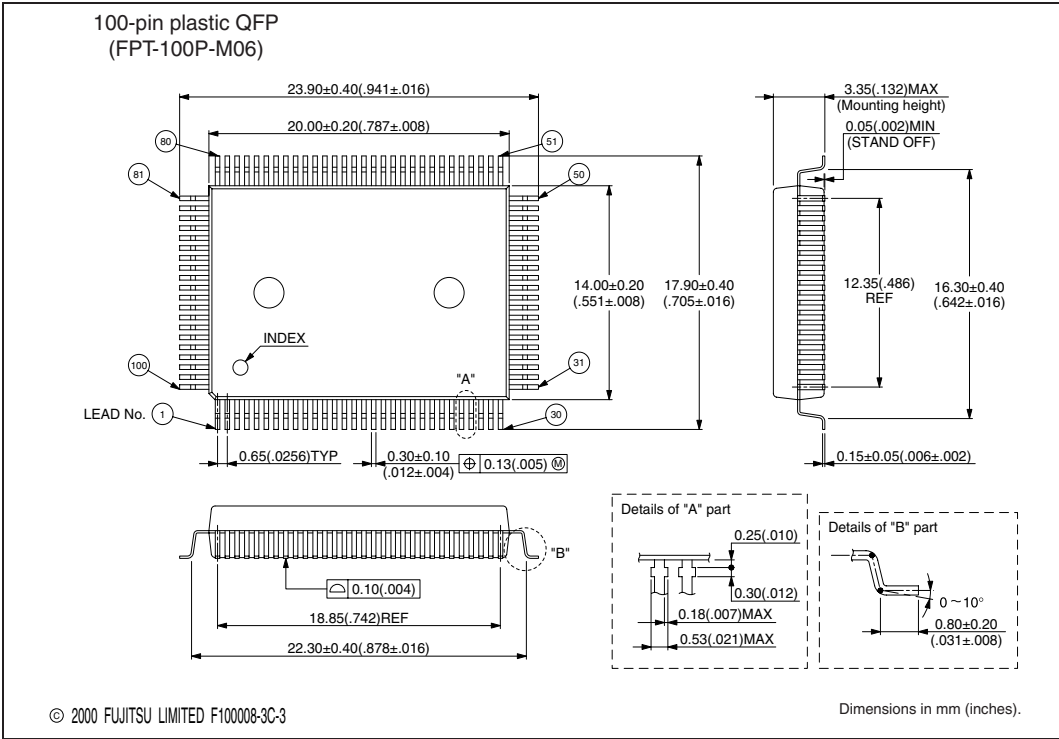
Figure 1.5-1 "Package Dimensions" shows the package dimensions of the MB90590 series.

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

■ Package Dimensions

Figure 1.5-1 Package Dimensions

 <p>100-pin plastic QFP</p> <p>(FPT-100P-M06)</p>	Lead pitch	0.65 mm
	Package width × package length	14 × 20 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Length of flat portion of pins	0.80 mm



1.6 Pin Functions

Table 1.6-1 "Pin Functions" describes the pin functions of the MB90590 series.

■ Pin Functions

Table 1.6-1 Pin Functions

No.	Pin name	Circuit type	Function
82	X0	A	Oscillation input
83	X1		Oscillation output
77	\overline{RST}	B	Reset input
52	\overline{HST}	C	Hardware standby input
85 to 90	P00 to P05	D	General purpose I/O
	IN0 to IN5		Inputs for the Input Captures
91 to 96	P06 to P07 P10 to P13	D	General purpose I/O
	OUT0 to OUT5		Outputs for the Output Compares. To enable the signal outputs, the corresponding bits of the Port Direction registers should be set to "1".
97	P14	D	General purpose I/O
	RX1		RX input for CAN Interface 1
98	P15	D	General purpose I/O
	TX1		TX output for CAN Interface 1. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
99	P16	D	General purpose I/O
	SGO		SGO output for the Sound Generator. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
100	P17	D	General purpose I/O
	SGA		SGA output for the Sound Generator. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
1 to 4	P20 to P23	D	General purpose I/O
5 to 8	P24 to P27	D	General purpose I/O
	INT4 to INT7		External interrupt input for INT4 to INT7 These pin functions are not supported by MB90V590G

Table 1.6-1 Pin Functions (Continued)

No.	Pin name	Circuit type	Function
9 to 10	P30 to P31	D	General purpose I/O
12 to 13	P32 to P33	D	General purpose I/O
14	P34	D	General purpose I/O
	SOT0		SOT output for UART 0. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
15	P35	D	General purpose I/O
	SCK0		SCK input/output for UART 0. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
16	P36	D	General purpose I/O
	SIN0		SIN input for UART 0
17	P37	D	General purpose I/O
	SIN1		SIN input for UART 1
18	P40	D	General purpose I/O
	SCK1		SCK input/output for UART 1
19	P41	D	General purpose I/O
	SOT1		SOT output for UART 1
20	P42	D	General purpose I/O
	SOT2		SOT output for UART 2
21	P43	D	General purpose I/O
	SCK2		SCK input/output for UART 2
22	P44	D	General purpose I/O
	SIN2		SIN input for UART 2
24	P45	D	General purpose I/O
	SIN3		SIN input for the Serial I/O
25	P46	D	General purpose I/O
	SCK3		SCK input/output for the Serial I/O
26	P47	D	General purpose I/O
	SOT3		SOT output for the Serial I/O
28 to 33	P50 to P55	D	General purpose I/O
	PPG0 to PPG5, ADTG		Outputs for the Programmable Pulse Generators. Pin number 33 is also shared with ADTG input for the external trigger of the A/D Converter.

CHAPTER 1 OVERVIEW

Table 1.6-1 Pin Functions (Continued)

No.	Pin name	Circuit type	Function
38 to 41	P60 to P63	E	General purpose I/O
	AN0 to AN3		Inputs for the A/D Converter
43 to 46	P64 to P67	E	General purpose I/O
	AN4 to AN7		Inputs for the A/D Converter
47	P56	D	General purpose I/O
	TIN		TIN input for the 16-bit Reload Timer
48	P57	D	General purpose I/O
	TOT/WOT		TOT output for the 16-bit Reload Timer and WOT output for the Watch Timer. Only one of three output enable flags in these peripheral blocks can be set at a time. Otherwise the output signal has no meaning.
54 to 57	P70 to P73	F	General purpose I/O
	PWM1P0 PWM1M0 PWM2P0 PWM2M0		Output for Stepping Motor Controller channel 0.
59 to 62	P74 to P77	F	General purpose I/O
	PWM1P1 PWM1M1 PWM2P1 PWM2M1		Output for Stepping Motor Controller channel 1.
64 to 67	P80 to P83	F	General purpose I/O
	PWM1P2 PWM1M2 PWM2P2 PWM2M2		Output for Stepping Motor Controller channel 2.
69 to 72	P84 to P87	F	F, GGeneral purpose I/O
	PWM1P3 PWM1M3 PWM2P3 PWM2M3		Output for Stepping Motor Controller channel 3.
74	P90	D	General purpose I/O
	TX0		TX output for CAN Interface 0
75	P91	D	General purpose I/O
	RX0		RX input for CAN Interface 0
76	P92	D	General purpose I/O
	INT0		External interrupt input for INT0

Table 1.6-1 Pin Functions (Continued)

No.	Pin name	Circuit type	Function
78	P93	D	General purpose I/O
	INT1		External interrupt input for INT1
79	P94	D	General purpose I/O
	INT2		External interrupt input for INT2
80	P95	D	General purpose I/O
	INT3		External interrupt input for INT3
58 68	DV _{CC}	-	Dedicated power supply pins for the high current output buffers (Pin No. 54 to 72)
53 63 73	DV _{SS}	-	Dedicated ground pins for the high current output buffers (Pin No. 54 to 72)
34	AV _{CC}	-	Dedicated power supply pin for the A/D Converter
37	AV _{SS}	-	Dedicated ground pin for the A/D Converter
35	AVRH	-	Upper reference voltage input for the A/D Converter
36	AVRL	-	Lower reference voltage input for the A/D Converter
49 50	MD0 MD1	C	Test mode inputs. These pins should be connected to V _{CC}
51	MD2	G	Test mode input. This pin should be connected to V _{SS}
27	C	-	External capacitor pin. A capacitor of 0.1μF should be connected to this pin and V _{SS} .
23 84	V _{CC}	-	Power supply pins
11 42 81	V _{SS}	-	Ground pins

1.7 Input-Output Circuits

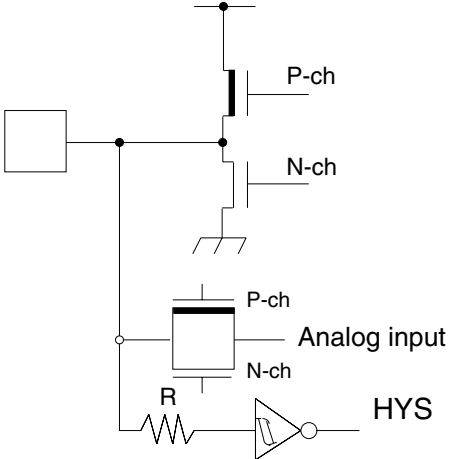
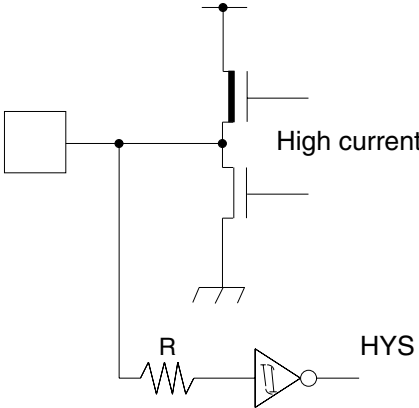
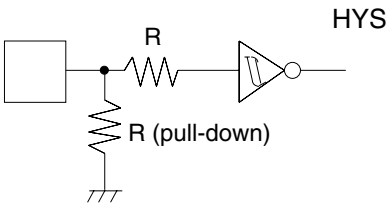
Table 1.7-1 "Input-output Circuits" lists the input-output circuits.

■ Input-output Circuits

Table 1.7-1 Input-output Circuits

Class	Circuit	Remarks
A		<ul style="list-style-type: none"> Oscillation feedback resistor: 1 MΩ approx.
B		<ul style="list-style-type: none"> Hysteresis input with pull-up Resistor Pull-up resistor: 50 kΩ approx.
C		<ul style="list-style-type: none"> Hysteresis input
D		<ul style="list-style-type: none"> CMOS output Hysteresis input

Table 1.7-1 Input-output Circuits (Continued)

Class	Circuit	Remarks
E		<ul style="list-style-type: none"> • CMOS output • Hysteresis input • Analog input
F		<ul style="list-style-type: none"> • CMOS high current output • Hysteresis input
G		<ul style="list-style-type: none"> • Hysteresis input with pull-down resistor Pull-down resistor: 50 kΩ approx. Flash version does not have pull-down resistor.

1.8 Handling Device

Special care is required for the following when handling the device:

- Preventing latch-up
 - Treatment of unused pins
 - Using external clock
 - Power supply pins (V_{CC}/V_{SS})
 - Pull-up/down resistors
 - Crystal Oscillator Circuit
 - Turning-on Sequence of Power Supply to A/D Converter and Analog Inputs
 - Connection of Unused Pins of A/D Converter
 - N.C. Pin
 - Precautions at power on
 - Initialization
 - Indeterminate outputs from ports 0 and 1
 - Using the "DIV A, Ri" and "DIVW A, RWi" instructions
 - Using REALOS
-

■ Handling the Device

○ Preventing latch-up

CMOS IC chips may suffer latch-up under the following conditions:

- A voltage higher than V_{CC} or lower than V_{SS} is applied to an input or output pin.
- A voltage higher than the rated voltage is applied between V_{CC} and V_{SS} .
- The AV_{CC} power supply is applied before the V_{CC} voltage.

Latch-up may increase the power supply current drastically, causing thermal damage to the device.

○ Treatment of unused pins

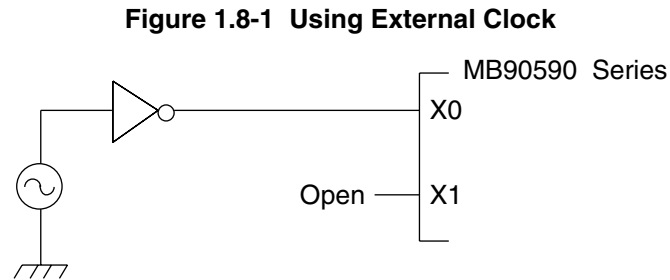
Leaving unused input pins open may result in misbehavior or latch up and possible permanent damage of the device. Therefore they must be pulled up or pulled down through resistors. In this case those resistors should be more than 2 K Ω .

Unused bidirectional pins should be set to the output state and can be left open, or the input state with the above described connection.

- **Using external clock**

To use external clock, drive the X0 pin and leave X1 pin open.

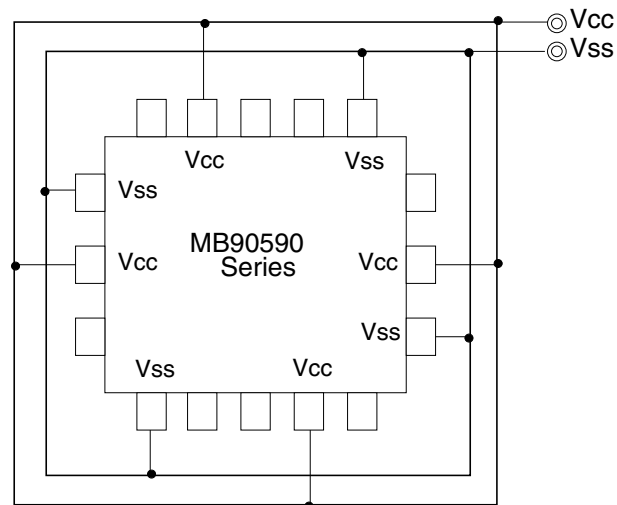
Figure 1.8-1 "Using External Clock" is a diagram of how to use external clock.



- **Power supply pins (V_{CC}/V_{SS})**

Ensure that all V_{CC} -level power supply pins are at the same potential. In addition, ensure the same for all V_{SS} -level power supply pins. (See the Figure 1.8-2 "Power Supply Pins (V_{CC}/V_{SS})".) If there are more than one V_{CC} or V_{SS} system, the device may operate incorrectly even within the guaranteed operating range.

Figure 1.8-2 Power Supply Pins (V_{CC}/V_{SS})



- **Pull-up/down resistors**

The MB90590 Series does not support internal pull-up/down resistors. Use external components where needed.

- **Crystal Oscillator Circuit**

Noises around X0 or X1 pins may be possible causes of abnormal operations. Make sure to provide bypass capacitors via shortest distance from X0, X1 pins, crystal oscillator (or ceramic resonator) and ground lines, and make sure, to the utmost effort, that lines of oscillation circuit not cross the lines of other circuits.

It is highly recommended to provide a printed circuit board art work surrounding X0 and X1 pins with a ground area for stabilizing the operation.

CHAPTER 1 OVERVIEW

○ **Turning-on Sequence of Power Supply to A/D Converter and Analog Inputs**

Make sure to turn on the A/D converter power supply (AV_{CC} , $AVRH$, $AVRL$) and analog inputs (AN0 to AN7) after turning-on the digital power supply (V_{CC}).

Turn-off the digital power after turning off the A/D converter supply and analog inputs. In this case, make sure that the voltage not exceed $AVRH$ or AV_{CC} (turning on/off the analog and digital power supplies simultaneously is acceptable).

○ **Connection of Unused Pins of A/D Converter**

Connect unused pins of A/D converter to $AV_{CC} = V_{CC}$, $AV_{SS} = AVRH = V_{SS}$.

○ **N.C. Pin**

The N.C. (internally connected) pin must be opened for use.

○ **Precautions at power on**

To prevent a malfunction of the internal step-down circuit, the voltage rise time at power-on should be 50 μ s or more (between 0.2 V and 2.7 V).

○ **Initialization**

In the device, there are internal registers which is initialized only by a power-on reset. To initialize these registers turning on the power again.

○ Indeterminate outputs from ports 0 and 1 (Except MB90F594G, MB90F591G, and MB90591G)

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

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

*: Other than P06, P07, P10 to P13, P16, P17. (The output of these pin become indeterminate only.)

Pay attention to the port output timing shown as follow

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

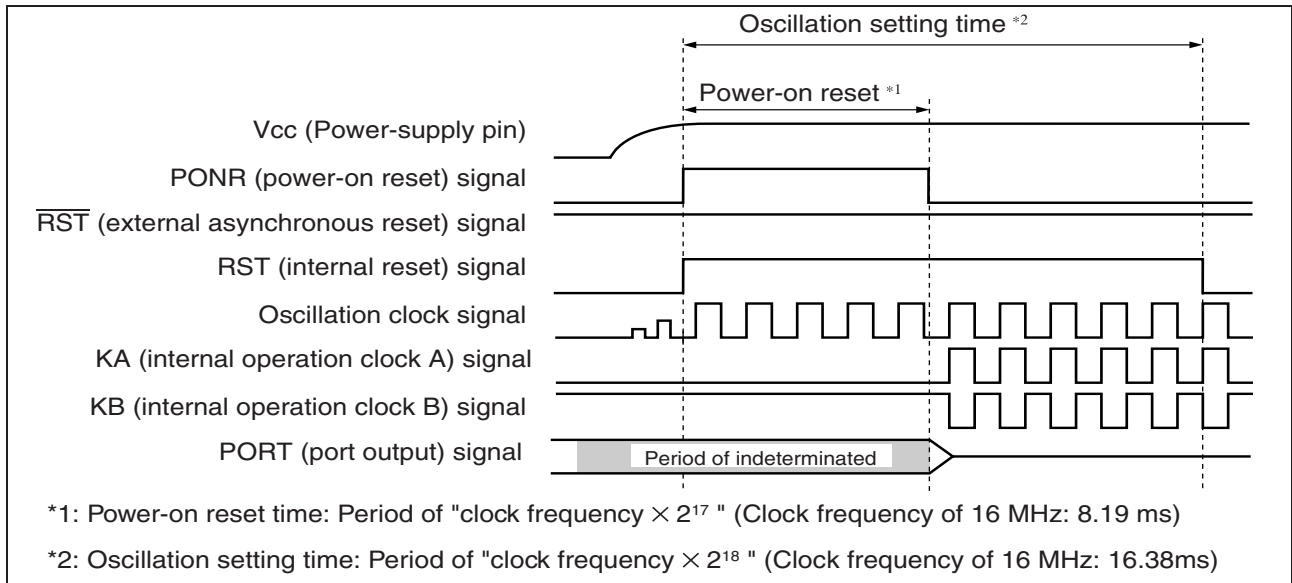
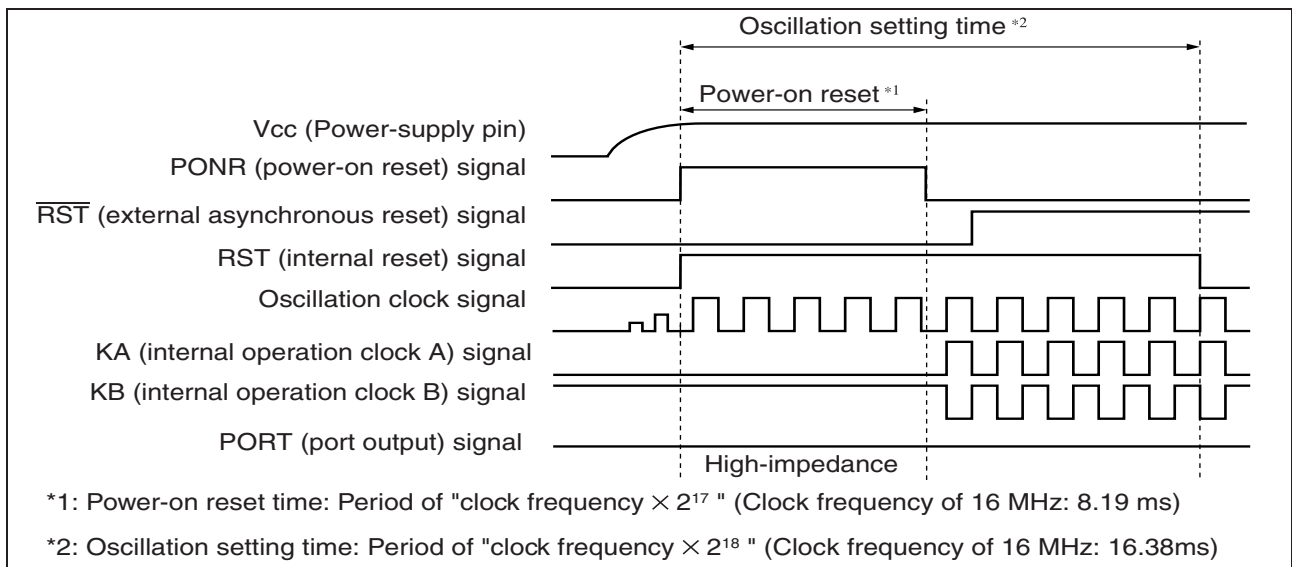


Figure 1.8-4 High-impedance output from ports 0 and 1 ($\overline{\text{RST}}$ pin is "L")



CHAPTER 1 OVERVIEW

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

Before using the multiplication and division instructions using signs ("DIV A, Ri" and "DIVW A, RWi"), set "00_H" in the corresponding bank registers (DTB, ADB, USB, and SSB). If the corresponding bank registers (DTB, ADB, USB, and SSB) are set to other than "00_H", the remainder in the execution results of the instruction is not stored in the register of the instruction operand. For more information, see Section 2.11 "Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions".

○ Using REALOS

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

○ Notes on during operation of PLL clock mode

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

CHAPTER 2 CPU

This chapter explains the CPU.

- 2.1 "Outline of CPU"
- 2.2 "Memory Space"
- 2.3 "Memory Space Map"
- 2.4 "Linear Addressing"
- 2.5 "Bank Addressing Types"
- 2.6 "Multi-byte Data in Memory Space"
- 2.7 "Registers"
- 2.8 "Register Bank"
- 2.9 "Prefix Codes"
- 2.10 "Interrupt Disable Instructions"
- 2.11 "Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions"

2.1 Outline of CPU

The F²MC-16LX CPU core is a 16-bit CPU designed for applications that require high-speed real-time processing, such as home-use or vehicle-mounted electronic appliances. The F²MC-16LX instruction set is designed for controller applications, and is capable of high-speed, highly efficient control processing.

■ Outline of CPU

In addition to 16-bit data, the F²MC-16LX CPU core can process 32-bit data by using an internal 32-bit accumulator. (32-bit data can be processed with some instructions.) Up to 16 Mbytes of memory space (expandable) can be used, which can be accessed by either the linear pointer or bank method. The instruction system, based on the F²MC-8 A-T architecture, has been reinforced by adding instructions compatible with high-level languages, expanding addressing modes, reinforcing multiplication and division instructions, and enhancing bit processing. The features of the F²MC-16LX CPU are explained below.

- **Minimum instruction execution time: 62.5 ns (at 4-MHz oscillation, 4 times clock multiplication)**

- **Maximum memory space: 16 Mbytes, accessed in linear or bank mode**

- **Instruction set optimized for controller applications**
 - Rich data types: Bit, byte, word, long word
 - Extended addressing modes: 23 types
 - High-precision operation (32-bit length) based on 32-bit accumulator
- **Powerful interrupt functions**
Eight priority levels (programmable)
- **CPU-independent automatic transfer**
Up to 16 channels of the extended intelligent I/O service
- **Instruction set compatible with high-level language (C)/multitasking**
System stack pointer/instruction set symmetry/barrel-shift instructions
- **Improved execution speed: 4-byte queue**

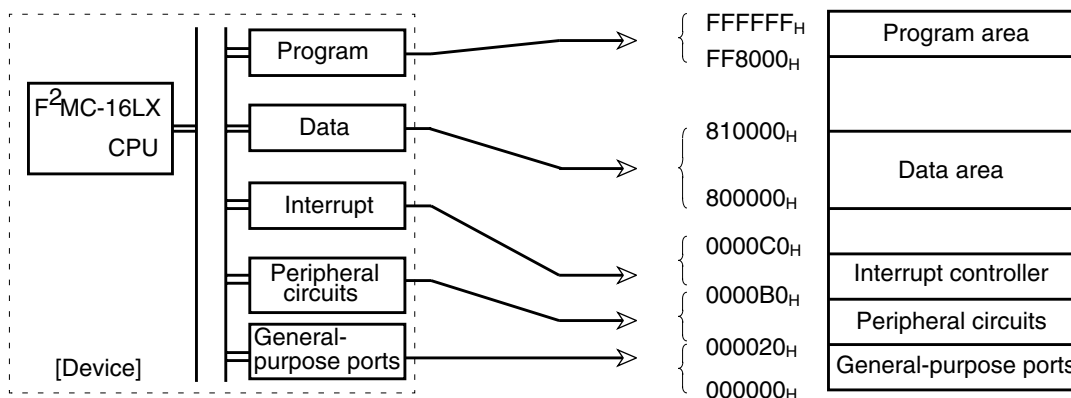
2.2 Memory Space

An F²MC-16LX CPU has a 16-Mbyte memory space. All data program input and output managed by the F²MC-16LX CPU are located in this 16-Mbyte memory space. The CPU accesses the resources by indicating their addresses using a 24-bit address bus.

■ Outline of CPU Memory Space

Figure 2.2-1 "Sample Relationship between F²MC-16LX System and Memory Map" shows a sample relationship between the F²MC-16LX system and memory map.

Figure 2.2-1 Sample Relationship between F²MC-16LX System and Memory Map



■ Address Generation Types

The F²MC-16LX has the following two addressing:

○ Linear addressing

An entire 24-bit address is specified by an instruction.

○ Bank addressing

The eight high-order bits of an address are specified by an appropriate bank register, and the remaining 16 low-order bits are specified by an instruction.

2.3 Memory Space Map

The memory space of the MB90590 Series is shown in Figure 2.3-1 "Memory Space Map".

■ Memory Space Map

The high-order portion of bank 00 gives the image of the FF bank ROM to make the small model of the C compiler effective. Since the low-order 16 bits are the same, the table in ROM can be referenced without using the far specification in the pointer declaration.

For example, an attempt to access 00C000_H accesses the value at FFC000_H in ROM.

The ROM area in bank FF exceeds 48 Kbytes, and its entire image cannot be shown in bank 00.

The image between FF4000_H and FFFFFFF_H is visible in bank 00, while the image between FF0000_H and FF3FFF_H is visible only in bank FF.

Figure 2.3-1 Memory Space Map

MB90V590G		MB90F594A/MB90F594G/MB90594G		MB90F591A/MB90591 MB90F591G/MB90591G	
FFFFFF _H	ROM (FF bank)	FFFFFF _H	ROM (FF bank)	FFFFFF _H	ROM (FF bank)
FF0000 _H FEFFFF _H	ROM (FE bank)	FF0000 _H FEFFFF _H	ROM (FE bank)	FF0000 _H FEFFFF _H	ROM (FE bank)
FE0000 _H FDFFFF _H	ROM (FD bank)	FE0000 _H FDFFFF _H	ROM (FD bank)	FE0000 _H FDFFFF _H	ROM (FD bank)
FD0000 _H FCFFFF _H	ROM (FC bank)	FD0000 _H FCFFFF _H	ROM (FC bank)	FD0000 _H FCFFFF _H	
FC0000 _H FBFFFF _H	ROM (FB bank)	FC0000 _H	ROM (FC bank)	FC0000 _H FBFFFF _H	ROM (FB bank)
FB0000 _H FAFFFF _H	ROM (FA bank)			FB0000 _H FAFFFF _H	ROM (FA bank)
FA0000 _H F9FFFF _H	ROM (F9 bank)			FA0000 _H F9FFFF _H	ROM (F9 bank)
F90000 _H				F90000 _H	
00FFFF _H	ROM (Image of FF bank)	00FFFF _H	ROM (Image of FF bank)	00FFFF _H	ROM (Image of FF bank)
004000 _H		004000 _H		004000 _H	
0028FF _H	RAM 2K			0028FF _H	RAM 2K
002100 _H 0020FF _H				002100 _H 0020FF _H	
001FFF _H	Peripheral	001FFF _H	Peripheral	001FFF _H	Peripheral
001900 _H 0018FF _H	RAM 6K	001900 _H 0018FF _H	RAM 6K	001900 _H 0018FF _H	RAM 6K
000100 _H		000100 _H		000100 _H	
0000BF _H 000000 _H	Peripheral	0000BF _H 000000 _H	Peripheral	0000BF _H 000000 _H	Peripheral

2.4 Linear Addressing

There are two types of linear addressing:

- **24-bit operand specification:** Directly specifies a 24-bit address using operands.
- **32-bit register indirect specification:** Indirectly specifies the 24 low-order bits of a 32-bit general-purpose register value as the address.

■ 24-bit Operand Specification

Figure 2.4-1 "Example of Linear Method (24-bit Register Operand Specification)" shows an example of 24-bit operand specification. Figure 2.4-2 "Example of Linear Method (32-bit Register Indirect Specification)" shows an example of 32-bit register indirect specification.

Figure 2.4-1 Example of Linear Method (24-bit Register Operand Specification)

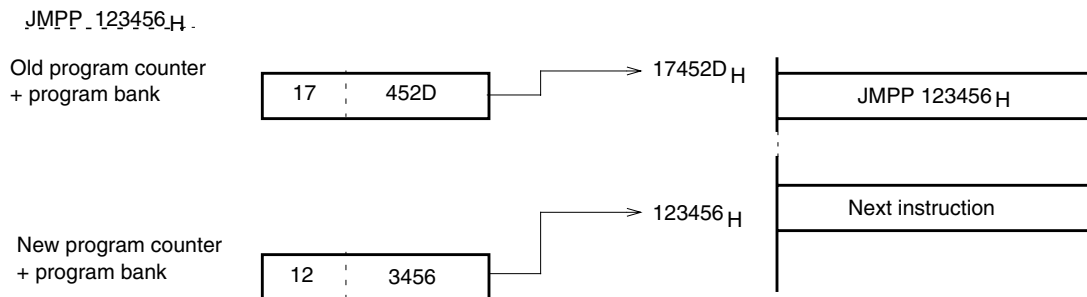
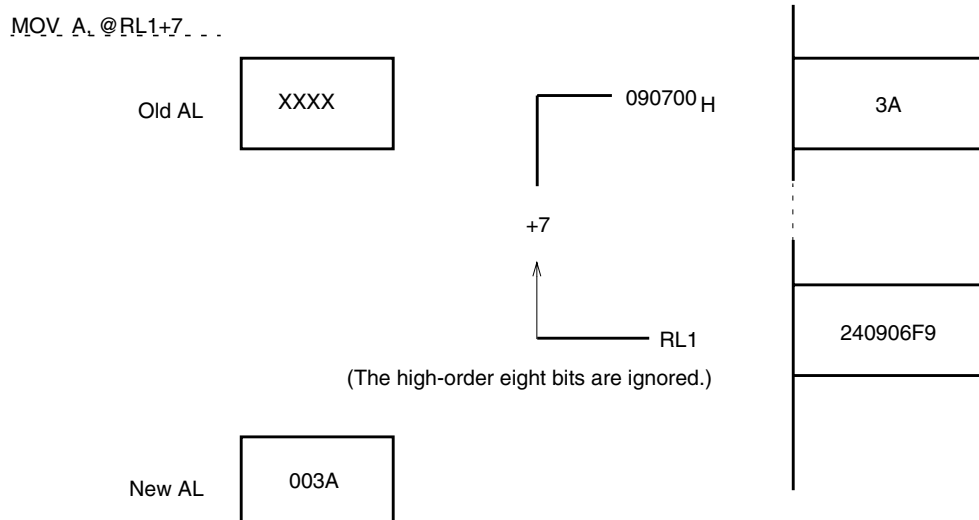


Figure 2.4-2 Example of Linear Method (32-bit Register Indirect Specification)



2.5 Bank Addressing Types

In the bank method, the 16-Mbyte space is divided into 256 64-Kbyte banks. The following five bank registers are used to specify the banks corresponding to each space:

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

■ Bank Addressing Types

○ Program bank register (PCB)

The 64-Kbyte bank specified by the PCB is called a program (PC) space. The PC space contains instruction codes, vector tables, and immediate value data, for example.

○ Data bank register (DTB)

The 64-Kbyte bank specified by the DTB is called a data (DT) space. The DT space contains readable/writable data, and control/data registers for internal and external resources.

○ User stack bank register (USB)/system stack bank register (SSB)

The 64-Kbyte bank specified by the USP or SSP is called a stack (SP) space. The SP space is accessed when a stack access occurs during a push/pop instruction or interrupt register saving. The S flag in the condition code register determines the stack space to be accessed.

○ **Additional bank register (ADB)**

The 64-Kbyte bank specified by the ADB is called an additional (AD) space. The AD space, for example, contains data that cannot fit into the DT space.

Table 2.5-1 "Default Space" lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency. To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank space corresponding to the specified prefix code.

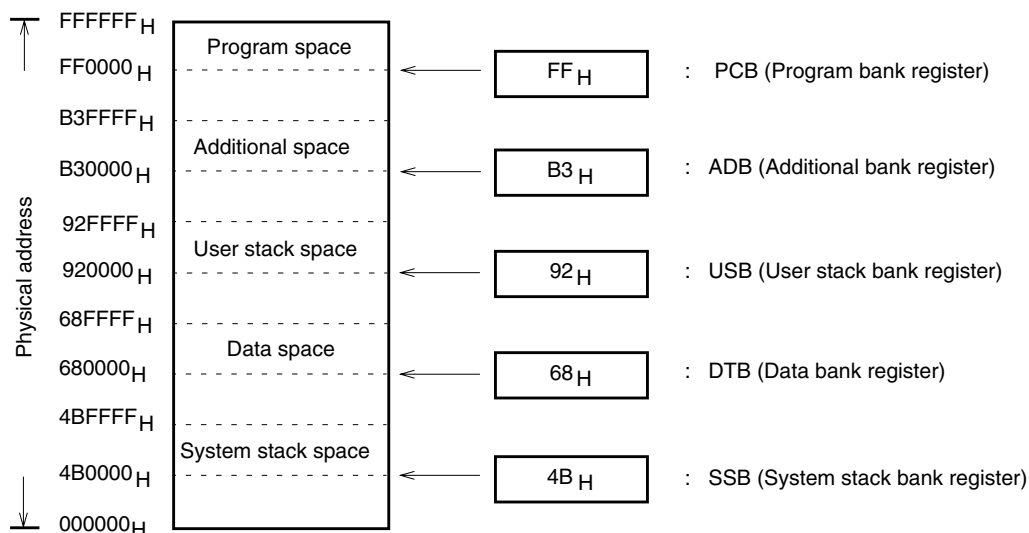
After reset, the DTB, USB, SSB, and ADB are initialized to 00_H. The PCB is initialized to a value specified by the reset vector. After reset, the DT, SP, and AD spaces are allocated in bank 00_H (000000_H to 00FFFF_H), and the PC space is allocated in the bank specified by the reset vector.

Table 2.5-1 Default Space

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

Figure 2.5-1 "Physical Addresses of Each Space" is an example of a memory space divided into register banks.

Figure 2.5-1 Physical Addresses of Each Space



2.6 Multi-byte Data in Memory Space

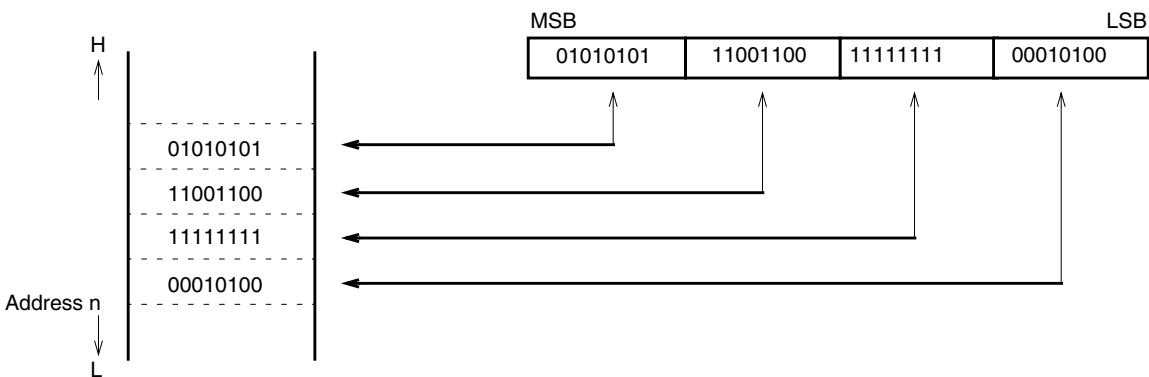
Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the low-order 16 bits are transferred before the high-order 16 bits.

If a reset signal is input immediately after the low-order bits are written, the high-order bits might not be written.

Multi-byte Data Allocation in Memory Space

Figure 2.6-1 "Sample Allocation of Multi-byte Data in Memory" is a diagram of multi-byte data configuration in memory. The low-order eight bits of a data item are stored at address n, then address n+1, address n+2, address n+3, etc.

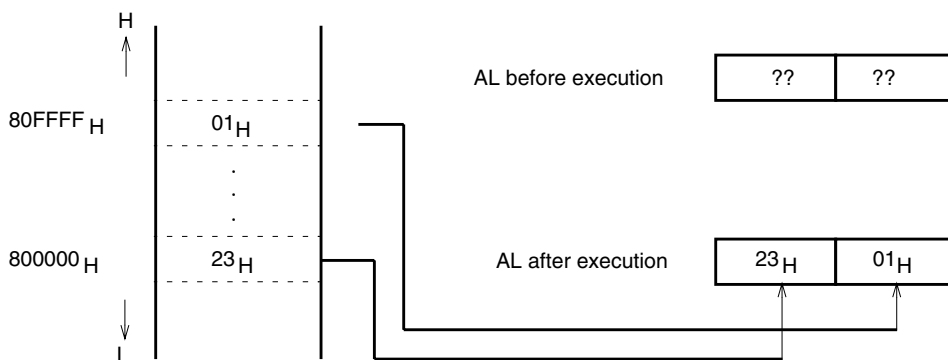
Figure 2.6-1 Sample Allocation of Multi-byte Data in Memory



Accessing Multi-byte Data

Fundamentally, accesses are made within a bank. For an instruction accessing a multi-byte data item, address FFFF_H is followed by address 0000_H of the same bank. Figure 2.6-2 "Execution of MOVW A, 080FFFF_H" is an example of an instruction accessing multi-byte data.

Figure 2.6-2 Execution of MOVW A, 080FFFF_H



2.7 Registers

The F²MC-16LX registers are largely classified into two types: special registers in the CPU and general-purpose registers in memory. The special registers are dedicated internal hardware of the CPU, and they have specific use defined by the CPU architecture. The general-purpose registers share the CPU address space with RAM. The general-purpose registers are the same as the special registers in that they can be accessed without using an address. The applications of the general-purpose registers can be specified by the user however, as is ordinary memory space.

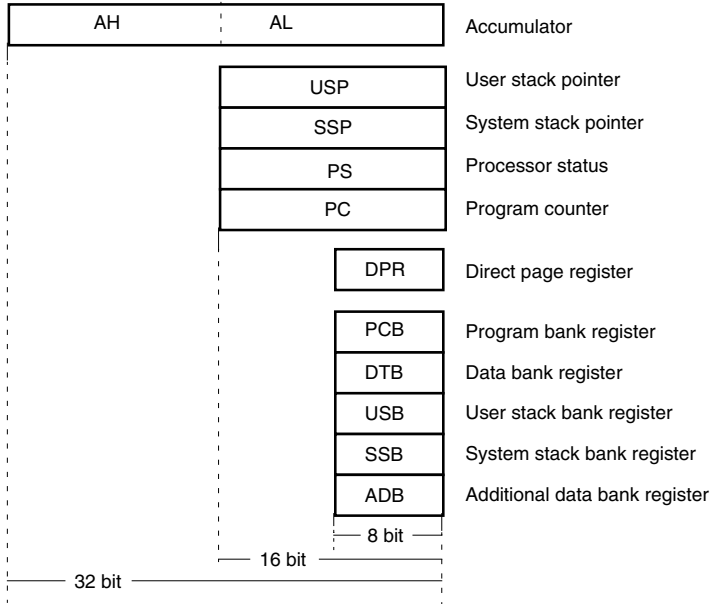
■ Special Registers

The F²MC-16LX CPU core has the following 13 special registers:

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

Figure 2.7-1 "Special Registers" is a diagram of the special registers.

Figure 2.7-1 Special Registers

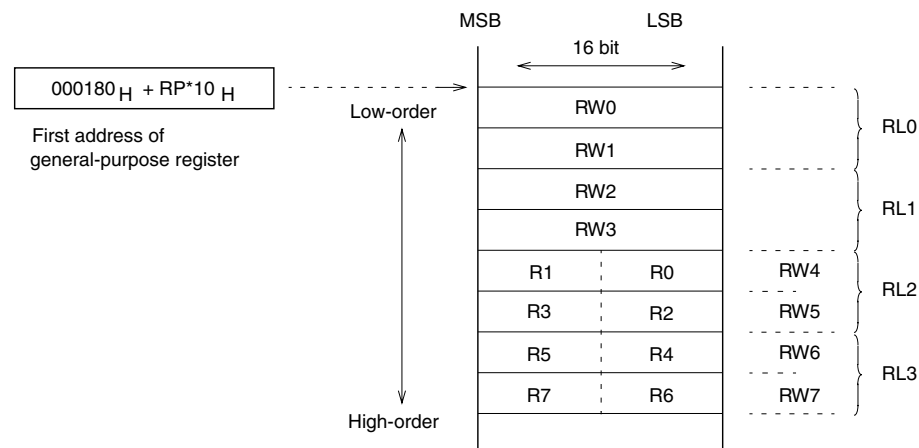


■ General-purpose Registers

The F²MC-16LX general-purpose registers are located from addresses 000180_H to 00037F_H (maximum configuration) of main storage. The register bank pointer (RP) indicates which of the above addresses are currently being used as a register bank. Each bank has the following three types of registers. These registers are mutually dependent as described in Figure 2.7-2 "General-purpose Registers".

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

Figure 2.7-2 General-purpose Registers



The relationship between the high-order and low-order bytes of a byte or word register is expressed as follows:

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

The relationship between the high-order and low-order bytes of RL_i and RW can be expressed as follows:

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

2.7.1 Accumulator (A)

The accumulator (A) register consists of two 16-bit arithmetic operation registers (AH and AL), and is used as a temporary storage for operation results and transfer data.

■ Accumulator (A)

The A register consists of two 16-bit arithmetic operation registers (AH and AL). The A register is used as a temporary storage for operation results and transfer data. During 32-bit data processing, AH and AL are used together. Only AL is used for word processing in 16-bit data processing mode or for byte processing in 8-bit data processing mode (see Figure 2.7-3 "32-bit Data Transfer" and Figure 2.7-4 "AL-AH Transfer"). The data stored in the A register can be operated upon with the data in memory or registers (Ri, Rwi, or Rli). In the same manner as with the F²MC-8L, when a word or shorter data item is transferred to AL, the previous data item in AL is automatically sent to AH (data preservation function). The data preservation function and operation between AL and AH help improve processing efficiency.

When a byte or shorter data item is transferred to AL, the data is sign-extended or zero-extended and stored as a 16-bit data item in AL. The data in AL can be handled either as word or byte long.

When a byte-processing arithmetic operation instruction is executed on AL, the high-order eight bits of AL before operation are ignored. The high-order eight bits of the operation result all become zeroes.

The A register is not initialized by a reset. The A register holds an undefined value immediately after a reset.

Figure 2.7-3 32-bit Data Transfer

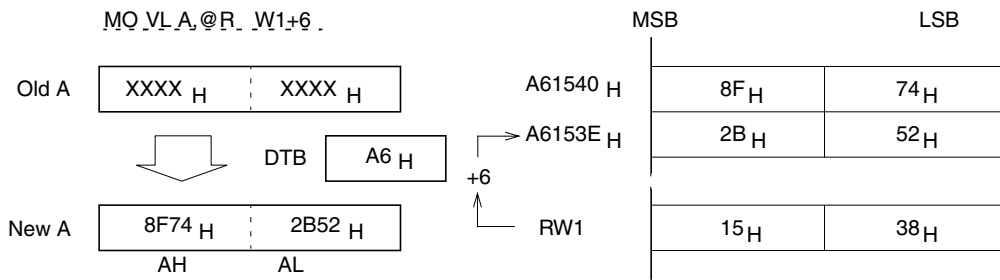
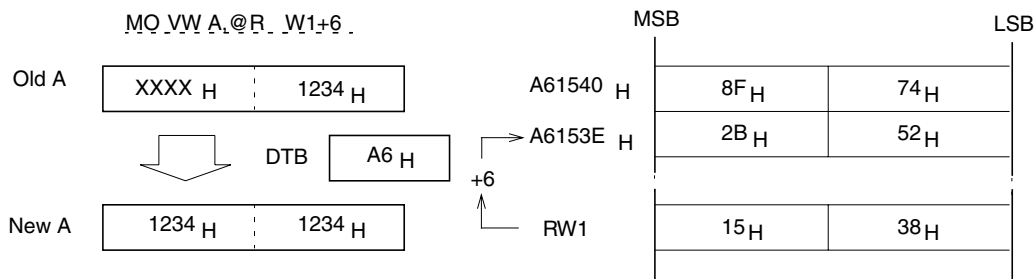


Figure 2.7-4 AL-AH Transfer



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

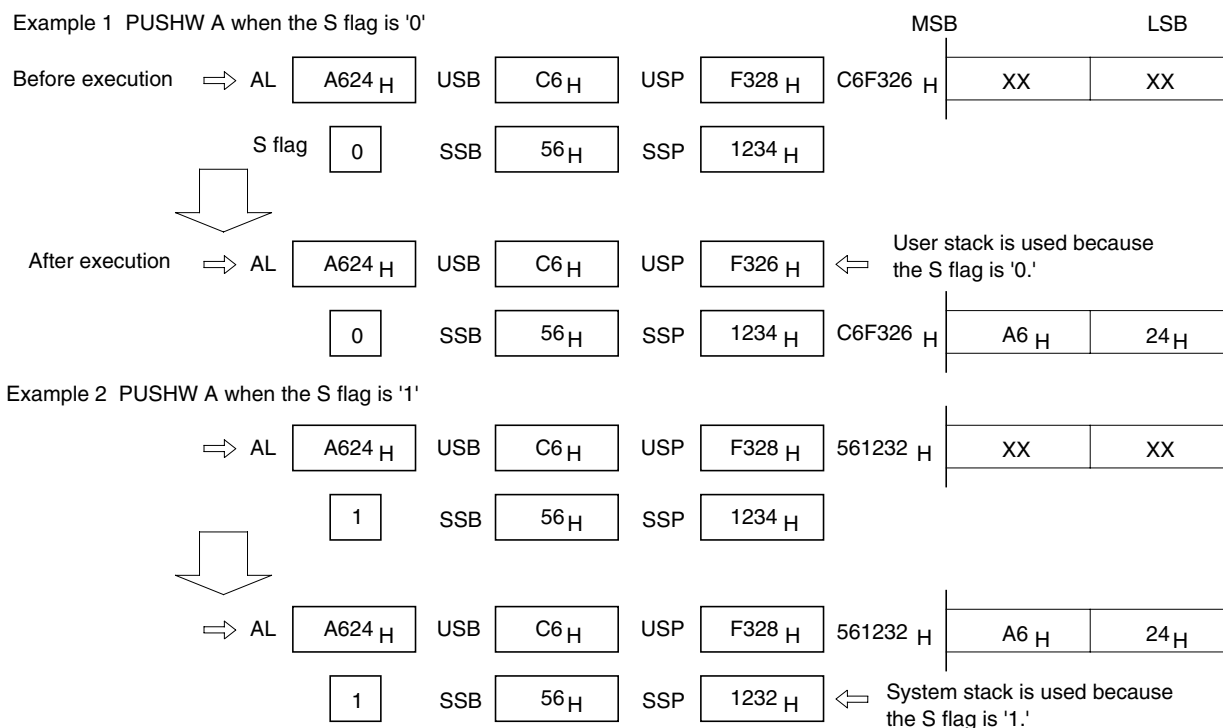
USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data when a push/pop instruction or subroutine is executed.

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

USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data in the event of a push/pop instruction or subroutine execution. The USP and SSP registers are used by stack instructions. The USP register is enabled when the S flag in the processor status register is '0,' and the SSP register is enabled when the S flag is '1' (see Figure 2.7-5 "Stack Manipulation Instruction and Stack Pointer"). Since the S flag is set when an interrupt is accepted, register values are always saved in the memory area indicated by SSP during interrupt processing. SSP is used for stack processing in an interrupt routine, while USP is used for stack processing outside an interrupt routine. If the stack space is not divided, use only the SSP.

During stack processing, the high-order eight bits of an address are indicated by SSB (for SSP) or USB (for USP). USP and SSP are not initialized by a reset. Instead, they hold undefined values.

Figure 2.7-5 Stack Manipulation Instruction and Stack Pointer



Note:

Specify an even-numbered address in the stack pointer whenever possible.

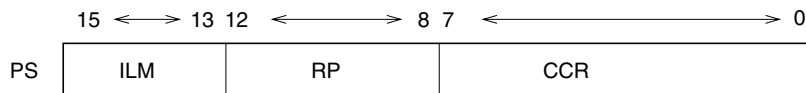
2.7.3 Processor Status (PS)

The PS register consists of the bits controlling the CPU Operation and the bits indicating the CPU status.

■ Processor Status (PS)

As shown in Figure 2.7-6 "Processor Status (PS) Structure", the high-order byte of the PS register consists of a register bank pointer (RP) and an interrupt level mask register (ILM). The RP indicates the start address of a register bank. The low-order byte of the PS register is a condition code register (CCR), containing the flags to be set or reset depending on the results of instruction execution or interrupt occurrences.

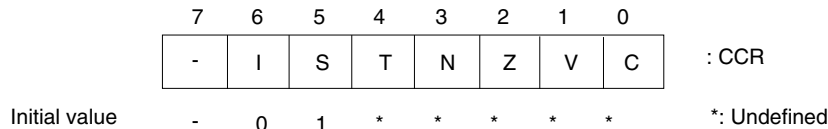
Figure 2.7-6 Processor Status (PS) Structure



■ Condition Code Register (CCR)

Figure 2.7-7 "Condition Code Register (CCR) Configuration" is a diagram of condition code register configuration.

Figure 2.7-7 Condition Code Register (CCR) Configuration



○ **I: Interrupt enable flag:**

Interrupts other than software interrupts are enabled when the I flag is 1 and are masked when the I flag is 0. The I flag is cleared by a reset.

○ **S: Stack flag:**

When the S flag is 0, USP is enabled as the stack manipulation pointer.
 When the S flag is 1, SSP is enabled as the stack manipulation pointer.
 The S flag is set by an interrupt reception or a reset.

○ **T: Sticky bit flag:**

1 is set in the T flag when there is at least one '1' in the data shifted out from the carry after execution of a logical right/arithmetic right shift instruction. Otherwise, 0 is set in the T flag. In addition, '0' is set in the T flag when the shift amount is zero.

○ **N: Negative flag:**

The N flag is set when the MSB of the operation result is '1,' and is otherwise cleared.

○ **Z: Zero flag:**

The Z flag is set when the operation result is all zeroes, and is otherwise cleared.

○ **V: Overflow flag:**

The V flag is set when an overflow of a signed value occurs as a result of operation execution and is otherwise cleared.

○ **C: Carry flag:**

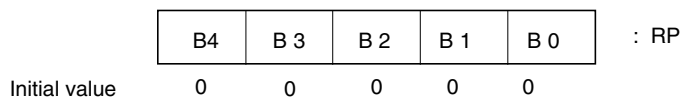
The C flag is set when a carry-up or carry-down from the MSB occurs as a result of operation execution, and is otherwise cleared.

■ **Register Bank Pointer (RP)**

The RP register indicates the relationship between the general-purpose registers of the F²MC-16LX and the internal RAM addresses. Specifically, the RP register indicates the first memory address of the currently used register bank in the following conversion expression: $[00180_{\text{H}} + (\text{RP}) * 10_{\text{H}}]$ (see Figure 2.7-8 "Register Bank Pointer (RP)"). The RP register consists of five bits, and can take a value between 00H and 1FH. Register banks can be allocated at addresses from 000180_H to 00037_H in memory.

Even within that range, however, the register banks cannot be used as general-purpose registers if the banks are not in internal RAM. The RP register is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the RP register; however, only the low-order five bits of that data are used.

Figure 2.7-8 Register Bank Pointer (RP)



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

The ILM register consists of three bits, indicating the CPU interrupt masking level. An interrupt request is accepted only when the level of the interrupt is higher than that indicated by these three bits. Level 0 is the highest priority interrupt, and level 7 is the lowest priority interrupt (see Table 2.7-1 "Levels Indicated by the Interrupt Level Mask (ILM) Register"). Therefore, for an interrupt to be accepted, its level value must be smaller than the current ILM value. When an interrupt is accepted, the level value of that interrupt is set in ILM. Thus, an interrupt of the same or lower level cannot be accepted subsequently. ILM is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the ILM register, but only the low-order three bits of that data are used.

Figure 2.7-9 Interrupt Level Register (ILM)

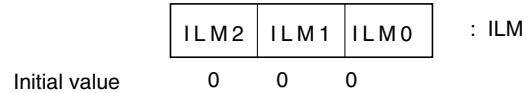


Table 2.7-1 Levels Indicated by the Interrupt Level Mask (ILM) Register

ILM2	ILM1	ILM0	Level value	Acceptable interrupt level
0	0	0	0	Interrupt disabled
0	0	1	1	0 only
0	1	0	2	Level value smaller than 1
0	1	1	3	Level value smaller than 2
1	0	0	4	Level value smaller than 3
1	0	1	5	Level value smaller than 4
1	1	0	6	Level value smaller than 5
1	1	1	7	Level value smaller than 6

2.7.4 Program Counter (PC)

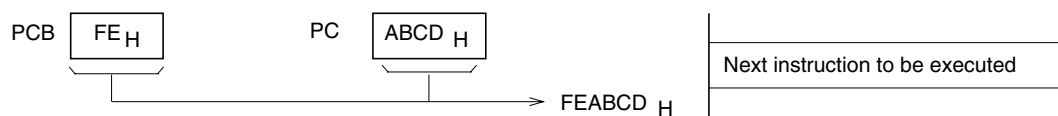
The PC register is a 16-bit counter that indicates the low-order 16 bits of the memory address of an instruction code to be executed by the CPU. The high-order eight bits of the address are indicated by the PCB. The PC register is updated by a conditional branch instruction, subroutine call instruction, interrupt, or reset.

The PC register can also be used as a base pointer for operand access.

■ Program Counter (PC)

Figure 2.7-10 "Program Counter" shows the program counter.

Figure 2.7-10 Program Counter



2.8 Register Bank

A register bank consists of eight words. The register bank can be used as the following general-purpose registers for arithmetic operations: byte registers R0 to R7, word registers RW0 to RW7, and long word registers RL0 to RL3. In addition, the register bank can be used as instruction pointers.

■ Register Bank

Table 2.8-1 "Register Functions" lists the functions of the registers. Table 2.8-2 "Relationship between Registers" indicates the relationship between the registers.

In the same manner as for an ordinary RAM area, the register bank values are not initialized by a reset. The status before a reset is maintained. When the power is turned on, however, the register bank will have an undefined value.

Table 2.8-1 Register Functions

R0 to R7	Used as operands of instructions. Note: R0 is also used as a counter for barrel shift or normalization instructions.
RW0 to RW7	Used as pointers. Used as operands of instructions. Note: RW0 is used as a counter for string instructions.
RL0 to RL3	Used as long pointers. Used as operands of instructions.

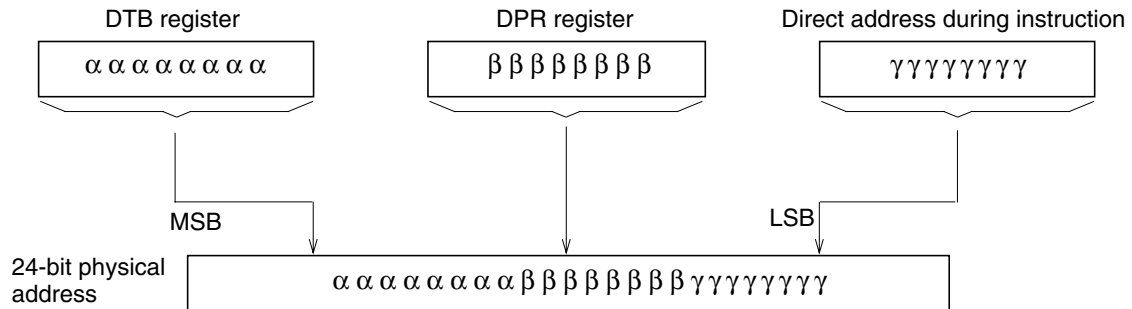
Table 2.8-2 Relationship between Registers

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

○ **Direct page register (DPR) <Initial value: 01_H>**

DPR specifies addr8 to addr15 of the instruction operands in direct addressing mode as shown in Figure 2.8-1 "Generating a Physical address in Direct Addressing Mode". DPR is eight bits long, and is initialized to 01_H by a reset. DPR can be read or written to by an instruction.

Figure 2.8-1 Generating a Physical address in Direct Addressing Mode



○ **Program counter bank register (PCB) <Initial value: Value in reset vector>**

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

○ **User stack bank register(USB) <Initial value: 00_H>**

○ **System stack bank register(SSB) <Initial value: 00_H>**

○ **Additional data bank register(ADB) <Initial value: 00_H>**

Each bank register indicates the memory bank where the PC, DT, SP (user), SP (system), or AD space is allocated. All bank registers are one byte long. PCB is initialized to 00_H by a reset. Bank registers other than PCB can be read or written to. PCB can be read but cannot be written to.

PCB is updated when the JMPP, CALLP, RETP, RETIQ, or RETF instruction branching to the entire 16-Mbyte space is executed or when an interrupt occurs. For operation of each register, see Section 2.2 "Memory space".

2.9 Prefix Codes

Placing a prefix code before an instruction partially changes the operation of the instruction. Three types of prefix codes can be used: bank select prefix, common register bank prefix, and flag change disable prefix.

■ Bank Select Prefix

The memory space used for accessing data is determined for each addressing mode.

When a bank select prefix is placed before an instruction, the memory space used for accessing data by that instruction can be selected regardless of the addressing mode.

Table 2.9-1 "Bank Select Prefix" lists the bank select prefixes and the corresponding memory spaces.

Table 2.9-1 Bank Select Prefix

Bank select prefix	Space selected
PCB	PC space
DTB	Data space
ADB	AD space
SPB	Either the SSP or USP space is used according to the stack flag value.

Use the following instructions with care:

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

The bank register specified by an operand is used regardless of the prefix.

○ **Stack manipulation instructions (PUSHW, POPW)**

SSB or USB is used according to the S flag regardless of the prefix.

○ **I/O access instructions**

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

The IO space of the bank is used regardless of the prefix.

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

The instruction is executed normally, but the prefix affects the next instruction.

○ **POPW PS**

SSB or USB is used according to the S flag regardless of the prefix. The prefix affects the next instruction.

- **MOV ILM,#imm8**

The instruction is executed normally, but the prefix affects the next instruction.

- **RETI**

SSB is used regardless of the prefix.

- **Common Register Bank Prefix (CMR)**

To simplify data exchange between multiple tasks, the same register bank must be accessed relatively easily regardless of the RP value. When CMR is placed before an instruction that accesses a register bank, that instruction accesses the common bank (the register bank selected when RP=0) at addresses from 000180_H to 00018F_H regardless of the current RP value. Use the following instructions with care:

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

If an interrupt request occurs during execution of a string instruction with a prefix code, the prefix code becomes invalid when the string instruction is resumed after the interrupt is processed. Thus, the string instruction is executed falsely after the interrupt is processed. Do not prefix any of the above string instructions with CMR.

- **Flag change instructions (AND CCR,#imm8, OR CCR,#imm8, POPW PS)**

The instruction is executed normally, but the prefix affects the next instruction.

- **MOV ILM,#imm8**

The instruction is executed normally, but the prefix affects the next instruction.

- **Flag Change Disable Prefix (NCC)**

To disable flag changes, use the flag change disable prefix code (NCC). Placing NCC before an instruction disables flag changes associated with that instruction. Use the following instructions with care:

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

If an interrupt request occurs during execution of a string instruction with a prefix code, the prefix code becomes invalid when the string instruction is resumed after the interrupt is processed. Thus, the string instruction is executed incorrectly after the interrupt is processed. Do not prefix any of the above string instructions with NCC.

- **Flag change instructions (AND CCR,#imm8, OR CCR,#imm8, POPW PS)**

The instruction is executed normally, but the prefix affects the next instruction.

- **Interrupt instructions (INT #vct8, INT9, INT addr16, INTP addr24, RETI)**

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

- **JCTX @A**

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

- **MOV ILM,#imm8**

The instruction is executed normally, but the prefix affects the next instruction.

2.10 Interrupt Disable Instructions

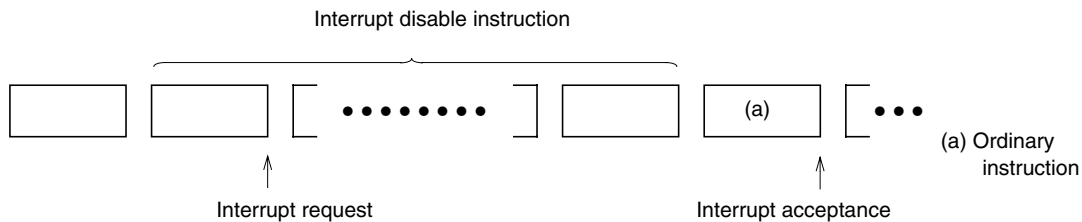
Interrupt requests are not sampled for the following ten instructions:

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

■ Interrupt Disable Instructions

If a valid interrupt request occurs during execution of any of the above instructions, the interrupt can be processed only when an instruction other than the above is executed. For details, see Figure 2.10-1 "Interrupt Disable Instruction".

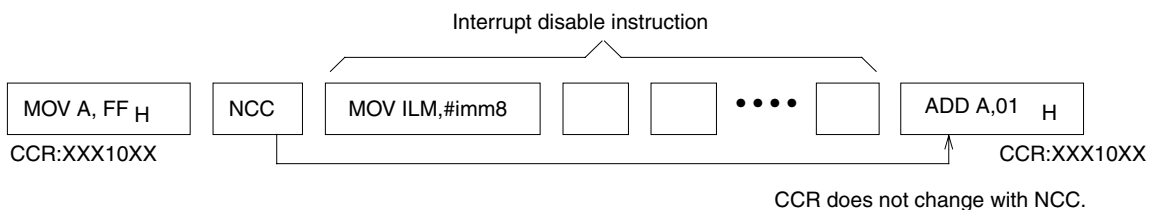
Figure 2.10-1 Interrupt Disable Instruction



■ Restrictions on Interrupt Disable Instructions and Prefix Instructions

When a prefix code is placed before an interrupt disable instruction, the prefix code affects the first instruction after the code other than the interrupt disable instruction. For details, see Figure 2.10-2 "Interrupt Disable Instructions and Prefix Codes".

Figure 2.10-2 Interrupt Disable Instructions and Prefix Codes

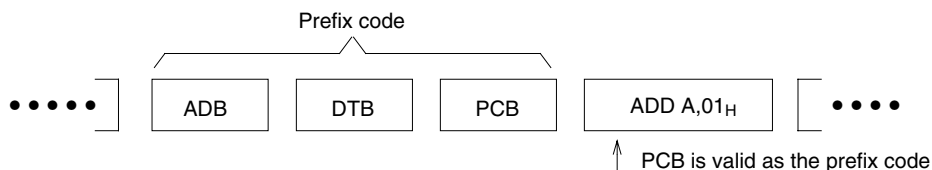


■ Consecutive Prefix Codes

When competitive prefix codes are placed consecutively, the latter becomes valid.

In the figure below, competitive prefix codes are PCB, ADB, DTB, and SPB. For details, see Figure 2.10-3 "Consecutive Prefix Codes".

Figure 2.10-3 Consecutive Prefix Codes



2.11 Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions

Set "00_H" in the bank register before using the "DIV A, Ri" and "DIVW A, RWi" Instructions.

■ Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions

Table 2.11-1 Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions (i = 0 to 7)

Instruction	Bank register affected by the execution of the instructions listed on the left	Address that stores the remainder
DIV A, R0	DTB	(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 8 _H : Lower 16 bits)
DIV A, R1		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 9 _H : Lower 16 bits)
DIV A, R4		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + C _H : Lower 16 bits)
DIV A, R5		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + D _H : Lower 16 bits)
DIVW A, RW0		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 0 _H : Lower 16 bits)
DIVW A, RW1		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 2 _H : Lower 16 bits)
DIVW A, RW4		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + 8 _H : Lower 16 bits)
DIVW A, RW5		(DTB: Upper 8 bits) + (0180 _H + RP x 10 _H + A _H : Lower 16 bits)
DIV A, R2	ADB	(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + A _H : Lower 16 bits)
DIV A, R6		(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + E _H : Lower 16 bits)
DIVW A, RW2		(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + 4 _H : Lower 16 bits)
DIVW A, RW6		(ADB: Upper 8 bits) + (0180 _H + RP x 10 _H + E _H : Lower 16 bits)
DIV A, R3	USB SSB *1	(USB *2: Upper 8 bits) + (0180 _H + RP x 10 _H + B _H : Lower 16 bits)
DIV A, R7		(USB *2: Upper 8 bits) + (0180 _H + RP x 10 _H + F _H : Lower 16 bits)
DIVW A, RW3		(USB *2: Upper 8 bits) + (0180 _H + RP x 10 _H + 6 _H : Lower 16 bits)
DIVW A, RW7		(USB *2: Upper 8 bits) + (0180 _H + RP x 10 _H + E _H : Lower 16 bits)

*1 Depends on the S bit of the CCR register.

*2 In the event that the S bit of the CCR register is zero

If the value of the bank registers (DTB, ADB, USB, and SSB) is "00_H", the remainder after division is stored in the register of the instruction operands. Otherwise, the upper eight bits is specified by the bank register corresponding to the register of the instruction operand, and the lower 16 bits is the same as the address of the register of the instruction operand. The remainder is stored in the bank register specified by the upper eight bits.

Example:

If "DIV A,R0" is executed with DTB = "053_H" and RP = "03_H", the address of R0 is "0180_H" + RP ("03_H") x "10_H" + "08_H" (R0 corresponding address) = "0001B8_H". Since the data bank register (DTB) is specified by "DIV A,R0" as the bank register, the remainder is stored in address "05301B8_H", which was obtained by adding the bank address "053_H".

Note:

For information about the bank register and Ri and RWi registers, see Section 2.7 "Registers".

■ Use of the "DIV A, Ri" and "DIVW A, RWi" Instructions without Precautions

To enable users to develop programs without having to take precautions for using the "DIV A,Ri" and "DIVW A,RWi" instructions, special compilers and assemblers are available. The special compiler does not generate the instructions in Table 2.11-1 "Precautions for Use of "DIV A,Ri" and "DIVW A,RWi" Instructions (i = 0 to 7)". The special assemblers have a function that replaces the instructions in Table 2.11-1 "Precautions for Use of "DIV A,Ri" and "DIVW A,RWi" Instructions (i = 0 to 7)" with equivalent instruction strings. For the MB90590 series, use the following types of compilers and assemblers:

○ Compiler

- cc907 V02L06 or later, or fcc907s V30L02 or later

○ Assembler

- asm907a V03L04 or later, or fasm907s V30L04 (Rev. 300004) or later

CHAPTER 3 INTERRUPTS

This chapter explains the interrupt functions and operations.

3.1 "Outline of Interrupts"

3.2 "Interrupt Vector"

3.3 "Interrupt Control Registers (ICR)"

3.4 "Interrupt Flow"

3.5 "Hardware Interrupts"

3.6 "Software Interrupts"

3.7 "Extended Intelligent I/O Service (EI²OS)"

3.8 "Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI²OS)"

3.9 "Exceptions"

3.1 Outline of Interrupts

The F²MC-16LX has interrupt functions that terminate the currently executing processing and transfer control to another specified program when a specified event occurs. There are four types of interrupt functions:

- **Hardware interrupt:** Interrupt processing due to an internal resource event
- **Software interrupt:** Interrupt processing due to a software event occurrence instruction
- **Extended intelligent I/O service (EI²OS):** Transfer processing due to an internal resource event
- **Exception:** Termination due to an operation exception

■ Hardware Interrupts

A hardware interrupt is activated by an interrupt request from an internal resource. A hardware interrupt request occurs when both the interrupt request flag and the interrupt enable flag in an internal resource are set. Therefore, an internal resource must have an interrupt request flag and interrupt enable flag to issue a hardware interrupt request.

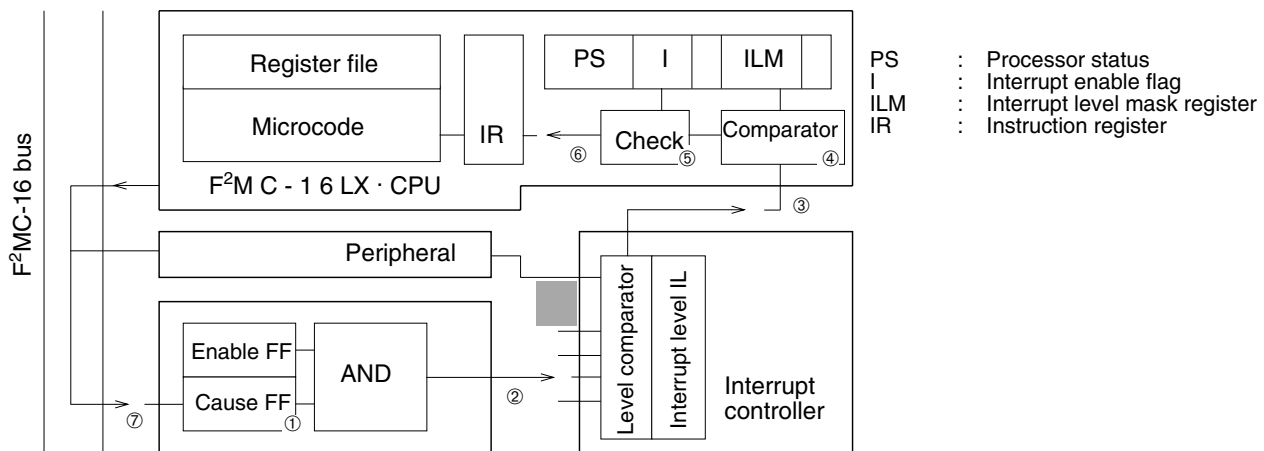
○ Specifying an interrupt level

An interrupt level can be specified for the hardware interrupt. To specify an interrupt level, use the level setting bits (IL0, IL1, and IL2) of the interrupt controller.

○ Masking a hardware interrupt request

A hardware interrupt request can be masked by using the I flag of the processor status register (PS) in the CPU and the ILM bits (IL0, IL1, and IL2). When an unmasked interrupt request occurs, the CPU saves 12 bytes of data that consists of registers PS, PC, PCB, DTB, ADB, DPR, and A in the memory area indicated by the SSB and SSP registers.

Figure 3.1-1 Overview of Hardware Interrupts

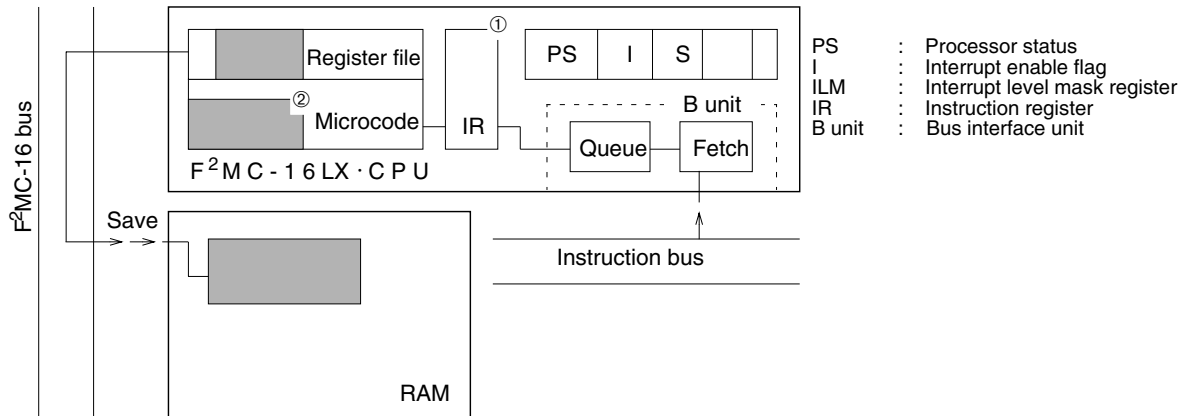


■ Software Interrupts

Interrupts requested by executing the INT instruction are software interrupts. An interrupt request by the INT instruction does not have an interrupt request or enable flag. An interrupt request is issued always by executing the INT instruction.

No interrupt level is assigned to the INT instruction. Therefore, ILM is not updated when the INT instruction is used. Instead, the I flag is cleared and the continuing interrupt requests are suspended.

Figure 3.1-2 Overview of Software Interrupts



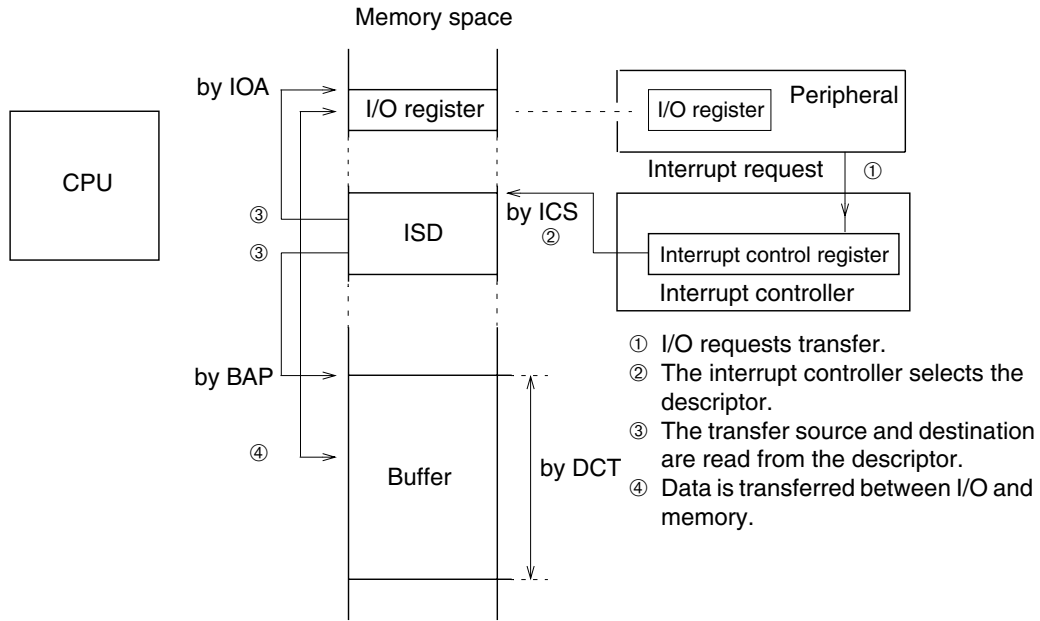
■ Extended Intelligent I/O Service (EI²OS)

The extended intelligent I/O service automatically transfers data between an internal resource and memory. This processing is traditionally performed by an interrupt processing program, but the EI²OS enables data to be transferred in a manner similar to a DMA (direct memory access) operation.

To activate the extended intelligent I/O service function from an internal resource, the interrupt control register (ICR) of the interrupt controller must have an extended intelligent I/O service enable flag (ISE).

The extended intelligent I/O service is started when an interrupt request occurs with 1 specified in the ISE flag. To generate a normal interrupt using a hardware interrupt request, set the ISE flag to 0.

Figure 3.1-3 Overview of the Extended Intelligent I/O Service (EI²OS)



■ Exceptions

Exception processing is basically the same as interrupt processing. When an exception is detected between instructions, exception processing is performed. In general, exception processing occurs as a result of an unexpected operation. Therefore, use exception processing only for debugging programs or for activating recovery software in an emergency.

3.2 Interrupt Vector

An interrupt vector uses the same area for both hardware and software interrupts. For example, interrupt request number INT42 is used for a delayed hardware interrupt and for software interrupt INT #42. Therefore, the delayed interrupt and INT #42 call the same interrupt processing routine. Interrupt vectors are allocated between addresses FFFC00_H and FFFFFFF_H as shown in Table 3.2-1 "Interrupt Vectors".

■ Interrupt Vector

Table 3.2-1 Interrupt Vectors

Interrupt request	Vector address L	Vector address H	Vector address bank	Mode register
INT 0 ^(*1)	FFFFFC _H	FFFFFD _H	FFFFFE _H	Unused
INT 1 ^(*1)	FFFFF8 _H	FFFFF9 _H	FFFFFA _H	Unused
·	·	·	·	·
·	·	·	·	·
·	·	·	·	·
INT 7 ^(*1)	FFFFE0 _H	FFFFE1 _H	FFFFE2 _H	Unused
INT 8 ^(*2)	FFFFDC _H	FFFFDD _H	FFFFDE _H	FFFFDF _H
INT 9	FFFFD8 _H	FFFFD9 _H	FFFFDA _H	Unused
INT 10 ^(*3)	FFFFD4 _H	FFFFD5 _H	FFFFD6 _H	Unused
INT 11	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Unused
·	·	·	·	·
·	·	·	·	·
·	·	·	·	·
INT 254	FFFC04 _H	FFFC05 _H	FFFC06 _H	Unused
INT 255	FFFC00 _H	FFFC01 _H	FFFC02 _H	Unused

*1: When PCB is FF_H, the vector area for the CALLV instruction is the same as that for INT #vct8 (#0 to #7).

Care must be taken when using the vector for the CALLV instruction.

*2: The vector is a reset vector.

*3: The vector is an exception processing vector.

■ Listing of Interrupt Vectors

See Table D-1 "MB90590 Interrupt Vectors" in APPENDIX D "List of MB90590 Interrupt Vectors" for a list of the MB90590 interrupt vectors.

3.3 Interrupt Control Registers (ICR)

The interrupt control registers are in the interrupt controller. Each interrupt control register has a corresponding I/O that has an interrupt function. The interrupt control registers have the following three functions:

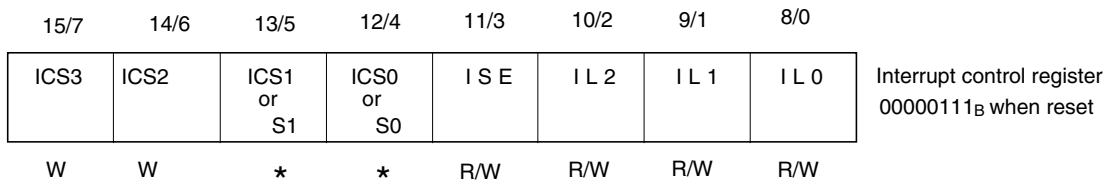
- Setting an interrupt level for corresponding peripherals
- Selecting whether to use an ordinary interrupt or extended intelligent I/O service for the corresponding peripherals
- Selecting the extended intelligent I/O service channel

Do not access an interrupt control register by using a read-modify-write instruction, as doing so causes a malfunction.

■ Interrupt Control Register (ICR)

Figure 3.3-1 "Interrupt Control Register (ICR)" is a diagram of the bit configuration of an interrupt control register.

Figure 3.3-1 Interrupt Control Register (ICR)



Note:

ICS3 to ICS0 are valid only when EI²OS is activated. Set '1' in ISE to activate EI²OS, and set '0' in ISE not to activate it. When EI²OS is not to be activated, any value can be set in ICS3 to ICS0. * '1' is read always.

ICS1 and ICS0 are valid for write only. S1 and S0 are valid for read only.

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

These bits are readable and writable, and specify the interrupt level of the corresponding internal resources. Upon a reset, these bits are initialized to level 7 (no interrupt). Table 3.3-1 "Interrupt Level Setting Bits and Interrupt Levels" describes the relationship between the interrupt level setting bits and interrupt levels.

Table 3.3-1 Interrupt Level Setting Bits and Interrupt Levels

ILM2	ILM1	ILM0	Level
0	0	0	0 (Strongest)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (Weakest)
1	1	1	7 (No interrupt)

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

The ISE bit is readable and writable. In response to an interrupt request, EI²OS is activated when '1' is set in the ISE bit and an interrupt sequence is activated when '0' is set in the ISE bit. Upon completion of EI²OS, the ISE bit is cleared to a zero. If the corresponding peripheral does not have the EI²OS function, the ISE bit must be set to '0' on the software side.

Upon a reset, the ISE bit is initialized to '0'.

[bits 15 to 12] [bits 7 to 4]: ICS 3 to 0 (extended intelligent I/O service channel select bits)

ICS3 to ICS0 are write-only bits. These bits specify the EI²OS channel. The values set in these bits determined the intelligent I/O service descriptor addresses in memory, which is explained later. The ICS bits are initialized by a reset.

Table 3.3-2 "ICS bits, Channel Numbers, and Descriptor Addresses" describes the correspondence between the ICS bits, channel numbers, and descriptor addresses.

Table 3.3-2 ICS bits, Channel Numbers, and Descriptor Addresses

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

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

S0 and S1 are read-only bits. The values set in these bits indicate the end condition of EI²OS. These bits are initialized to '00' upon a reset.

Table 3.3-3 "S Bits and End Conditions" shows the relationship between the S bits and the end conditions.

Table 3.3-3 S Bits and End Conditions

S1	S0	End condition
0	0	EI ² OS running or not activated
0	1	Termination by count
1	0	Reserved
1	1	Termination by request from resource

3.4 Interrupt Flow

Figure 3.4-1 "Interrupt Flow" shows the interrupt flow.

■ Interrupt Flow

Figure 3.4-1 Interrupt Flow

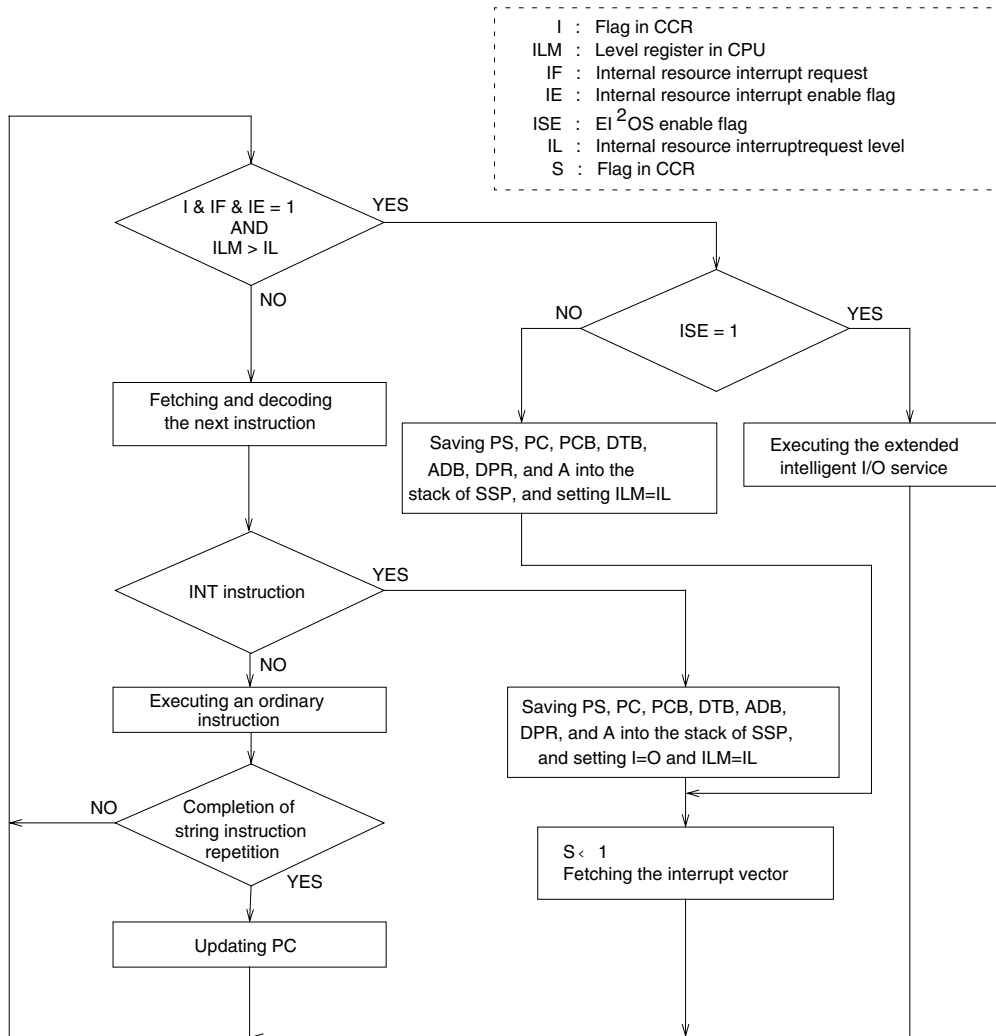
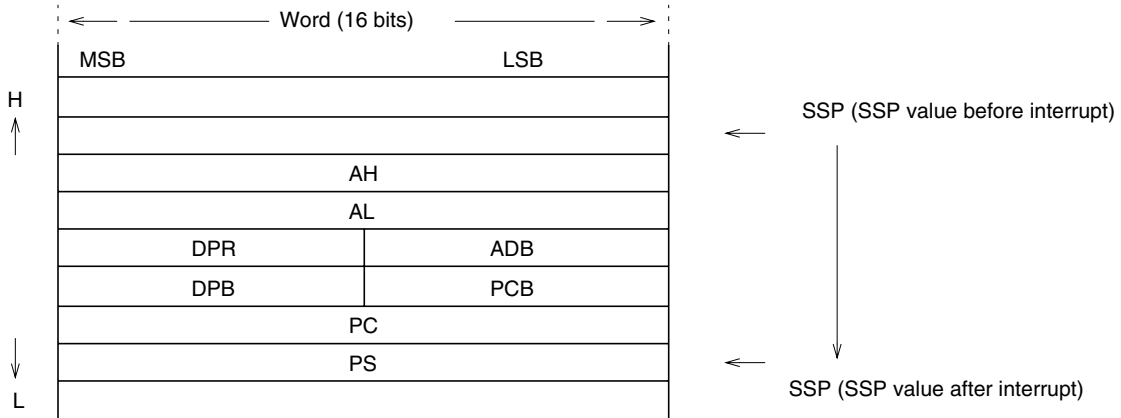


Figure 3.4-2 Register Saving during Interrupt Processing



3.5 Hardware Interrupts

In response to an interrupt request signal from an internal resource, the CPU pauses current program execution and transfers control to the interrupt processing program defined by the user. This function is called the hardware interrupt function.

■ Hardware Interrupts

A hardware interrupt occurs when the relevant conditions are satisfied as a result of two operations: comparison between the interrupt request level and the value in the interrupt level mask register (ILM) of PS in the CPU, and hardware reference to the I flag value of PS.

The CPU performs the following processing when a hardware interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- Sets ILM in the PS register. The currently requested interrupt level is automatically set.
- Fetches the corresponding interrupt vector value and branches to the processing indicated by that value.

■ Structure of Hardware Interrupt

Hardware interrupts are handled by the following three sections:

○ Internal resources

Interrupt enable and request bits: Used to control interrupt requests from resources.

○ Interrupt controller

ICR: Assigns interrupt levels and determines the priority levels of simultaneously requested interrupts.

○ CPU

I and ILM: Used to compare the requested and current interrupt levels and to identify the interrupt enable status.

Microcode: Interrupt processing step

The status of these sections are indicated by the resource control registers for internal resources, the ICR for the interrupt controller, and the CCR value for the CPU. To use a hardware interrupt, set the three sections beforehand by using software.

The interrupt vector table referenced during interrupt processing is assigned to addresses FFFC00_H to FFFFFF_H in memory. These addresses are shared with software interrupts.

Table D-2 "Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers" in Appendix D lists the assignments for the MB90590 Series.

3.5.1 Hardware Interrupt Operation

An internal resource that has the hardware interrupt request function has an interrupt request flag and interrupt enable flag. The interrupt request flag indicates whether an interrupt request exists, and the interrupt enable flag indicates whether the relevant internal resource requests an interrupt to the CPU. The interrupt request flag is set when an event occurs that is unique to the internal resource. When the interrupt enable flag indicates "enable", the resource issues an interrupt request to the interrupt controller.

■ Hardware Interrupt Operation

When two or more interrupt requests are received at the same time, the interrupt controller compares the interrupt levels (IL) in ICR, selects the request at the highest level (the smallest IL value), then reports that request to the CPU. If multiple requests are at the same level, the interrupt controller selects the request with the lowest interrupt number. The relationship between the interrupt requests and ICRs is determined by the hardware.

The CPU compares the received interrupt level and the ILM in the PS register. If the interrupt level is smaller than the ILM value and the I bit of the PS register is set to 1, the CPU activates the interrupt processing microcode after the currently executing instruction is completed. The CPU references the ISE bit of the ICR of the interrupt controller at the beginning of the interrupt processing microcode, checks that the ISE bit is 0 (interrupt), and activates the interrupt processing body.

The interrupt processing body saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP, fetches three bytes of interrupt vector and loads them onto PC and PCB, updates the ILM of PS to a level value of the received interrupt, sets the S flag, then performs branch processing. As a result, the interrupt processing program defined by the user is executed next.

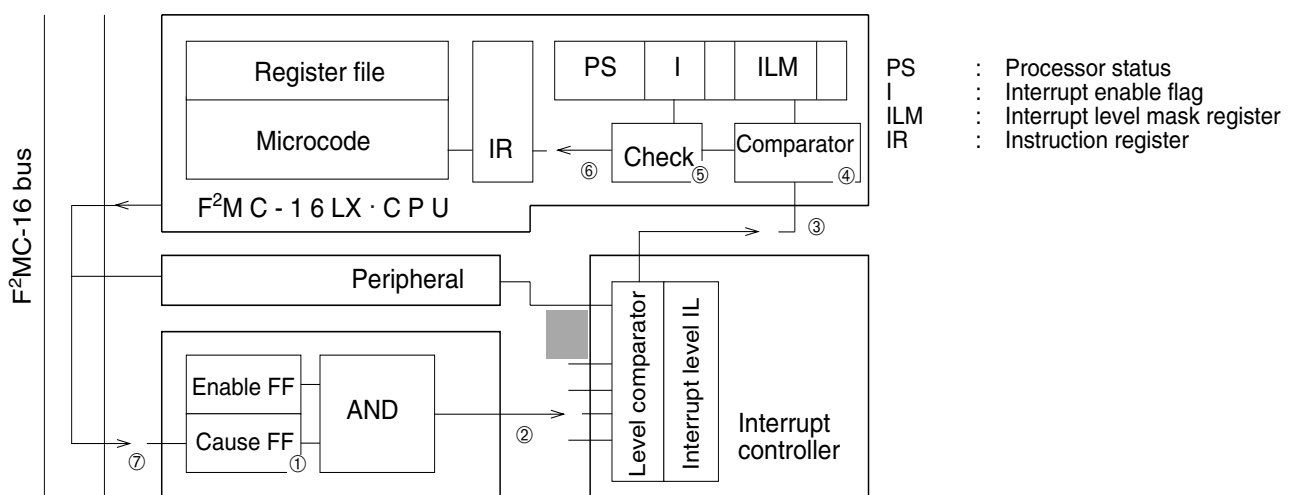
Figure 3.5-1 "Occurrence and Release of Hardware Interrupt" illustrates the flow from the occurrence of a hardware interrupt until there is no interrupt request in the interrupt processing program.

3.5.2 Occurrence and Release of Hardware Interrupt

Figure 3.5-1 "Occurrence and Release of Hardware Interrupt" shows the processing flow from occurrence of a hardware interrupt to release of the interrupt request in an interrupt processing program.

■ Occurrence and Release of Hardware Interrupt

Figure 3.5-1 Occurrence and Release of Hardware Interrupt



1. An interrupt cause occurs in a peripheral.
2. The interrupt enable bit in the peripheral is referenced. If interrupts are enabled, the peripheral issues an interrupt request to the interrupt controller.
3. Upon reception of the interrupt request, the interrupt controller determines the priority levels of simultaneously requested interrupts. Then, the interrupt controller transfers the interrupt level of the corresponding interrupt to the CPU.
4. The CPU compares the interrupt level requested by the interrupt controller with the ILM bit of the processor status register.
5. If the comparison shows that the requested level is higher than the current interrupt processing level, the I flag value of the same processor status register is checked.
6. If the check in step 5. shows that the I flag indicates interrupt enable status, the requested level is written to the ILM bit. Interrupt processing is performed as soon as the currently executing instruction is completed, then control is transferred to the interrupt processing routine.
7. When the interrupt cause of step 1. is cleared by software in the user interrupt processing routine, the interrupt request is completed.

CHAPTER 3 INTERRUPTS

The time required for the CPU to execute the interrupt processing in steps 6. and 7. is shown below.

Interrupt start: $24 + 6 \times$ (Table 3.3-2 "ICS bits, Channel Numbers, and Descriptor Addresses" machine cycles)

Interrupt return: $15 + 6 \times$ (Table 3.3-2 "ICS bits, Channel Numbers, and Descriptor Addresses" machine cycles) RETI instruction

Table 3.5-1 Compensation Values for Interrupt Processing Cycle Count

Address indicated by the stack pointer	Cycle count compensation value
Internal area, even-numbered address	0
Internal area, odd-numbered address	+2

3.5.3 Multiple interrupts

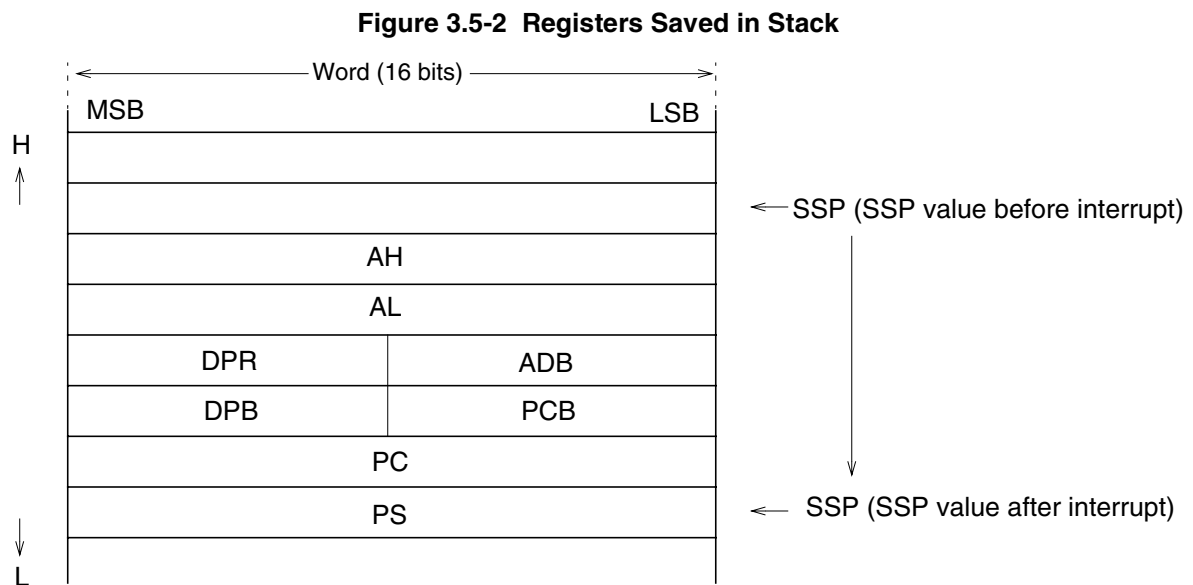
As a special case, no hardware interrupt request can be accepted while data is being written to the I/O area. This is intended to prevent the CPU from operating falsely because of an interrupt request issued while an interrupt control register for a resource is being updated.

If an interrupt occurs during interrupt processing, a higher-level interrupt is processed first.

■ Multiple Interrupts

The F²MC-16LX CPU supports multiple interrupts. If an interrupt of a higher level occurs while another interrupt is being processed, control is transferred to the high-level interrupt after the currently executing instruction is completed. After processing of the high-level interrupt is completed, the original interrupt processing is resumed. An interrupt of the same or lower level may be generated while another interrupt is being processed. If this happens, the new interrupt request is suspended until the current interrupt processing is completed, unless the ILM value or I flag is changed by an instruction. The extended intelligent I/O service cannot be activated from multiple sources; while an extended intelligent I/O service is being processed, all other interrupt requests or extended intelligent I/O service requests are suspended.

Figure 3.5-2 "Registers Saved in Stack" shows the order of the registers saved in the stack.



3.6 Software Interrupts

In response to execution of a special instruction, control is transferred from the program currently executed by the CPU to the interrupt processing program defined by the user. This is called the software interrupt function. A software interrupt occurs always when the software interrupt instruction is executed.

■ Software Interrupts

The CPU performs the following processing when a software interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- Sets I in the PS register. Interrupts are automatically disabled.
- Fetches the corresponding interrupt vector value, then branches to the processing indicated by that value.

A software interrupt request issued by the INT instruction has no interrupt request or enable flag. A software interrupt request is always issued by executing the INT instruction.

The INT instruction does not have an interrupt level. Therefore, the INT instruction does not update ILM. The INT instruction clears the I flag to suspend subsequent interrupt requests.

■ Structure of Software Interrupts

Software interrupts are handled within the CPU:

CPU.....Microcode: Interrupt processing step

■ List of MB90590 Interrupt Vectors

Table D-1 "MB90590 Interrupt Vectors" lists the interrupt vectors of the MB90590 series.

As shown in Table D-1 "MB90590 Interrupt Vectors", software interrupts share the same interrupt vector area with hardware interrupts.

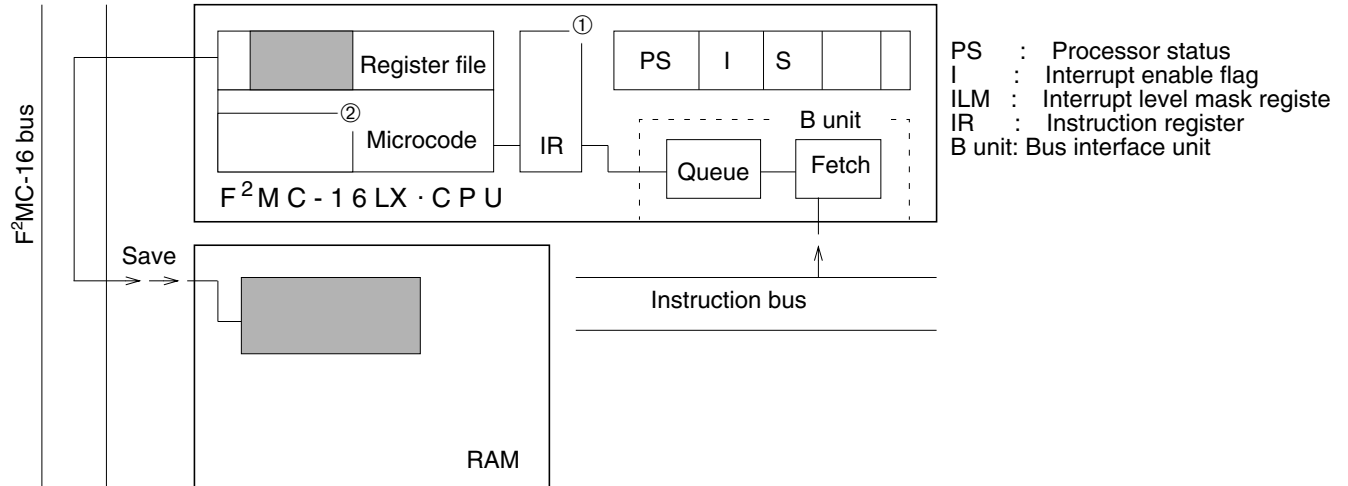
For example, interrupt request number INT 12 is used for external interrupt #0 to #7 of a hardware interrupt as well as for INT #12 of a software interrupt. Therefore, external interrupt #0 and INT #12 call the same interrupt processing routine.

■ Software Interrupt Operation

When the CPU fetches and executes the software interrupt instruction, the software interrupt processing microcode is activated. The software interrupt processing microcode saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP. The microcode then fetches three bytes of interrupt vector and loads them onto PC and PCB, resets the I flag, and sets the S flag. Then, the microcode performs branch processing. As a result, the interrupt processing program defined by the user application program is executed next.

Figure 3.6-1 "Occurrence and Release of Software Interrupt" illustrates the flow from the occurrence of a software interrupt until there is no interrupt request in the interrupt processing program.

Figure 3.6-1 Occurrence and Release of Software Interrupt



1. The software interrupt instruction is executed.
2. Special CPU registers in the register file are saved according to the microcode corresponding to the software interrupt instruction.
3. The interrupt processing is completed with the RETI instruction in the user interrupt processing routine.

■ Others

When the program bank register (PCB) is FFH, the CALLV instruction vector area overlaps the table of the INT #vct8 instruction. When designing software, ensure that the CALLV instruction does not use the same address as that of the #vct8 instruction.

Table D-2 "Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers" shows the relationship of interrupt cause, interrupt vector, and interrupt control register in the MB90590 series.

3.7 Extended Intelligent I/O Service (EI²OS)

The EI²OS function automatically transfers data between input and output and memory. An interrupt processing program was conventionally used for such processing, but EI²OS enables data transfer to be performed like DMA (direct memory access).

■ Extended Intelligent I/O Service (EI²OS)

EI²OS has the following advantages over the conventional method:

- The program size can be small because it is not necessary to write a transfer program.
- No internal register is used for transfer, eliminating the need for register saving and increasing the transfer speed.
- Transfer can be terminated from I/O, preventing unnecessary data from being transferred.
- The buffer address may either be incremented or left unupdated.
- The I/O register address may either be incremented or left unupdated.

At the end of EI²OS, processing automatically branches to an interrupt processing routine after the end condition is set. Thus, the user can identify the end condition.

To implement EI²OS, the hardware is distributed in two blocks. Each block has the following registers and descriptors.

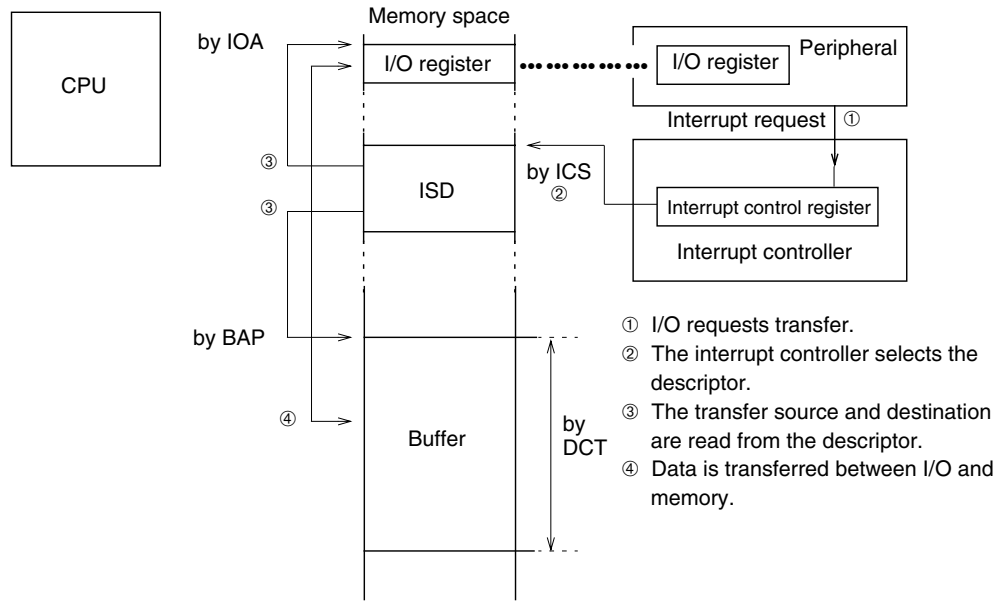
- Interrupt control register: Exists in the interrupt controller and indicates the ISD address.
- Extended intelligent I/O service descriptor (ISD): Exists in RAM and holds the transfer mode, I/O address, number of transfers, and buffer address.

Note:

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

Figure 3.7-1 "Outline of Extended Intelligent I/O Service" outlines the extended intelligent I/O service.

Figure 3.7-1 Outline of Extended Intelligent I/O Service

**Note:**

The area that can be specified by IOA is between 000000_H and 00FFFF_H.

The area that can be specified by BAP is between 000000_H and FFFFFFF_H.

The maximum transfer count that can be specified by DCT is 65,536.

■ Structure

EI²OS is handled by the following four sections:

Internal resources

Interrupt enable and request bits: Used to control interrupt requests from resources.

Interrupt controller

ICR: Assigns interrupt levels, determines the priority levels of simultaneously requested interrupts, and selects the EI²OS operation.

CPU

I and ILM: Used to compare the requested and current interrupt levels and to identify the interrupt enable status

Microcode: EI²OS processing step

RAM

Descriptor: Describes the EI²OS transfer information.

3.7.1 Extended Intelligent I/O Service Descriptor (ISD)

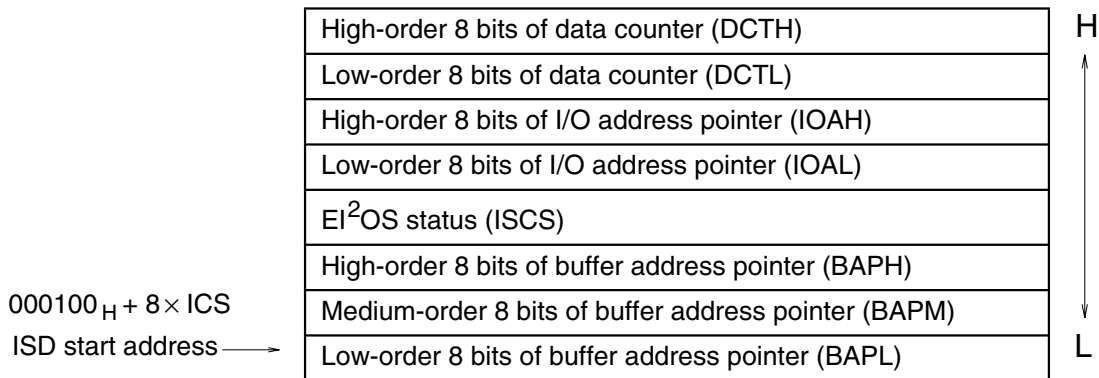
The extended intelligent I/O service descriptor exists between 000100_H and 00017F_H in internal RAM, and consists of the following items:

- Data transfer control data
- Status data
- Buffer address pointer

■ Extended Intelligent I/O Service Descriptor (ISD)

Figure 3.7-2 "Extended Intelligent I/O Service Descriptor Configuration" shows the configuration of the extended intelligent I/O service descriptor.

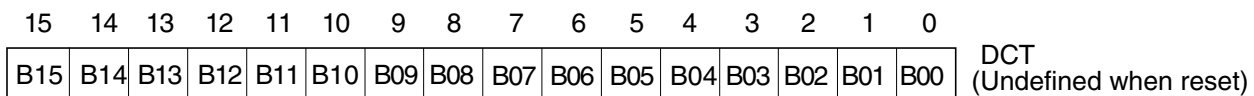
Figure 3.7-2 Extended Intelligent I/O Service Descriptor Configuration



■ Data Counter (DCT)

This is a 16-bit register that works as a counter corresponding to the number of data items transferred. This counter is decremented by one before data transfer. EI²OS is terminated when this counter reaches 0. Figure 3.7-3 "Data Counter Configuration" is a diagram of the data counter configuration.

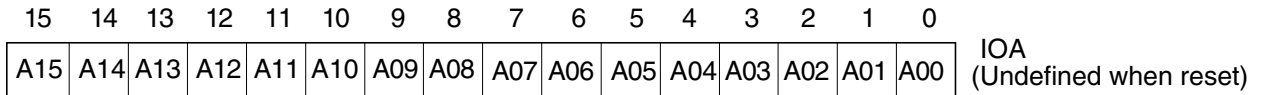
Figure 3.7-3 Data Counter Configuration



■ I/O Register Address Pointer (IOA)

This is a 16-bit register that indicates the low-order address (A15 to A0) of the buffer and I/O register used for data transfer. The high-order address (A23 to A16) are all zeroes, and any I/O between addresses 000000_H and 00FFFF_H can be specified. Figure 3.7-4 "I/O Register Address Pointer Configuration" is a diagram of the IOA configuration.

Figure 3.7-4 I/O Register Address Pointer Configuration



■ Buffer Address Pointer (BAP)

This 24-bit register holds the address used for the next EI²OS transfer. BAP exists for each EI²OS channel. Therefore, each EI²OS channel can be used for transfer with anywhere in the 16-Mbyte space. If the BF bit of ISCS is set to '0' (update enabled), only the low-order 16 bits of BAP changes and BAPH does not change.

3.7.2 EI²OS Status Register (ISCS)

This eight-bit register indicates the update direction (increment/decrement), transfer data format (byte/word), and transfer direction of the buffer address pointer and the I/O register address pointer. This register also indicates whether the buffer address pointer or I/O register address pointer is updated or fixed.

■ EI²OS Status Register (ISCS)

Figure 3.7-5 "ISCS Configuration" is a diagram of the ISCS configuration.

Figure 3.7-5 ISCS Configuration

7	6	5	4	3	2	1	0	ISCS (Undefined when reset)
Reserved	Reserved	Reserved	IF	BW	BF	DIR	SE	

* Always write 0 to bits 7 to 5 of ISCS.

Each bit is described below.

[bit 4] IF : Specify whether the I/O register address pointer is updated or fixed.

0 : The I/O register address pointer is updated after data transfer.

1 : The I/O register address pointer is not updated after data transfer.

Note:

Only increment is allowed.

[bit 3] BW : Specify the transfer data length.

0 : Byte

1 : Word

[bit 2] BF : Specify whether the buffer address pointer is updated or fixed.

0 : The buffer address pointer is updated after data transfer.

1 : The buffer address pointer is not updated after data transfer.

Note:

Only the low-order 16 bits of the buffer address are updated. Only increment is allowed.

[bit 1] DIR : Specify the data transfer direction.

0 : I/O --> Buffer

1 : Buffer --> I/O

[bit 0] SE : Control the termination of the extended intelligent I/O service based on resource requests.

0 : The extended intelligent I/O service is not terminated by a resource request.

1 : The extended intelligent I/O service is terminated by a resource request.

3.8 Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI²OS)

Figure 3.8-1 "EI²OS Operation Flow" is a diagram of the EI²OS operation flow. Figure 3.8-2 "EI²OS Use Flow" is a diagram of the EI²OS use procedure.

■ EI²OS Operation Flow

Figure 3.8-1 EI²OS Operation Flow

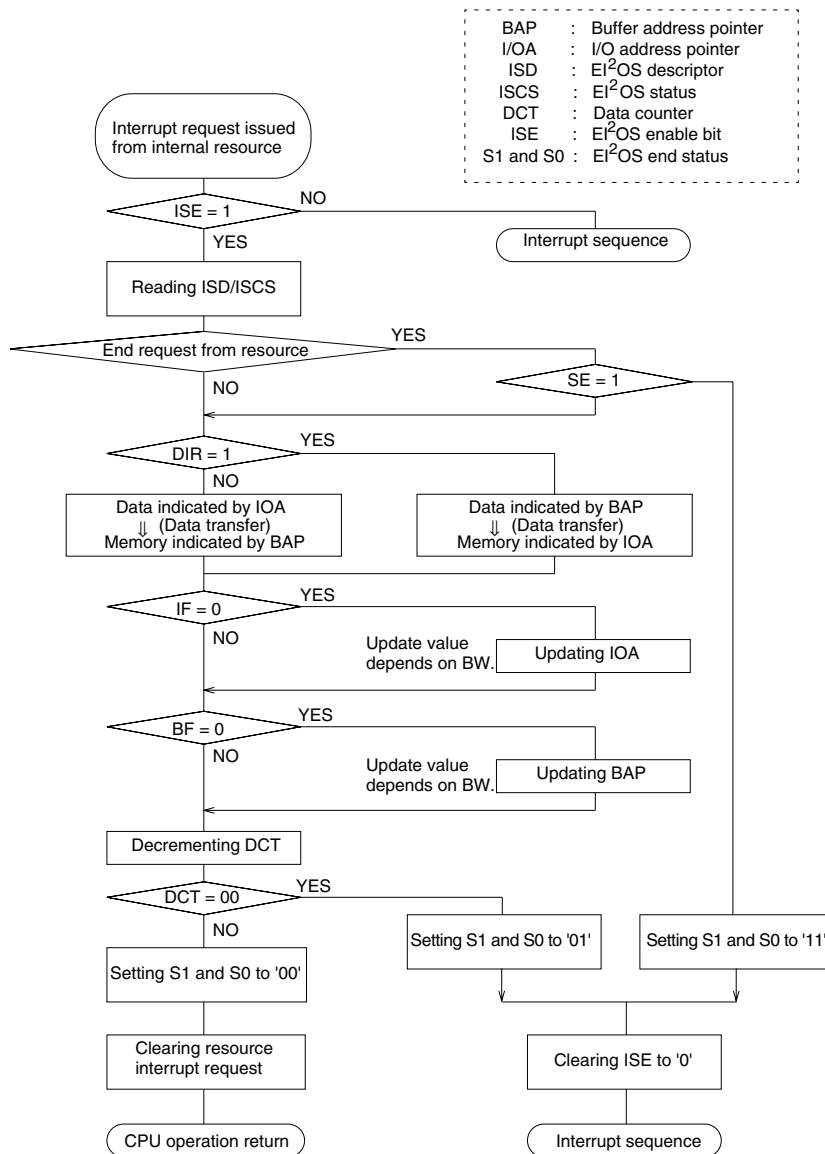
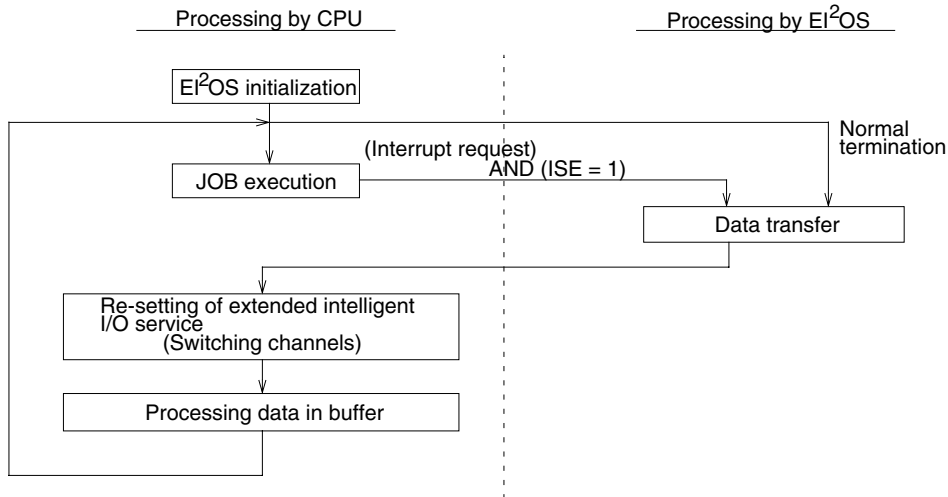


Figure 3.8-2 EI²OS Use Flow



The extended EI²OS execution time for each flow is described below.

○ **When data transfer continues (when the stop condition is not satisfied)**

(Table 3.8-1 "Execution Time when the Extended EI²OS Continues" + Table 3.8-2 "Data Transfer Compensation Values for Extended EI²OS Execution Time") machine cycles

○ **When a stop request is issued from a resource**

(36 + 6 × Table 3.D-2) machine cycles

○ **When the counting is completed**

(Table 3.8-1 "Execution Time when the Extended EI²OS Continues" + Table 3.8-2 "Data Transfer Compensation Values for Extended EI²OS Execution Time" + (21 + 6 × Table 3.D-2)) machine cycles

Table 3.8-1 Execution Time when the Extended EI²OS Continues

ISCS SE bit		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

Table 3.8-2 Data Transfer Compensation Values for Extended EI²OS Execution Time

I/O address pointer			Internal access	
			B/E	O
Buffer address pointer	Internal access	B/E	0	+2
		O	+2	+4

B: Byte data transfer

E: Even address word transfer

O: Odd address word transfer

3.9 Exceptions

The F²MC-16LX performs exception processing when the following event occurs:

■ Execution of an Undefined Instruction

Exception processing is fundamentally the same as interrupt processing. When an exception is detected between instructions, exception processing is performed separately from ordinary processing. In general, exception processing is performed as a result of an unexpected operation. Fujitsu recommends using exception processing only for debugging or for activating emergency recovery software.

■ Exception due to Execution of an Undefined Instruction

The F²MC-16LX handles all codes that are not defined in the instruction map as undefined instructions. When an undefined instruction is executed, processing equivalent to the INT 10 software interrupt instruction is performed. Specifically, the AL, AH, DPR, DTB, ADB, PCB, PC, and PS values are saved into the system stack, and processing branches to the routine indicated by the interrupt number 10 vector. In addition, the I flag is cleared and the S flag is set. The PC value saved in the stack is the address at which the undefined instruction is stored. Processing can be restored by the RETI instruction, but is of no use, however, because the same exception occurs again.

CHAPTER 4 DELAYED INTERRUPT

This chapter explains the functions and operations of the delayed interrupt.

- 4.1 "Outline of Delayed Interrupt Module"
- 4.2 "Delayed Interrupt Register"
- 4.3 "Delayed Interrupt Operation"

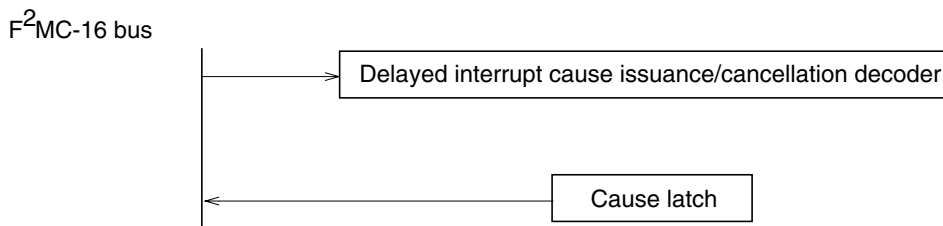
4.1 Outline of Delayed Interrupt Module

The delayed interrupt source module is used to generate interrupts for switching tasks. Using this module, interrupt requests to the F²MC-16LX CPU can be issued and canceled by software.

■ Block Diagram of Delayed Interrupt

Figure 4.1-1 "Block Diagram" is a block diagram of the delayed interrupt source module.

Figure 4.1-1 Block Diagram



■ Notes on Operation

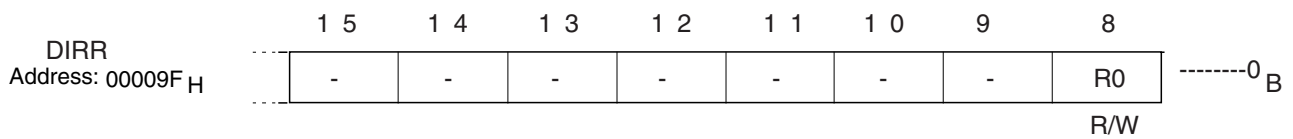
This lock is set by writing '1' to the corresponding bit of DIRR, and is cleared by writing '0' to the same bit. Therefore, interrupt processing is reactivated immediately after control returns from interrupt processing, unless the software is designed so that the cause of the interrupt is cleared within the interrupt processing routine.

4.2 Delayed Interrupt Register

DIRR controls issuance and cancellation of delayed interrupt requests. Writing "1" to this register issues a delayed interrupt request, and writing "0" cancels the delayed interrupt request. Upon a reset, the request is canceled.

■ Delayed Interrupt Cause Issuance/Cancellation Register (DIRR: Delayed Interrupt Request Register)

In DIRR, either "0" or "1" can be written to the reserved bit area. However, it is recommended that a set bit or clear bit instruction be used to access this register for future expansions.



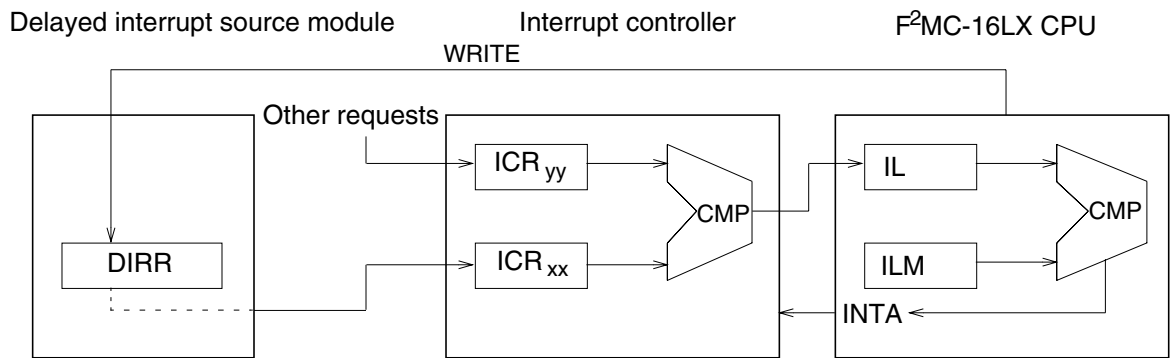
4.3 Delayed Interrupt Operation

When the CPU writes "1" to the relevant bit of DIRR by software, the request latch in the delayed interrupt source module is set and an interrupt request is issued to the interrupt controller.

■ **Delayed Interrupt Occurrence**

When the CPU writes '1' to the relevant bit of DIRR by software, the request latch in the delayed interrupt source module is set and an interrupt request is issued to the interrupt controller. If this interrupt has the highest priority or if there is no other interrupt request, the interrupt controller issues an interrupt request to the F²MC-16LX CPU. The F²MC-16LX CPU compares the ILM bit of its internal CCR register and the interrupt request, and starts the hardware interrupt processing microprogram as soon as the current instruction is completed if the interrupt level of the request is higher than that of the ILM bit. The interrupt processing routine for this interrupt is thus executed.

Figure 4.3-1 Delayed Interrupt Issuance



CHAPTER 5 CLOCK AND RESET

This chapter explains the functions and operations of clocks and resets.

5.1 "Clock Generator"

5.2 "Reset Cause Occurrence"

5.3 "Reset Causes"

5.1 Clock Generator

The clock generator controls internal clock operation, including such functions as sleep, timer, stop, and PLL multiplication. This internal clock is called the machine clock, and one cycle of the machine clock is called a machine cycle. A clock based on the source oscillation is called the main clock, and a clock based on the internal VCO oscillation is called the PLL clock.

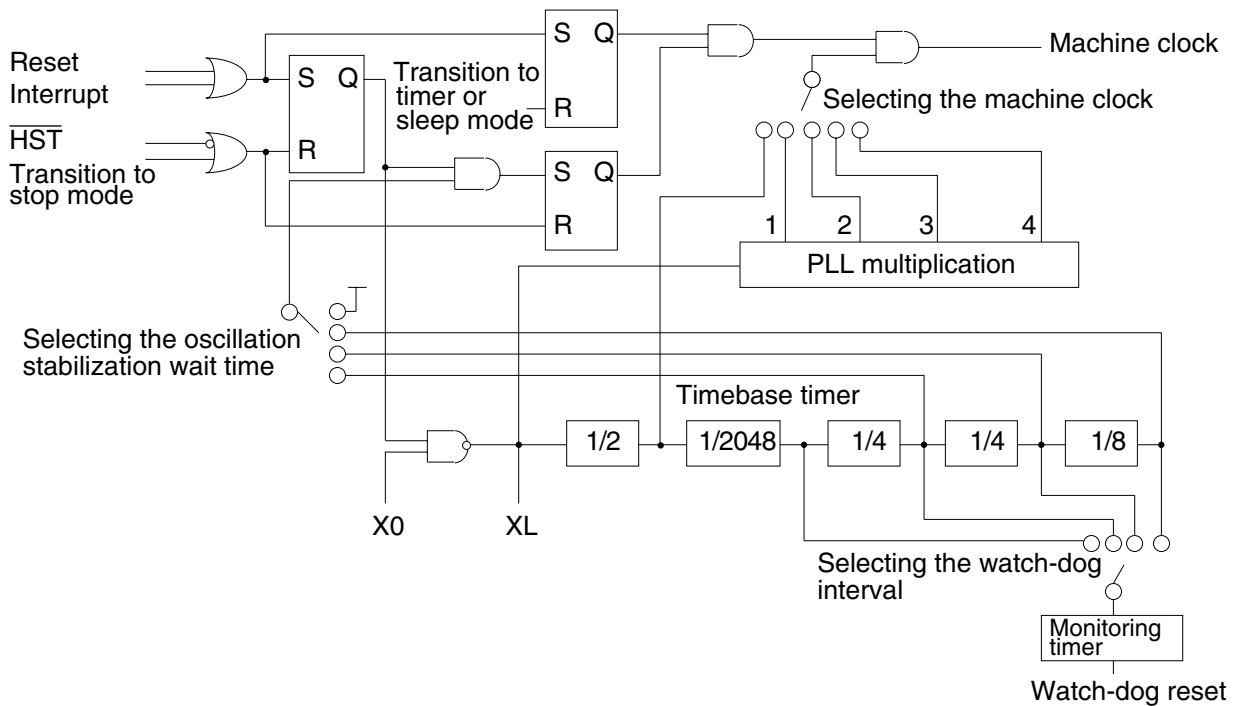
■ Notes on Clock Generator

When the operating voltage is 5V, the OSC source oscillation can be between 3MHz and 5MHz. When an external clock source is used, its frequency can be between 3MHz and 16MHz. The highest operating frequency for the CPU and peripheral resource circuits is 16 MHz, however. Normal operation is not guaranteed if a multiplication factor resulting in a higher frequency than 16 MHz is specified. For example, if the external clock frequency is 16 MHz, only 1 can be specified as the multiplication factor.

The lowest operating frequency of the VCO oscillation is 4 MHz, and an oscillation below 4 MHz must not be specified.

Figure 5.1-1 "Clock Generator Circuit Block Diagram" is a block diagram of the clock generator circuit.

Figure 5.1-1 Clock Generator Circuit Block Diagram



5.2 Reset Cause Occurrence

When a reset cause occurs, F²MC-16LX terminates the currently executing processing and waits for reset release. A reset is caused by the following factors:

Reset Cause Occurrence

A reset is caused by the following factors:

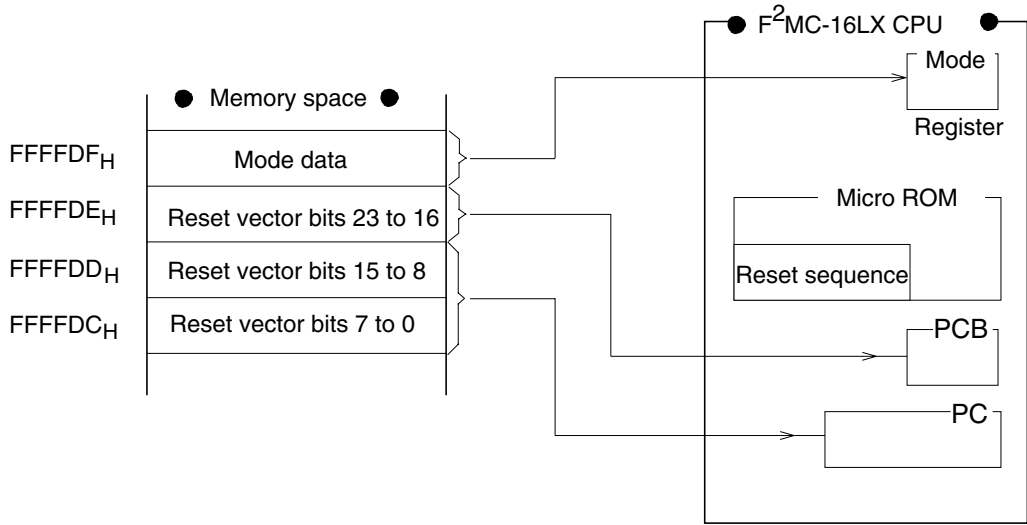
- Power-on reset
- Hardware standby release
- Watch-dog timer overflow
- External reset request via \overline{RST} pin
- Reset request by software

Operation after Reset Release

When a reset cause is removed, the F²MC-16LX immediately outputs the address in which the reset vector is stored, then fetches the reset vector and mode data. The reset vector and mode data are assigned to the four bytes between FFFFDC_H and FFFFDF_H. After reset is released, the reset vector and mode data are transferred to the registers by the hardware as described in Figure 5.2-1 "Source and Destination of Reset Vector and Mode Data".

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

Figure 5.2-1 Source and Destination of Reset Vector and Mode Data



Note:

For MB90F594A, MB90F594G, MB90F591A and MB90F591G, the reset vector and mode data have predetermined values by the hard-wired logic.

For more information, refer to Section 24.9 "Reset Vector Address in Flash Memory".

■ Registers not Initialized by Reset Input

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

Table 5.2-1 "Registers not Initialized by Reset Input" lists registers not initialized by each reset cause.

Table 5.2-1 Registers not Initialized by Reset Input

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

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

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

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

CG1,CG0: Set the intermittent CPU operation.

Y: Initialized

N: Not initialized (previous status maintained)

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

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

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

Table 5.2-2 "Registers not Initialized by Reset Input" lists registers that are not initialized by reset input using \overline{HST} plus \overline{RST} . Note that the operation status after the reset is released differs depending on the reset input type, \overline{HST} plus \overline{RST} reset input, or only \overline{RST} input, as listed in Table 5.2-2 "Registers not Initialized by Reset Input".

Table 5.2-2 Registers not Initialized by Reset Input

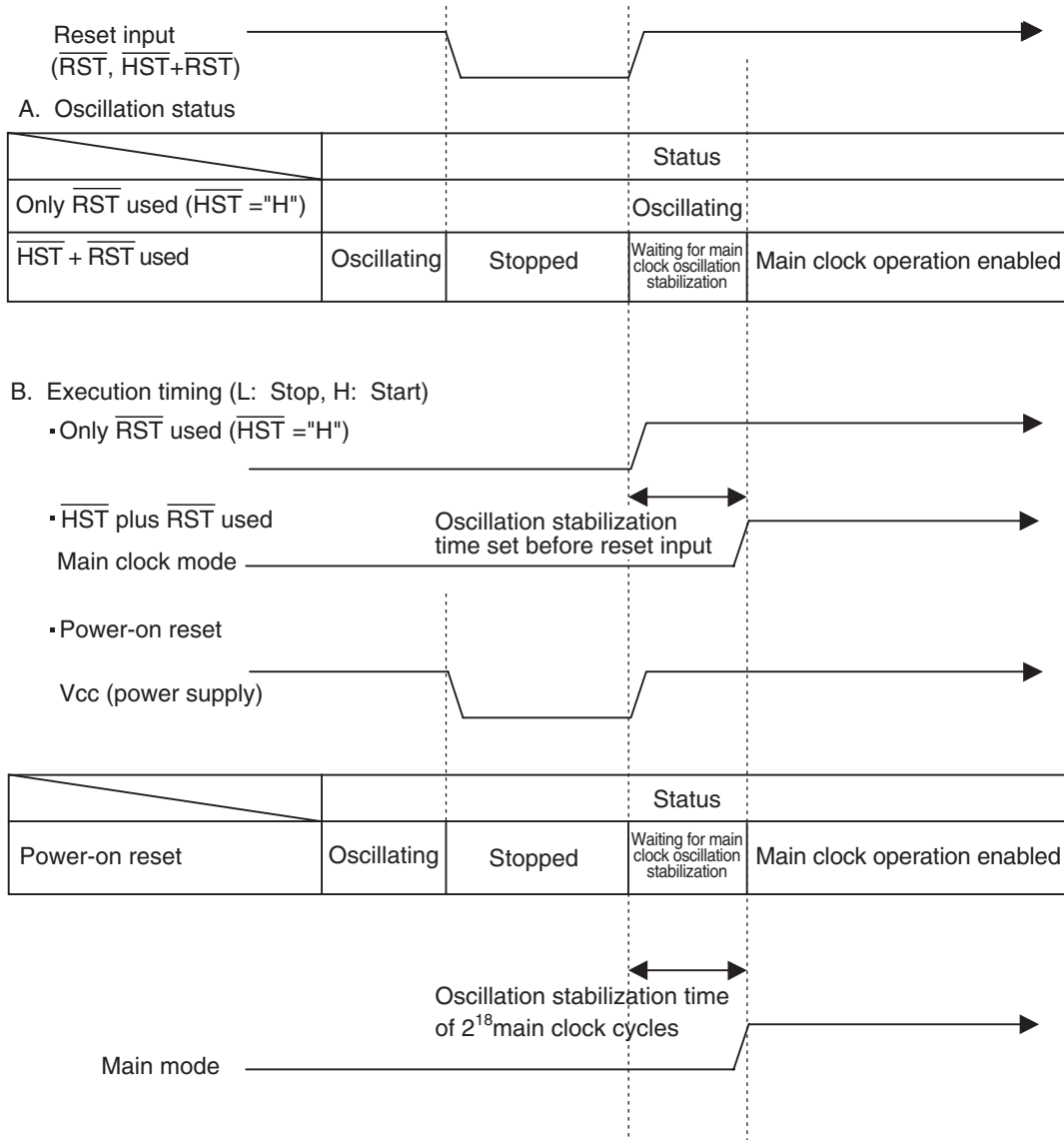
Type of reset	CKSCR					LPMCR	
	WS1	WS0	MCS	CS1	CS0	CG1	CG0
$\overline{HST} + \overline{RST}$	N	N	N	N	N	Y	Y

Y: Initialized

N: Not initialized (previous status maintained)

Figure 5.2-2 Operation Transition by Reset Input

[Operation Transition by Reset Input]



5.3 Reset Causes

Table 5.3-1 "Reset Causes" lists the five reset causes. The machine clock and watch-dog function are initialized differently for each reset cause. The reset cause register indicates the reset cause.

■ Reset Causes

Table 5.3-1 Reset Causes

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

* In stop or hardware standby mode, a reset input allows for oscillation stabilization time regardless of the reset cause.

* The oscillation stabilization time for a power-on reset is fixed to 2^{18} cycles of source oscillation. For other types of reset, the oscillation stabilization wait time is determined by CS1 and CS0 of the clock selection register.

As shown in Figure 5.3-1 "Reset Cause bit Block Diagram" each reset cause has a corresponding flip-flop. The contents of the flip-flop can be obtained by reading the watch-dog timer control register. If identifying the reset cause is required after the reset is released, ensure that the value read from the watch-dog timer control register is processed by software and processing branches to an appropriate program. Figure 5.3-2 "WDTC (Watch-Dog Timer Control) Register" is a diagram of the watch-dog timer control register.

Figure 5.3-1 Reset Cause bit Block Diagram

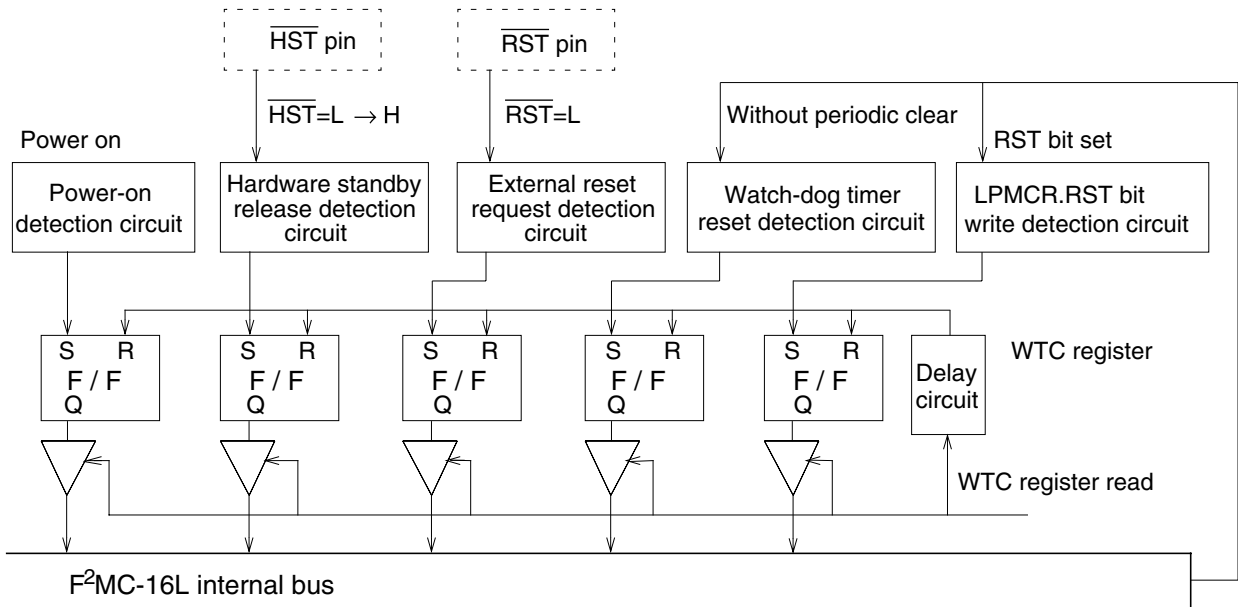


Figure 5.3-2 WDTC (Watch-Dog Timer Control) Register

	7	6	5	4	3	2	1	0	Bit No.
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)	

When there are multiple reset causes, the corresponding reset cause bits in the watch-dog timer control register are set. Therefore, if an external reset request and a watch-dog reset occur at the same time, both the ERST and WRST bits are set to 1.

A power-on reset is an exception; while the PONR bit is 1, the values of other bits do not indicate the correct reset causes. Therefore, design software so that the other reset cause bit values are ignored while the PONR bit is set to 1.

Table 5.3-2 Reset Cause Bits

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

(An asterisk (*) in the table means that the previous value is maintained.)

CHAPTER 6 LOW-POWER CONTROL CIRCUIT

This chapter explains the functions and operations of the low-power control circuits.

6.1 "Outline of Low-Power Control Circuit"

6.2 "Registers"

6.3 "Low-Power Mode Operation"

6.4 "Intermittent CPU Operation"

6.5 "Switching Machine Clocks"

6.6 "Status Transition of Clock Selection"

6.1 Outline of Low-Power Control Circuit

The MB90590 Series supports various operation modes to help reduce the power dissipation.

The operation modes include PLL clock mode, PLL sleep mode, watch mode, main clock mode, main sleep mode, stop mode, and hardware standby mode. Modes other than PLL clock mode are classified as low-power modes.

■ Outline of Lower-power Control Circuit

In main clock mode or main sleep mode, the main clock (OSC oscillation clock) is used. The operation clock is generated by dividing the main clock by two, and the PLL clock (VCO oscillation clock) is stopped.

In PLL sleep mode or main sleep mode, only the CPU operation clock is stopped. All other clocks are in operation.

In watch mode, only the timebase timer is in operation. In stop mode or hardware standby mode, oscillation is stopped. The data can be maintained at the lowest power consumption possible.

The intermittent CPU operation function is used to intermittently enable the clock supplied to the CPU when a register, internal memory, or internal resource is accessed. CPU execution is slowed while high-speed clock is supplied to the internal resources, enabling processing at low-power consumption.

The PLL clock multiplication factor can be selected from 1, 2, 3, and 4 by setting the CS1 and CS0 bits.

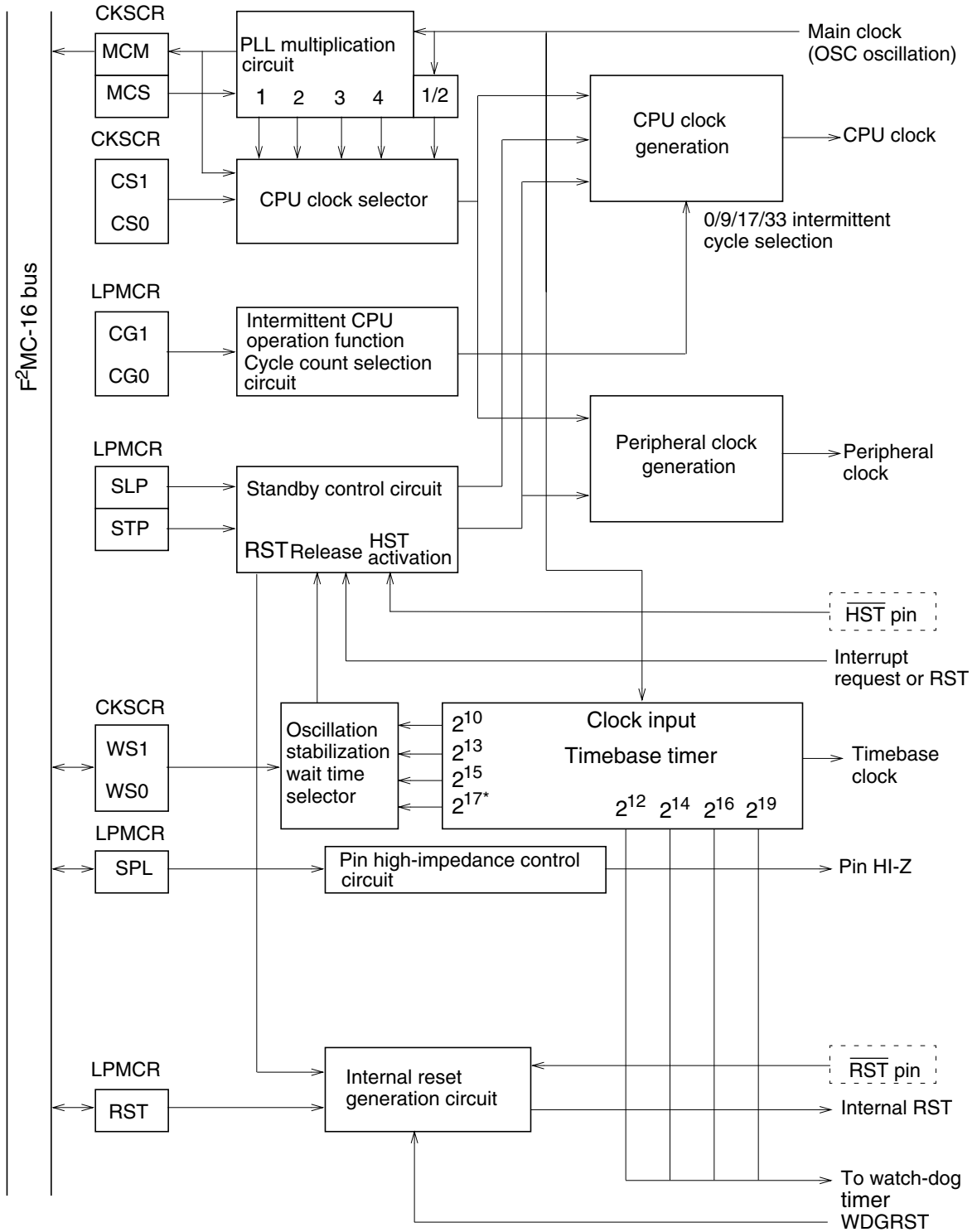
The oscillation stabilization wait time for the main clock upon release of stop or hardware standby mode can be set by the WS1 and WS0 bits.

Note:

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

■ Block Diagram

Figure 6.1-1 Low-power Control Circuit and Clock Generator



6.2 Registers

A low-power control circuit has the following two registers:

- Low-power mode control register (LPMCR)
- Clock selection register (CKSCR)

■ Low-Power Mode Control Register

Address:	7	6	5	4	3	2	1	0	Bit No.
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

Address:	15	14	13	12	11	10	9	8	Bit No.
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)	

6.2.1 Low-Power Mode Control Register (LPMCR)

In association with the clock selection register, the low-power mode control register sets various operation modes to reduce power consumption.

■ Low-Power Mode Control Register (LPMCR)

Address:	7	6	5	4	3	2	1	0	↔ Bit No.
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)	

[bit 7] STP

Writing "1" to this bit starts the watch mode (CKSCR.MCS=0) or stop mode (CKSCR.MCS=1). Writing "0" performs no operation. This bit is cleared to "0" upon a reset, watch mode release, or stop mode release. This is a write-only bit. "0" is always read from this bit.

[bit 6] SLP

Writing "1" to this bit starts sleep mode. Writing "0" performs no operation. This bit is cleared to "0" upon a reset, clock release, or stop release.

Writing "1" to the STP and SLP bits simultaneously starts clock or stop mode. This is a write-only bit. "0" is always read from this bit.

[bit 5] SPL

When "0" is written to this bit, the external pin level in watch mode or stop mode is maintained. When "1" is written to this bit, the external pin in clock or stop mode is set to high impedance. This bit is cleared to "0" upon a reset. This bit is readable and writable.

It is important to note that when SPL is set to 0 and the microcontroller is in the stop mode or the watch mode (STP=1), all inputs must be provided with stable digital values. Otherwise it results in current consumption at the input buffers. (A/D analog inputs are exception)

Generally it is recommended to set the SPL bit to 1 when the microcontroller is in the stop mode or the watch mode in order to disable all input buffers.

[bit 4] RST

Writing "0" to this bit generates internal reset signals for three machine cycles. Writing "1" performs no operation. "1" is always read from this bit.

[bit 3] Reserved

This bit must be set to "1".

[bits 2 and 1] CG1 and CG0

These bits are used to set the clock pause cycle count during intermittent CPU operation.

These bits are initialized to "00" upon a reset by power-on, hardware standby, or watch-dog. These bits are not initialized by any other type of reset. These bits are readable and writable.

The intermittent CPU operation function pauses the clock to the CPU when a register, internal memory, or internal resource is accessed, thus delaying the activation of the internal bus cycle. CPU execution is slowed while high-speed clock is supplied to an internal resource, enabling processing at low-power consumption.

Table 6.2-1 CG Bit Setting

CG1	CG0	CPU clock pause cycle count
0	0	0 cycle (CPU clock = Resource clock)
0	1	9 cycles (CPU clock: Resource clock = 1:3 to 4 approx.)
1	0	17 cycles (CPU clock: Resource clock = 1:5 to 6 approx.)
1	1	33 cycles (CPU clock: Resource clock = 1:9 to 10 approx.)

[bit 0] Reserved

This bit must be set to "0".

Note:

To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode or watch mode, disable the output of peripheral functions, and set the STP bit of the low-power mode control register (LPMCR) to 1.

This applies to the following pins:

P06/OUT0, P07/OUT1, P10/OUT2, P11/OUT3, P12/OUT4, P13/OUT5, P15/TX1, P16/SG0, P17/SGA, P34/SOT0, and P35/SCK0

6.2.2 Clock Selection Register (CKSCR)

The clock selection register sets and controls the CPU machine clock, and sets the oscillation stabilization wait time when power is turned on or oscillation is restored.

■ Clock Selection Register (CKSCR)

Address:	15	14	13	12	11	10	9	8	⇐ Bit No.
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)	

[bit 14] MCM

This bit indicates whether the main clock or PLL clock is selected as the machine clock. "0" indicates that the PLL clock is selected, and "1" indicates that the main clock is selected. When MCS=0 and MCM=1, the system is waiting for the PLL clock oscillation to stabilize. The PLL clock oscillation stabilization wait time is fixed to 2^{13} main clock cycles.

[bits 13 and 12] WS1 and WS0

These bits are used to set the main clock oscillation stabilization wait time upon release of stop or hardware standby mode.

These bits are initialized to "11" upon a power-on reset. These bits are not initialized by any other type of reset. These bits are readable and writable.

Table 6.2-2 WS Bit Setting

WS1	WS0	Oscillation stabilization wait time (at 4 MHz source oscillation)
0	0	Approx. 256 μ s (2^{10} counts of source oscillation)
0	1	Approx. 2.05 ms (2^{13} counts of source oscillation)
1	0	Approx. 8.19 ms (2^{15} counts of source oscillation)
1	1	Approx. 32.77 ms (2^{17} counts of source oscillation)

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

More stabilization time of 2^{17} is added to the default duration only upon with the power-on reset. Therefore, after power-on there will be about 65.54ms of the stabilization time.

[bit 11] Reserved

This bit must be set to "1".

[bit 10] MCS

This bit is used to select the main clock or PLL clock as the machine clock. Writing "0" selects the PLL clock and writing "1" selects the main clock. When this bit is updated from "1" to "0", the PLL clock oscillation stabilization wait period is created by automatically clearing the timebase timer. The oscillation stabilization wait time for the PLL clock is fixed to 2^{13} main clock cycles. (The oscillation wait time is about 2 ms at 4 MHz source oscillation.)

When the main clock is selected, the operation clock is generated by dividing the main clock by two. (The operation clock is 2 MHz at 4 MHz source oscillation.)

This bit is initialized to "1" by the power-on reset, hardware standby, or watch-dog reset. But it is not initialized by the external reset from the \overline{RST} pin or by the software reset (the RST bit in the LPMCR register).

Note:

When updating the MCS bit from "1" to "0", ensure that the timebase timer interrupt is masked by the TBIE bit or the ILM bit of the CPU.

[bits 9 and 8] CS1 and CS0

These bits are used to select the multiplication factor of the PLL clock.

These bits are initialized to "00" upon a power-on reset. These bits are not initialized by any other type of reset.

Write is disabled when "0" is written to the MCS bit. To update the CS bit, set "1" in the MCS bit (to start main clock mode).

These bits are readable and writable.

Table 6.2-3 CS Bit Setting

CS1	CS0	Machine clock (at 4 MHz source oscillation)
0	0	4 MHz (Operation frequency = OSC oscillation frequency)
0	1	8 MHz (Operation frequency = OSC oscillation frequency *2)
1	0	12 MHz (Operation frequency = OSC oscillation frequency *3)
1	1	16 MHz (Operation frequency = OSC oscillation frequency *4)

Note:

When the operating voltage is 5 V, the OSC source oscillation can be between 3 MHz and 5 MHz. When an external clock source is used, its frequency can be between 3MHz and 16MHz. Since the highest operating frequency for the CPU and peripheral resource circuits is 16 MHz, however, normal operation is not guaranteed if a multiplication factor resulting in a higher frequency than 16 MHz is specified. For example, if the external clock frequency is 16 MHz, only 1 can be specified as the multiplication factor.

The lowest operating frequency of the VCO oscillation is 4 MHz, and an oscillation below 4 MHz must not be specified.

6.3 Low-Power Mode Operation

Table 6.3-1 "Low-power mode status" lists the chip status in each operation mode.

■ Low-power Mode Operation

Table 6.3-1 Low-power mode status

	Transition condition	Oscillation & T.B.T	PLL	CPU	Watch Timer	Other Peripheral	Pin	Release method
Main sleep	MCS=1 SLP=1	Operating	Stopped	Stopped	Operating	Operating	Operating	External Reset Interrupt
PLL sleep	MCS=0 SLP=1	Operating	Operating	Stopped	Operating	Operating	Operating	External Reset Interrupt
Watch (SPL=0)	MCS=0 STP=1	Operating	Stopped	Stopped	Operating	Stopped	Held ^(*)	External Reset External Interrupt
Watch (SPL=1)	MCS=0 STP=1	Operating	Stopped	Stopped	Operating	Stopped	HI-Z	External Reset External Interrupt
Stop (SPL=0)	MCS=1 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Held ^(*)	External Reset External Interrupt
Stop (SPL=1)	MCS=1 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	External Reset External Interrupt
Hardware standby	$\overline{\text{HST}}=\text{L}$	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	$\overline{\text{HST}}=\text{H}$

*1: When the SPL is set to 0 in the stop mode or the watch mode, all inputs must be provided with stable digital values. Otherwise it results in current consumption at the input buffers. (A/D analog inputs are exception)

Note:

To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode or watch mode, disable the output of peripheral functions, and set the STP bit of the low-power mode control register (LPMCR) to 1.

This applies to the following pins:

P06/OUT0, P07/OUT1, P10/OUT2, P11/OUT3, P12/OUT4, P13/OUT5, P15/TX1, P16/SG0, P17/SGA, P34/SOT0, and P35/SCK0

CHAPTER 6 LOW-POWER CONTROL CIRCUIT

■ Note: Low-power Mode Control Register Access

Writing data to the low-power mode control register starts low-power mode (stop or sleep mode). In this case, use an instruction shown in Table 6.3-2 "List of Instructions Used for Transition to Low-power Mode". If any other instruction is used to start low-power mode, malfunction may result. Any instruction can be used to control functions other than transition of the low-power mode control register to low-power mode.

To write data to the low-power mode control register in word length, ensure that the data is written to an even-number address. If low-power mode is started by writing data to an odd-number address, malfunction may result.

Table 6.3-2 List of Instructions Used for Transition to Low-power Mode

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

■ Notes on the transition to low-power mode

To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode or watch mode, use the following procedure:

1. Disable the output of peripheral functions.
2. Set the SPL bit of the low-power mode control register (LPMCR) to 1, and set the STP bit to 1.

6.3.1 Sleep Mode

In sleep mode, only the clock supplied to the CPU is stopped. As a result, the CPU terminates while peripheral circuits keep operating.

■ Transition to Sleep Mode

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

If an interrupt request has been issued when "1" is written to the SLP bit, the standby control circuit does not enter sleep mode. Therefore, the CPU executes the next instruction if the interrupt cannot be accepted, or immediately branches to the interrupt processing routine if the interrupt can be accepted.

In sleep mode, the values of special registers such as the accumulator and the internal RAM are maintained.

■ Releasing Sleep Mode

The standby control circuit releases sleep mode in the event of a reset input or an interrupt. If sleep mode is released by a reset, the reset status takes effect after sleep mode is released.

If a peripheral circuit or similar issues an interrupt request of a higher interrupt level than 7 in sleep mode, the standby control circuit releases sleep mode. After sleep mode is released, processing is handled as normal interrupt processing. The CPU executes interrupt processing if the interrupt can be accepted according to the I flag, ILM, and the interrupt control register (ICR). If the interrupt cannot be accepted, processing continues from the instruction following the instruction that was being executed before the transition to sleep mode.

Note:

Usually, interrupt processing is started after the instruction following the instruction that was executed during the transition to sleep mode.

6.3.2 Watch Mode

Watch mode stops operations other than the source oscillation, timebase timer, and watch timer, resulting in almost all functions of the chip being stopped.

■ Transition to Watch Mode

The standby control circuit is set to watch mode when the MCS bit of the clock selection register is "0" and "1" is written to the STP bit of the low-power mode control register. In watch mode, all operations are stopped except for the source oscillation and timebase timer. Most functions of the chip stop.

Using the STP bit of the low-power mode control register, the I/O pin may be maintained at the immediately preceding status or at high impedance in watch mode.

If an interrupt request has been issued when "1" is written to the STP bit, the standby control circuit does not enter watch mode.

In watch mode, the values of special registers such as the accumulator and the internal RAM are maintained.

Note:

To set a pin to high impedance when the pin is shared by a peripheral function and a port in watch mode, disable the output of peripheral functions, and set the STP bit of the low-power mode control register (LPMCR) to 1.

This applies to the following pins:

P06/OUT0, P07/OUT1, P10/OUT2, P11/OUT3, P12/OUT4, P13/OUT5, P15/TX1, P16/SG0, P17/SGA, P34/SOT0, and P35/SCK0

■ Releasing Watch Mode

The standby control circuit releases watch mode in the event of a reset input or an interrupt. If watch mode is released by a reset, the reset status takes effect after watch mode is released.

To return from watch mode, the standby control circuit initially releases watch mode, then enters the PLL clock oscillation stabilization wait state. The MCS bit is not cleared by an external reset, so the reset sequence is performed using the main clock if the reset period is shorter than the PLL clock oscillation stabilization wait period. The PLL clock oscillation stabilization wait period is 2^{13} to 3×2^{13} main clock cycles depending on the timebase timer status, because the timebase timer is not cleared.

If a peripheral circuit or similar issues an interrupt request of a higher interrupt level than 7 in watch mode, the standby control circuit releases watch mode. After the watch mode is released, processing is handled as normal interrupt processing. The CPU executes interrupt processing if the interrupt can be accepted according to the I flag, ILM, and the interrupt control register (ICR). If the interrupt cannot be accepted, processing continues from the instruction following the instruction that was being executed during transition to watch mode.

Note:

Usually, interrupt processing is started after the instruction following the instruction that was being executed during the transition to watch mode.

The standby control circuit enters PLL clock oscillation stabilization wait status when watch mode is released. If the PLL clock is not used, write "1" to the MCS bit by an instruction immediately following the reset or interrupt.

6.3.3 Stop Mode

Stop mode stops the source oscillation, resulting in all functions of the chip being stopped. Data can be maintained at the lowest power consumption possible.

■ Transition to Stop Mode

The standby control circuit is set to stop mode when the MCS bit of the clock selection register is "1" and "1" is written to the STP bit of the low-power mode control register. In stop mode, the source oscillation is stopped and all functions of the chip are stopped. Therefore, data can be maintained at the lowest power consumption possible.

Using the SPL bit of the LPMCR, the I/O pins can be maintained at the immediately preceding status or at high impedance in stop mode. When the SPL bit is set to 0, all inputs must be provided with stable digital values. Otherwise it results in current consumption at the input buffers. (A/D analog inputs are exception)

If an interrupt request has been issued when "1" is written to the STP bit, the standby control circuit does not enter the stop mode.

In stop mode, the values of special registers such as the accumulator and the internal RAM are maintained.

Note:

To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode, disable the output of peripheral functions, and set the STP bit of the low-power mode control register (LPMCR) to 1.

This applies to the following pins:

P06/OUT0, P07/OUT1, P10/OUT2, P11/OUT3, P12/OUT4, P13/OUT5, P15/TX1, P16/SG0, P17/SGA, P34/SOT0, and P35/SCK0

■ Releasing Stop Mode

The standby control circuit releases stop mode in the event of a reset input or an interrupt. If stop mode is released by a reset, the reset status takes effect after stop mode is released.

If a peripheral circuit or similar issues an interrupt request of a higher interrupt level than 7 in stop mode, the standby control circuit releases stop mode. After stop mode is released, the processing is handled as normal interrupt processing after the main clock oscillation stabilization wait period specified by the WS1 and WS0 bits of CKSCR. The CPU executes interrupt processing if the interrupt can be accepted according to the I flag, ILM, and the interrupt control register (ICR). If the interrupt cannot be accepted, processing continues from the instruction following the instruction that was being executed during transition to stop mode.

■ Setting the Oscillation Stabilization Wait Time

Use the WS1 and WS0 bits to specify the oscillation stabilization wait time when stop mode or hardware standby mode is released. Specify the oscillation stabilization wait time according to the types and characteristics of the oscillator circuit and oscillator device connected to the X0 and X1 pins.

These bits are not initialized upon a reset, except for a power-on reset. Upon a power-on reset, these bits are initialized to "11". Therefore, at power-on, the oscillation stabilization wait time is about 2^{17} counts of source oscillation.

Note:

Usually, interrupt processing is started after the instruction following the instruction that was being executed during the transition to stop mode.

6.3.4 Hardware Standby Mode

In the hardware standby mode, oscillation is stopped and all I/O pins are set to high impedance while the $\overline{\text{HST}}$ pin is at "L" level, regardless of other statuses (including reset).

■ Transition to Hardware Standby Mode

The standby control circuit can be set in hardware standby mode from any status by setting the $\overline{\text{HST}}$ pin at "L" level. In hardware standby mode, oscillation is stopped and all I/O pins are set to high impedance while the $\overline{\text{HST}}$ pin is at "L" level, regardless of other status including reset.

In hardware standby mode, the internal RAM contents are maintained but the special registers such as the accumulator are initialized.

■ Releasing Hardware Standby Mode

Hardware standby mode can be released only by the $\overline{\text{HST}}$ pin. When the $\overline{\text{HST}}$ pin is set at "H" level, the standby control circuit releases hardware standby mode, enables the internal reset signal, and enters oscillation stabilization wait status. After the oscillation stabilization wait period, the standby control circuit releases the internal reset, and consequently the CPU starts execution from the reset sequence.

■ Setting the Oscillation Stabilization Wait time

Use the WS1 and WS0 bits to specify the oscillation stabilization wait time when stop mode or hardware standby mode is released. Specify the oscillation stabilization wait time according to the types and characteristics of the oscillator circuit and oscillator device connected to the X0 and X1 pins.

These bits are not initialized upon a reset, except for a power-on reset. Upon a power-on reset, these bits are initialized to "11". Therefore, at power-on, the oscillation stabilization wait time is about 2^{17} counts of source oscillation.

6.4 Intermittent CPU Operation

The intermittent CPU operation function pauses the clock supplied to the CPU when a register, or internal memory (ROM, RAM, I/O, or resource) is accessed, delaying the activation of the internal bus cycle. The CPU execution speed is decreased while a high-speed clock is supplied to internal resources, thus enabling processing at low-power consumption.

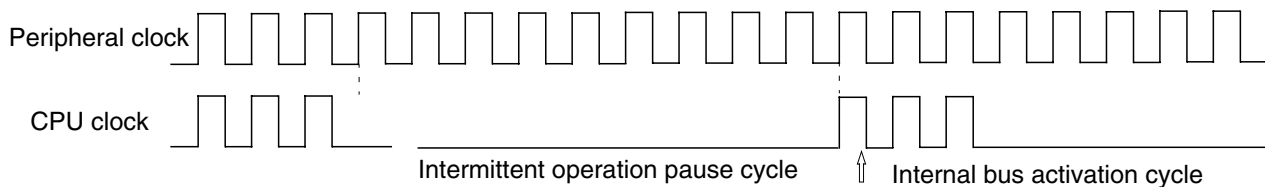
■ Intermittent CPU Operation

Figure 6.4-1 "Intermittent CPU Operation" is a diagram of intermittent CPU operation. For intermittent CPU operation, the CG1 and CG0 bits are used to select the cycle count for clock pausing.

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

An instruction execution time using the intermittent CPU operation function can be obtained by adding a compensation value to the ordinary execution time. The compensation value is obtained by multiplying the number of accesses to a register, internal memory, or internal resource by the cycle count for pausing.

Figure 6.4-1 Intermittent CPU Operation



6.5 Switching Machine Clocks

Writing to the MCS bit in the CKSCR register switches the machine clock from the main clock to the PLL clock.

■ Switching between Main Clock and PLL Clock

Write data to the MCS bit of the CKSCR register to switch between the main clock and PLL clock.

When the MCS bit is changed from "1" to "0", the PLL clock takes over the main clock after the PLL clock oscillation stabilization wait time (2^{13} machine clock cycles).

When the MCS bit is changed from "0" to "1", the main clock takes over the PLL clock when the edges of the PLL and main clocks match (after about 1 to 8 PLL clock cycles).

Writing to the MCS bit does not change the machine clock immediately. To manipulate a resource that depends on the machine clock, always reference the MCM bit before hand to check that the machine clock has been switched.

Note:

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

■ Initializing the Machine Clock

The MCS bit cannot be initialized by a reset that uses the $\overline{\text{RST}}$ external reset pin or the RST bit. The MCS bit is initialized to "1" by any other reset.

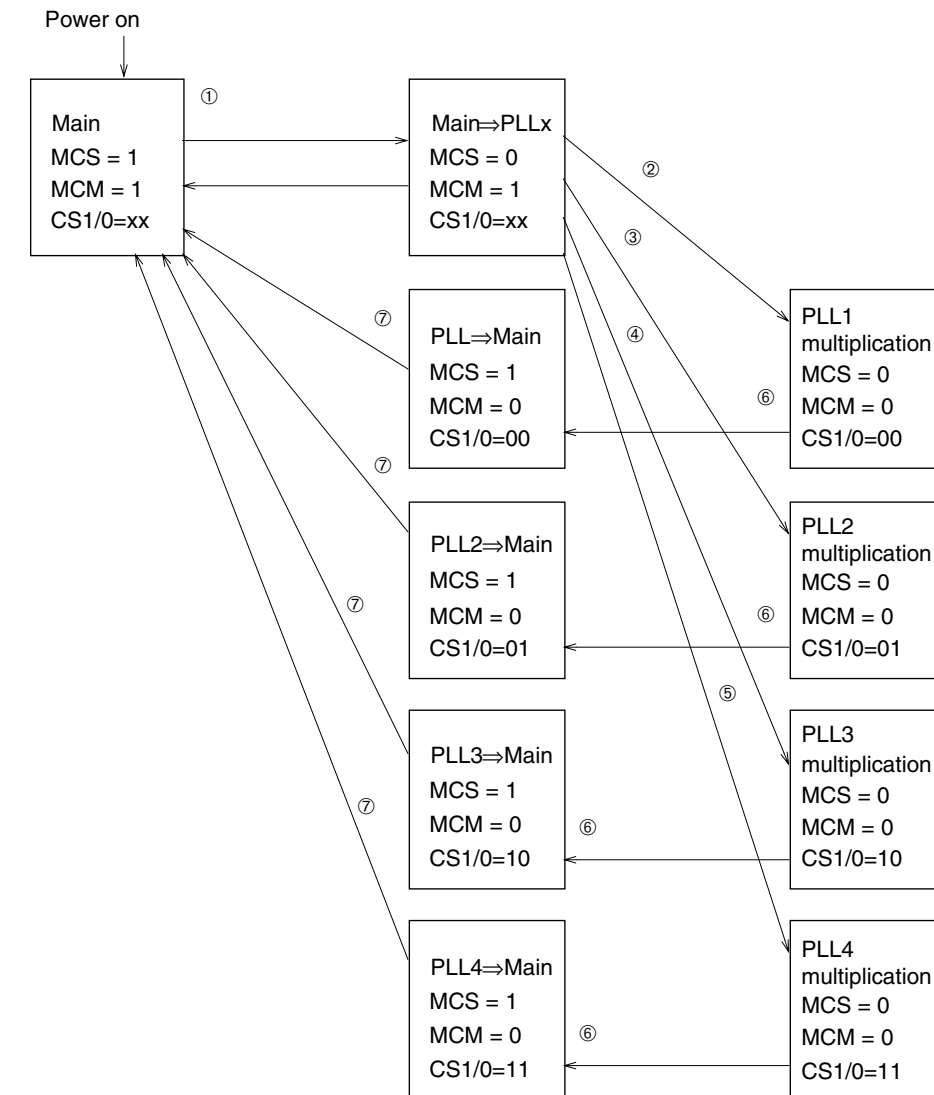
6.6 Status Transition of Clock Selection

The oscillation stabilization wait time for the PLL clock is fixed at 2^{13} main clock cycles. (The oscillation wait time is about 2 ms at a source oscillation of 4 MHz.)

■ Status Transition of Clock Selection

Figure 6.6-1 "Status Transition of Clock Selection" is a diagram of status transition of clock selection.

Figure 6.6-1 Status Transition of Clock Selection



- ① MCS bit clear
- ② End of PLL clock oscillation stabilization wait & CS1/O=00
- ③ End of PLL clock oscillation stabilization wait & CS1/O=01
- ④ End of PLL clock oscillation stabilization wait & CS1/O=10
- ⑤ End of PLL clock oscillation stabilization wait & CS1/O=11
- ⑥ MCS bit set (including hardware standby and watch-dog reset)
- ⑦ Synchronization timing between PLL clock and main clock

Note:

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

CHAPTER 7 MEMORY ACCESS MODES

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

7.1 "Outline of Memory Access Modes"

7.2 "Mode Pins"

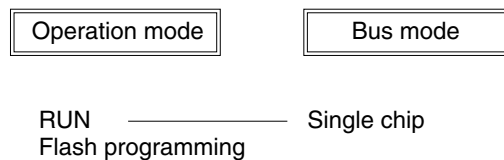
7.3 "Mode Data"

7.1 Outline of Memory Access Modes

In the F²MC-16LX, the following two memory access modes are provided for each of the access methods and access areas:

- Operation mode
 - Bus mode
-

■ Memory Access Modes



For the MB90590 Series, the external bus function is not supported. Therefore the following part of this document is not fully supported. In user applications, please use the MB90590 Series in the single chip mode.

To set the MB90590 Series into the single chip mode, the mode inputs (MD2 to 0) should be "011" and the most significant two bits of the mode data (M1 and M0) should be "00".

○ Operation mode

Operation mode means the mode for controlling the device operation status. The operation mode is specified by the MD_x mode setting pin and the Ex bit in mode data. By selecting an operation mode, normal operation, internal test program activation, or special test function activation can be performed.

○ Bus mode

Bus mode means the mode for controlling the internal ROM operation and external access function. The bus mode is specified by the MD_x mode setting pin and the M_x bit in mode data. The MD_x mode setting pin specifies the bus mode for reading the reset vector and mode data, and the M_x bit in mode data specifies the bus mode for normal operation.

7.2 Mode Pins

Table 7.2-1 "Mode Pins and Modes" describes the operations specified by combinations of the MD2 to MD0 external pins.

■ Mode pins

Table 7.2-1 Mode Pins and Modes

Mode pin setting			Mode name	Reset vector access area	External data bus width	Remarks
MD2	MD1	MD0				
0	0	0	Specification not allowed			
0	0	1				
0	1	0				
0	1	1	Internal vector mode	Internal	(Mode data)	Reset sequence and later segments are controlled based on mode data.
1	0	0	Specification not allowed			
1	0	1				
1	1	0	Flash memory serial programming ^(*1)	-	-	-
1	1	1	Flash memory	-	-	Mode for use of a parallel programmer

*1: Data cannot be written only by setting the flash serial programming mode by mode pins. Other must be set. For details, see Chapter 25 "Examples of MB90F594A/MB90F594G/MB90F591A/MB90F591G Serial Programming Connection".

7.3 Mode Data

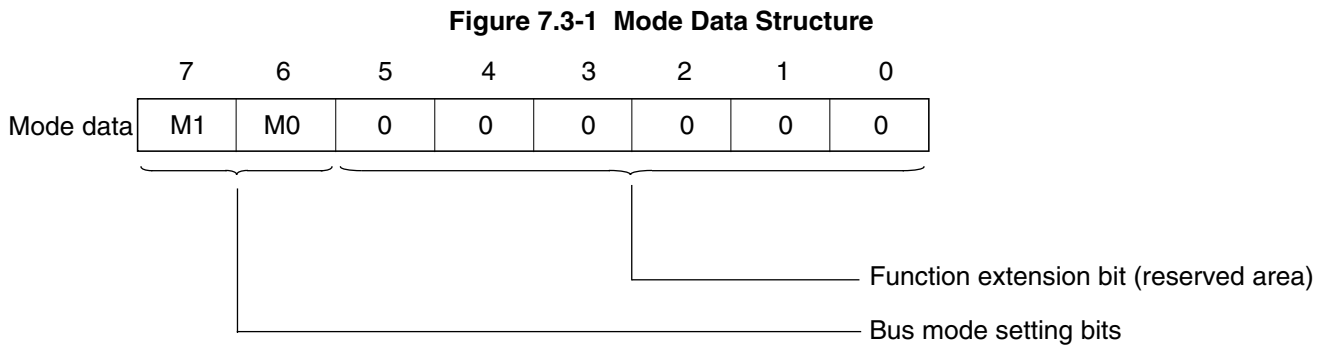
Mode data is stored at $FFFFDF_H$ of main memory and used for controlling the CPU operation. This data is fetched during a reset sequence and stored in the mode register inside the device. The mode register value can be changed only by a reset sequence.

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

Always set the reserved bits to "0".

■ Mode Data

Figure 7.3-1 "Mode Data Structure" is a diagram of the setting of the bits.



■ Bus Mode Setting Bits

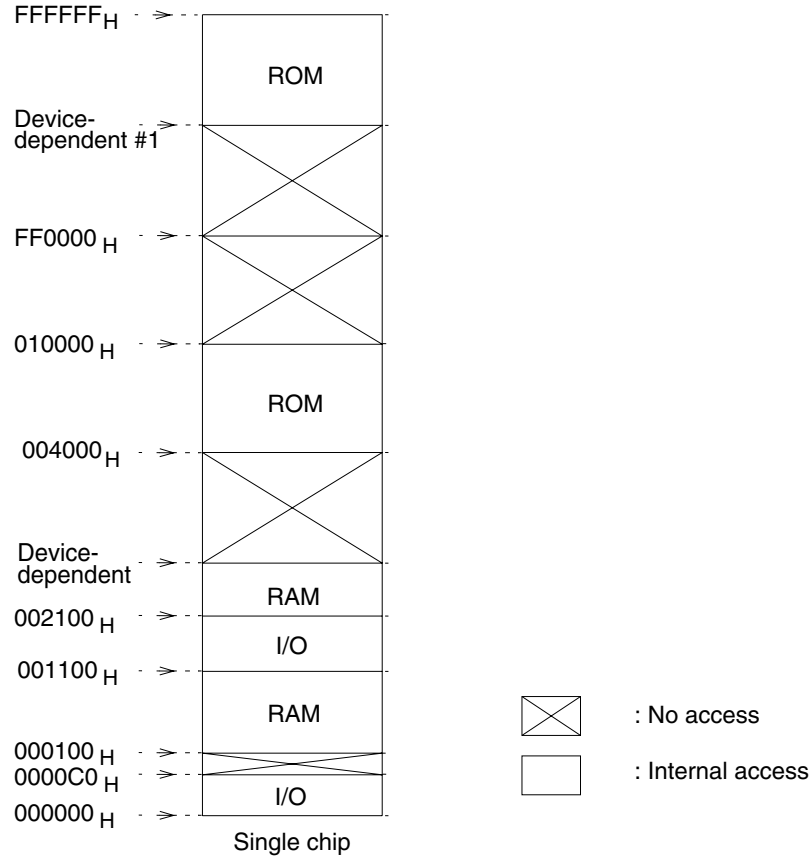
These bits are used to specify the operation mode after the reset sequence is completed. Table 7.3-1 "Bus Mode Setting Bits and Functions" shows the relationship between the bits and the functions.

Table 7.3-1 Bus Mode Setting Bits and Functions

M1	M0	Function	Remarks
0	0	Single chip mode	
0	1	(Inhibited)	
1	0		
1	1		

Figure 7.3-2 "Access Areas and Physical Addresses in each Bus Mode" is a diagram of the correspondence between the access areas and physical addresses for each bus mode.

Figure 7.3-2 Access Areas and Physical Addresses in each Bus Mode



Note:

"Device-dependent" means an address that is determined depending on the device.

■ **Recommended Setting**

Table 7.3-2 "Sample Recommended Setting of Mode Pins and Mode Data" lists a sample recommended setting of mode pins and mode data.

Table 7.3-2 Sample Recommended Setting of Mode Pins and Mode Data

Sample setting	MD2	MD1	MD0	M1	M0
Single chip	0	1	1	0	0

Note:

For the MB90590 series devices with Flash memory, the mode data have predetermined values by the hard-wired logic.

For more information, refer to Section 24.9 "Reset Vector Address in Flash Memory".

CHAPTER 8 I/O PORTS

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

8.1 "I/O Port"

8.2 "I/O Port Registers"

8.1 I/O Ports

Each pin of the ports can be specified as input or output using the direction register if the corresponding peripheral does not use the pin. When a pin is specified as input, the logic level at the pin is read. When a pin is specified as output, the data register value is read. The above also applies to a read operation for the read-modify-write instructions.

Only for Port 0, Port 1, Port 2 and Port 3, the corresponding bits of the Port Direction registers should be set to "1" in order to enable peripheral signal outputs.

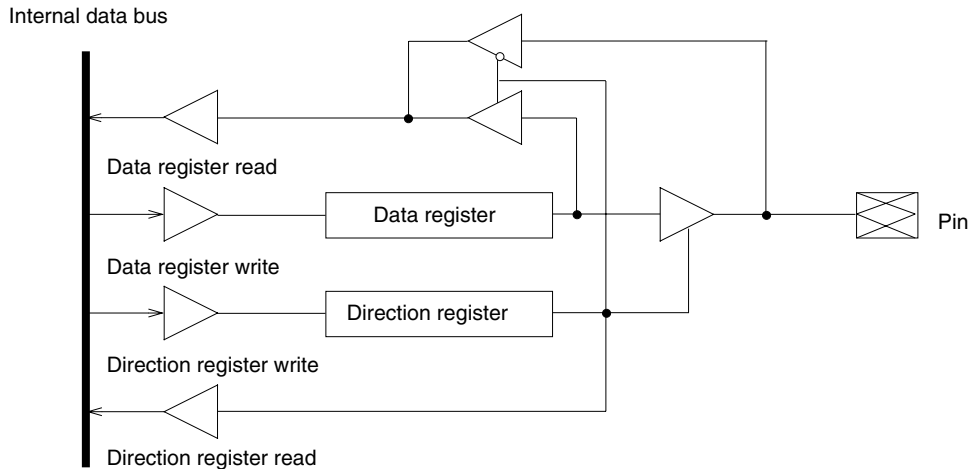
■ I/O Ports

When a pin is used as an output of other peripheral function, the peripheral output value is read regardless of the direction register value.

It is generally recommended that the read-modify-write instructions should not be used for setting the data register prior to setting the port as an output. This is because the read-modify-write instruction in this case results reading the logic level at the port rather than the register value.

Figure 8.1-1 "I/O Port Block Diagram" is a block diagram of the I/O ports.

Figure 8.1-1 I/O Port Block Diagram



8.2 I/O Port Registers

There are three types of I/O port registers:

- Port data register (PDR0 to 9)
- Port direction register (DDR0 to 9)
- Analog input enable register (ADER)

■ I/O Port Registers

Figure 8.2-1 "I/O Port Registers" shows the I/O port registers.

Figure 8.2-1 I/O Port Registers

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000000 H	P07	P06	P05	P04	P03	P02	P01	P00	Port 0 data register (PDR0)
Address : 000001 H	P17	P16	P15	P14	P13	P12	P11	P10	Port 1 data register (PDR1)
Address : 000002 H	P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address : 000003 H	P37	P36	P35	P34	P33	P32	P31	P30	Port 3 data register (PDR3)
Address : 000004 H	P47	P46	P45	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address : 000005 H	P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address : 000006 H	P67	P66	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address : 000007 H	P77	P76	P75	P74	P73	P72	P71	P70	Port 7 data register (PDR7)
Address : 000008 H	P87	P86	P85	P84	P83	P82	P81	P80	Port 8 data register (PDR8)
Address : 000009 H	—	—	P95	P94	P93	P92	P91	P90	Port 9 data register (PDR9)

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000010 H	D07	D06	D05	D04	D03	D02	D01	D00	Port 0 direction register (DDR0)
Address : 000011 H	D17	D16	D15	D14	D13	D12	D11	D10	Port 1 direction register (DDR1)
Address : 000012 H	D27	D26	D25	D24	D23	D22	D21	D20	Port 2 direction register (DDR2)
Address : 000013 H	D37	D36	D35	D34	D33	D32	D31	D30	Port 3 direction register (DDR3)
Address : 000014 H	D47	D46	D45	D44	D43	D42	D41	D40	Port 4 direction register (DDR4)
Address : 000015 H	D57	D56	D55	D54	D53	D52	D51	D50	Port 5 direction register (DDR5)
Address : 000016 H	D67	D66	D65	D64	D63	D62	D61	D60	Port 6 direction register (DDR6)
Address : 000017 H	D77	D76	D75	D74	D73	D72	D71	D70	Port 7 direction register (DDR7)
Address : 000018 H	D87	D86	D85	D84	D83	D82	D81	D80	Port 8 direction register (DDR8)
Address : 000019 H	—	—	D95	D94	D93	D92	D91	D90	Port 9 direction register (DDR9)

Bit	15	14	13	12	11	10	9	8	
Address : 00001B H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 6 analog input enable register (ADER)

8.2.1 Port Data Register

Note that R/W for I/O ports differ from R/W for memory in the following points:

Input mode

Read: The level at the corresponding pin is read.

Write: Data is written to an output latch.

Output mode

Read: The data register latch value is read.

Write: Data is written to an output latch and output to the corresponding pin.

■ Port data Register

Figure 8.2-2 "Port Data Registers" shows the port data registers.

Figure 8.2-2 Port Data Registers

	7	6	5	4	3	2	1	0	Initial value	Access
PDR0 Address: 000000 H	P07	P06	P05	P04	P03	P02	P01	P00	Undefined	R/W *1
PDR1 Address: 000001 H	15	14	13	12	11	10	9	8	Undefined	R/W *1
PDR2 Address: 000002 H	P27	P26	P25	P24	P23	P22	P21	P20	Undefined	R/W *1
PDR3 Address: 000003 H	15	14	13	12	11	10	9	8	Undefined	R/W *1
PDR4 Address: 000004 H	P47	P46	P45	P44	P43	P42	P41	P40	Undefined	R/W *1
PDR5 Address: 000005 H	15	14	13	12	11	10	9	8	Undefined	R/W *1
PDR6 Address: 000006 H	P67	P66	P65	P64	P63	P62	P61	P60	Undefined	R/W *1
PDR7 Address: 000007 H	15	14	13	12	11	10	9	8	Undefined	R/W *1
PDR8 Address: 000008 H	7	6	5	4	3	2	1	0	Undefined	R/W *1
PDR9 Address: 000009 H	15	14	13	12	11	10	9	8	Undefined	R/W *1
	—	—	P95	P94	P93	P92	P91	P90	Undefined	R/W *1

8.2.2 Port Direction Register

When a pin is used as a port, the corresponding pin is controlled as described below:

0: Input mode

1: Output mode

■ Port Direction Register

Figure 8.2-3 "Port Direction Registers" shows the port direction registers.

Figure 8.2-3 Port Direction Registers

	7	6	5	4	3	2	1	0	Initial value	Access
DDR0 Address: 000010 H	D07	D06	D05	D04	D03	D02	D01	D00	00000000 B	R/W
DDR1 Address: 000011 H	15	14	13	12	11	10	9	8	00000000 B	R/W
DDR2 Address: 000012 H	D27	D26	D25	D24	D23	D22	D21	D20	00000000 B	R/W
DDR3 Address: 000013 H	15	14	13	12	11	10	9	8	00000000 B	R/W
DDR4 Address: 000014 H	D47	D46	D45	D44	D43	D42	D41	D40	00000000 B	R/W
DDR5 Address: 000015 H	15	14	13	12	11	10	9	8	00000000 B	R/W
DDR6 Address: 000016 H	D67	D66	D65	D64	D63	D62	D61	D60	00000000 B	R/W
DDR7 Address: 000017 H	15	14	13	12	11	10	9	8	00000000 B	R/W
DDR8 Address: 000018 H	D87	D86	D85	D84	D83	D82	D81	D80	00000000 B	R/W
DDR9 Address: 000019 H	—	—	D95	D94	D93	D92	D91	D90	__000000 B	R/W

Note:

The Port Direction Registers for Ports 0 and 1 will stay undefined during Power-On reset and will be initialized to 00_H after the completion of Power-On reset. For this reason, the Port 0 and 1 outputs become undefined during Power-On reset.

8.2.3 Analog Input Enable Register

This register controls the port 6 pins as described below:

0: Port input/output mode

1: Analog input mode

If an external pin is used as an analog input for the A/D converter, the corresponding bit should be set to "1".

■ Analog Input Enable Register

Figure 8.2-4 "Analog Input Enable Register" shows the analog input enable register.

Figure 8.2-4 Analog Input Enable Register

bit	15	14	13	12	11	10	9	8	
Address: 00001B _H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	11111111 B

CHAPTER 9 TIMEBASE TIMER

This chapter explains the functions and operations of the timebase timer.

- 9.1 "Outline of Timebase Timer"
- 9.2 "Timebase Timer Control Register"
- 9.3 "Operations of Timebase Timer"

9.1 Outline of Timebase Timer

The timebase timer consists of an 18-bit timebase counter and a control register. The 18-bit timebase counter divides the system clock. The timebase timer issues interrupts at specified intervals based on carry signals of the timebase counter.

■ Outline of Timebase Timer

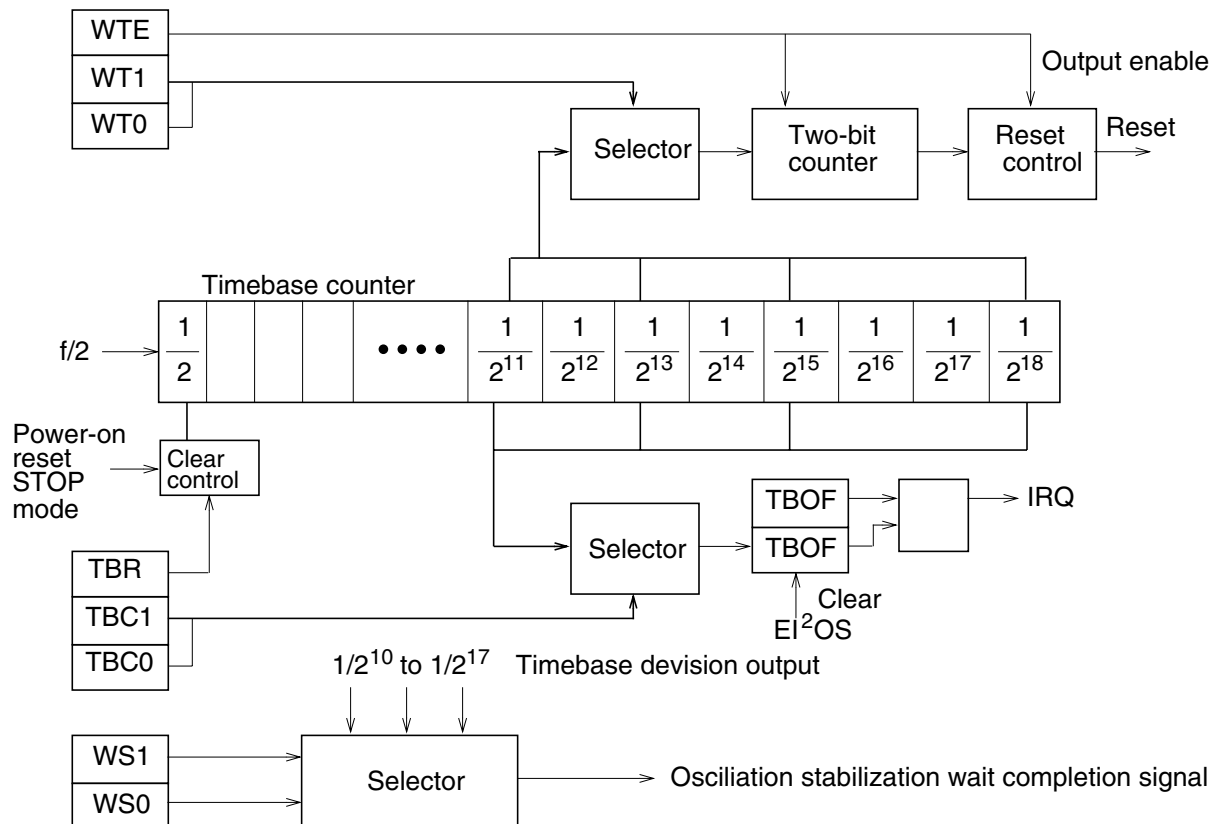
When the power is turned on, the timebase counter can be cleared to all zeroes by setting the stop mode or by software (writing "0" to the TBR bit). The timebase counter is incremented while the source oscillation is input.

The timebase counter can be used as a timer for supplying clock to the watch-dog timer or for waiting for the oscillation to stabilize.

■ Block Diagram of Timebase Timer

Figure 9.1-1 "Block Diagram of Timebase Timer" shows a block diagram of the timebase timer.

Figure 9.1-1 Block Diagram of Timebase Timer



9.2 Timebase Timer Control Register

The timebase timer control register controls interrupts of the timebase timer and can clear the timebase counter.

■ Timebase Timer Control Register (TBTC)

bit	15	14	13	12	11	10	9	8	Initial value
TBTC Address: 0000A9 _H	Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	1- -00100 _B
	W			R/W	R/W	W	R/W	R/W	

[bit 15] Reserved

This is a reserved bit. When writing data to this register, ensure that "1" is written to this bit.

[bit 12] TBIE

This bit is used to enable interval interrupts based on the timebase timer. Writing "1" to this bit enables interrupts, and writing "0" disables interrupts. This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 11] TBOF

This is an interrupt request flag for the timebase timer. While the TBIE bit is "1", an interrupt request is issued when "1" is written to TBOF. This bit is set to "1" for each interval specified with the TBC1 and TBC0 bits.

This bit is cleared by writing "0", transition to stop or hardware standby mode, or a reset. Writing "1" has no effect.

"1" is always read by a read-modify-write instruction.

[bit 10] TBR

This bit clears all bits of the timebase timer counter to "0".

Writing "0" clears the timebase counter.

Writing "1" has no effect.

"1" is always read from this bit.

CHAPTER 9 TIMEBASE TIMER

[bits 9 and 8] TBC1 and TBC0

These bits are used to set the timebase timer interval.

Table 9.2-1 "Selecting the Timebase Timer Interval" lists the specifiable intervals.

Table 9.2-1 Selecting the Timebase Timer Interval

TBC1	TBC0	Interval at 4 MHz source oscillation
0	0	1.024 ms
0	1	4.096 ms
1	0	16.384 ms
1	1	131.072 ms

9.3 Operations of Timebase Timer

The timebase timer functions as a watch-dog timer clock source, timer for waiting for the oscillation to stabilize, and interval timer for generating interrupts at specified intervals.

■ Timebase Counter

The timebase counter consists of an 18-bit counter for a clock generated by dividing the source oscillation input by two. This clock is used to generate the machine clock. While the source oscillation is input, the timebase counter keeps counting. The timebase counter is cleared by a power-on reset, transition to stop or hardware standby mode, or writing "0" to the TBR bit of the TBTC register.

■ Interval Interrupt Function

Interrupts are generated at specified intervals according to the carry signals of the timebase counter. The TBOF flag is set at the intervals specified with the TBC1 and TBC0 bits of the TBTC register. The flag is written to reference to the time at which the timebase timer is cleared last.

Upon transition to stop or hardware standby mode, the timebase timer is used as a timer for waiting for the oscillation to stabilize upon recovery. Therefore, the TBOF flag is immediately cleared upon mode transition.

CHAPTER 10 WATCH-DOG TIMER

This chapter explains the functions and operations of the watch-dog timer.

10.1 "Outline of Watch-Dog Timer"

10.2 "Watch-dog Timer Operations"

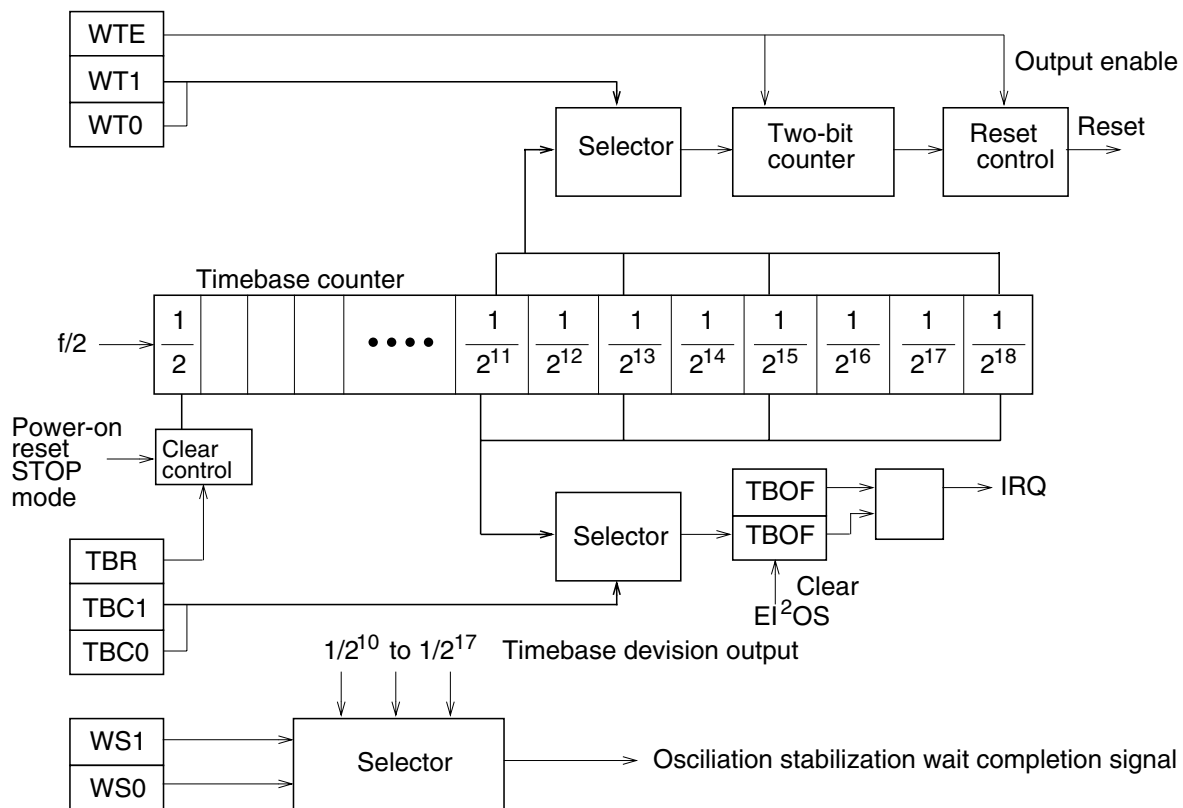
10.1 Outline of Watch-Dog Timer

The watch-dog timer consists of a two-bit watch-dog counter, control register, and watch-dog reset controller. The two-bit watch-dog counter uses the carry signals of an 18-bit timebase counter as a clock source.

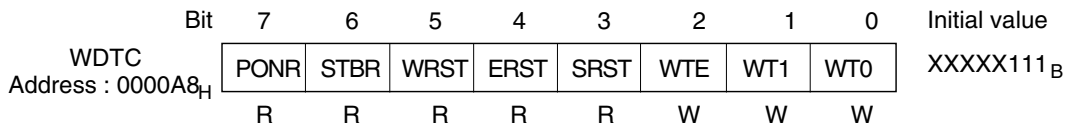
■ Watch-dog Timer Block Diagram

Figure 10.1-1 "Watch-dog Timer Block Diagram" is a diagram of the configuration of the watch-dog timer.

Figure 10.1-1 Watch-dog Timer Block Diagram



■ Watch-dog Timer Control Register (WDTC)



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

These flags indicate the reset causes. The flags are set upon a reset as described in Table 10.1-1 "Reset Cause Registers".

All bits are cleared to "0" after the WDTC register is read. These bits are read-only bits. For details, see Section 5.2 "Reset Cause Occurrence".

Table 10.1-1 Reset Cause Registers

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

(*: The previous value is maintained.)

[bit 2] WTE

While the watch-dog timer is stopped, writing "0" to this bit activates the watch-dog timer. Subsequently, writing "0" clears the watch-dog timer counter. Writing "1" has no effect.

The watch-dog timer is stopped by power-on, hardware standby, or reset by watch-dog timer. "1" is always read from this bit.

[bits 1 and 0] WT1 and WT0

These bits are used to select the watch-dog timer interval. Only the data items written during watch-dog timer activation are valid. Data items that are written outside watch-dog timer activation are ignored. Table 10.1-2 "Watch-dog Timer Interval Selection Bit" lists the interval settings.

These bits are write-only bits.

Table 10.1-2 Watch-dog Timer Interval Selection Bit

WT1	WT0	Interval (at a source oscillation of 4 MHz)		Main clock cycle count
		Minimum	Maximum	
0	0	approx. 3.58 ms	approx. 4.61 ms	2^{14} plus or minus 2^{11} cycles
0	1	approx. 14.33 ms	approx. 18.43 ms	2^{16} plus or minus 2^{13} cycles
1	0	approx. 57.23 ms	approx. 73.73 ms	2^{18} plus or minus 2^{15} cycles
1	1	approx. 458.7 ms	approx. 589.82 ms	2^{21} plus or minus 2^{18} cycles

Note:

The interval becomes the maximum when the timebase counter is not reset during watch-dog timer operation.

10.2 Watch-dog Timer Operation

The watch-dog timer function enables detection of program malfunction. If the watch-dog timer is not accessed within the specified time due to, for example, a program malfunction, the watch-dog timer resets the system.

■ Activation

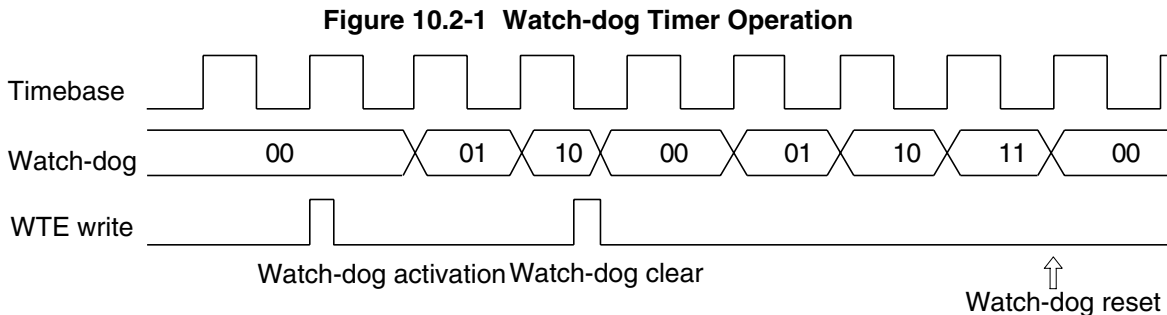
The watch-dog timer is activated by writing "0" to the WTE bit of the WDTC register while the watch-dog timer is stopped. At the same time, the WT1 and WT0 bits are used to set the watch-dog timer reset interval. Only the interval setting specified during activation is valid.

■ Watch-dog Counter

Once the watch-dog timer is activated, the watch-dog timer counter must be periodically cleared within the program. Writing "0" to the WTE bit of the WDTC register clears the watch-dog counter. The watch-dog counter consists of a two-bit counter which uses the carry signals of the timebase counter as a clock source. Therefore, the watch-dog reset time may become shorter than the setting if the timebase counter is cleared.

The watch-dog counter is cleared not only by writing to the WTE bit but also by a reset and transition to the sleep or stop mode. (The watch-dog counter is not cleared by transition to watch mode.)

Figure 10.2-1 "Watch-dog Timer Operation" is a diagram of the watch-dog timer operation.



■ Watch-dog Stop

Once activated, the watch-dog timer is initialized and stopped only by power-on, hardware standby, or reset by watch-dog. Reset by an external pin or software merely clears the watch-dog counter without stopping the watch-dog function.

CHAPTER 11 16-BIT I/O TIMER

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

11.1 "Outline of 16-Bit I/O Timer"

11.2 "16-Bit I/O Timer Registers"

11.3 "16-bit Free-running Timer"

11.4 "Output Compare"

11.5 "Input Capture"

11.1 Outline of 16-Bit I/O Timer

The MB90590 Series contains one 16-bit free-running timer module, three output compare modules, and three input capture modules and supports six input channels and six output channels. The following sections only describes the 16-bit free-running timer, Output Compare 0/1 and Input Capture 0/1.

The remaining modules have the identical functions and the register addresses should be found in the I/O map.

■ 16-bit Free-running Timer

The 16-bit free-running timer consists of a 16-bit up counter, control register, and prescaler. The values output from this timer counter are used as the base timer for input capture and output compare.

○ **Four counter clocks are available.**

Internal clock: $\phi/4$, $\phi/16$, $\phi/64$, $\phi/256$

○ **An interrupt can be generated upon a counter overflow or a match with compare register 0.**

○ **The counter value can be initialized to '0000_H' upon a reset, software clear, or match with compare register 0.**

■ Output Compare (2 Channels per One Module)

The output compare module consists of two 16-bit compare registers, compare output latch, and control register.

When the 16-bit free-running timer value matches the compare register value, the output level is reversed and an interrupt is issued.

○ **The two compare registers can be used independently.**

Output pins and interrupt flags corresponding to compare registers

○ **Output pins can be controlled based on pairs of the two compare registers.**

Output pins can be reversed by using the two compare registers.

○ **Initial values for output pins can be set.**

○ **Interrupts can be generated upon a compare match.**

■ **Input Capture (2 Channels per one Module)**

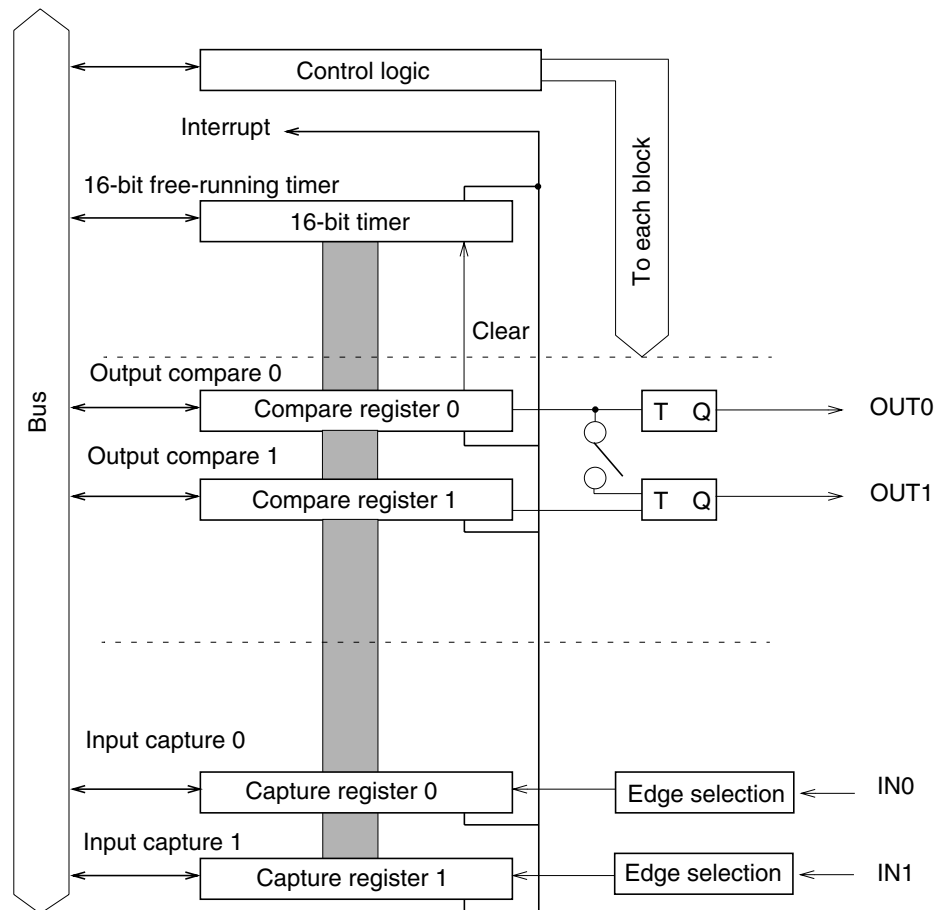
The input capture module consists of two 16-bit capture registers and control registers corresponding to two independent external input pins. The 16-bit free-running timer value can be stored in the capture register and an interrupt is issued simultaneously upon detection of an edge of a signal input from an external input pin.

- **The detection edge of an external input signal can be specified.**
Rising, falling, or both edges
- **Two input channels can operate independently.**
- **An interrupt can be issued upon a valid edge of an external input signal.**
The intelligent I/O service can be activated upon an input capture interrupt.

■ **Block Diagram of 16-bit I/O Timer**

Figure 11.1-1 "Block Diagram of 16-bit I/O Timer" shows a block diagram of the 16-bit I/O timer.

Figure 11.1-1 Block Diagram of 16-bit I/O Timer

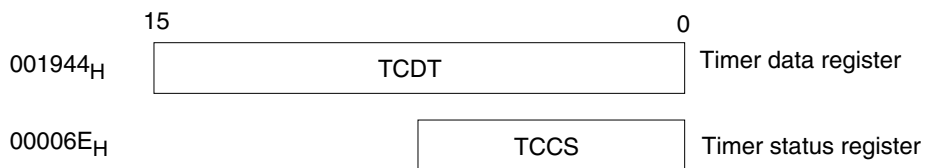


11.2 16-Bit I/O Timer Registers

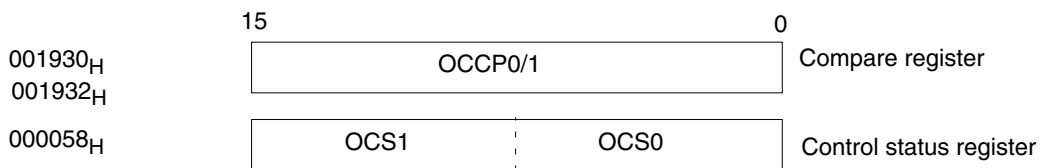
The 16-bit I/O timer has the following three registers:

- 16-bit free-running timer register
- 16-bit output compare register
- 16-bit input capture register

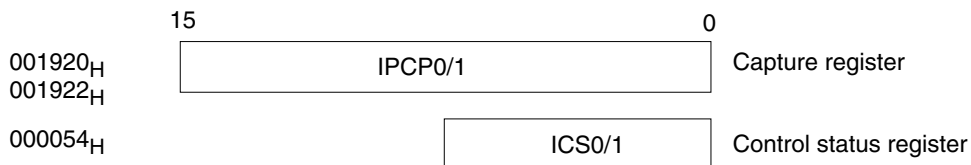
■ 16-bit Free-running Timer



■ 16-bit Output Compare



■ 16-bit Input Capture

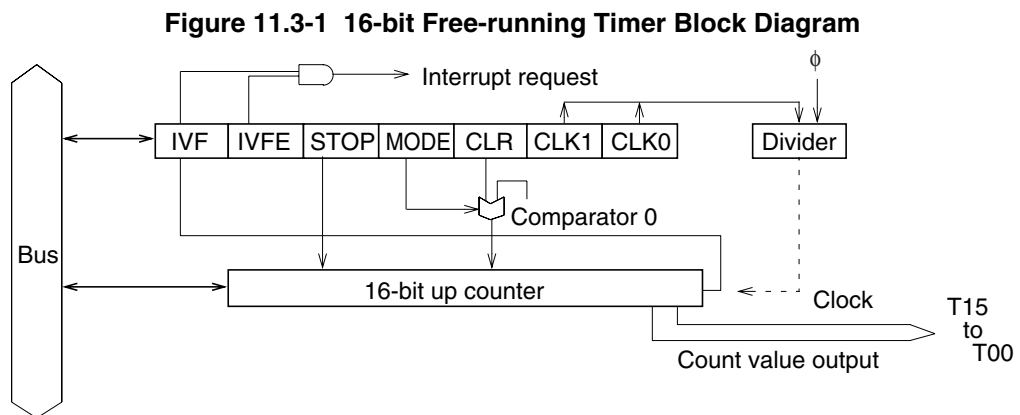


11.3 16-bit Free-running Timer

The 16-bit free-running timer consists of a 16-bit up counter and a control status register. The count values of this timer are used as the base timer for the output compares and input captures.

- Four counter clock frequencies are available.
- An interrupt can be generated upon a counter value overflow.
- The counter value can be initialized upon a match with compare register 0, depending on the mode.

■ 16-bit Free-running Timer Block Diagram

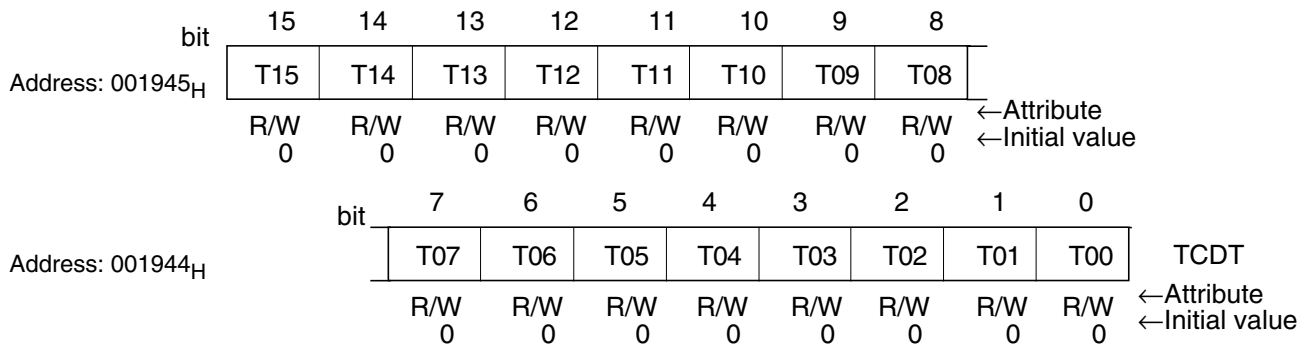


11.3.1 Data Register

The data register can read the count value of the 16-bit free-running timer. The counter value is cleared to "0000" upon a reset. The timer value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (STOP=1).

The data register must be accessed by the word access instructions.

■ Data Register



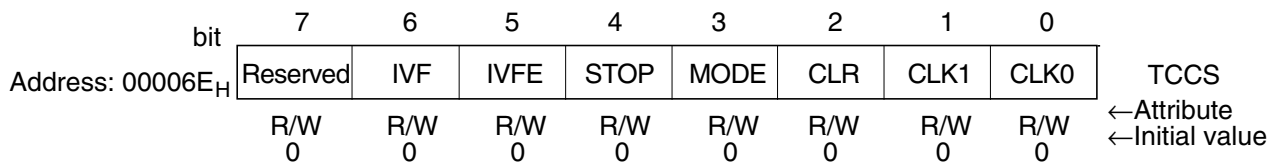
The 16-bit free-running timer is initialized upon the following factors:

- Reset
- Clear bit (CLR) of control status register
- A match between compare register 0 and the timer counter value.

11.3.2 Control Status Register

The control status register sets the operation mode of the 16-bit free-running timer, starts and stops the 16-bit free-running timer, and controls interrupts.

■ Control Status Register



[bit 7] Reserved bit

Always write "0" to this bit.

[bit 6] IVF

This bit is an interrupt request flag of the 16-bit free-running timer.

If the 16-bit free-running timer overflows, or if the counter is cleared by a match with compare register 0, "1" is set to this bit.

An interrupt is issued if the interrupt request enable bit (bit 5: IVFE) is set.

This bit is cleared by writing "0". Writing "1" has no effect.

"1" is always read by a read-modify-write instruction.

0	No interrupt request (initial value)
1	Interrupt request

[bit 5] IVFE

IVFE is an interrupt enable bit of the 16-bit free-running timer. While this bit is "1", an interrupt is issued if "1" is set to the interrupt flag (bit 5: IVF).

0	Interrupt disabled (initial value)
1	Interrupt enabled

[bit 4] STOP

The STOP bit is used to stop the 16-bit free-running timer.

Writing "1" to this bit stops the timer. Writing "0" starts the timer.

0	Counter enabled (operation) (initial value)
1	Counter disabled (stop)

Note:

The output compare operation stops when the 16-bit free-running timer stops.

[bit 3] MODE

The MODE bit is used to set the reset condition of the 16-bit free-running timer.

When "0" is set, the counter value can be initialized by RESET or a clear bit (bit 2: CLR).

When "1" is set, the counter value can be initialized by a match with compare register 0 in addition to RESET and a clear bit (bit 2: CLR).

0	Initialization by reset or clear bit (initial value)
1	Initialization by reset, clear bit, or compare register 0

Note:

The clear bit and the match with compare register initializes the timer when the timer value changes.

[bit 2] CLR

The CLR bit initializes the operating 16-bit free-running timer value to "0000".

When "1" is set, the counter value is initialized to "0000". Writing "0" has no effect. "0" is always read from this bit. The counter value is initialized when the count value changes.

0	No effect (initial value)
1	The counter value is initialized to "0000".

Note:

To initialize the counter value while the timer is stopped, write "0000" to the data register.

[bits 1 and 0] CLK1 and CLK0

CLK1 and CLK0 are used to select the count clock for the 16-bit free-running timer. The clock is updated immediately after a value is written to these bits. Therefore, ensure that the output compare and input capture operations are stopped before a value is written to these bits.

CLK1	CLK0	Count clock	$\phi=16$ MHz	$\phi=8$ MHz	$\phi=4$ MHz	$\phi=1$ MHz
0	0	$\phi/4$	0.25 μ s	0.5 μ s	1 μ s	4 μ s
0	1	$\phi/16$	1 μ s	2 μ s	4 μ s	16 μ s
1	0	$\phi/64$	4 μ s	8 μ s	16 μ s	64 μ s
1	1	$\phi/256$	16 μ s	32 μ s	64 μ s	256 μ s

ϕ = Machine clock

11.3.3 16-bit Free-running Timer Operation

The 16-bit free-running timer starts counting from counter value "0000" after the reset is released. The counter value is used as the reference time for the 16-bit output compare and 16-bit input capture operations.

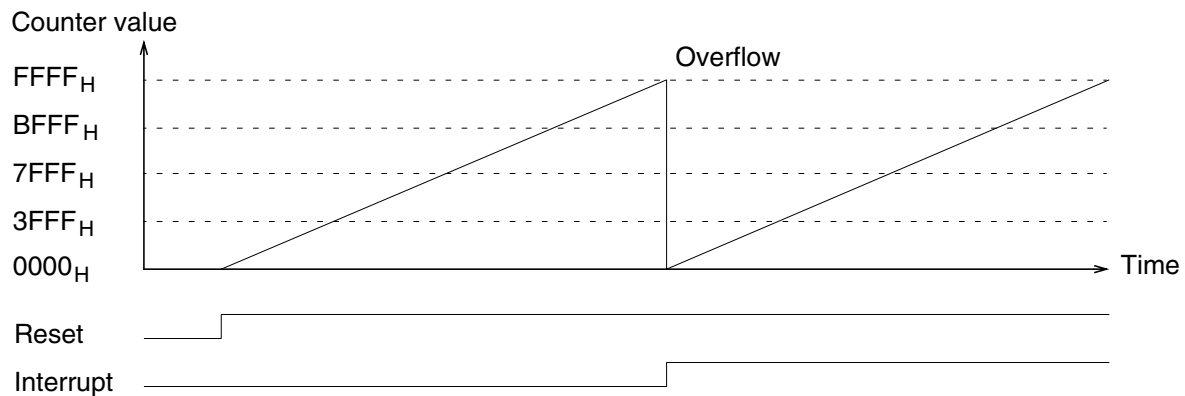
■ 16-bit Free-running Timer Operation

The counter value is cleared in the following conditions:

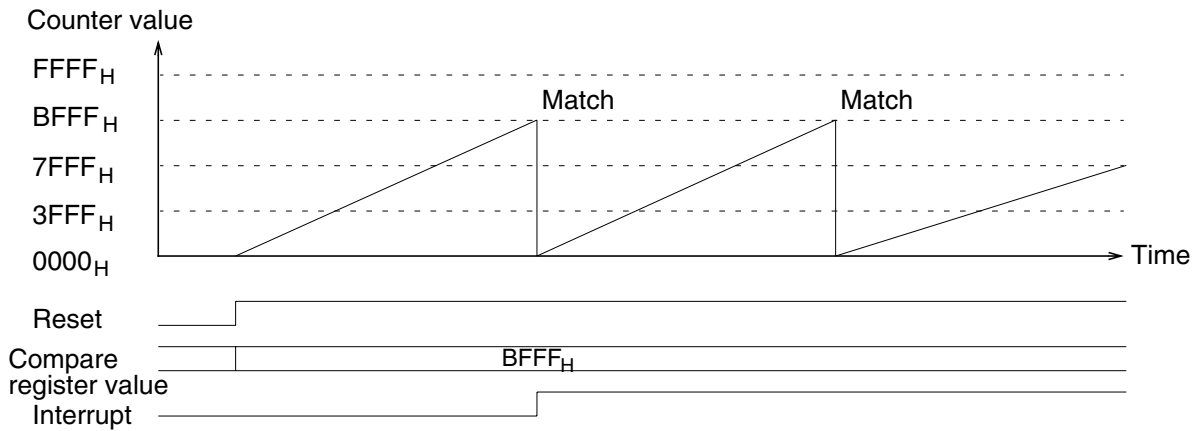
- When an overflow occurs.
- When a match with the output compare register 0 occurs. (This depends on the mode.)
- When "1" is written to the CLR bit of the TCCS register during operation.
- When "0000" is written to the TCDC register during stop.
- Reset

An interrupt can be generated when an overflow occurs or when the counter is cleared by a match with the compare register 0. (Compare match interrupts can be used only in an appropriate mode.)

■ Clearing the Counter by an Overflow



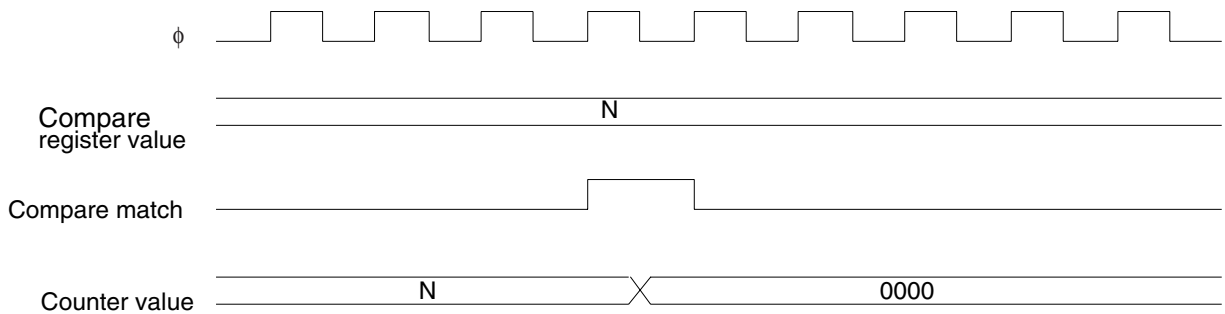
■ Clearing the Counter upon a Match with Output Compare Register 0



■ 16-bit Free-running Timer Timing

○ 16-bit free-running timer clear timing (match with the compare register 0)

The counter can be cleared upon a reset, software clear, or a match with the compare register 0. By a reset or software clear, the counter is immediately cleared. By a match with compare register 0, the counter is cleared in synchronization with the count timing.



11.4 Output Compare

The output compare module consists of two 16-bit compare registers, two compare output pins, and control register. If the value written to the compare register of this module matches the 16-bit free-running timer value, the output level of the pin can be reversed and an interrupt can be issued.

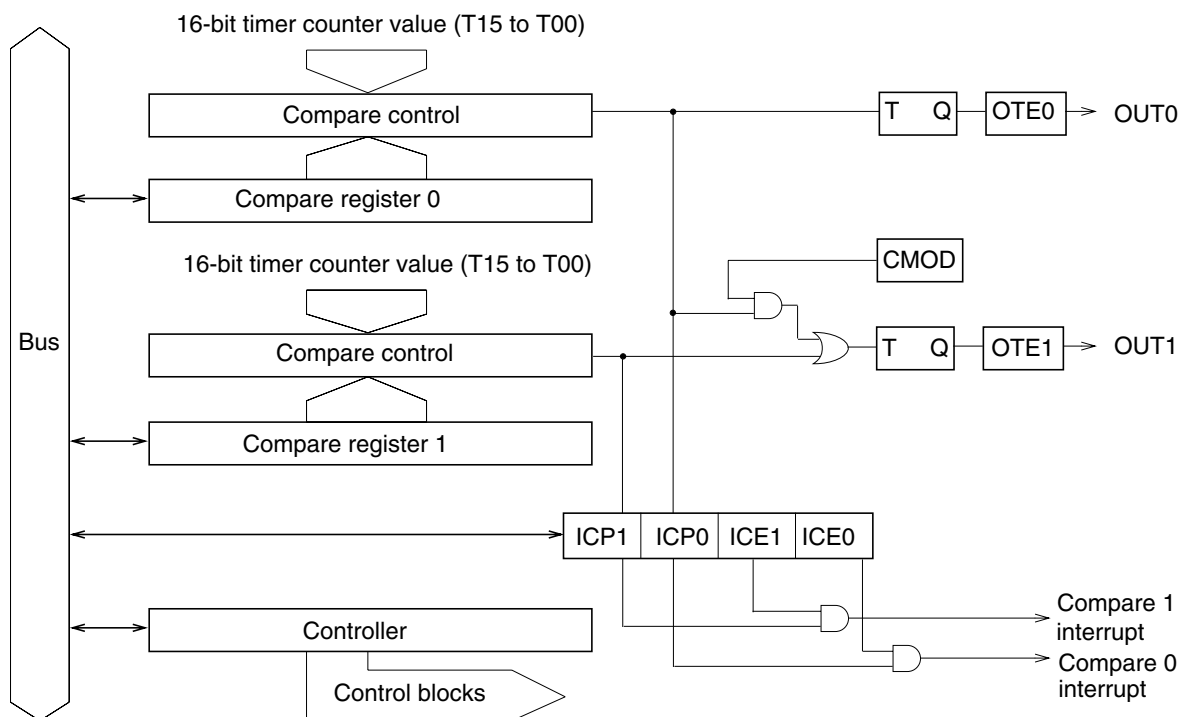
■ Output Compare

- Two compare registers exist that can be used independently. Depending on the setting, the two compare registers can be used to control pin outputs.
- The initial value for the pin output can be specified.
- An interrupt can be issued upon a match as a result of comparison.

■ Output Compare Block Diagram

Figure 11.4-1 "Output Compare Block Diagram" shows a block diagram of output compare.

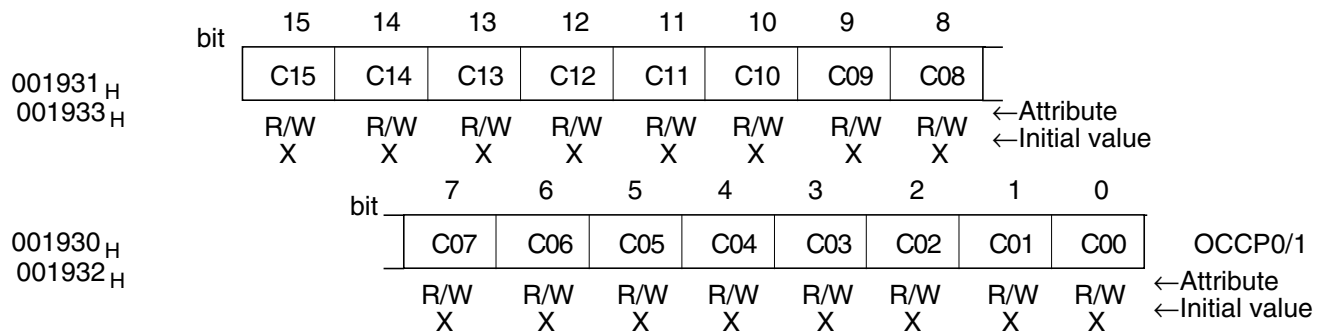
Figure 11.4-1 Output Compare Block Diagram



11.4.1 Output Compare Register

These 16-bit compare registers are compared with the 16-bit free-running timer. Since the initial register values are undefined, set appropriate value before enabling the operation. These registers must be accessed by the word access instructions. When the value of the register matches that of the 16-bit free-running timer, a compare signal is generated and the output compare interrupt flag is set. If output is enabled, the output level corresponding to the compare register is reversed.

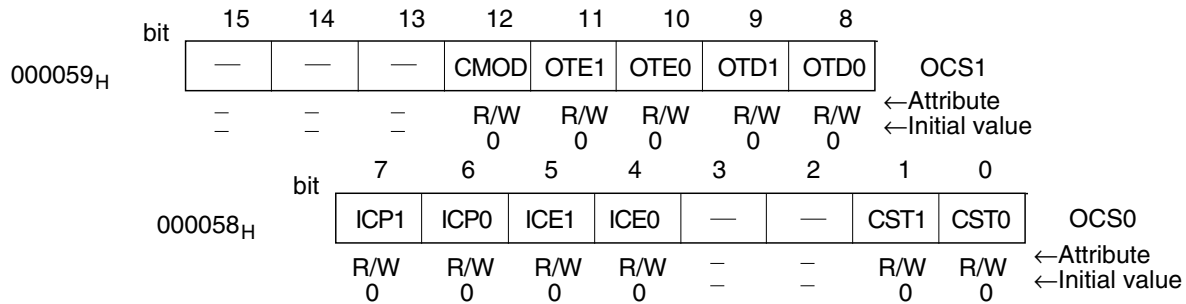
■ Output Compare Register



11.4.2 Control Status Register of Output Compare

The control status register sets the operation mode of output compare, starts and stops output compare, controls interrupts, and sets the external output pins.

■ Control Status Register of Output Compare



[bits 15, 14, and 13] Unused bits

[bit 12] CMOD

CMOD is used to switch the pin output level reverse mode upon a match while pin output is enabled (OTE1=1 or OTE0=1).

- When CMOD=0 (initial value), the output level of the pin corresponding to the compare register is reversed.
 - OUT0: The level is reversed upon a match with compare register 0.
 - OUT1: The level is reversed upon a match with compare register 1.
- When CMOD=1, the output level is reversed for the compare register 0 in the same manner as for CMOD=0. The output level of the pin corresponding to compare register 1 (OUT1), however, is reversed upon a match with compare register 0 or 1. If compare registers 0 and 1 have the same value, the same operation as with a single compare register is performed.
 - OUT0: The level is reversed upon a match with compare register 0.
 - OUT1: The level is reversed upon a match with compare register 0 or 1.

[bits 11 and 10] OTE1 and OTE0

These bits are used to enable the output compare output pins. The initial value for these bits is "0".

0	General-purpose port (initial value)
1	Output compare pin output

Note:

OTE1: Corresponds to output compare 1 (OUT1).

OTE0: Corresponds to output compare 0 (OUT0).

When they are specified as outputs, the corresponding bits of the Port Direction registers should also be set to "1".

[bits 9 and 8] OTD1 and OTD0

These bits are used to change the pin output level when the output compare pin output is enabled. The initial value of the compare pin output is "0". Ensure that the compare operation is stopped before a value is written. When read, these bits indicate the output compare pin output value.

0	Sets "0" for the compare pin output. (initial value)
1	Sets "1" for the compare pin output.

Note:

OTD1: Corresponds to output compare 1.

OTD0: Corresponds to output compare 0.

[bits 7 and 6] ICP1 and ICP0

These bits are used as output compare interrupt flags. "1" is set to these bits when the compare register value matches the 16-bit free-running timer value. While the interrupt request bits (ICE1 and ICE0) are enabled, an output compare interrupt occurs when the ICP1 and ICP0 bits are set. These bits are cleared by writing "0".

Writing "1" has no effect. "1" is always read by a read-modify-write instruction.

0	No compare match (initial value)
1	Compare match

Note:

ICP1: Corresponds to output compare 1.

ICP0: Corresponds to output compare 0.

[bits 5 and 4] ICE1 and ICE0

These bits are used as output compare interrupt enable flags. While the "1" is written to these bits, an output compare interrupt occurs when an interrupt flag (ICP1 or ICP0) is set.

0	Output compare interrupt disabled (initial value)
1	Output compare interrupt enabled

Note:

ICE1: Corresponds to output compare 1.

ICE0: Corresponds to output compare 0.

[bits 3 and 2] Unused bits**[bits 1 and 0] CST1 and CST0**

These bits are used to enable the comparison with 16-bit free-running timer.

0	Compare operation disabled (initial value)
1	Compare operation enabled

Ensure that a value is written to the compare register before the compare operation is enabled.

Note:

CST1: Corresponds to output compare 1.

CST0: Corresponds to output compare 0.

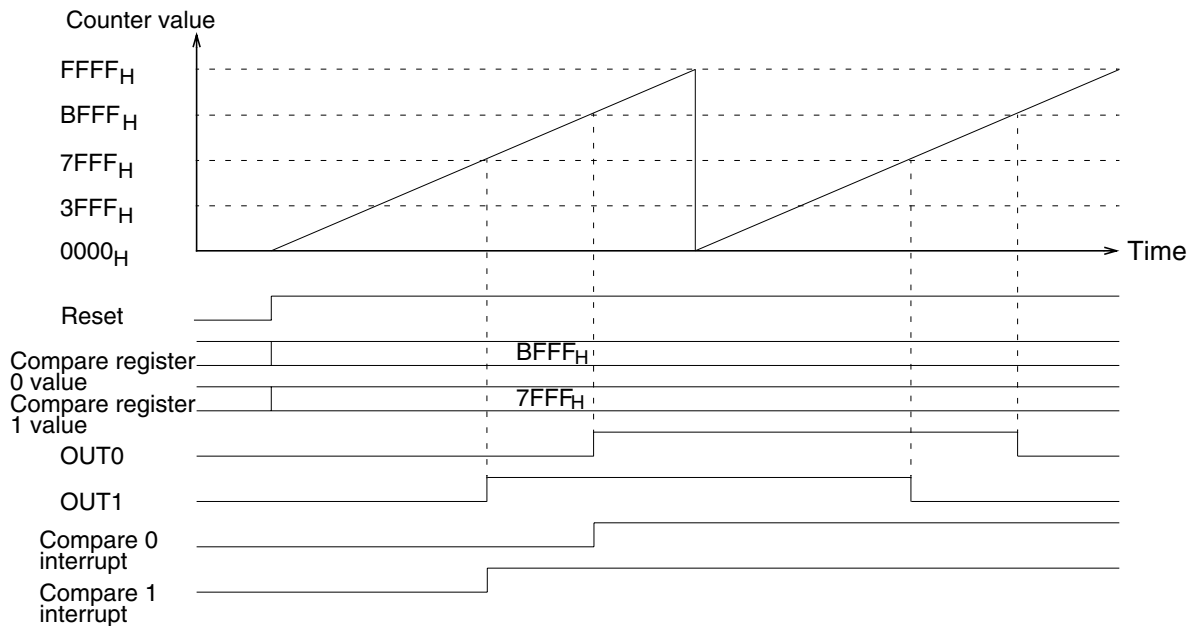
Since output compare is synchronized with the 16-bit free-running timer clock, stopping the 16-bit free-running timer stops compare operation.

11.4.3 16-bit Output Compare Operation

In the 16-bit output compare operation, an interrupt request flag can be set and the output level can be reversed when the specified compare register value matches the 16-bit free-running timer value.

■ Sample of Output Waveform when Compare Registers 0 and 1 are Used (The Initial Output Value is 0.)

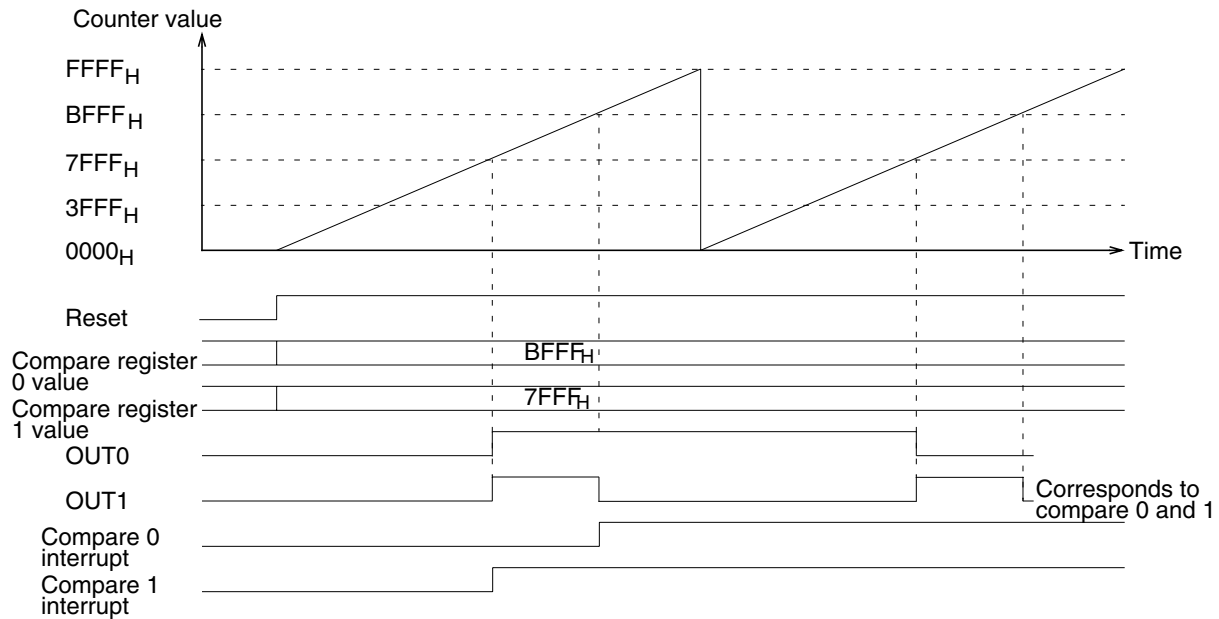
Figure 11.4-2 Sample of Output Waveform when Compare Registers 0 and 1 are Used



The output level can be changed using two compare registers (when CMOD=1).

■ Sample of a Output Waveform with Two Compare Registers (The Initial Output Value is '0')

Figure 11.4-3 Sample of a Output Waveform with Two Compare Registers (The Initial Output Value is '0')

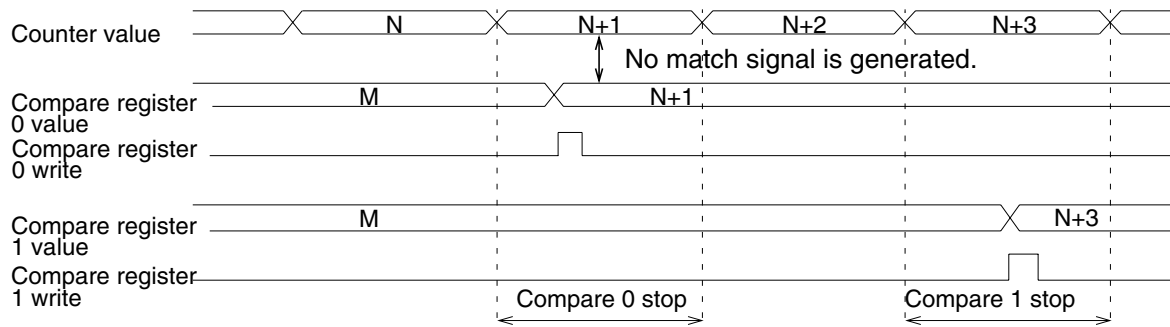


■ Output Compare Timing

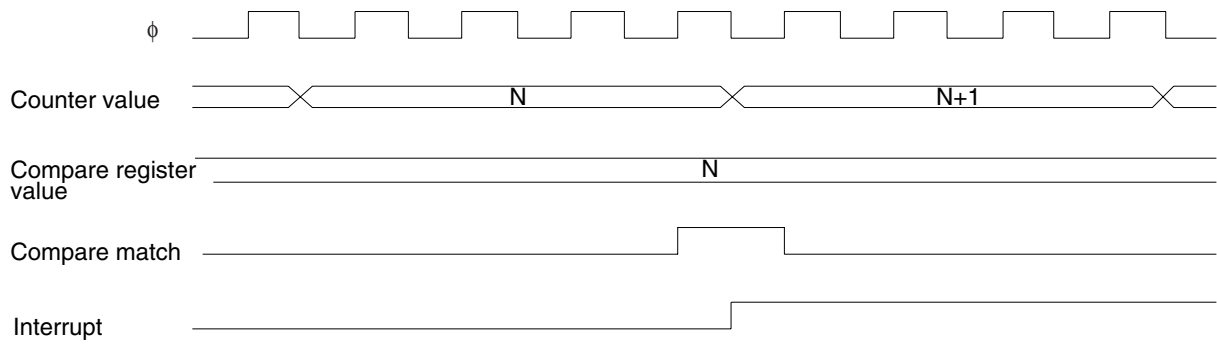
In output compare operation, a compare match signal is generated when the free-running timer value matches the specified compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing upon a compare match is synchronized with the counter count timing.

○ Compare operation upon update of compare register

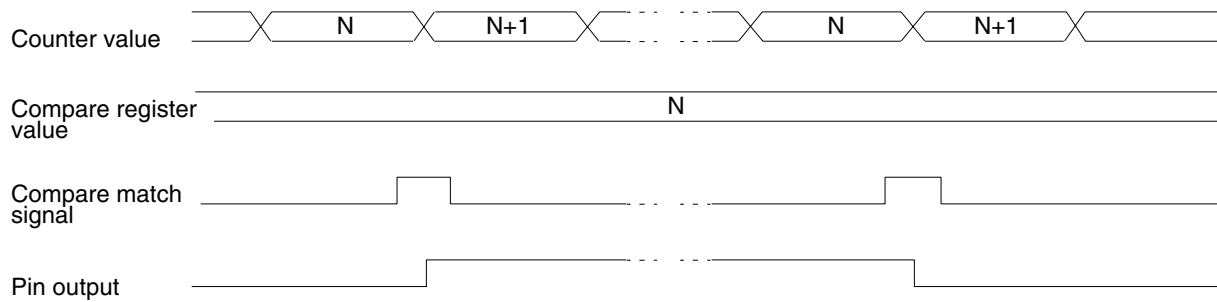
When the compare register is updated, comparison with the counter value is not performed.



○ **Interrupt timing**



○ **Output pin change timing**



11.5 Input Capture

Input capture detects a rising or falling edge or both edges of an external input signal and stores a 16-bit free-running timer value at that time in a register. In addition, input capture can generate an interrupt upon detection of an edge. Input capture consists of an input capture data register and a control register.

■ Input Capture

Each input capture has a corresponding external input pin.

- The valid edge of an external input can be selected from the following three types:

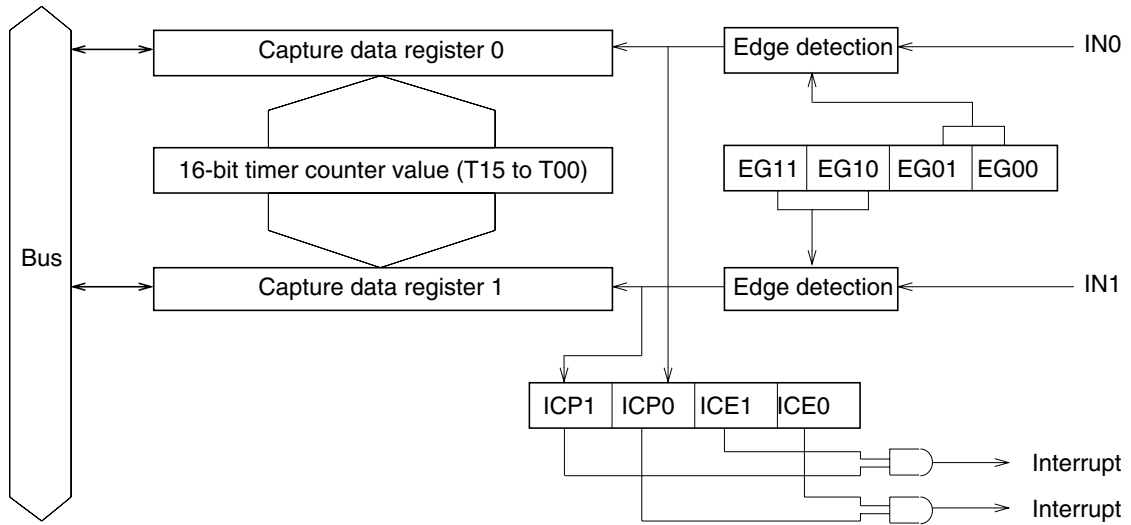
Rising edge	↑
Falling edge	↓
Both edges	↑ ↓

- An interrupt can be generated upon detection of a valid edge of an external input.

■ Input Capture Block Diagram

Figure 11.5-1 "Input Capture Block Diagram" shows a block diagram of input capture.

Figure 11.5-1 Input Capture Block Diagram



11.5.1 Input Capture Register Details

Input capture has the two registers listed. These registers store a value from the 16-bit free running timer when a valid edge of the corresponding external pin input waveform is detected. (The registers must be accessed in word mode. No values can be written to the registers.)

- Input capture data register
- Input capture control register

■ Input Capture Data Register

bit		15	14	13	12	11	10	9	8		
001921 _H		CP15	CP14	CP13	CP12	CP12	CP11	CP09	CP08		
001923 _H		R	R	R	R	R	R	R	R	←Attribute	
		X	X	X	X	X	X	X	X	←Initial value	
bit		7	6	5	4	3	2	1	0		
001920 _H		CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	IPCP0/1	
001922 _H		R	R	R	R	R	R	R	R	←Attribute	
		X	X	X	X	X	X	X	X	←Initial value	

■ Control Status Register

bit		7	6	5	4	3	2	1	0		
000054 _H		ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	ICS01	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	←Attribute	
		0	0	0	0	0	0	0	0	←Initial value	

[bits 7 and 6] ICP1 and ICP0

These bits are used as input capture interrupt flags. "1" is set to this bit upon detection of a valid edge of an external input pin. While the interrupt enable bits (ICE0 and ICE1) are set, an interrupt can be generated upon detection of a valid edge.

These bits are cleared by writing "0". Writing "1" has no effect. "1" is always read by a read-modify-write instruction.

0	No valid edge detection (initial value)
1	Valid edge detection

Note:

ICP0: Corresponds to input capture 0.

ICP1: Corresponds to input capture 1.

[bits 5 and 4] ICE1 and ICE0

These bits are used to enable input capture interrupts. While these bits are "1", an input capture interrupt is generated when the interrupt flag (ICP0 or ICP1) is set.

0	Interrupt disabled (initial value)
1	Interrupt enabled

Note:

ICE0: Corresponds to input capture 0.

ICE1: Corresponds to input capture 1.

[bits 3, 2, 1, and 0] EG11, EG10, EG01, and EG00

These bits are used to specify the valid edge polarity of the external inputs. These bits are also used to enable input capture operation.

EG11 EG01	EG10 EG00	Edge detection polarity
0	0	No edge detection (stop) (initial value)
0	1	Rising edge detection ↑
1	0	Falling edge detection ↓
1	1	Both edge detection ↑ ↓

Note:

EG01 and EG00: Correspond to input capture 0.

EG11 and EG10: Correspond to input capture 1.

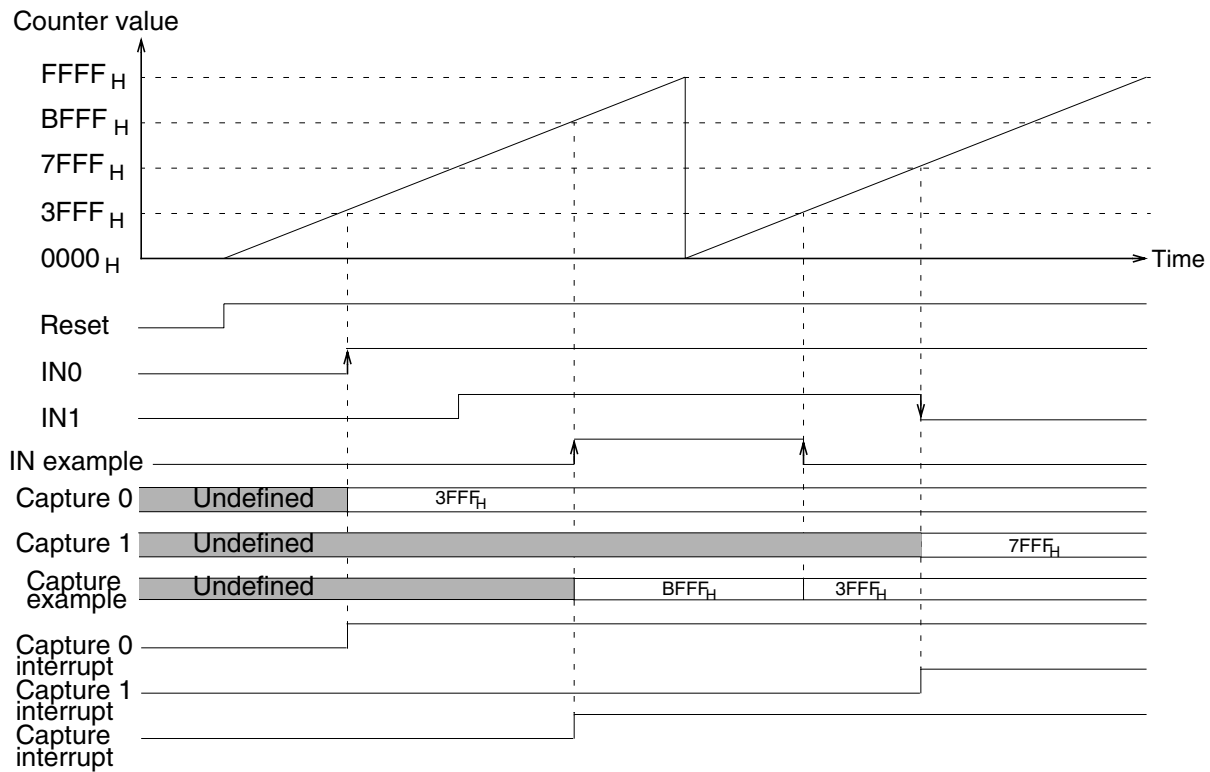
11.5.2 16-bit Input Capture Operation

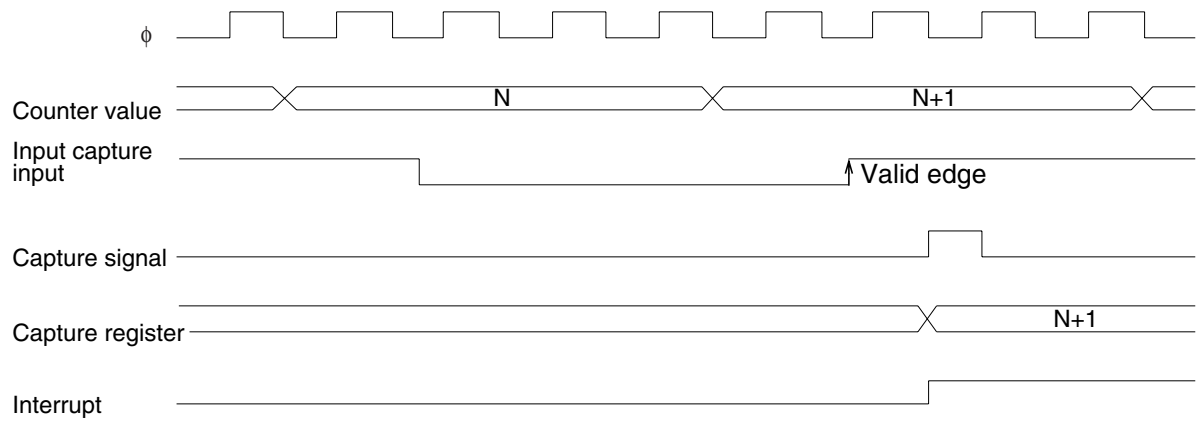
In 16-bit input capture operation, an interrupt can be generated upon detection of at the specified edge, fetching the 16-bit free-running timer value and writing it to the capture register.

■ Sample of Input Capture Fetch Timing

- Capture 0: Rising edge
- Capture 1: Falling edge
- Capture example: Both edges

Figure 11.5-2 Sample of Input Capture Fetch Timing



■ Input Capture Input Timing**○ Capture timing for input signals**

CHAPTER 12 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)

This chapter explains the functions and operations of the 16-bit reload timer (with the event count function).

- 12.1 "Outline of 16-bit Reload Timer (with Event Count Function)"
- 12.2 "16-bit Reload Timer (with Event Count Function)"
- 12.3 "Internal Clock and External Clock Operations of 16-bit Reload Timer"
- 12.4 "Underflow Operation of 16-bit Reload Timer"
- 12.5 "Output Pin Functions of 16-bit Reload Timer"
- 12.6 "Counter Operation State"

12.1 Outline of 16-Bit Reload Timer (with Event Count Function)

The 16-bit reload timer consists of a 16-bit down-counter, a 16-bit reload register, one input pin (TIN) and one output pin (TOT), and a control register. The input clock can be selected from one external clock and three types of internal clock.

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

The output pin (TOT) outputs a toggle output waveform in reload mode and outputs a square waveform indicating counting in one-shot mode. The input pin (TIN) is used for event input in event count mode, and can be used for trigger input or gate input in internal clock mode.

The MB90590 Series has two 16-bit reload timers. However the TIN input and TOT output external pins are shared between the two timers.

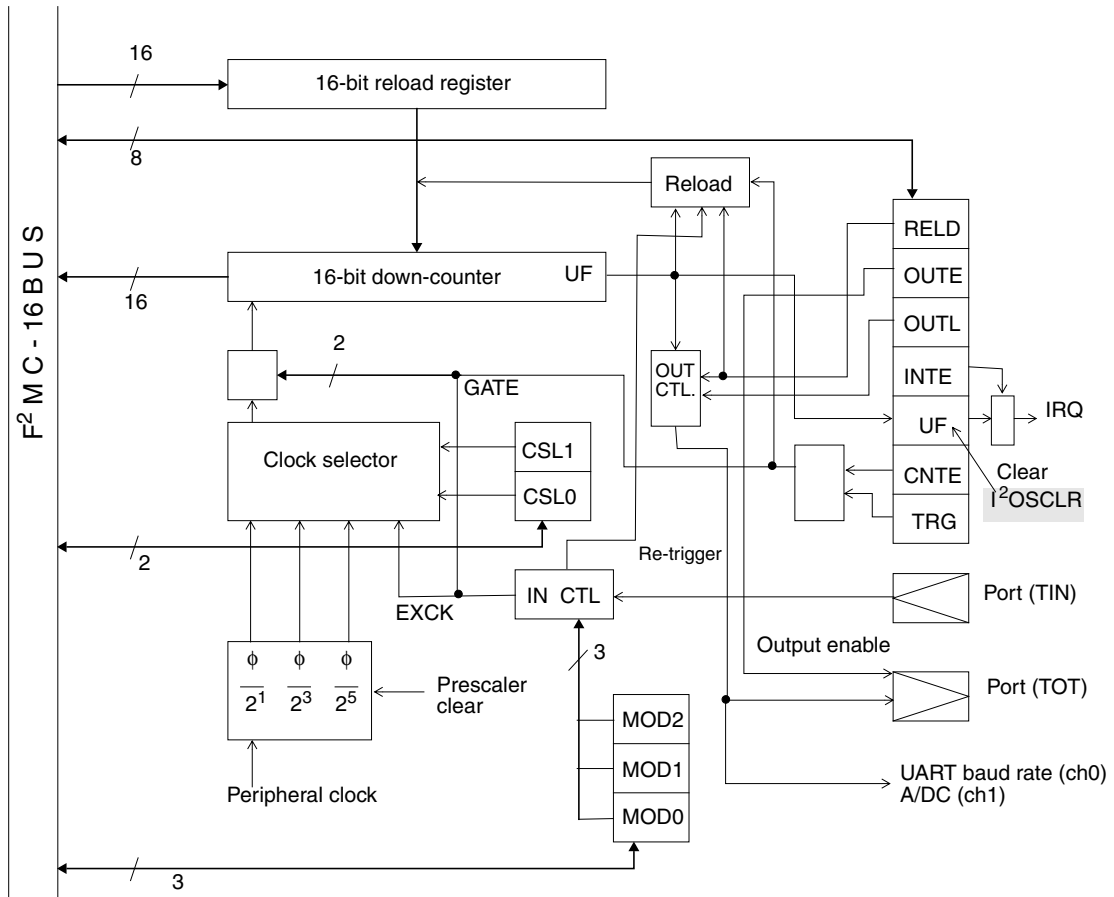
■ Intelligent I/O Service (EI²OS) Function and Interrupts

The timer includes a circuit that supports EI²OS. The timer can activate EI²OS when an underflow occurs. EI²OS can be used with both timers on this product. However, as both timers (ch0 and ch1) are connected to the same interrupt control register (ICRx) in the interrupt controller, ch0 and ch1 cannot be assigned to different EI²OS services. Also, as the two timers have different interrupt vectors, they can be assigned to two different interrupt services. However, as ch0 and ch1 share an interrupt control register as described above, the same interrupt level applies to both channels.

■ Block Diagram of 16-bit Reload Timer

Figure 12.1-1 "Block Diagram of 16-bit Reload Timer" shows a block diagram of the 16-bit reload timer.

Figure 12.1-1 Block Diagram of 16-bit Reload Timer

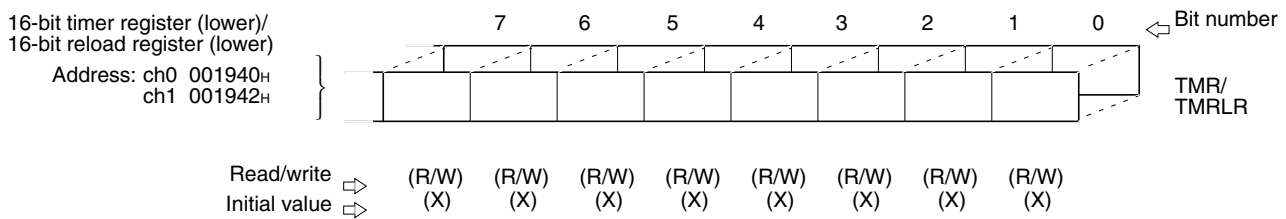
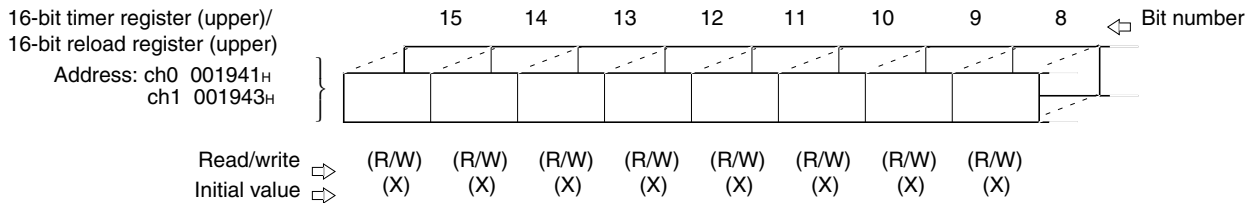
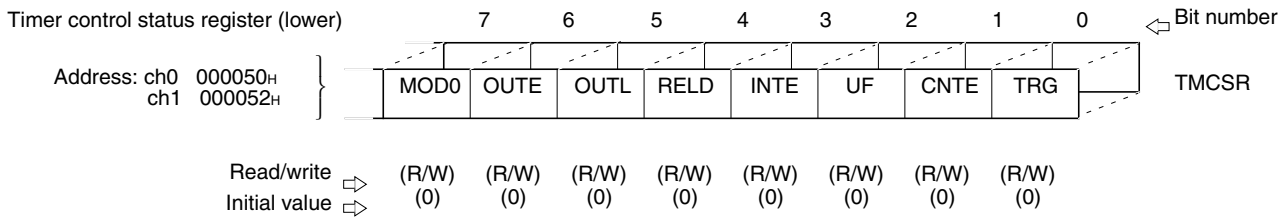
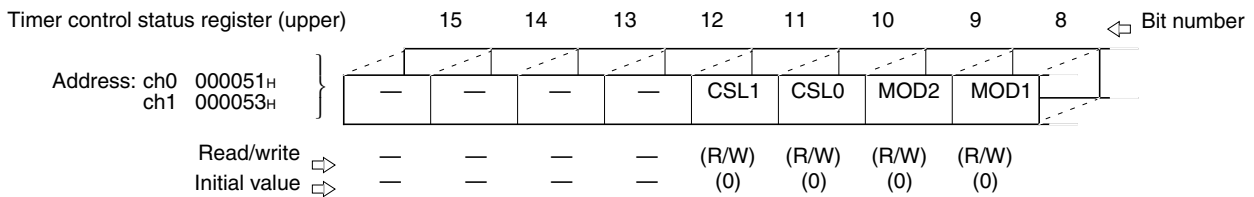


12.2 16-Bit Reload Timer (with Event Count Function)

The 16-bit reload timer has the following two types of registers:

- Timer control register (TMCSR)
- 16-bit timer register (TMR)/16-bit reload register (TMRLR)

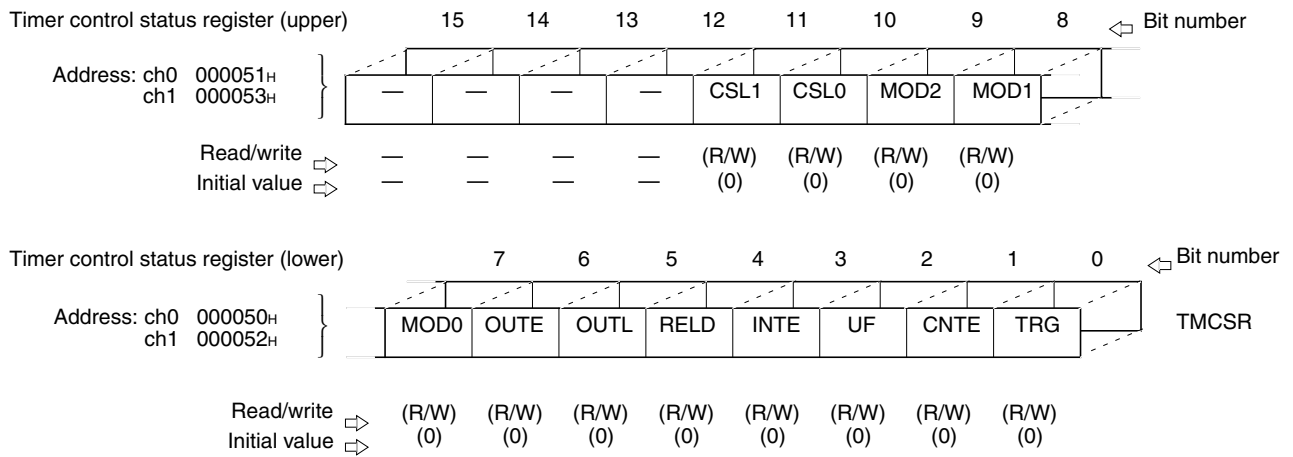
■ 16-bit Reload Timer Register



12.2.1 Timer Control Status Register (TMCSR)

Controls the operation mode and interrupts for the 16-bit timer. Only modify bits other than UF, CNTE, and TRG when CNTE = "0".

Register Layout of Timer Control Register (TMCSR)



Register Contents of Timer Control Register (TMCSR)

[Bits 11, 10] CSL1, CSL0 (Clock select 1, 0)

The count clock select bits. Table 12.2-1 "Clock Sources for CSL Bit Settings" lists the selected clock sources.

Table 12.2-1 Clock Sources for CSL Bit Settings

CSL1	CSL0	Clock Source (Machine cycle $\phi = 16$ MHz)
0	0	$\phi/2^1$ (0.125 μ s)
0	1	$\phi/2^3$ (0.5 μ s)
1	0	$\phi/2^5$ (2.0 μ s)
1	1	External event count mode

[Bits 9, 8, 7] MOD2, MOD1, MOD0

These bits set the operation mode and I/O pin functions.

The MOD2 bit selects the I/O functions. When MOD2 = "0", the input pin functions as a trigger input. In this case, the reload register contents is loaded to the counter when an active edge is input to the input pin and count operation proceeds. When MOD2 = "1", the timer operates in gate counter mode and the input pin functions as a gate input. In this mode, the counter only counts while an active level is input to the input pin.

The MOD1 and 0 bits set the pin functions for each mode. Table 12.2-2 "MOD2, 1, 0 Bit Settings (1)" and Table 12.2-3 "MOD2, 1, 0 Bit Settings (2)" list the MOD2, 1, 0 bit settings.

Table 12.2-2 MOD2, 1, 0 Bit Settings (1)

MOD2	MOD1	MOD0	Input Pin Function	Active Edge or Level
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

Internal clock mode (CSL0, 1 = "00", "01", or "10")

Table 12.2-3 MOD2, 1, 0 Bit Settings (2)

MOD2	MOD1	MOD0	Input Pin Function	Active Edge or Level
×	0	0	-	-
	0	1	Trigger input	Rising edge
	1	0	↑	Falling edge
	1	1	↑	Both edges

- Event counter mode (CSL0,1 = "11")
- Bits marked as × in the table can be set to any value.

[Bit 6] OUTE

Output enable bit. The TOT pin functions as a general-purpose port when this bit is "0" and as the timer output pin when this bit is "1". In reload mode, the output waveform toggles. In one-shot mode, TOT outputs a square waveform that indicates that counting is in progress.

[Bit 5] OUTL

This bit sets the output level for the TOT pin.

Table 12.2-4 OUTE, RELD, and OUTL Settings

OUTE	RELD	OUTL	Output Waveform
0	×	×	General-purpose port
1	0	0	Output an "H" level square waveform during counting.
1	0	1	Output an "L" level square waveform during counting.
1	1	0	Toggle output. Starts with "L" level output.
1	1	1	Toggle output. Starts with "H" level output.

[Bit 4] RELD (Reload)

This bit enables reload operations. When RELD is "1", the timer operates in reload mode. In this mode, the timer loads the reload register contents into the counter and continues counting whenever an underflow occurs (when the counter value changes from 0000_H to FFFF_H). When RELD is "0", the timer operates in one-shot mode. In this mode, the count operation stops when an underflow occurs due to the counter value changing from 0000_H to FFFF_H.

[Bit 3] INTE (Interrupt enable)

Timer interrupt request enable bit. When INTE is "1", an interrupt request is generated when the UF bit changes to "1". When INTE is "0", no interrupt request is generated, even when the UF bit changes to "1".

[Bit 2] UF (Underflow)

Timer interrupt request flag. UF is set to "1" when an underflow occurs (when the counter value changes from 0000_H to FFFF_H). Cleared by writing "0" or by the intelligent I/O service. Writing "1" to this bit has no meaning. Read as "1" by read-modify-write instructions.

[Bit 1] CNTE (Count enable)

Timer count enable bit. Writing "1" to CNTE sets the timer to wait for a trigger. Writing "0" stops count operation.

[Bit 0] TRG (Trigger)

Software trigger bit. Writing "1" to TRG applies a software trigger, causing the timer to load the reload register contents to the counter and start counting. Writing "0" has no meaning. Reading always returns "0". Applying a trigger using this register is only valid when CNTE = "1". Writing "1" has no effect if CNTE = "0".

12.2.2 Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)

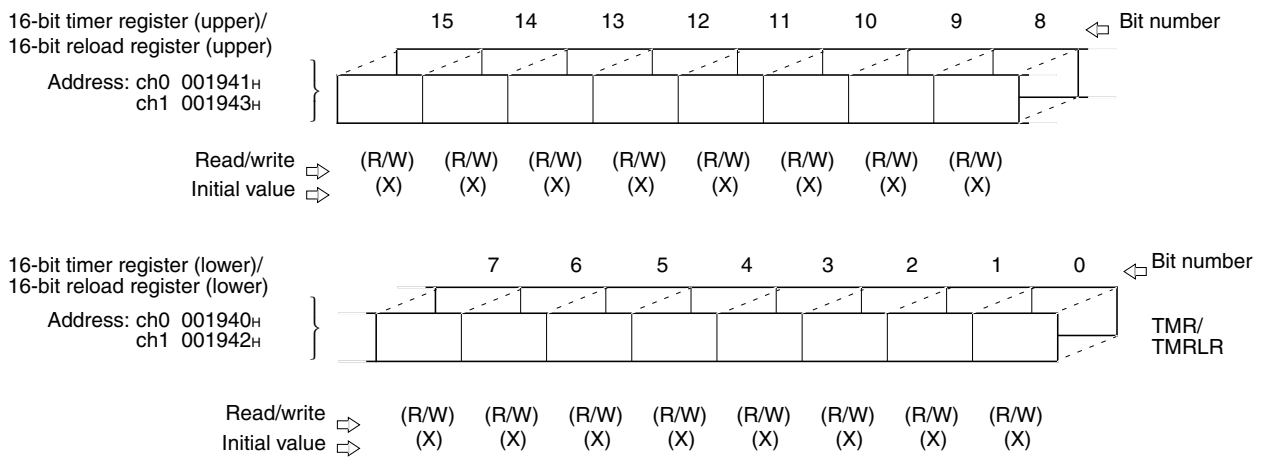
TMR contents (for reading)

Reading this register reads the count value of the 16-bit timer. The initial value is undefined. Always read this register using the word access instructions.

TMRLR contents (for writing)

The 16-bit reload register holds the initial count value. The initial value is undefined. Always write to this register using the word access instructions.

■ Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)



12.3 Internal Clock and External Clock Operations of 16-bit Reload Timer

The machine clock divided by 2^1 , 2^3 , or 2^5 can be selected as the clock sources for operating the timer from an internal divide clock. The external input pin can be selected as either a trigger input or gate input by a register setting.

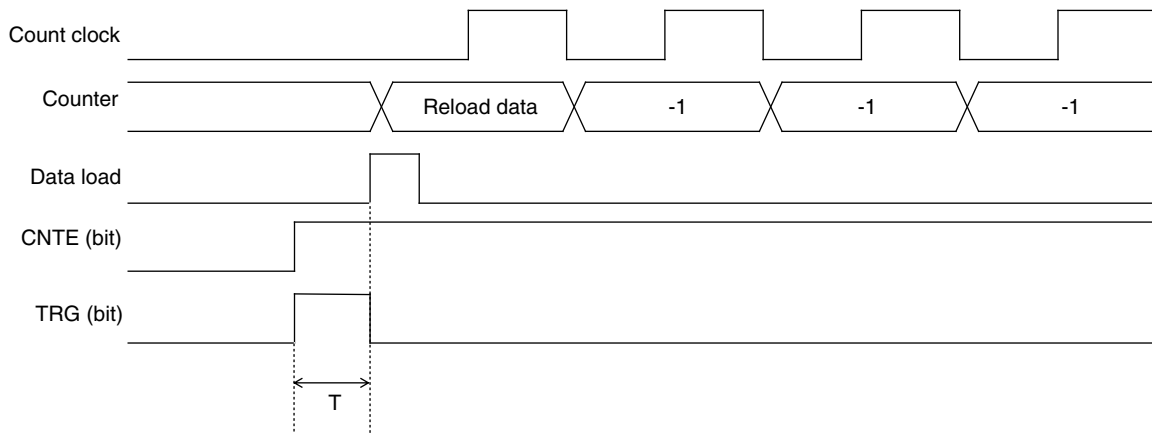
If an external clock is selected, the TIN pin functions as an external event input pin to count the number of valid edges set in the register.

■ Internal Clock Operation of 16-bit Reload Timer

Writing "1" to both the CNTE and TRG bits in the control register enables and starts counting at one time. Using the TRG bit as a trigger input is always available when the timer is enabled (CNTE = "1"), regardless of the operation mode.

Figure 12.3-1 "Activation and Operation of 16-bit Reload Timer Counter" shows counter activation and counter operation. A time period T (T: machine cycle) is required from the counter start trigger being input until the reload register data is loaded into counter.

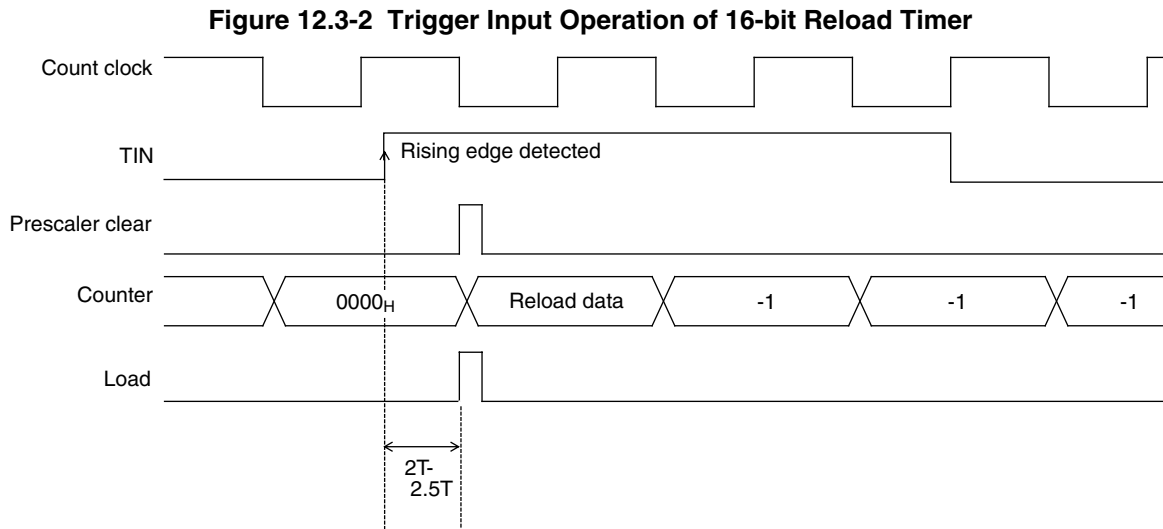
Figure 12.3-1 Activation and Operation of 16-bit Reload Timer Counter



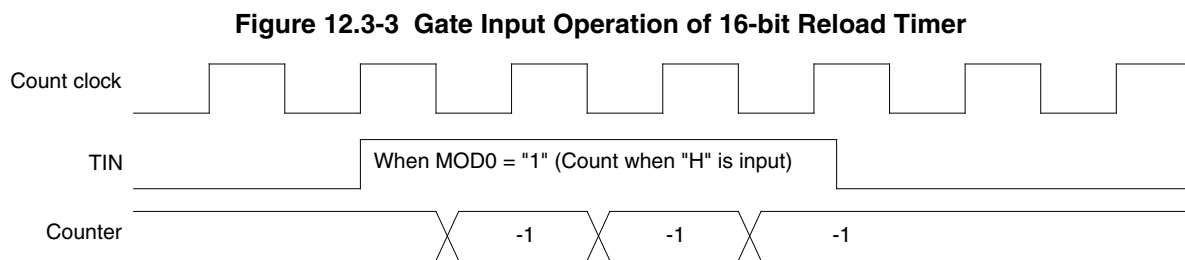
■ Input Pin Functions of 16-bit Reload Timer (in Internal Clock Mode)

The TIN pin can be used as either a trigger input or a gate input when an internal clock is selected as the clock source. When used as a trigger input, input of an active edge causes the timer to load the reload register contents to the counter and then start count operation after clearing the internal prescaler. Input a pulse width of at least $2T$ (T is the machine cycle) to TIN.

Figure 12.3-2 "Trigger Input Operation of 16-bit Reload Timer" shows the operation of trigger input.



When used as a gate input, the counter only counts while the active level specified by the MOD0 bit of the control register is input to the TIN pin. In this case, the count clock continues to operate unless stopped. The software trigger can be used in gate mode, regardless of the gate level. Input a pulse width of at least $2T$ (T is the machine cycle) to the TIN pin. Figure 12.3-3 "Gate Input Operation of 16-bit Reload Timer" shows the operation of gate input.



■ External Event Counter

The TIN pin functions as an external event input pin when an external clock is selected. The counter counts on the active edge specified in the register. Input a pulse width of at least $4T$ (T is the machine cycle) to the TIN pin.

12.4 Underflow Operation of 16-bit Reload Timer

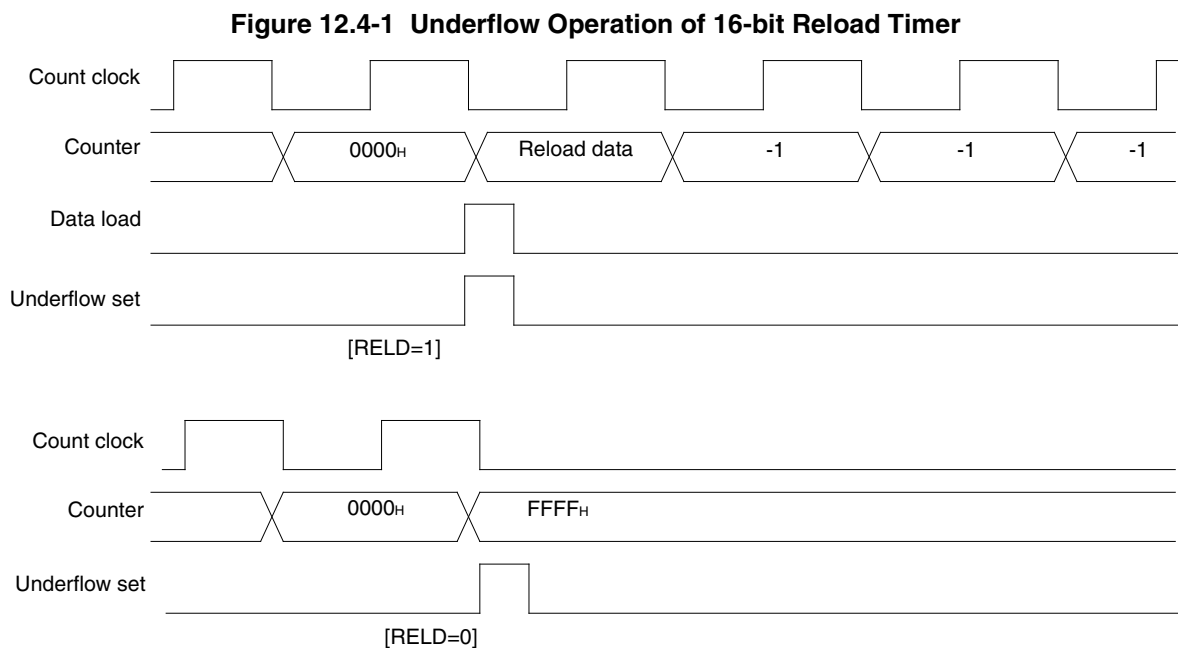
An underflow is defined for this timer as the time when the counter value changes from 0000_H to FFFF_H. Therefore, an underflow occurs after (reload register setting + 1) counts.

■ Underflow Operation of 16-bit Reload Timer

If the RELD bit in the control register is "1" when the underflow occurs, the contents of the reload register is loaded into the counter and counting continues. When RELD is "0", counting stops with the counter at FFFF_H.

The UF bit in the control register is set when the underflow occurs. If the INTE bit is "1" at this time, an interrupt request is generated.

Figure 12.4-1 "Underflow Operation of 16-bit Reload Timer" shows the operation when an underflow occurs.



12.5 Output Pin Functions of 16-bit Reload Timer

In reload mode, the TOT pin performs toggle output (inverts at each underflow). In one-shot mode, the TOT pin functions as a pulse output that outputs a particular level while the count is in progress.

■ Output Pin Functions of 16-bit Reload Timer

The OUTL bit of the control register sets the output polarity. When OUTL = "0", the initial value for toggle output is "0" and the one-shot pulse output is "1" while the count is in progress. The output waveforms are opposite when OUTL = "1".

Figure 12.5-1 "Output Pin Function of 16-bit Reload Timer (1)"

and Figure 12.5-2 "Output Pin Function of 16-bit Reload Timer (2)" show the output pin functions.

Figure 12.5-1 Output Pin Function of 16-bit Reload Timer (1)

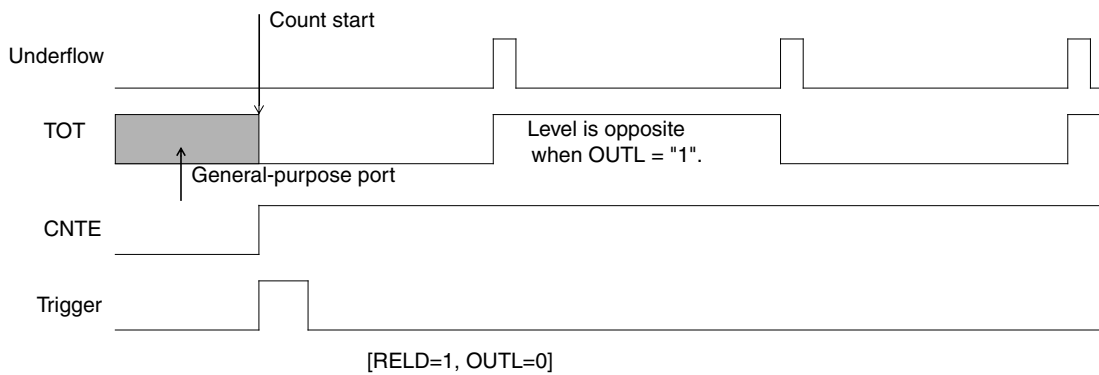
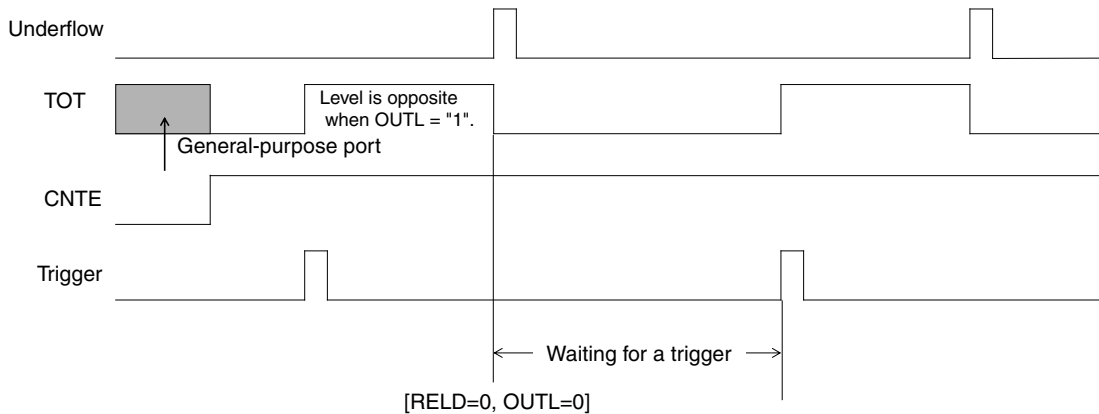


Figure 12.5-2 Output Pin Function of 16-bit Reload Timer (2)



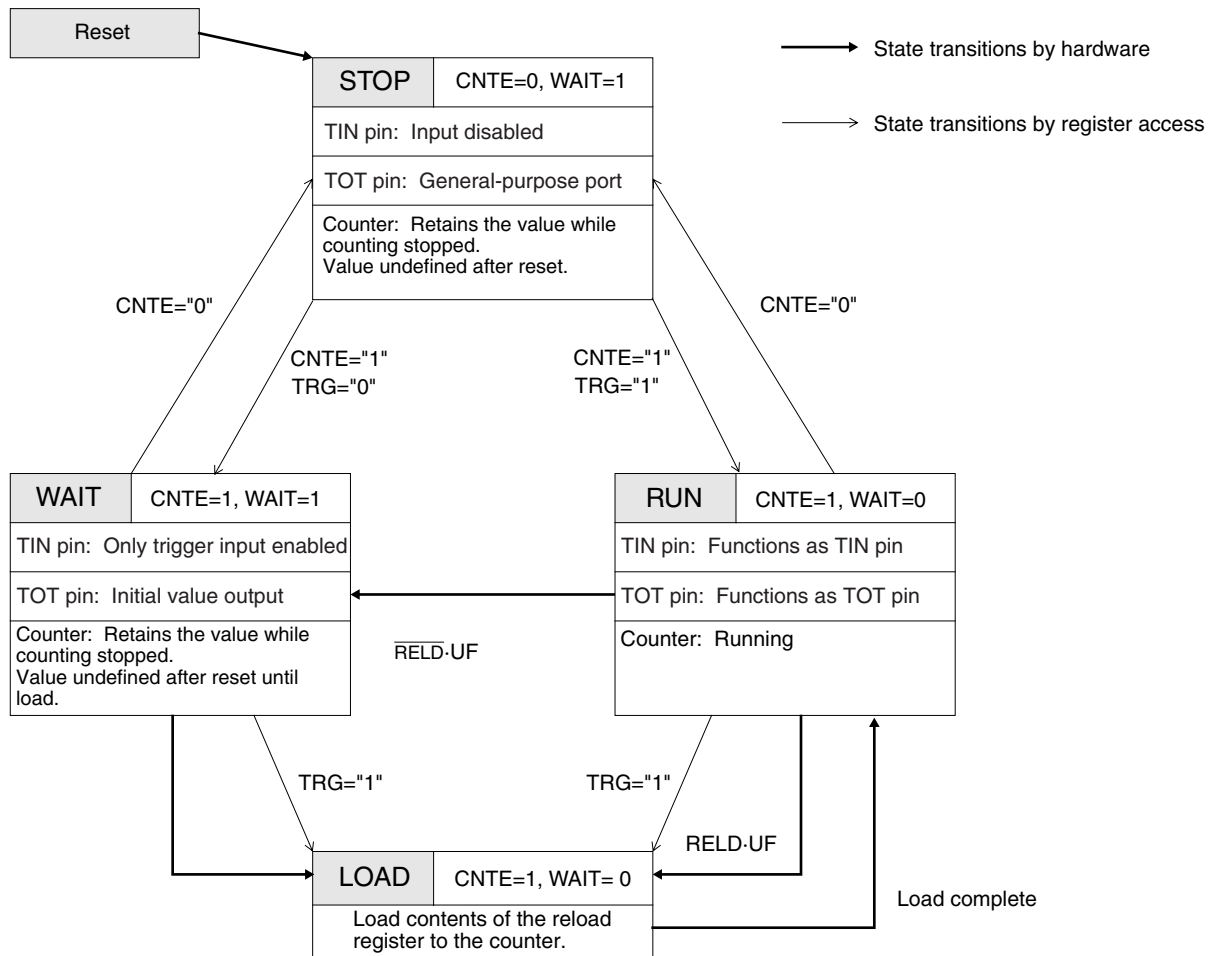
12.6 Counter Operation State

The counter state is determined by the CNTE bit in the control register and the internal WAIT signal. Available states are: CNTE = "0" and WAIT = "1" (STOP state), CNTE = "1" and WAIT = "1" (WAIT state for trigger), and CNTE = "1" and WAIT = "0" (RUN state).

■ Counter Operation State

Figure 12.6-1 "Counter State Transitions" shows the transitions between each state.

Figure 12.6-1 Counter State Transitions



CHAPTER 13 WATCH TIMER

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

13.1 "Outline of Watch Timer"

13.2 "Watch Timer Registers"

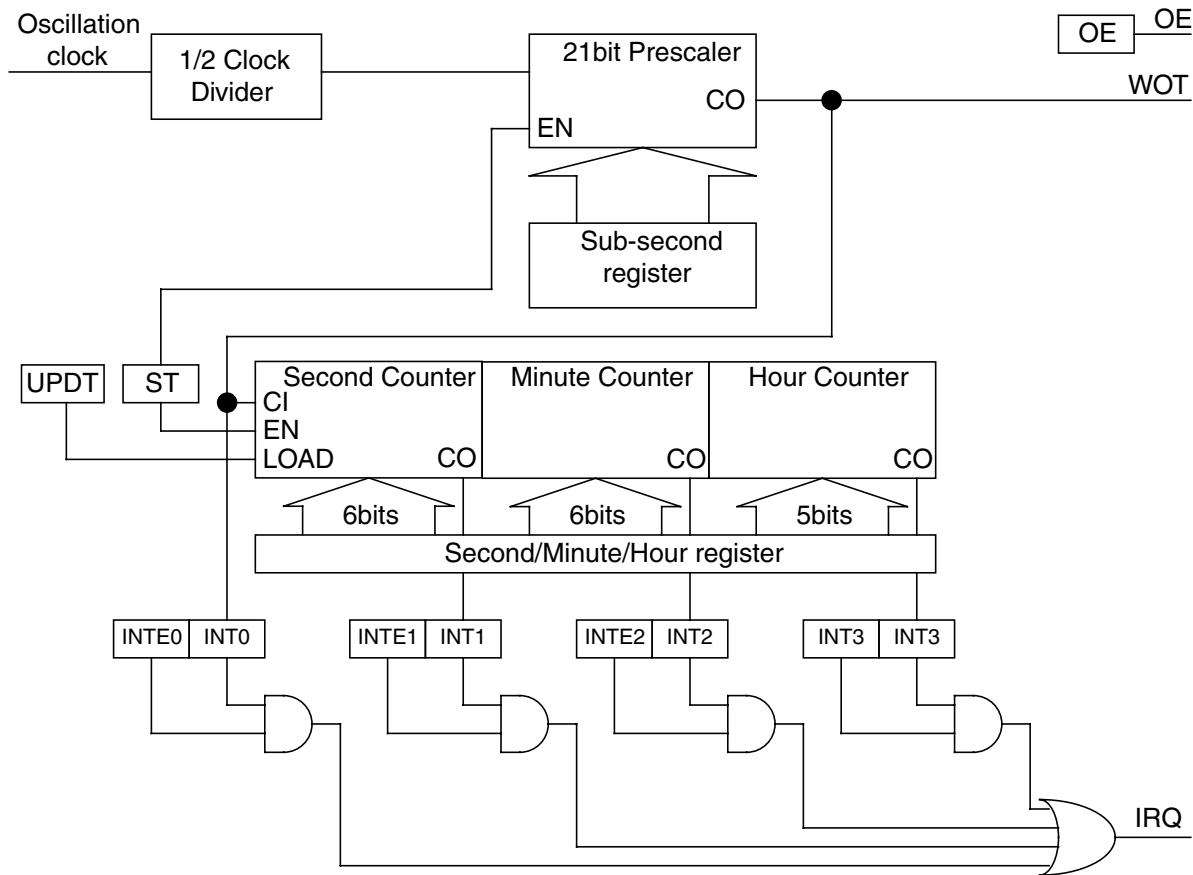
13.1 Outline of Watch Timer

The Watch Timer consists of the Timer Control register, Sub-second register, Second/Minute/Hour registers, 1/2 clock divider, 21-bit prescaler and Second/Minute/Hour counters. The oscillation frequency of the MCU is assumed to be at 4MHz for the aimed operation of the Watch Timer. The Watch Timer operates as the real-world timer and provides the real-world time information.

■ Block Diagram of Watch Timer

Figure 13.1-1 "Block Diagram of Watch Timer" shows a block diagram of the Watch Timer.

Figure 13.1-1 Block Diagram of Watch Timer

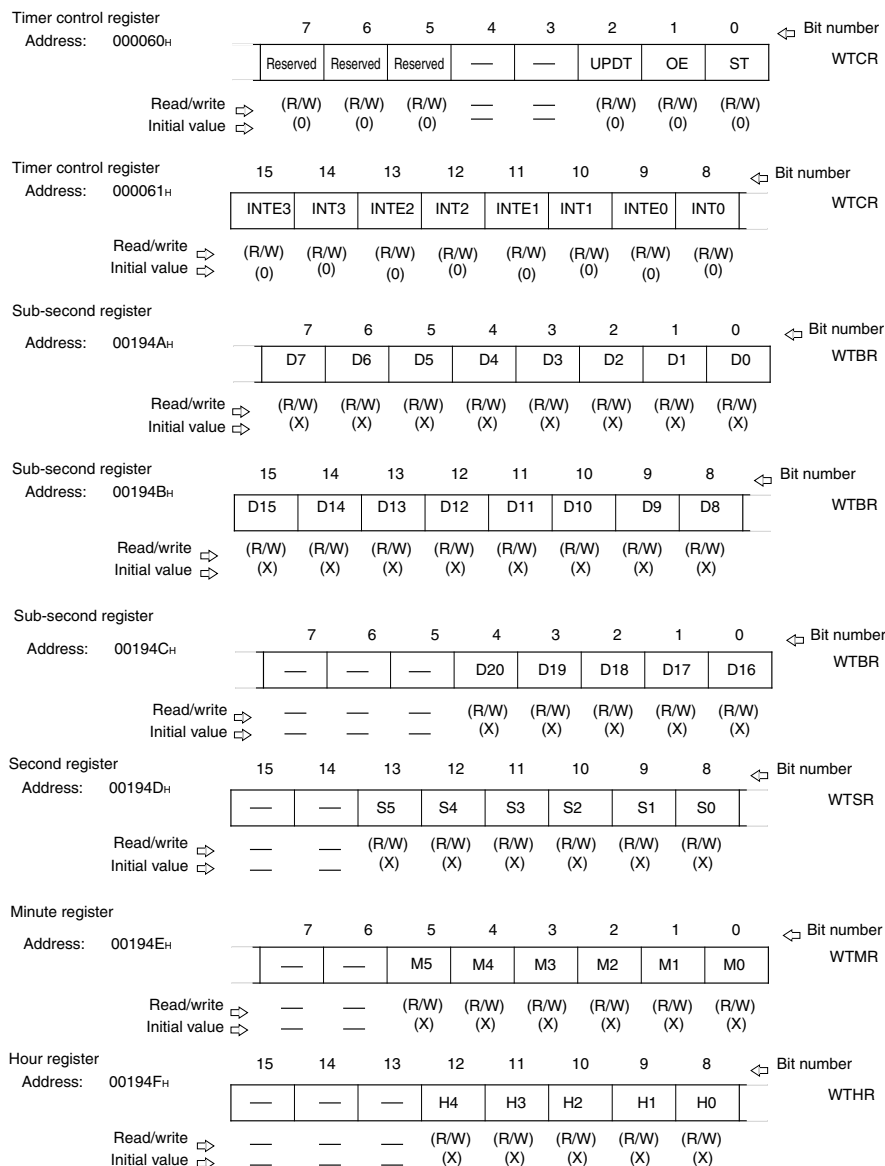


13.2 Watch Timer Registers

The Watch Timer has the following five types of registers:

- Timer control register (WTCR)
- Sub-second register (WTBR)
- Second register (WTSR)
- Minute register (WTMR)
- Hour register (WTHR)

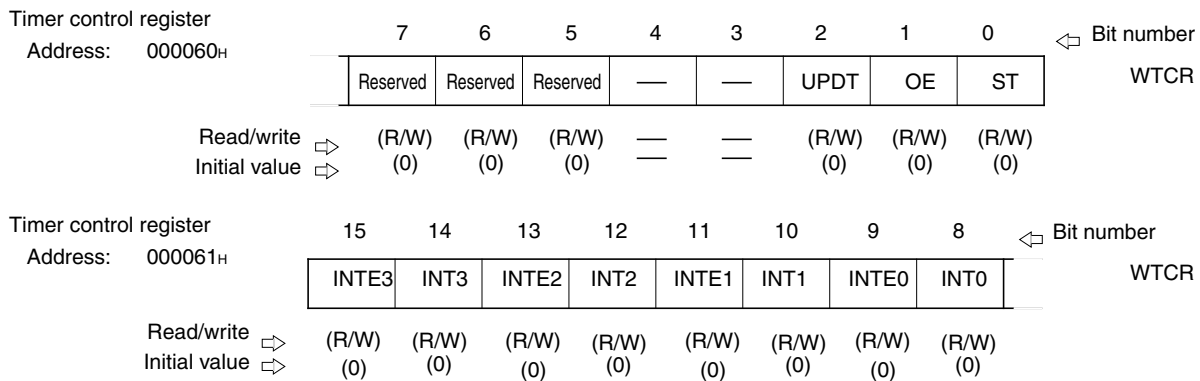
■ Watch Timer Registers



13.2.1 Timer Control Register

The timer control register starts and stops the Watch Timer, controls interrupts, and sets the external output pins.

■ Timer Control Register



[bits 15 to 8] INT3 to 0, INTE3 to 0: Interrupt flags and Interrupt enable flags

INT0 to INT3 are the interrupt flags. They are set when the second counter, minute counter and hour counter overflow respectively. If a INT bit is set while the corresponding INTE bit is "1", the Watch Timer signals an interrupt. These flags are intended to signal an interrupt every second/minute/hour/day.

Writing "0" to the INT bits clears the flags and writing "1" does not have any effect. Any read-modify-write instruction performed on the INT bit results reading "1".

[bits 7 to 5] Reserved bits

These are reserved bits. Always write "0" to these bits.

[bit 2] UPDT: Update bit

The UPDT bit is prepared for modifying the Second/Minute/Hour counter values.

To modify the counter values, write the modified data in the Second/Minute/Hour registers. Then set the UPDT bit to "1". The register values are loaded to the counter at the next CO signal from the 21-bit prescaler. The UPDT bit is reset by the hardware when the counter values are updated. However, if the set operation by software and the reset operation by hardware occur at the same time, the UPDT bit will not be reset.

Writing "0" to the UPDT bit does not have any effect. The result of reading by a read-modify-write instruction is always "0".

Note:

If this bit is set during "59 second", normal up count operation is executed and this bit is reset to "0" without reflecting the Second/Minute/Hour register values.

Writing "0" to the UPDT bit has no effect and a read-modify-write instruction results in reading "0".

[bit 1] OE: Output enable bit

When the OE bit is set to "1", the WOT external pin serves as the output for the Watch Timer. Otherwise it can be used as a general purpose I/O or for another peripheral block.

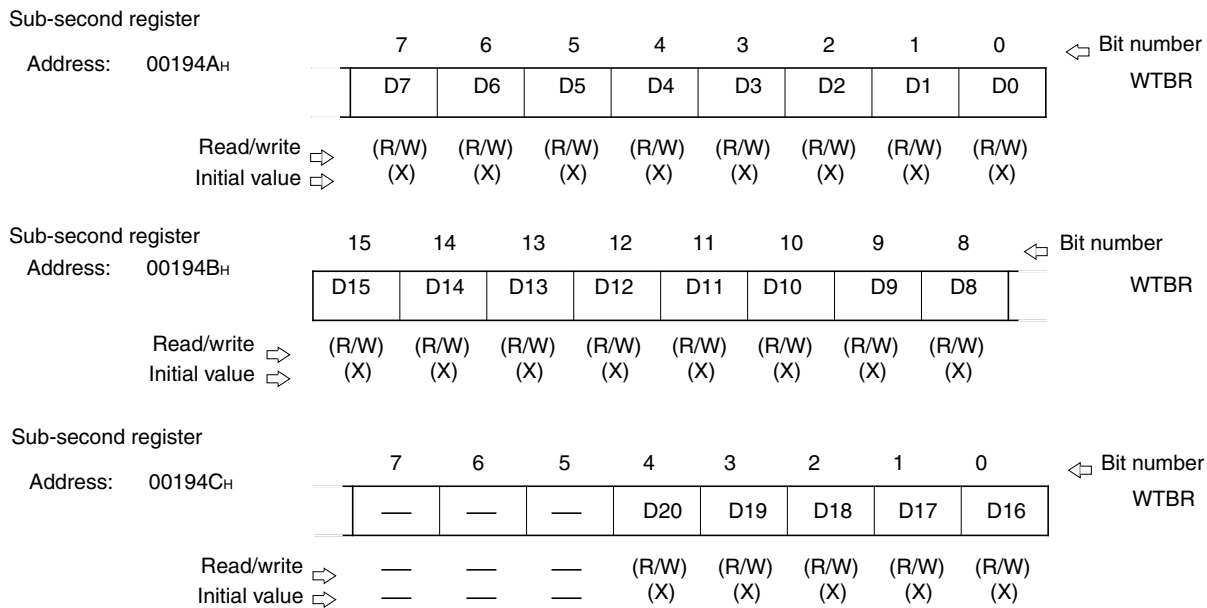
[bit 0] ST: Start bit

When the ST bit is set to "1", the Watch Timer loads Second/Minute/Hour values from the registers and starts its operation. When it is reset to "0", all the counters and the prescalers are reset to "0" and halts.

13.2.2 Sub-second Registers

The sub-second register stores a reload value for the 21-bit prescaler that divides the oscillation clock. The reload value is usually set so that the 21-bit prescaler will output exactly within a one-second cycle. This register is not initialized by reset, but 21-bit prescaler is initialized by reset.

■ Sub-second Register



[bit 20 to 0] D20 to D0

The Sub-second register stores the reload value for the 21-bit prescaler. This value is reloaded after the reload counter reaches "0". Note that when modifying the all three bytes, make sure the reload operation will not be performed in between the write instructions. Otherwise the 21-bit prescaler loads the incorrect value of the combination of new data and old data bytes. It is generally recommended that the Sub-Second register are updated while the ST bit is "0". If the sub-second registers are set to "0", the 21-bit prescaler does not operate at all.

The input clock frequency always equals the oscillation clock frequency and it is intended to be 4MHz. The reload value of the 21-bit prescaler is typically set to Hex1E847F which equals to $2^7 * 5^6 - 1$. Therefore the combination of these two prescalers is intended to provide a clock signal of exact one second.

13.2.3 Second/Minute/Hour Registers

The Second/Minute/Hour registers stores the time information. It is a binary representation of the second, minute and hour.

Reading any of these register always results in the corresponding counter value.

These registers are write associative however, the written data is loaded in the counters after the UPDT bit is set to "1". These registers and counter are initialized by reset.

■ Second/Minute/Hour Registers

Second register		15	14	13	12	11	10	9	8	↩ Bit number
Address: 00194D _H		—	—	S5	S4	S3	S2	S1	S0	WTSR
Read/write ↗	↘	—	—	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↗	↘	—	—	(X)	(X)	(X)	(X)	(X)	(X)	
Minute register		7	6	5	4	3	2	1	0	↩ Bit number
Address: 00194E _H		—	—	M5	M4	M3	M2	M1	M0	WTMR
Read/write ↗	↘	—	—	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↗	↘	—	—	(X)	(X)	(X)	(X)	(X)	(X)	
Hour register		15	14	13	12	11	10	9	8	↩ Bit number
Address: 00194F _H		—	—	—	H4	H3	H2	H1	H0	WTMR
Read/write ↗	↘	—	—	—	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↗	↘	—	—	—	(X)	(X)	(X)	(X)	(X)	

Since there are three byte-registers, make sure the obtained values from the registers are consistent.

i.e. Obtained value of "1 hour, 59 minute, 59 second" could be "0 hour 59 minute, 59 second" or "1 hour, 0 minute, 0 second" or "2 hour, 0 minute, 0 second".

Also when the operation clock of the MCU is the half of the oscillation clock (When the PLL is stopped), the read values from these registers may be corrupt. This is due to the synchronization of the read operation and the count operation. Therefore it is recommended is use a second interrupt to trigger the read instructions.

CHAPTER 14 8/16-BIT PPG

This chapter explains the 8/16-bit PPG and explains its functions.

- 14.1 "Outline of 8/16-bit PPG"
- 14.2 "Block Diagram of 8/16-bit PPG"
- 14.3 "8/16-bit PPG Registers"
- 14.4 "Operations of 8/16-bit PPG"
- 14.5 "Selecting a Count Clock for 8/16-bit PPG"
- 14.6 "Controlling Pin Output of 8/16-bit PPG Pulses"
- 14.7 "8/16-bit PPG Interrupts"
- 14.8 "Initial Values of 8/16-bit PPG Hardware"

14.1 Outline of 8/16-bit PPG

The 8/16-bit Programmable Pulse Generator (PPG) consists of two eight-bit down counters, four eight-bit reload registers, one 16-bit control register, two external pulse output signals, and two interrupt outputs. The following functions are implemented:

■ Function of 8/16-bit PPG

○ **8-bit PPG output, 2-channel independent operation mode:**

Two independent channels of PPG output operation are implemented.

○ **16-bit PPG output operation mode:**

One channel of 16-bit PPG output operation is implemented.

○ **8+8-bit PPG output operation mode:**

8-bit PPG output operation is implemented at specified intervals, using channel 0 output as channel 1 clock input.

○ **PPG output operation:**

Pulse waves are output at specified intervals and duty ratio. With an external circuit, this module can be used as a D/A converter.

The MB90590 Series contains six PPG's. The following sections only describe the functionality of the PPG 0/1. The remaining PPG's have the identical function and the register addresses should be found in the I/O map.

14.2 Block Diagram of 8/16-bit PPG

Figure 14.2-1 "8-bit PPG ch0 Block Diagram" shows a block diagram of the 8/16-bit PPG (ch0). Figure 14.2-2 "8-bit PPG ch1 Block Diagram" shows a block diagram of the 8/16-bit PPG (ch1).

■ Block Diagram of 8/16-bit PPG

Figure 14.2-1 8-bit PPG ch0 Block Diagram

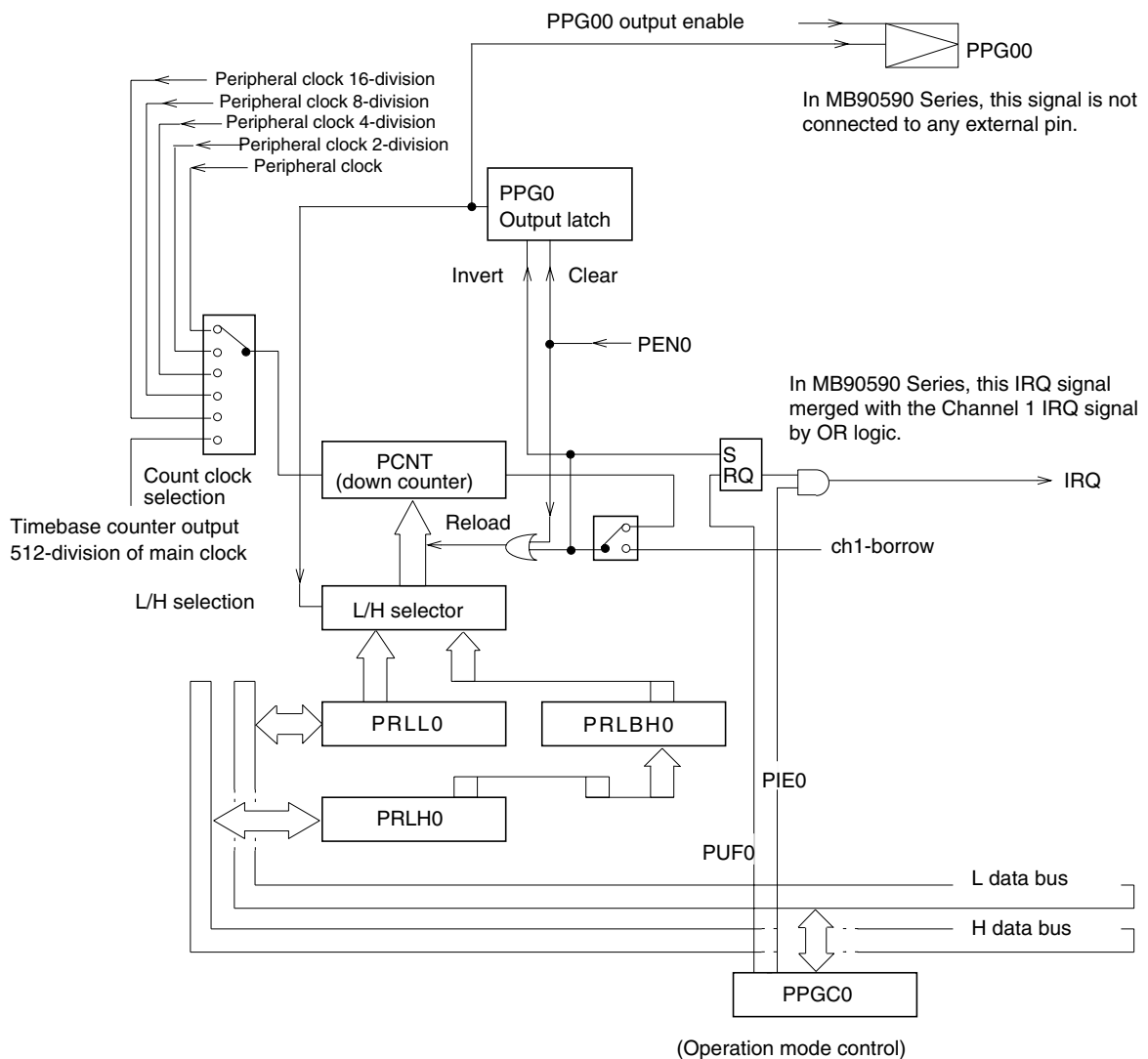
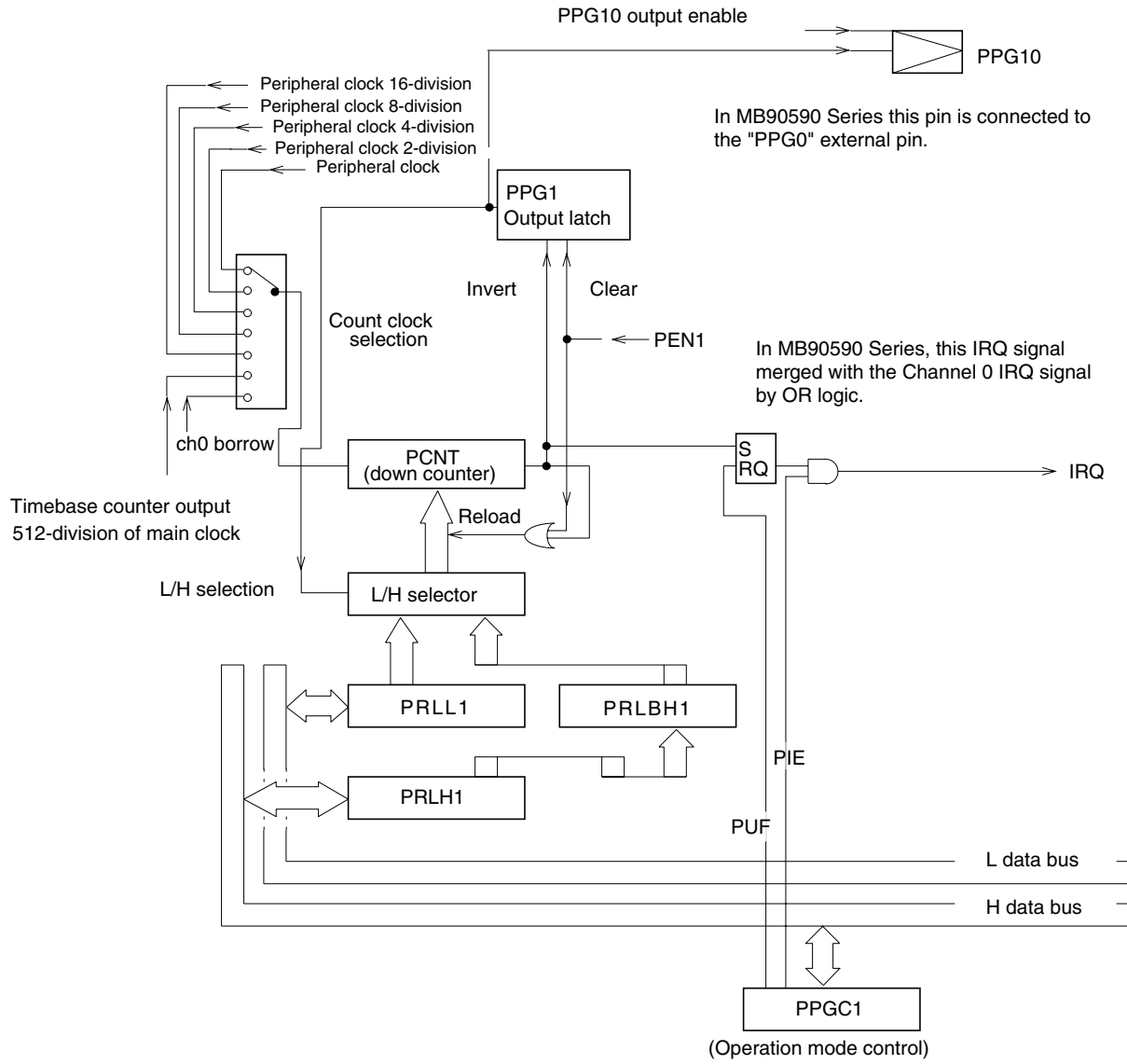


Figure 14.2-2 8-bit PPG ch1 Block Diagram

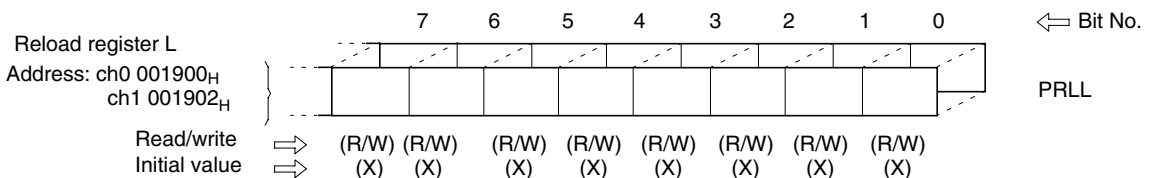
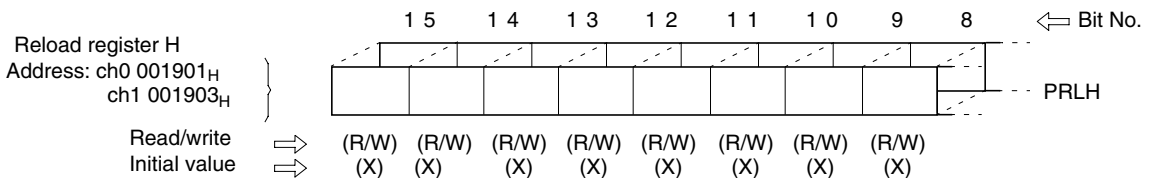
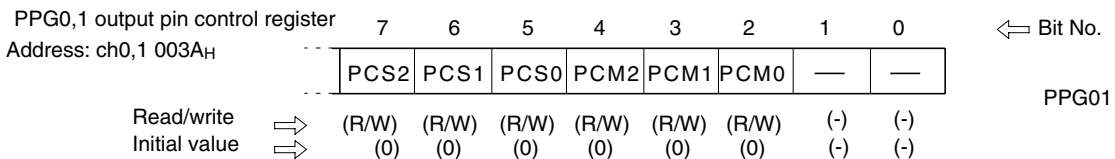
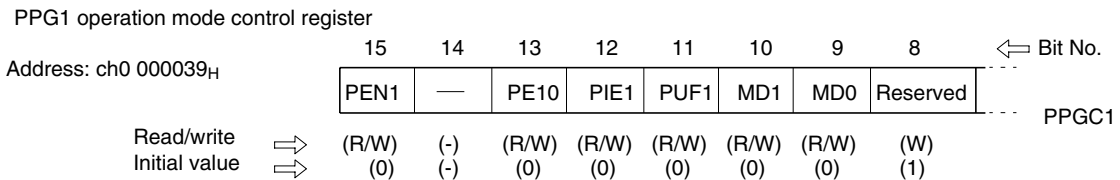
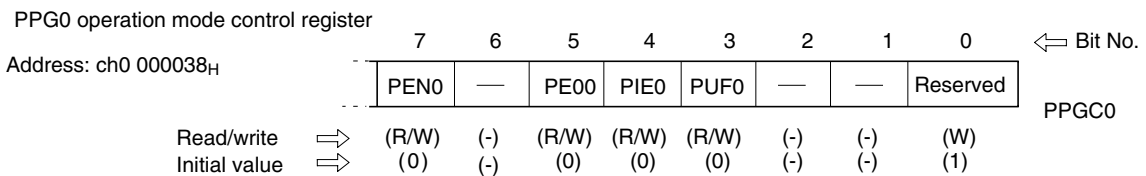


14.3 8/16-bit PPG Registers

The 8/16-bit PPG has the following five types of registers:

- PPG0 operation mode control register
- PPG1 operation mode control register
- PPG0, 1 output pin control register
- Reload register H
- Reload register L

■ 8/16-bit PPG Registers



14.3.1 PPG0 Operation Mode Control Register (PPGC0)

PPGC0 is a five-bit control register that selects the operation mode of the block, controls pin outputs, selects count clock, and controls triggers.

■ PPG0 Operation Mode Control Register (PPGC0)

PPG0 operation mode control register
Address: ch0, 000038_H

	7	6	5	4	3	2	1	0	Bit No.
	PEN0	-	PE00	PIE0	PUF0	-	-	Reserved	PPGC0
Read/write	⇒ (R/W)	(-)	(R/W)	(R/W)	(R/W)	(-)	(-)	(W)	
Initial value	⇒ (0)	(-)	(0)	(0)	(0)	(-)	(-)	(1)	

[bit 7] PEN0 (PPG enable): Operation enable bit

This bit enables the counter operation of the PPG.

PEN0	Operation
0	Stop ("L" level output maintained)
1	PPG operation enabled

Setting this bit to 1 enables the counter operation.

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 5] PE00 (PPG output enable 00): PPG00 pin output enable bit

This bit controls the PPG00 pulse output external pin as described below.

0	General-purpose port pin (pulse output disabled)
1	PPG00 = pulse output pin (pulse output enabled)

This bit is initialized to "0" upon a reset. This bit is readable and writable.

For MB90590 Series, this bit should always be set to "0".

[bit 4] PIE0 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

0	Interrupt disabled
1	Interrupt enabled

While this bit is "1", an interrupt request is issued as soon as PUF0 is set to "1". No interrupt request is issued while this bit is set to "0".

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 3] PUF0 (PPG underflow flag): PPG counter underflow bit

This bit indicates the PPG counter underflow as described below.

0	PPG counter underflow is not detected.
1	PPG counter underflow is detected.

In 8-bit PPG 2-channel mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the ch0 counter value becoming from 00_H to FF_H. In 16-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the Channel 0 and 1 counter value becoming from 0000_H to FFFF_H. To set this bit to "0", write "0". Writing "1" to this bit has not effect. Upon a read operation during a read-modify-write instruction, "1" is read.

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 0]

This is a reserved bit. When setting PPGC0, always set this bit to 1.

14.3.2 PPG1 Operation Mode Control Register (PPGC1)

PPGC0 is a seven-bit control register that selects the operation mode of the block, controls pin outputs, selects count clock, and controls triggers.

■ PPG1 Operation Mode Control Register (PPGC1)

PPG1 operation mode control register Address: ch1 000039 _H	1 5	1 4	1 3	1 2	1 1	1 0	9	8	↔ Bit No.
	PEN1	-	PE10	PIE1	PUF1	MD1	MD0	Reserved	PPGC1
Read/write	⇒ (R/W)	⇒ (-)	⇒ (R/W)	⇒ (R/W)	⇒ (R/W)	⇒ (R/W)	⇒ (R/W)	⇒ (W)	
Initial value	⇒ (0)	⇒ (-)	⇒ (0)	⇒ (0)	⇒ (0)	⇒ (0)	⇒ (0)	⇒ (1)	

[bit 15] PEN1 (PPG enable): Operation enable bit

This bit enables the counter operation of the PPG.

PEN1	Operation
0	Stop ("L" level output maintained)
1	PPG operation enabled

Setting this bit to 1 enables the counter operation.

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 13] PE10 (PPG output enable 10): PPG10 pin output enable bit

This bit controls the PPG10 pulse output external pin as described below.

0	General-purpose port pin (pulse output disabled)
1	PPG10 = pulse output pin (pulse output enabled)

This bit is initialized to "0" upon a reset. This bit is readable and writable.

For MB90590 Series, the pulse signal is output to the "PPG0" external pin.

[bit 12] PIE1 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

0	Interrupt disabled
1	Interrupt enabled

While this bit is "1", an interrupt request is issued as soon as PUF1 is set to "1". No interrupt request is issued while this bit is set to "0".

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 11] PUF1 (PPG underflow flag): PPG counter underflow bit

This bit indicates the PPG counter underflow as described below.

0	PPG counter underflow is not detected.
1	PPG counter underflow is detected.

In 8-bit PPG 2-channel mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the Channel 1 counter value becoming from 00_H to FF_H. In 16-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the Channel 0 and 1 counter value becoming from 0000_H to FFFF_H. To set this bit to "0", write "0". Writing "1" to this bit has not effect. Upon a read operation during a read-modify-write instruction, "1" is read.

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 10, 9] MD1, 0 (PPG count mode): Operation mode selection bit

These bits selects the PPG timer operation mode as described below.

MD1	MD0	Operation mode
0	0	8-bit PPG 2ch independent mode
0	1	8-bit prescaler + 8-bit PPG 1ch mode
1	0	Reserved
1	1	16-bit PPG 1ch mode

These bits are initialized to "00" upon a reset. These bits are readable and writable.

Note:

Do not set "10" in these bits.

To write "01" to these bits, ensure that "01" is not written to the PEN0 bit of PPGC0 or PEN1 bit of PPGC1. Write "11" or "00" in both the PEN0 and PEN1 bits simultaneously.

To write "11" to these bits, update PPGC0 and PPGC1 by word transfer and write "11" or "00" to both the PEN0 and PEN1 bits simultaneously.

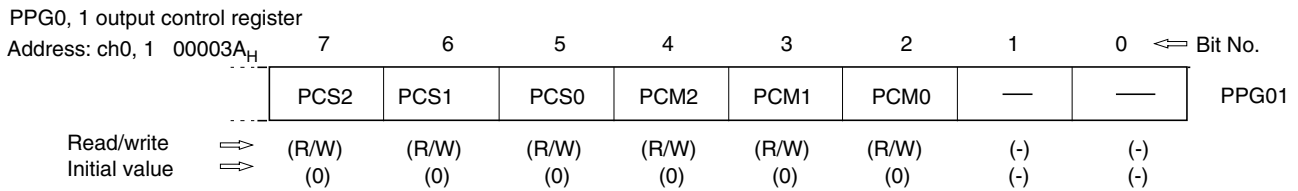
[bit 8]

This is a reserved bit. When setting PPGC1, always write "1" to this bit.

14.3.3 PPG0, 1 Output Control Register (PPG01)

The PPG0, 1 output control register (PPG01) is an 8-bit control register that controls the pin output of the 8/16-bit PPG.

■ PPG0, 1 Clock Select Register (PPG01)



[bits 7 to 5] PCS2 to 0 (PPG count select): Count clock selection bit

These bits select the operation clock for the down counter of Channel 1 as described below.

PCS2	PCS1	PCS0	Operation mode
0	0	0	Peripheral clock (62.5 ns machine clock, 16 MHz)
0	0	1	Peripheral clock/2 (125 ns machine clock, 16 MHz)
0	1	0	Peripheral clock/4 (250 ns machine clock, 16 MHz)
0	1	1	Peripheral clock/8 (500 ns machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1 μs machine clock, 16 MHz)
1	0	1	Clock input from the timebase timer (128 μs, 4 MHz source oscillation)

These bits are initialized to "000" upon a reset. These bits are readable and writable.

Note:

In 8-bit prescaler + 8-bit PPG mode or in 16-bit PPG mode, ch1 PPG operates in response to a counter clock from ch0. Therefore, the setting in these bits has no effect.

[bits 4 to 2] PCM2 to 0 (PPG count mode): Count clock selection bit

These bits select the operation clock for the down counter of Channel 0 as described below.

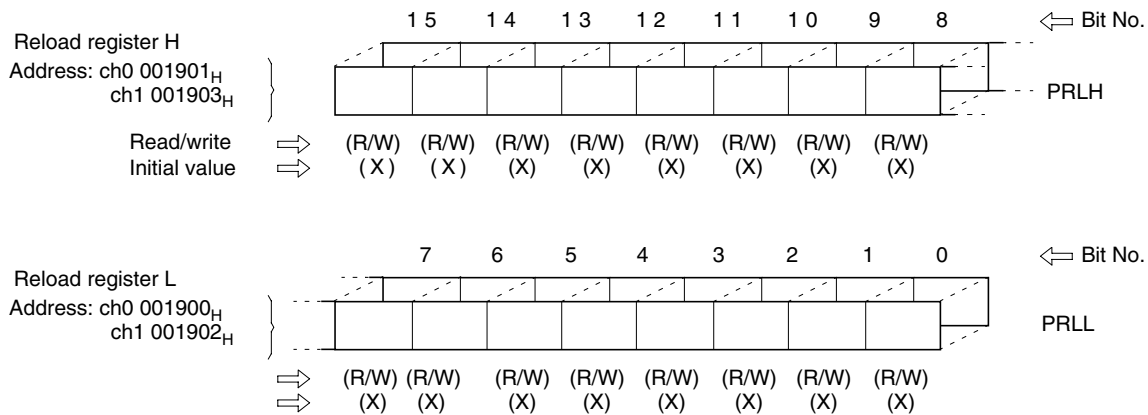
PCM2	PCM1	PCM0	Operation mode
0	0	0	Peripheral clock (62.5 ns machine clock, 16 MHz)
0	0	1	Peripheral clock/2 (125 ns machine clock, 16 MHz)
0	1	0	Peripheral clock/4 (250 ns machine clock, 16 MHz)
0	1	1	Peripheral clock/8 (500 ns machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1 μ s machine clock, 16 MHz)
1	0	1	Clock input from the timebase timer (128 μ s, 4 MHz source oscillation)

These bits are initialized to "000" upon a reset. These bits are is readable and writable.

14.3.4 Reload Register (PRLH/PRLH)

The reload registers (PRL and PRLH) are 8-bit registers that store reload values for the PCNT down counters. The PRL and PRLH registers are readable and writable.

■ Reload Register (PRL/PRLH)



Register name	Function
0	Holds the L side reload value.
1	Holds the H side reload value.

Note:

In 8-bit prescaler + 8-bit PPG mode, different values in PRL and PRLH of Channel 0 may cause the PPG waveform of ch1 to vary in each cycle. Write the same value to PRL and PRLH of ch0.

14.4 Operations of 8/16-bit PPG



One 8/16-bit PPG consists of two channels of 8-bit PPG units. These two channels can be used in three modes: independent two-channel mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

■ Operations of 8/16-bit PPG

Each of the 8-bit PPG units has two eight-bit reload registers. One reload register is for the L pulse width (PRL) and the other is for the H pulse width (PRLH). The values stored in these registers are reloaded into the 8-bit down counter (PCNT), from the PRL and PRLH in turn. The pin output value is inverted upon a reload caused by counter borrow. This operation results in the pulses of the specified L pulse width and H pulse width.

Table 14.4-1 "Reload Operation and Pulse Output" lists the relationship between the reload operation and pulse outputs.

Table 14.4-1 Reload Operation and Pulse Output

Reload operation	Pin output change		
PRLH --> PCNT	PPG00/10 [0 --> 1]		Rise
PRL --> PCNT	PPG00/10 [1 --> 0]		Fall

When 1 is set in bit 4 (PIE0) of PPGC0 or in bit 12 (PIE1) of PPGC1, an interrupt request is output upon a borrow from 00 to FF (from 0000 to FFFF in 16-bit PPG mode) of each counter.

■ Operation Modes of 8/16-bit PPG

This block can be used in three modes: independent two-channel mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

○ Independent two-channel mode

The two channels of 8-bit PPG units operate independently. The PPG00 pin is connected to the ch0 PPG output, while the PPG10 pin is connected to the ch1 PPG output.

○ 8-bit prescaler + 8-bit PPG mode

ch0 is used as an 8-bit prescaler while the count in ch1 is based on borrow outputs from ch0. Thus, 8-bit PPG waveforms can be output with arbitrary length of cycle time. The PPG00 pin is connected to the ch0 prescaler output, while the PPG10 pin is connected to the ch1 PPG output.

○ 16-bit PPG 1ch mode

ch0 and ch1 are connected and used as a single 16-bit PPG. The PPG00 and PPG10 pins are connected to the 16-bit PPG output.

For the MB90590 Series, the output signal from the Channel 0 PPG is not connected to any external pin.

■ 8/16-bit PPG Output Operation

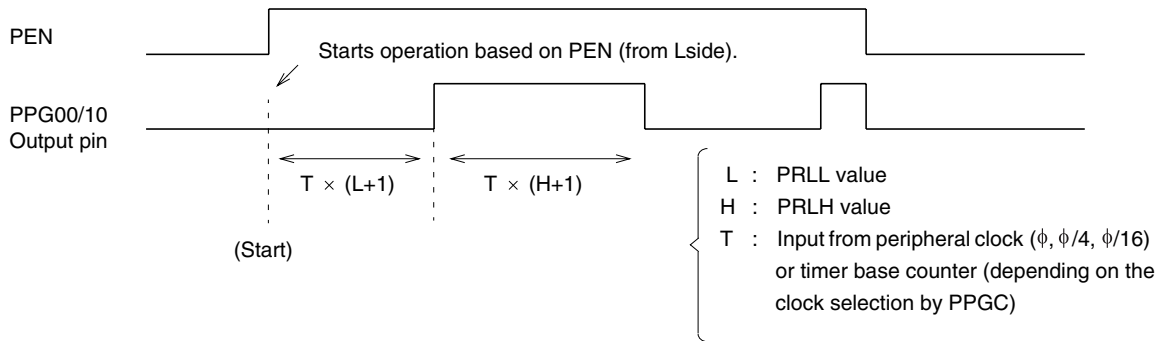
In this block, the ch0 PPG is activated to start counting when "1" is written to bit 7 (PEN0) of the PPGC0 (PMW operation mode control register). Similarly, the ch1 PPG is activated to start counting when "1" is written to bit 15 (PEN1) of the PPGC1 register. Once the operation has started, counting is terminated by writing "0" to bit 7 (PEN0) of PPGC0 or in bit 15 (PEN1) of PPGC1. Once the counting is terminated, the output is maintained at the L level.

For the MB90590 Series, the output signal from the Channel 0 PPG is not connected to any external pin.

In 8-bit prescaler + 8-bit PPG mode, do not set ch1 to be in operation while ch0 operation is stopped.

In 8/16-bit PPG mode, ensure that bit 7 (PEN0) of PPGC0 (PMW operation mode control register) and bit 15 (PEN1) of PPGC1 register are started or stopped simultaneously. The figure below is a diagram of PPG output operation. During PPG operation, a pulse wave is continuously output at a frequency and duty ratio (the ratio of the H-level period of the pulse wave to the L-level period). PPG continues operation until stop is specified explicitly.

Figure 14.4-1 PPG Output Operation, Output Waveform



■ Relationship Between 8/16-bit PPG Reload Value and Pulse Width

The width of the output pulse is determined by adding 1 to the reload register value and multiplying it by the count clock cycle. Note that when the reload register value is 00_H during 8-bit PPG operation or 0000_H during 16-bit PPG operation, the pulse width is equivalent to one count clock cycle. In addition, note that when the reload register value is FF_H during 8-PPG operation, the pulse width is equivalent to 256 count clock cycles. When the reload register value is $FFFF_H$ during 16-bit PPG operation, the pulse width is equivalent to 65536 count clock cycles.

$$\begin{aligned}
 P_l &= T \times (L+1) \\
 P_h &= T \times (H+1)
 \end{aligned}
 \left. \begin{array}{l}
 L : \text{PRL value} \\
 H : \text{value} \\
 T : \text{Input clock cycle} \\
 P_h : \text{High pulse width} \\
 P_l : \text{Low pulse width}
 \end{array} \right\}$$

14.5 Selecting a Count Clock for 8/16-bit PPG

The count clock used for the operation is supplied from the peripheral clock or the timebase timer. The count clock can be selected from six choices.

■ Selecting a Count Clock for 8/16-bit PPG

Select ch0 clock at bit 4 to 2 (PCM2 to 0) of the PPG01 register, and ch1 clock at bit 7 to 5 (PCS2 to 0) of the PPG01 register.

The clock is selected from a peripheral clock 1/16 to 1 times higher than a machine clock or an input clock from the timebase timer.

In 8-bit prescaler + 8-bit PPG mode or 16-bit PPG mode, however, the setting in the PCS2 to 0 has no effect.

When the timebase timer input is used, the first count cycle after a trigger or a stop may be shifted. The cycle may also be shifted if the timebase counter is cleared during operation of this module.

In 8-bit prescaler + 8-bit PPG mode, if ch1 is activated while ch0 is in operation and ch1 is stopped, the first count cycle may be shifted.

14.6 Controlling Pin Output of 8/16-bit PPG Pulses

The pulses generated by this module can be output from external pins PPG00 and PPG10.

■ Controlling Pin Output of 8/16-bit PPG Pulses

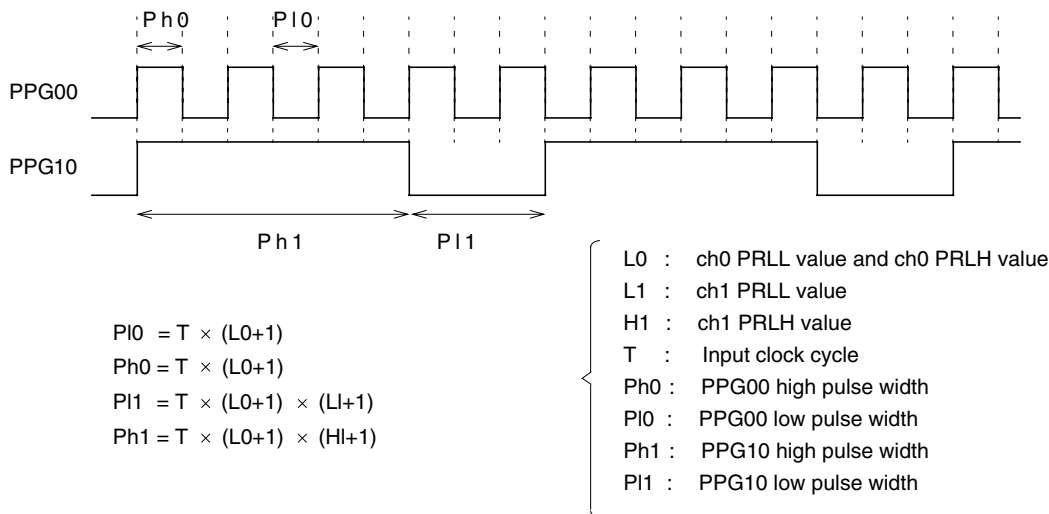
To output the pulses from an external pin, write "1" to the bit corresponding to each pin (PPGC0: PE00, PPGC1: PE10). When "0" is written to these bits (default), the pulses are not output from the corresponding external pins; the pins work as general-purpose ports.

In 16-bit PPG mode, the same waveform is output from PPG00 and PPG10. Thus, the same output can be obtained by enabling both external pin.

In 8-bit prescaler + 8-bit PPG mode, the 8-bit prescaler toggle output waveform is output from PPG00, while the 8-bit PPG waveform is output from PPG10. Figure 14.6-1 "8+8 PPG Output Operation Waveform" is a diagram of output waveforms in this mode.

For the MB90590 Series, the output signal from the Channel 0 PPG is not connected to any external pin.

Figure 14.6-1 8+8 PPG Output Operation Waveform



Note:

Set the same value in ch0 PRL and ch0 PRLH.

14.7 8/16-bit PPG Interrupts

For the 8/16-bit PPG, an interrupt becomes active when the reload value counts out and a borrow occurs.

■ 8/16-bit PPG Interrupts

In 8-bit PPG 2ch mode or 8-bit prescaler + 8-bit PPG mode, an interrupt is requested by a borrow in each counter. In 16-bit PPG mode, PUG0 and PUF1 are simultaneously set by a borrow in the 16-bit counter. Therefore, enable only PIE0 or PIE1 to unify the interrupt causes. In addition, simultaneously clear the interrupt flags for PUF0 and PUF1.

14.8 Initial Values of 8/16-bit PPG Hardware

The hardware components of this block are initialized to the following values when reset:

■ Initial Values of 8/16-bit PPG Hardware

○ Registers

- PPGC0 --> $0X000XX1_B$
- PPGC1 --> $0X000001_B$
- PPG10 --> $XXXXXX00_B$

○ Pulse outputs

- PPG00 --> "L"
- PPG10 --> "L"
- PE00 --> PPG00 output disabled
- PE10 --> PPG10 output disabled

○ Interrupt requests

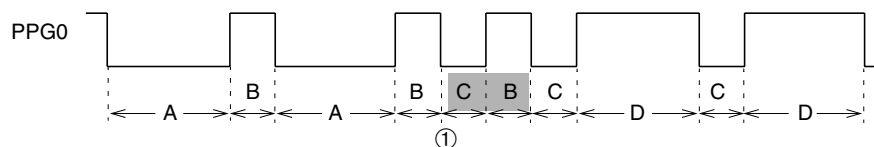
- IRQ0 --> "L"
- IRQ1 --> "L"

Hardware components other than the above are not initialized.

Note:

In a mode other than 16-bit PPG mode, it is recommended to use a word transfer instruction to write data in reload registers PRL and PRLH. If two byte transfer instructions are used to write a data item to these registers, a pulse of unexpected cycle time may be output depending on the timing.

Figure 14.8-1 Write Timing for 8/16-bit PPG Reload Registers (PRL and PRLH)

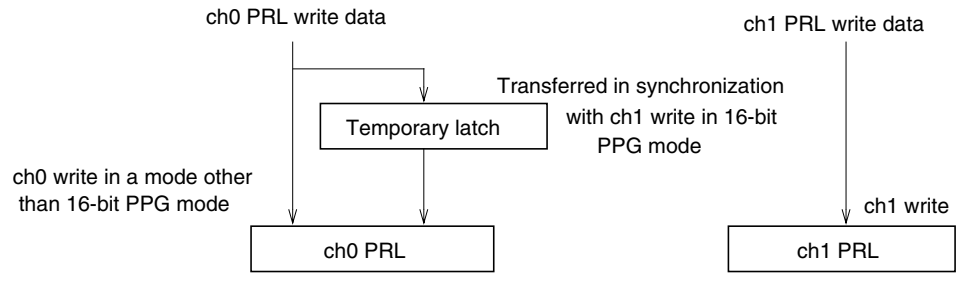


Assume that PRL is updated from A to C before point 1 in the time chart above, and PRLH is updated from B to D after point 1. Since the PRL values at point 1 are PRL=C and PRLH=B, a pulse of L side count value C and H side count value B is output only once.

Similarly, to write data in PRL of ch0 and ch1 in 16-bit PPG mode, use a long word transfer instruction, or use word transfer instructions in the order of ch0 and then ch1. In this mode, the data is only temporarily written to ch0 PRL. Then, the data is actually written into ch0 PRL when the ch1 PRL is written to.

In a mode other than 16-bit PPG mode, ch0 and ch1 PRL are written independently.

Figure 14.8-2 PRL Write Operation Block Diagram



CHAPTER 15 DTP/EXTERNAL INTERRUPTS

This chapter explains the functions and operations of the DTP/external interrupts.

- 15.1 "Outline of DTP/External Interrupts"
- 15.2 "DTP/External Interrupt Registers"
- 15.3 "Operations of DTP/External Interrupts"
- 15.4 "Switching Between External Interrupt and DTP Requests"
- 15.5 "Notes on Using DTP/External Interrupts"

15.1 Outline of DTP/External Interrupts

The data transfer peripheral (DTP) is located between an external peripheral and the F²MC-16LX CPU. The DTP receives a DMA request or interrupt request from the external peripheral, transfers the request to the F²MC-16LX CPU to activate the intelligent I/O service or interrupt processing.

■ Outline of DTP/External Interrupts

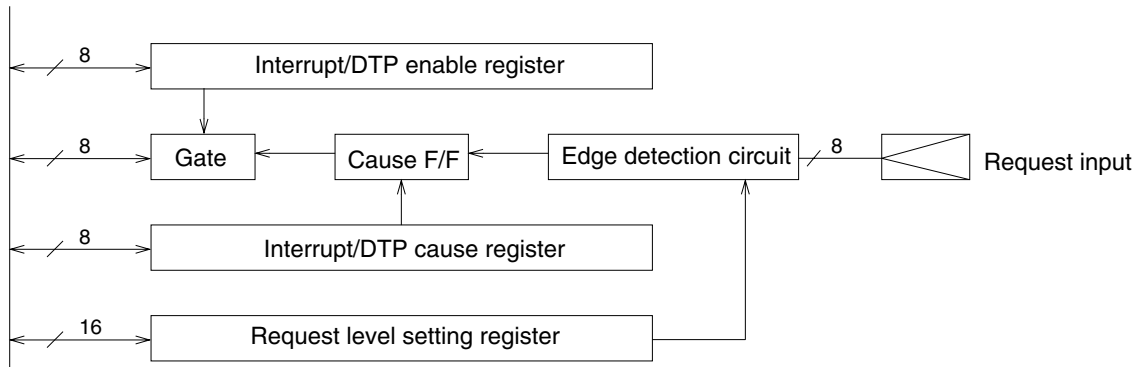
For the intelligent I/O service, "H" and "L" request levels are available. For an external interrupt request, four request levels are available: "H", "L", rising edge, and falling edge.

For the MB90590 Series, the external bus interface is not supported. Therefore the DTP/ External Interrupt can not serve as the data transfer peripheral. It can be only used as the External Interrupt.

For MB90V590G, there are only four external pins assigned to this block. Therefore the external interrupt channel 4 to 7 are not supported. These external interrupts should be disabled.

■ Block Diagram of DTP/External Interrupts

Figure 15.1-1 Block Diagram of DTP/External Interrupts



■ DTP/external Interrupts Registers

bit	7	6	5	4	3	2	1	0	
Address : 000030 _H	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	Interrupt/DTP enable register (ENIR)

bit	15	14	13	12	11	10	9	8	
Address : 000031 _H	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	Interrupt/DTP cause register (EIRR)

bit	7	6	5	4	3	2	1	0	
Address : 000032 _H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	Request level setting register (ELVR)

bit	15	14	13	12	11	10	9	8	
Address : 000033 _H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	Request level setting register (ELVR)

15.2 DTP/External Interrupt Registers

The DTP/external interrupts has the following three types of registers:

- Interrupt/DTP enable register (ENIR: Interrupt request enable register)
- Interrupt/DTP source register (EIRR: External interrupt request register)
- Request level setting register (ELVR: External level register)

■ Interrupt/DTP Enable Register (ENIR: Interrupt request enable register)

	7	6	5	4	3	2	1	0	Initial value
ENIR Address : 000030 _H	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

ENIR enables the function to issue a request to the interrupt controller using a device pin as an external interrupt/DTP request input. A pin corresponding to a "1" bit of this register is used as an external interrupt/DTP request input. A pin corresponding to a "0" bit holds the external interrupt/DTP request input cause, but does not issue a request to the interrupt controller.

■ Interrupt/DTP Source Register (EIRR: External interrupt request register)

	15	14	13	12	11	10	9	8	Initial value
EIRR Address : 000031 _H	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	XXXXXXXX _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/WThe objects differ for R and W.

The EIRR indicates the presence of external interrupt/DTP requests at the pins corresponding to the "1" bits of this register. Writing "0" to a bit of this register clears the corresponding request flag. Writing "1" has no effect. Reading this register with a read-modify-write instruction always results in the reading value "1".

Note:

If more than one external interrupt request output is enabled (EN7 to EN0 of ENIR are set to 1), clear to 0 only the bit for which the CPU accepted an interrupt (any of bits ER7 to ER0 that are set to 1). Do not clear the other bits without a valid reason.

■ Request Level Setting Register (ELVR: External level register)

	7	6	5	4	3	2	1	0	Initial value
Address : 000032 _H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	7	6	5	4	3	2	1	0	Initial value
Address : 000033 _H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

ELVR defines the request event at the external pin. Each pin is assigned two bits as described in Table 15.2-1 "Interrupt Request Detection Factor for LBx and LAx Pins". If a request is detected by the input level, the interrupt flag is set as long as the input is at the specified level even after the flag is reset by software.

Table 15.2-1 Interrupt Request Detection Factor for LBx and LAx Pins

LBx	LAx	Interrupt request detection factor
0	0	L level pin input
0	1	H level pin input
1	0	Rising edge pin input
1	1	Falling edge pin input

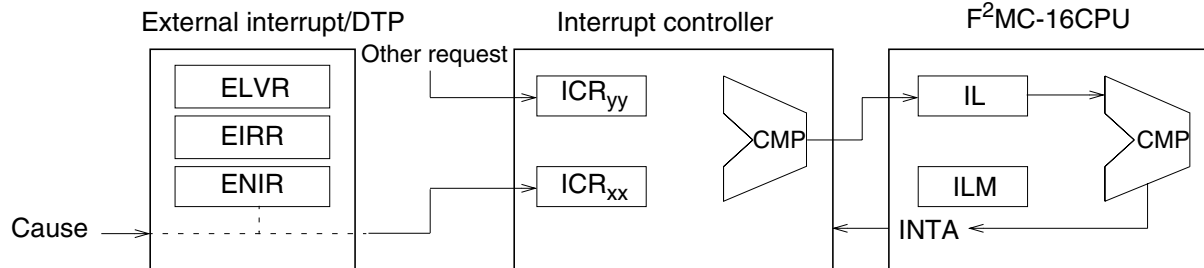
15.3 Operations of DTP/External Interrupts

When the interrupt flag is set, this block signals an interrupt to the interrupt controller. The interrupt controller judges the priority levels of the simultaneous interrupts, and issues an interrupt request to the F²MC-16LX CPU if the interrupt from this block has the highest priority. The F²MC-16LX CPU compares the ILM bits of its internal CCR register and the interrupt request. If the interrupt level of the request is higher than that indicated by the ILM bits, the F²MC-16LX CPU activates the hardware interrupt processing microprogram as soon as the currently executing instruction is terminated.

External Interrupt Operation

In the hardware interrupt processing microprogram, the CPU reads the ISE bit information from the interrupt controller, identifies that the request is for interrupt processing based on that information, and branches to the interrupt processing microprogram. The interrupt processing microprogram reads the interrupt vector area and issues an interrupt acknowledgment signal for the interrupt controller. Then, the microprogram transfers the jump destination address of the macro instruction generated from the vector to the program counter, and executes the user interrupt processing program.

Figure 15.3-1 External Interrupt



■ DTP operation

To activate the intelligent I/O service, the user program initially sets the address of a register, assigned between 000000_H and 0000FF_H, in the I/O address pointer of the intelligent I/O service descriptor. Then, the user program sets the start address of the memory buffer in the buffer address pointer.

The DTP operation sequence is almost the same as for external interrupts. The operation is identical until the CPU activates the hardware interrupt processing microprogram. Then, for the DTP, control is transferred to the intelligent I/O service processing microprogram, since the ISE bit read by the CPU within the hardware interrupt processing microprogram indicates the DTP. Once the intelligent I/O service is activated, a read or write signal is sent to the addresses external peripheral, and data is transferred between the peripheral and the chip. The external peripheral must cancel the interrupt request to this chip within three machine cycles after the transfer is made. When the transfer is completed, the descriptor is updated, and the interrupt controller generates a signal that clears the transfer cause. Upon receiving the signal to clear the transfer cause, this resource clears the flip-flop holding the cause and prepares for the next request from the pin. For details of the intelligent I/O service processing, refer to the MB90500 Programming Manual.

Figure 15.3-2 Timing to Cancel the External Interrupt at the End of DTP Operation

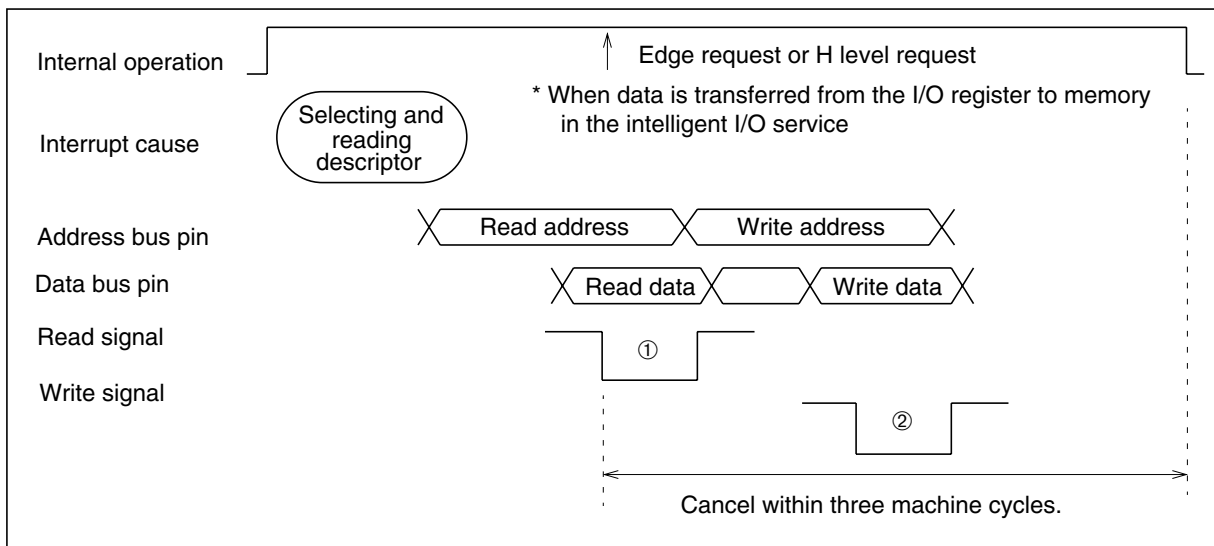
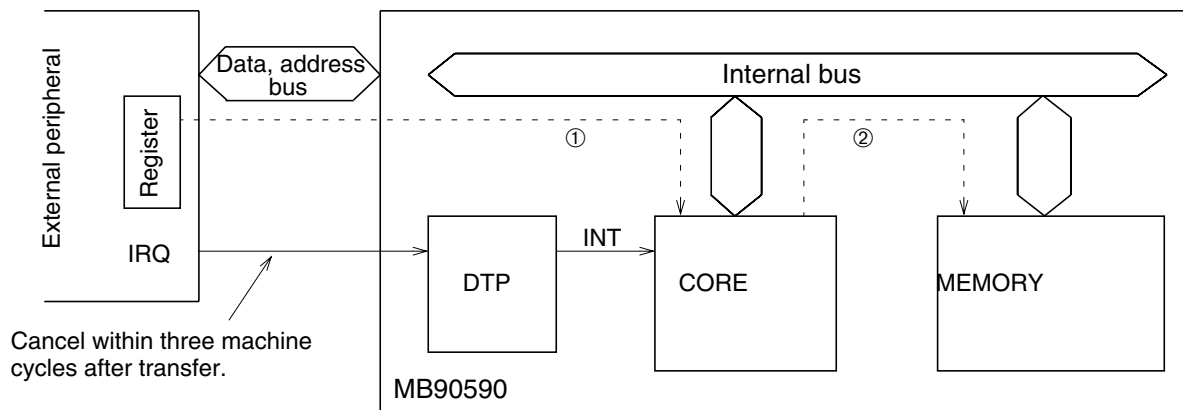


Figure 15.3-3 Sample Interface to the External Peripheral

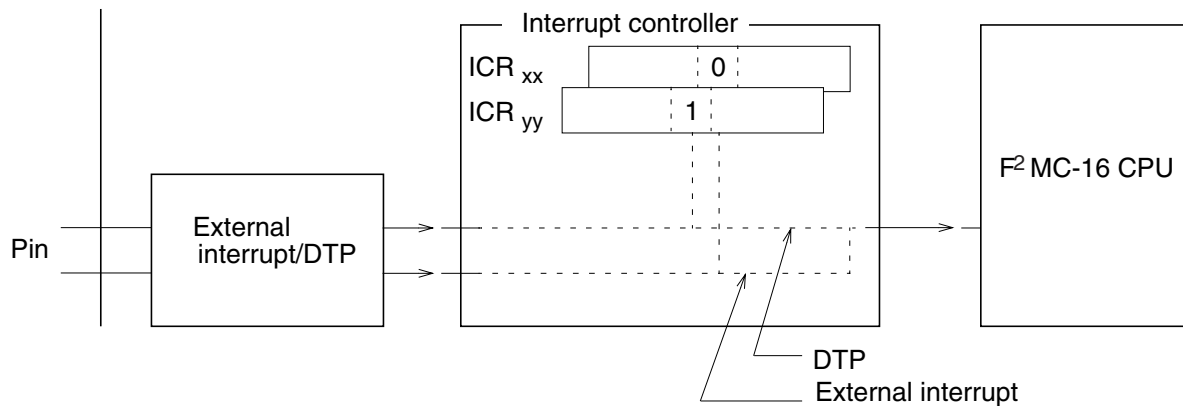


15.4 Switching between External Interrupt and DTP Requests

To switch between external interrupt and DTP requests, use the ISE bit in the ICR register corresponding to this block, which is in the interrupt controller. Each pin is individually assigned ICR. Thus, a pin is used for a DTP request if "1" is written to the ISE bit of the corresponding ICR, and is used for an external interrupt request if "0" is written to the bit.

■ Switching Between External Interrupt and DTP Requests

Figure 15.4-1 Switching Between External Interrupt and DTP Requests



15.5 Notes on Using DTP/External Interrupts

Note carefully the following items when using DTP/external interrupts:

- **Conditions on the externally connected peripheral when DTP is used**
 - **Recovery from standby**
 - **External interrupt/DTP operation procedure**
 - **External interrupt request level**
-

■ Notes on Using DTP/External Interrupts

○ **Conditions on the externally connected peripheral when DTP is used**

DTP supports only external peripherals that automatically clear a request once a transfer is completed. The system must be designed so that a transfer request is canceled within three machine cycles (provisional) after transfer operation starts. Otherwise, this resource assumes that a transfer request is issued.

○ **Recovery from standby**

To use an external interrupt to recover from the standby state in stop mode and watch mode, use an H or L level request as an input request. If an edge request is used, recovery from the standby state in stop mode and watch mode cannot be performed.

○ **External interrupt/DTP operation procedure**

To set registers in the external interrupt/DTP, follow the steps below:

1. Disable the bits corresponding to the enable register.
2. Set the bits corresponding to the request level setting register.
3. Clear the bits corresponding to the cause register.
4. Enable the bits corresponding to the enable register.

(Steps 3. and 4. can be simultaneously performed by word specification.)

To set a register in this resource, ensure that the enable register is disabled. Before enabling the enable register, ensure that the cause register is cleared. Clearing the cause register prevents a false interrupt cause from being determined while registers are set or interrupts are enabled.

○ **External interrupt request level**

To detect an edge for an edge request level, the pulse width must be at least three machine cycles.

As shown in Figure 15.5-1 "Clearing the Cause Hold Circuit Upon Level Set", when the request input level is related to the level setting, a request that is input from an external device to the interrupt controller is kept active even if the request is later canceled because a cause hold circuit has been installed. To cancel the request to the interrupt controller, the cause hold circuit must be cleared as shown in Figure 15.5-2 "Interrupt Cause and Interrupt Request to the Interrupt Controller While Interrupts are Enabled".

Figure 15.5-1 Clearing the Cause Hold Circuit Upon Level Set

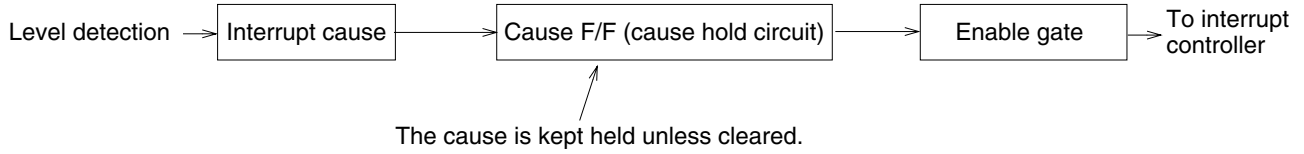
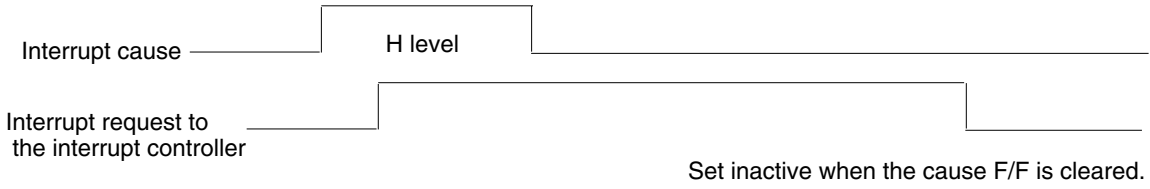


Figure 15.5-2 Interrupt Cause and Interrupt Request to the Interrupt Controller While Interrupts are Enabled



CHAPTER 16 A/D Converter

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

- 16.1 "Features of A/D Converter"
- 16.2 "Block Diagram of A/D Converter"
- 16.3 "A/D Converter Registers"
- 16.4 "Operations of A/D Converter"
- 16.5 "Conversion Using EI²OS"
- 16.6 "Conversion Data Protection"

16.1 Features of A/D Converter

The A/D converter converts analog input voltages into digital values. The A/D converter has the following features:

■ Features of A/D converter

- **Conversion time:**
26.3 μ s min. per channel (at 16 MHz machine clock)
- **RC sequential compare conversion with sample and hold circuit**
- **Resolution of 8 or 10 bits**
- **Analog input selected from eight channels by programming**
Single conversion mode: One channel is selected for conversion.
Scan conversion mode: Voltages in multiple consecutive channels are converted. Up to eight channels can be programmed.
Continuous conversion mode: Voltages at the specified channel are converted repeatedly.
Stop conversion mode: Voltages at the specified channel are converted, then the system pauses and stands by for the next activation. (The conversion start points can be synchronized.)
- **Interrupt request**
At the end of A/D conversion, a relevant interrupt request can be issued to the CPU. This interrupt can be used to activate the EI²OS, which automatically transfers A/D conversion result to memory. This feature is suitable for continuous processing.
- **Selectable activation cause**
The activation can be done by software, external trigger (falling edge), or timer (rising edge).

■ Analog Input Enable Register

Always write "1" to the ADER bit corresponding to a pin used as analog input.

bit	15	14	13	12	11	10	9	8	Initial value
Address: 00001B _H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Port 6 pins are controlled as described below.

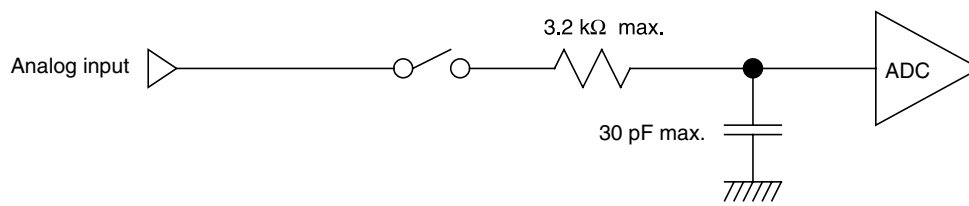
0: Port input/output mode

1: Analog input mode

"1" is set upon a reset.

■ Input Impedance

The sampling circuit of the A/D Converter can be represented with the equivalent circuit shown below.



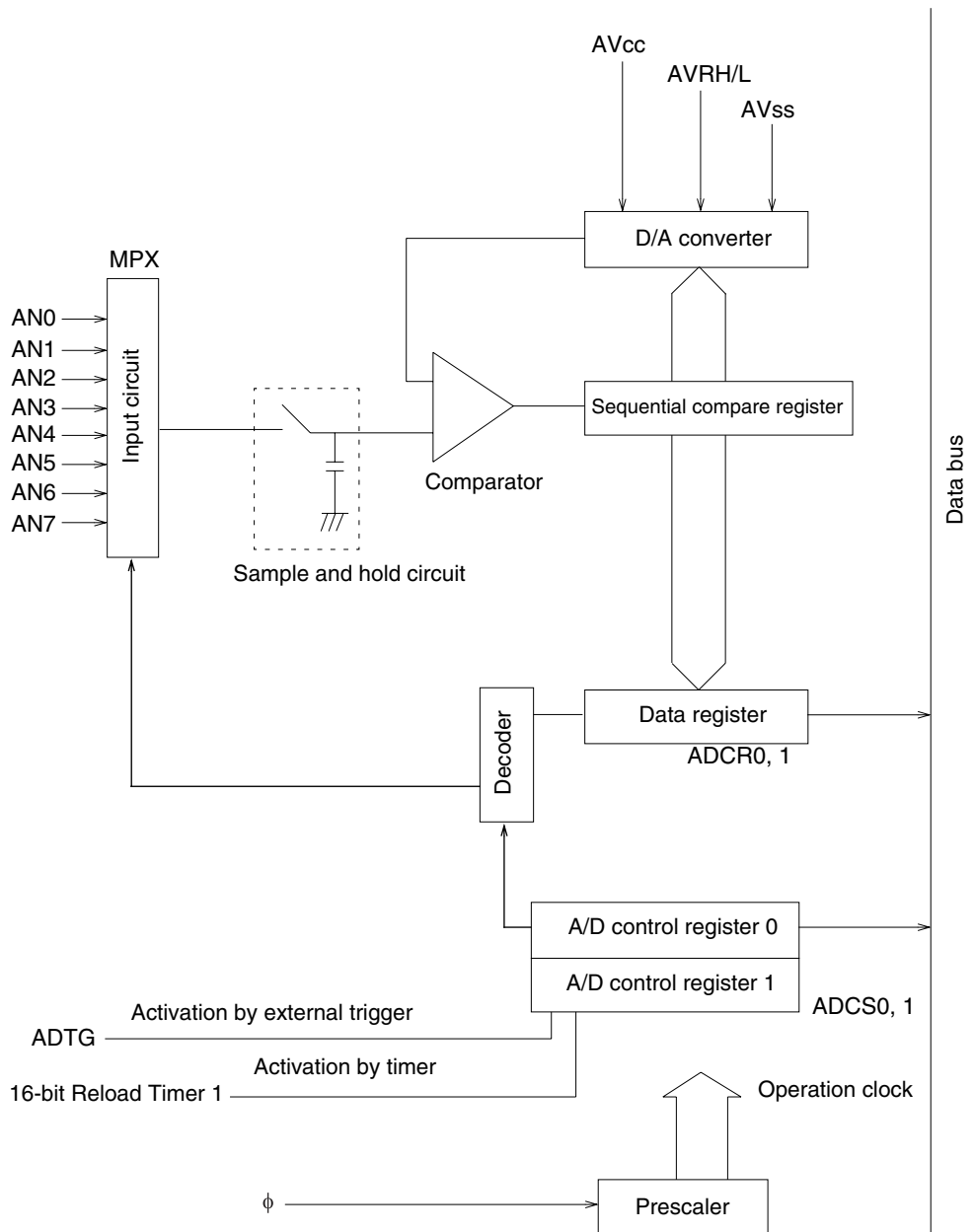
Driving impedance to an analog input should be lower than 15.5K ohm when the sampling time is set to 4μs (ST1=0 and ST0=0 at 16MHz machine clock). Otherwise the conversion accuracy will be worsened. If this is the case, set the sampling time longer (ST1=1 and ST0=1) or add external capacitor in order to compensate the driving impedance.

16.2 Block Diagram of A/D Converter

Figure 16.2-1 "Block Diagram of A/D Converter" shows a block diagram of the A/D converter.

■ Block Diagram of A/D Converter

Figure 16.2-1 Block Diagram of A/D Converter



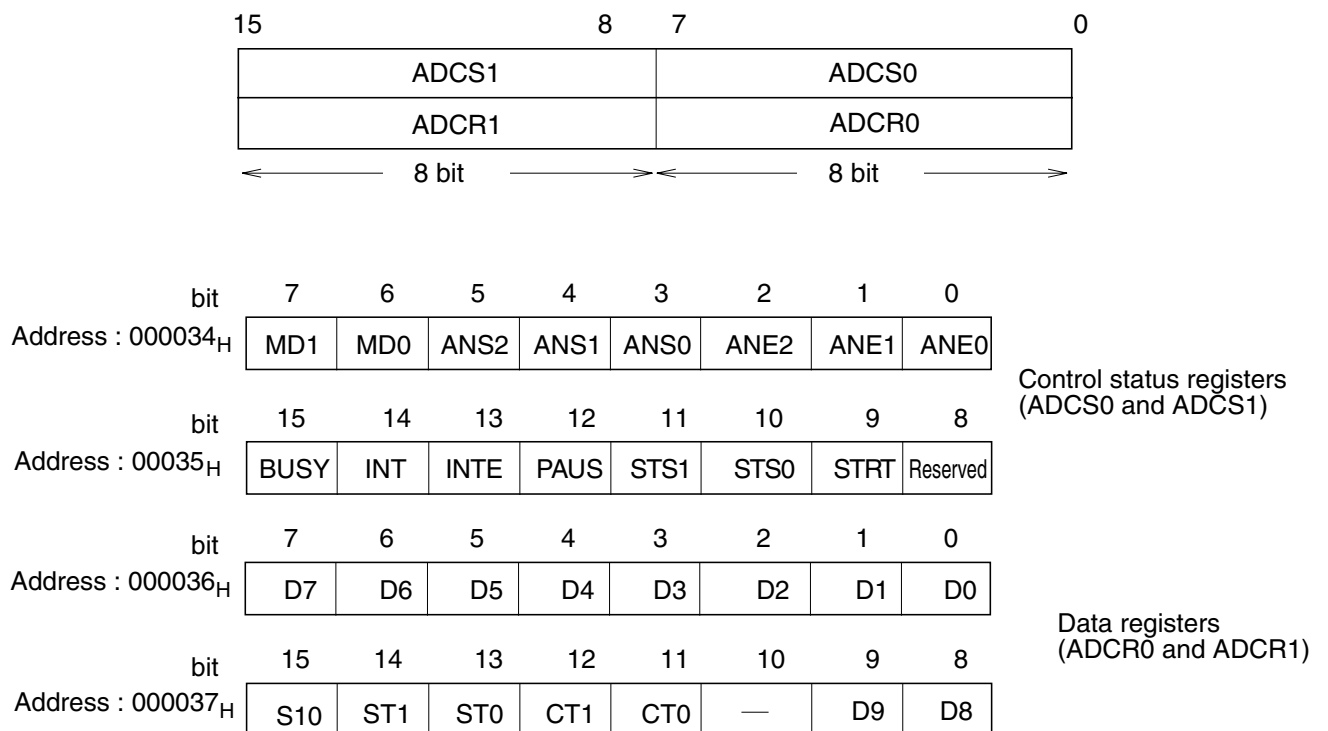
16.3 A/D Converter Registers

The A/D converter has the following two types of registers:

- Control status register
- Data register

■ A/D Converter Registers

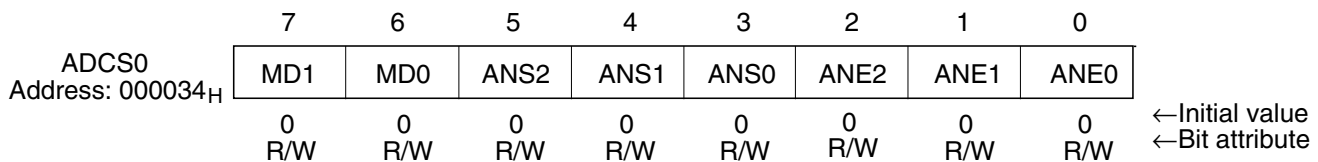
Figure 16.3-1 A/D Converter Register Configuration



16.3.1 Control Status Registers (ADCS0)

The control status register (ADCS0) controls the A/D converter and indicates the status. Do not rewrite ADCS0 during A/D conversion.

■ Control Status Registers (ADCS0)



[bits 7 and 6] MD1 and MD0 (A/D converter mode set):

Table 16.3-1 Operation Mode Setting

MD1	MD0	Operation mode
0	0	Single mode. Reactivation during operation is allowed.
0	1	Single mode. Reactivation during operation is not allowed.
1	0	Continuous mode. Reactivation during operation is not allowed.
1	1	Stop mode. Reactivation during operation is not allowed.

○ Single mode:

A/D conversion is continuously performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0. The conversion stops once it has been done for all these channels.

○ Continuous mode:

A/D conversion is repeatedly performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0.

○ Stop mode:

A/D conversion is performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0, pausing for each channel. The A/D conversion is resumed upon an activation.

Upon a reset, these bits are initialized to "00".

Note:

When activated in the continuous or stop mode, A/D conversion continues until it is stopped by the BUSY bit.

The conversion is stopped by writing "0" to the BUSY bit.

Reactivation disabled in single mode, continuous mode, and stop mode applies to all kinds of activation by software, an external trigger, and a timer.

[bits 5, 4, and 3] ANS2, ANS1, and ANS0 (Analog start channel set):

Use these bits to specify the start channel for A/D conversion.

When the A/D converter is activated, A/D conversion starts from the channel selected with these bits.

ANS2	ANS1	ANS0	Start channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

* Read

During A/D conversion, the current conversion channel is read from these bits. If the system is stopped in the stop mode, the last conversion channel is read.

* Upon a reset, these bits are initialized to "000".

[bits 2, 1, and 0] ANE2, ANE1, and ANE0 (Analog end channel set):

Use these bits to set the A/D conversion end channel.

ANE2	ANE1	ANE0	End channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Note:

When the same channel is written to ANE2 to ANE0 and ANS2 to ANS0, conversion is performed for one channel only (single conversion).

In the continuous or stop mode, operation returns to the start channel specified in ANS2 to ANS0 after the conversion is completed for the channel specified in ANE2 to ANE0.

If the ANS value is greater than the ANE value, conversion starts from the ANS channel. Then, once conversion is complete up to channel 7, operation returns to channel 0 and conversion is performed up to the ANE channel.

Upon a reset, these bits are initialized to "000".

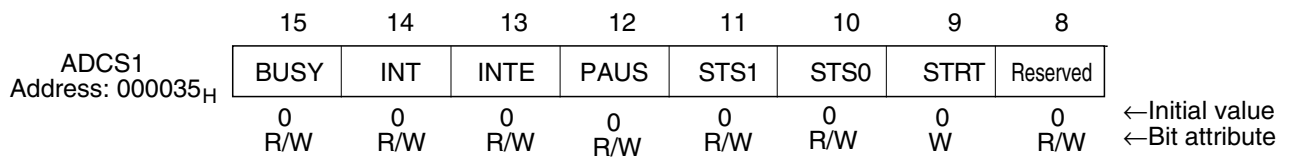
Example: ANS=6, ANE=3, single mode

Conversion is performed in the following sequence: CH6, CH7, CH0, CH1, CH2, CH3

16.3.2 Control Status Register (ADCS1)

The control status register (ADCS1) controls the A/D converter and indicates the status.

■ Control Status Register (ADCS1)



[bit 15] BUSY (busy flag and stop):

- Read

This bit indicates the A/D converter operation.

This bit is set when A/D conversion starts and is cleared when the conversion ends.

- Write

Writing "0" to this bit during A/D conversion forces the conversion to terminate.

The above feature is used for forced stop in continuous or stop mode.

"1" cannot be written to the BUSY bit. With a read-modify-write (RMW) instruction, "1" is read from this bit. In single mode, this bit is cleared at the end of A/D conversion.

In continuous or stop mode, this bit is not cleared until conversion is stopped by writing "0".

This bit is initialized to "0" upon a reset.

Do not perform a forced stop and activation by software simultaneously (BUSY = 0, STRT = 1).

[bit 14] INT (Interrupt):

This bit is set when conversion data is written to ADCR.

An interrupt request is issued if this bit is set while bit 5 (INTE) is "1". In addition, the EI²OS is activated if it is enabled. Writing "1" has no effect.

This bit is cleared by writing "0" or by the EI²OS interrupt clear signal.

Note: To clear this bit by writing "0", ensure that A/D conversion is not in progress.

This bit initialized to "0" upon a reset.

[bit 13] INTE (Interrupt enable):

This bit is used to enable or disable interrupts at the end of conversion.

- 0: Interrupts are disabled.
- 1: Interrupts are enabled.

Set this bit when using the EI²OS. The EI²OS is activated when an interrupt request is issued.

Upon a reset, this bit is initialized to "0".

[bit 12] PAUS (A/D conversion pause):

This bit is set when the A/D conversion is paused.

Only one register is available for storing the A/D conversion result. Therefore, unless the conversion results are transferred by the EI²OS, the result data would be continuously updated and destroyed in continuous conversion.

To prevent the above condition, the system is designed so that a data register value must be transferred by the EI²OS before the next conversion data is saved. A/D conversion pauses during that period. A/D conversion is resumed at the end of transfer by the EI²OS.

This register is valid only when the EI²OS is used.

Note:

For the conversion data protection function, see Section 16.4 "Operations of A/D Converter".

Upon a reset, this bit is initialized to "0".

[bits 11 and 10] STS1 and STS0 (Start source select):

Upon a reset, these bits are initialized to "00".

These bits are used to select the A/D conversion activation source.

STS1	STS0	Function
0	0	Activation by software
0	1	Activation by external pin trigger and software
1	0	Activation by timer and software
1	1	Activation by external pin trigger, timer, and software

In a mode allowing two or more activation factors, A/D conversion is activated by the source that occurs first.

The activation source setting changes as soon as it is updated. Thus, take care when updating it during A/D conversion.

Note:

The external pin trigger is detected by the falling edge. If this bit is updated to external trigger activation while the external trigger input level is "L", A/D may be activated at once.

When timer is selected, the 16-bit Reload Timer 1 is selected.

[bit 9] STRT (Start):

A/D conversion is activated when "1" is written to this bit.

To reactivate A/D conversion, write "1" to this bit again.

Upon a reset, this bit is initialized to "0".

In the stop mode, a reactivation during the operation is not supported. Check the BUSY bit before writing "1".

Do not perform a forced stop and activation by software simultaneously. (BUSY=0, STRT=1)

[bit 8] Reserved

This is a reserved bit. Always write "0" to this bit.

16.3.3 Data Registers (ADCR1 and ADCR0)

These registers are used to store the digital values produced as a result of the conversion. ADCR1 stores the most significant two bits of the conversion result, while ADCR0 stores the lower eight bits. These register values are updated each time conversion is completed. Usually, the final conversion value is stored in these bits.

■ Data Registers (ADCR1 and ADCR0)

ADCR0 bit	7	6	5	4	3	2	1	0	
Address : 000036 _H	D7	D6	D5	D4	D3	D2	D1	D0	Initial value XXXXXXXX
	R	R	R	R	R	R	R	R	
ADCR1 bit	15	14	13	12	11	10	9	8	
Address : 000037 _H	S10	ST1	ST0	CT1	CT0	—	D9	D8	Initial value 000010XX
	W	W	W	W	W		R	R	

"0" is always read from the bits 10 to 15 of ADCR1.

The conversion data protection function is available. See Section 2.7.4 "Program Counter (PC)". Ensure that no data is written to these registers during A/D conversion.

[bits 15] S10

This bit specifies the resolution of the conversion. When it is set to "0", the 10-bit A/D conversion is performed. Otherwise the 8-bit A/D conversion is performed and the result is stored in the D7 to D0.

Reading this bit always results in the reading value "0".

[bits 14 and 13] ST1 and ST0 (Sampling time):

ST1	ST0	Function
0	0	64 machine cycles (4 μs at 16 MHz)
0	1	Reserved
1	0	Reserved
1	1	4096 machine cycles (256 μs at 16 MHz)

These bits determines the duration of the voltage sampling time at the input.

Reading this bit group always results in the reading value "00".

[bits 12 and 11] CT1 and CT0 (Compare time):

CT1	CT0	Function
0	0	176 machine cycles (22 μ s at 8 MHz)
0	1	352 machine cycles (22 μ s at 16 MHz)
1	0	Reserved
1	1	Reserved

These bits determines the duration of the compare operation time.

Do not set to "00" unless the machine clock is 8MHz. Otherwise the conversion accuracy is not guaranteed.

Reading this bit group always results in the reading value "00".

16.4 Operations of A/D Converter

The A/D converter operates employs the sequential compare technique, and can be selected from 10-bit or 8-bit resolution.

Since the A/D converter has one register (16 bits) for storing the conversion result, the conversion data registers (ADCR0 and ADCR1) are updated each time conversion is completed. Thus, the A/D converter alone must not be used for the continuous conversion. Use the External intelligent I/O service (EI²OS) function to transfer converted data to memory while conversion is in progress.

The operation modes are explained below.

■ Single Mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits. The converter stops operation after the conversion is completed for the end channel specified with the ANE bits. If the start and end channels are the same (ANS=ANE), conversion is performed only for one channel.

Example:

```
ANS = 0 0 0 , ANE = 0 1 1
Start --> AN0 --> AN1 --> AN2 --> AN3 --> End
```

```
ANS = 0 1 0 , ANE = 0 1 0
Start --> AN2 --> End
```

■ Continuous Mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits. After the conversion is completed for the end channel specified with the ANE bits, conversion is repeated from the analog inputs of the ANS. If the start and end channels are the same (ANS=ANE), conversion for one channel is repeated.

Example:

```
ANS = 0 0 0 , ANE = 0 1 1
Start --> AN0 --> AN1 --> AN2 --> AN3 --> AN0 --> Repeat
```

```
ANS = 0 1 0 , ANE = 0 1 0
Start --> AN2 --> AN2 --> AN2 --> Repeat
```

In continuous mode, conversion is repeated until "0" is written to the BUSY bit. (Writing "0" to the BUSY bit forces the operation to end.) If the operation is terminated forcibly, conversion stops before conversion is completed. (Upon a forced stop, the conversion register stores the last data that has been converted completely.)

■ Stop Mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits, pausing each time conversion for one channel is completed. To release pausing, activate the converter again.

After the conversion is completed for the end channel specified with the ANE bits, conversion is repeated from the analog inputs of the ANS. If the start and end channels are the same (ANS=ANE), conversion is performed only for one channel.

Example:

ANS = 0 0 0 , ANE = 0 1 1

Start --> AN0 --> End --> Restart --> AN1 --> End --> Restarte --> AN2 --> End -->
--> Restart --> AN3 --> End --> Restart -->AN0 Repeat

ANS = 0 1 0 , ANE = 0 1 0

Start --> AN2 --> End --> Restart --> AN2 --> End --> Restarte --> AN2 Repeat

Only the activation sources specified with STS1 and STS0 are used.

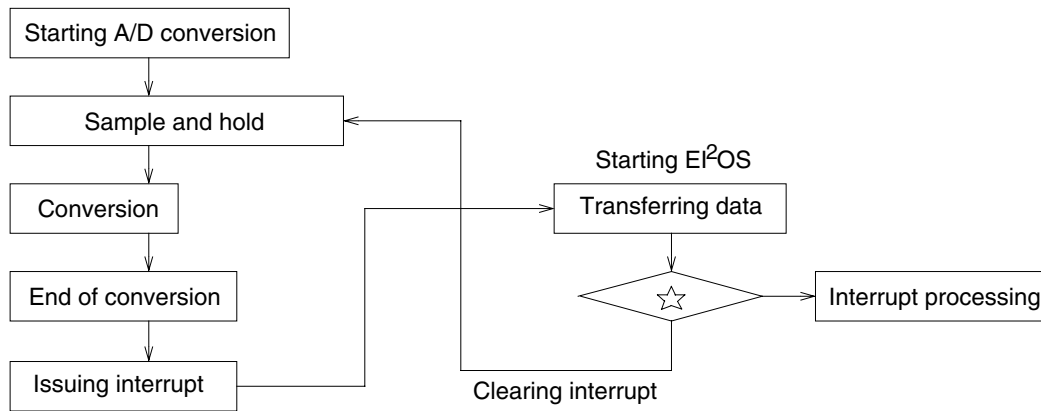
Using this mode, start of conversion can be synchronized with the activation source.

16.5 Conversion Using EI²OS

Figure 16.5-1 "A/D conversion processing flow from the start to converted data transfer (in continuous mode)" shows the processing flow from the start of A/D conversion to the transfer of converted data (in continuous mode).

■ Conversion Using EI²OS

Figure 16.5-1 A/D conversion processing flow from the start to converted data transfer (in continuous mode)



The portion indicated by the star (☆) is determined according to the EI²OS setting.

16.5.1 Starting EI²OS in Single Mode

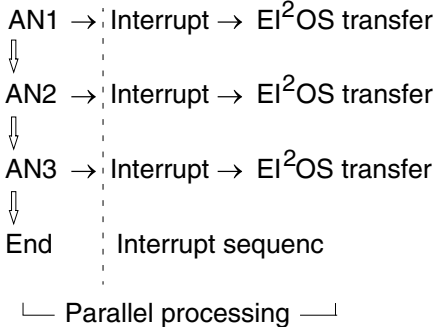
Follow the steps below to start the EI²OS in single mode.

- To terminate conversion after analog inputs AN1 to AN3 are converted
- To transfer conversion data sequentially to addresses 200H to 205H
- To start conversion by software
- To use the highest interrupt level

■ Starting EI²OS in Single Mode

Settings	Sample program	Function
EI ² OS setting	MOV ICR10 #08H	Specifies the highest interrupt level, EI ² OS activation upon an interrupt, and the descriptor address.
	MOV BAPL, #00H	Specifies the transfer destination address of converted data.
	MOV BAPM, #02H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is not terminated in response to a request from a resource.
	MOV I/OA, #36H	
	MOV DCT, #03H	EI ² OS transfer is performed three times. This count is the same as the conversion count.
A/D converter setting	MOV ADCS0 #0BH	Specifies single mode, start channel AN1, and end channel AN3.
	MOV ADCS1 #A2H	Specifies activation by software and start of A/D conversion.
Interrupt sequence	RET	Specifies return from an interrupt.

ICR10: Interrupt control register
 BAPL: Buffer address pointer, low-order
 BAPM: Buffer address pointer, medium-order
 BAPH: Buffer address pointer, high-order
 ISCS: EI²OS status register
 I/OA: I/O address counter
 DCT: Data counter



16.5.2 Starting EI²OS in Continuous Mode

Follow the steps below to start the EI²OS in continuous mode.

- To convert analog inputs AN3 to AN5 and obtain two conversion data items for each channel
- To transfer conversion data sequentially to addresses 600H to 60BH
- To start conversion by external edge input
- To use the highest interrupt level

■ Starting EI²OS in Continuous Mode

Settings	Sample program	Function
EI ² OS setting	MOV ICR10 #08H	Specifies the highest interrupt level, EI ² OS activation upon an interrupt, and the descriptor address.
	MOV BAPL, #00H	Specifies the transfer destination address of converted data.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is not terminated in response to a request from a resource.
	MOV I / OA, #36H	Transfer source address
	MOV DCT, #06H	EI ² OS transfer is performed six times. Data is transferred for three channels X 2.
A/D converter setting	MOV ADCS0 #9DH	Specifies continuous mode, start channel AN3, and end channel AN5.
	MOV ADCS1 #A4H	Specifies activation by external edge and start of A/D conversion.
Interrupt sequence	MOV ADCS1 #00H	Specifies return from an interrupt.
	RET	

ICR10: Interrupt control register

BAPL: Buffer address pointer, low-order

BAPM: Buffer address pointer, medium-order

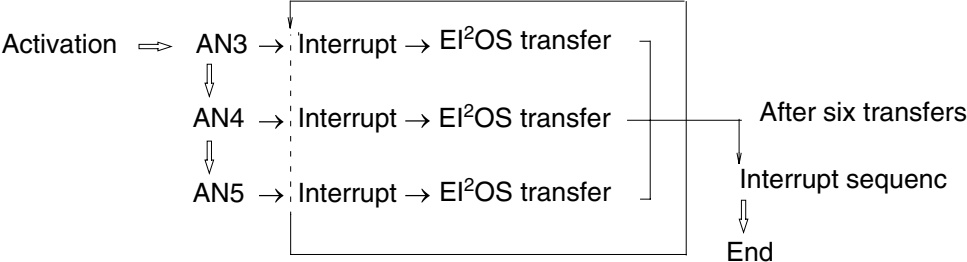
BAPH: Buffer address pointer, high-order

ISCS: EI²OS status register

I/OA: I/O address counter

DCT: Data counter

CHAPTER 16 A/D Converter



16.5.3 Starting EI²OS in Stop Mode

Follow the steps below to start the EI²OS in stop mode.

- To convert analog input AN3 12 times at fixed intervals
- To transfer conversion data sequentially to addresses 600H to 617H
- To start conversion by external edge input
- To use the highest interrupt level

■ Starting EI²OS in Stop Mode

Settings	Sample program	Function
EI ² OS setting	MOV ICR10 #08H	Specifies the highest interrupt level, EI ² OS activation upon an interrupt, and the descriptor address.
	MOV BAPL, #00H	Specifies the transfer destination address of converted data.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #18H	Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is not terminated in response to a request from a resource.
	MOV I / OA, #36H	Transfer source address
	MOV DCT, #0CH	EI ² OS transfer is performed 12 times.
A/D converter setting	MOV ADCS0 #DBH	Specifies stop mode, start channel AN3, and end channel AN3 (one-channel conversion).
	MOV ADCS1 #A4H	Specifies activation by external edge and start of A/D conversion.
Interrupt sequence	MOV ADCS1 #00H	Specifies return from an interrupt.
	RET	

ICR10: Interrupt control register

BAPL: Buffer address pointer, low-order

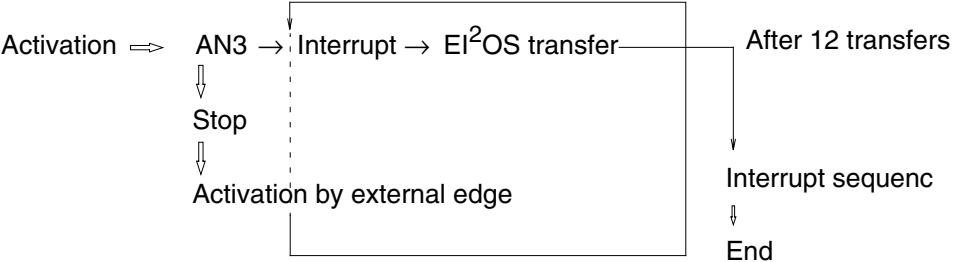
BAPM: Buffer address pointer, medium-order

BAPH: Buffer address pointer, high-order

ISCS: EI²OS status register

I/OA: I/O address counter

DCT: Data counter



16.6 Conversion Data Protection

The A/D converter has a conversion data protection function that enables continuous conversion and preservation of multiple data items using EI²OS.

Since there is only one conversion data register, its value is updated each time conversion is completed. Thus, continuous data conversion results in the loss of the previous data due to storage of the new data. To prevent this situation, the A/D converter pauses after conversion if the previous data item has not been transferred to memory by EI²OS. The converted data is not saved until the previous data is transferred to memory.

■ Conversion Data Protection

The pause is released after data is transferred to memory by EI²OS.

If the previous data has been transferred to memory, the A/D converter continues operation without pausing.

Note:

This function is related to the INT and INTE bits of ADCS1.

The data protection function operates only when interrupts are enabled (INTE=1).

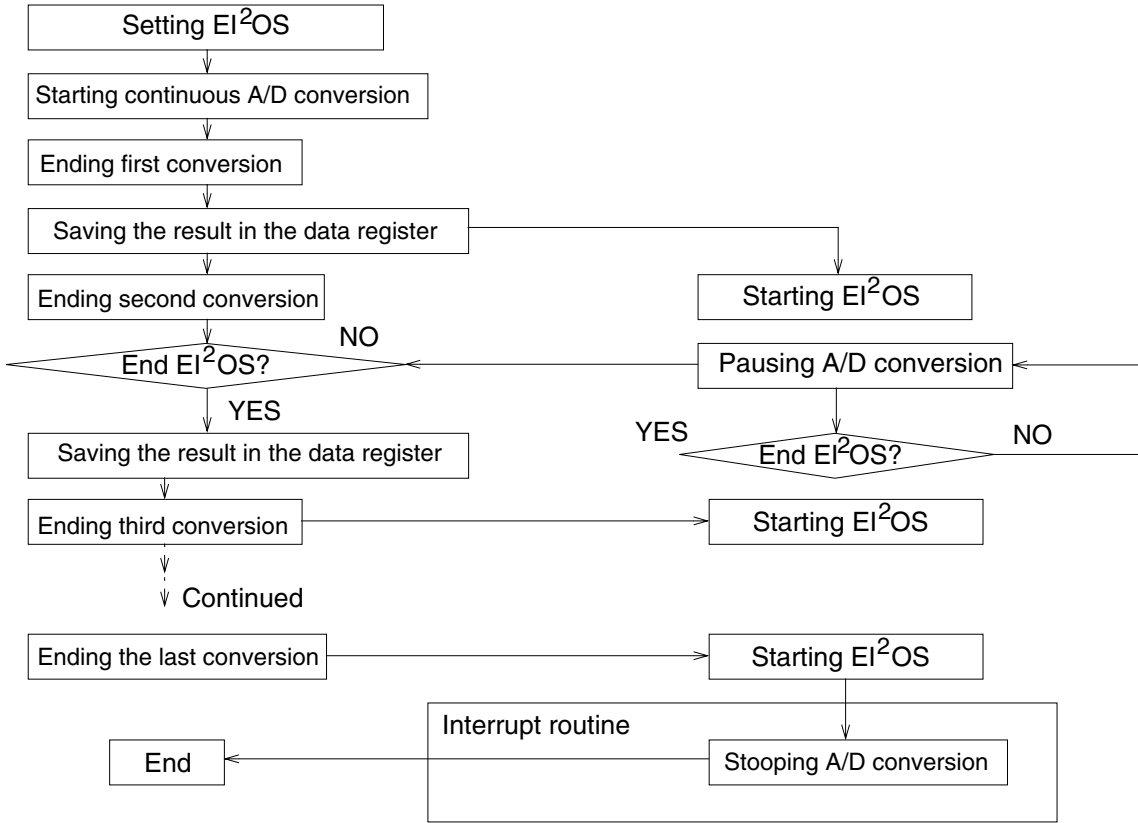
If interrupts are disabled (INTE=0), this function is disabled. Continuous A/D conversion results in loss of previous data, since the converted data items are saved to the register one after another.

If EI²OS is not used while interrupts are enabled (INTE=1), the INT bit is not cleared. Thus, the data protection function works and the A/D converter pauses. In this case, clearing the INT bit in the interrupt sequence releases the pause.

If the A/D converter is pausing during EI²OS operation, disabling interrupts may restart the A/D converter. In this case, the value in the conversion data register may be changed without being transferred.

Restarting the A/D converter while it is pausing destroys the standby data.

■ Flow of Data Protection Function (When EI²OS is Used)



■ Notes on using the conversion data protection function

To start the A/D converter upon an external trigger or internal timer, A/D activation factor bits STS1 and STS0 of the ADCS1 register are used. Ensure that the input values of the external trigger or internal timer are inactive. If the values are active, A/D conversion may start immediately.

When setting STS1 and STS0, ensure that "1" (input) is specified for ADTG and "0" (output) is specified for the internal timer (timer 2).

CHAPTER 17 UART0

This chapter explains the UART0 functions and operations.

- 17.1 "Feature of UART0"
- 17.2 "UART Block Diagram"
- 17.3 "UART Registers"
- 17.4 "UART0 Operation"
- 17.5 "Baud Rate"
- 17.6 "Internal and External Clock"
- 17.7 "Transfer Data Format"
- 17.8 "Parity Bit"
- 17.9 "Interrupt Generation and Flag Set Timings"
- 17.10 "UART0 Application Example"

17.1 Feature of UART0

The UART is a serial I/O port for asynchronous or CLK synchronous communication. The MB90590 Series contains three UART's. The following sections only describe the functionality of the UART 0. The remaining UART's have the identical function and the register addresses should be found in the I/O map.

■ Feature of UART0

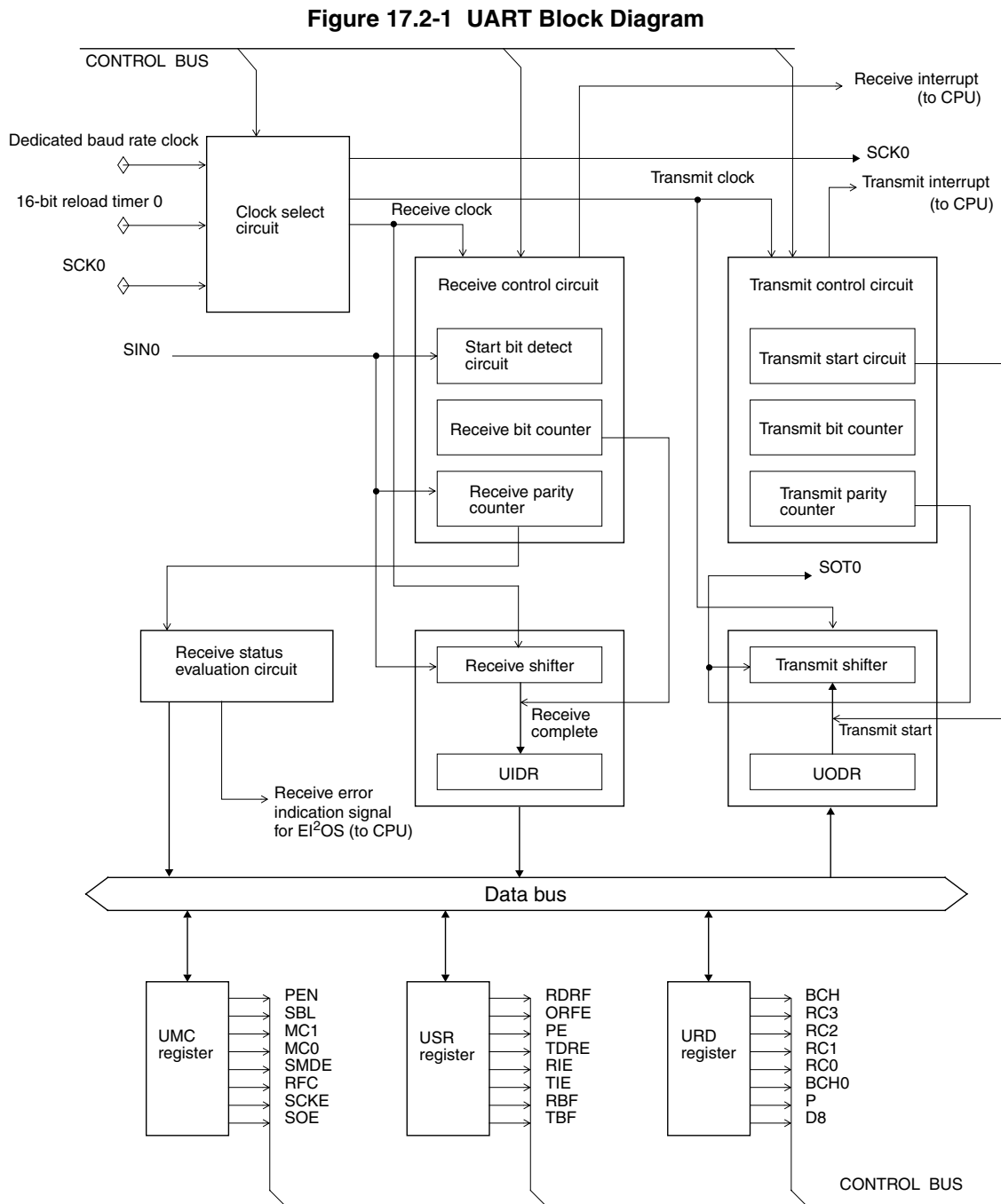
UART0 has the following features.

- Full duplex double buffer
- Supports CLK synchronous and CLK asynchronous start-stop data transfer.
- Multiprocessor mode support (mode 2)
- Internally dedicated baud rate generator (12 types)
- Supports flexible baud rate setting using an external clock input or internal timer.
- Variable data length (7 to 9 bits, [no parity]; 6 to 8 bits [with parity]).
- Error detect function (framing, overrun, and parity)
- Interrupt function (receive and transmit interrupts)
- NRZ type transfer format

17.2 UART Block Diagram

Figure 17.2-1 "UART Block Diagram" shows a block diagram of the UART.

■ UART Block Diagram



17.3 UART Registers

The UART has the following four registers:

- Serial mode control register
- Status register
- Input data register/output data register
- Rate and data register

■ UART Registers

Serial mode control register Address: ch0 000020H	7	6	5	4	3	2	1	0	Bit number
	PEN	SBL	MC1	MC0	SMDE	RFC	SCKE	SOE	UMC0
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	

Status register Address: ch0 000021H	15	14	13	12	11	10	9	8	Bit number
	RDRF	ORFE	PE	TDRE	RIE	TIE	RBF	TBF	USR0
Read/write	(R)	(R)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	
Initial value	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	

Input data register/ Output data register Address: ch0 000022H	7	6	5	4	3	2	1	0	Bit number
	D7	D6	D5	D4	D3	D2	D1	D0	UIDR0(read) UODR0(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)	

Rate and data register Address: ch0 000023H	15	14	13	12	11	10	9	8	Bit number
	BCH	RC3	RC2	RC1	RC0	BCH0	P	D8	URD0
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(X)	

17.3.1 Serial Mode Control Register (UMC)

UMC specifies the operation mode of UART0. Set the operation mode while operation is halted. However, the RFC bit can be accessed during operation.

■ Layout of Serial Mode Control Register (UMC)

Serial mode control register Address: ch0 000020H		7	6	5	4	3	2	1	0	Bit number UMC0
		PEN	SBL	MC1	MC0	SMDE	RFC	SCKE	SOE	
Read/write	⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value	⇒	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	

■ Serial Mode Control Register (UMC) Contents

[Bit 7] PEN (Parity enable)

Specifies whether to add (for transmit) or detect (for receive) a parity bit in serial data I/O. Set to "0" in mode 2.

0: Do not use parity

1: Use parity

[Bit 6] SBL (Stop bit length)

Specifies the number of stop bits for transmit data. For receive data, the first stop bit only is recognized and any second stop bit is ignored.

0: 1 bit length

1: 2 bits length

[Bits 5, 4] MC1, MC0 (Mode control)

These bits control the length of the transferred data. Table 17.3-1 "UART Operation Modes" lists the four transfer modes (data lengths) selectable by these bits.

Table 17.3-1 UART Operation Modes

Mode	MC1	MC0	Data Length* ¹
0	0	0	7 (6)
1	0	1	8 (7)
2 ^{*2}	1	0	8 + 1
3	1	1	9 (8)

*1: The figures enclosed in parentheses indicate the data length with parity.

*2: Mode 2 is used when a number of slave CPUs are connected to a single host CPU. As the receive parity check function cannot be used, set PEN in the UMC register to "0" (see Section 17.4 "UART0 Operation" for details). The transmit data length is 9 bits and no parity bit can be added.

[Bit 3] SMDE (Synchro mode enable)

This bit selects the transfer method.

0: Start-stop CLK synchronous transfer (clocked synchronous transfer using start and stop bits.)

1: Start-stop CLK asynchronous transfer

[Bit 2] RFC (Receiver flag clear)

Writing "0" to this bit clears the RDRF, ORFE, and PE flags in the USR register. Writing "1" has no effect. Reading always returns "1".

Note:

When receive interrupts are enabled during UART0 operation, only write "0" to RFC when either RDRF, ORFE, or PE is "1".

[Bit 1] SCKE (SCLK enable)

Writing "1" to this bit in CLK synchronous mode switches the port pin to the UART0 serial clock output pin and outputs the synchronizing clock. Set to 0 in CLK asynchronous mode or external clock mode.

0: The pin functions as a general purpose I/O port and does not output the serial clock. The pin functions as the external clock input pin when the port is set to input mode (DDR=0) and RC3 to 0 are set to "1111".

1: The pin functions as the UART0 serial clock output pin.

Note:

The corresponding bit of the Port Direction register should be set to "1" when the port pin is used as the clock output. This is for UART0 only.

[Bit 0] SOE (Serial output enable)

Writing "1" to this bit switches the port pin to the UART0 serial data output pin and enables serial output.

0: The pin functions as a port pin and does not output serial data.

1: The pin functions as the UART0 serial data output pin (SOT).

Note:

The corresponding bit of the Port Direction register should be set to "1" when the port pin is used as the serial output. This is for UART0 only.

17.3.2 Status Register (USR)

USR indicates the current state of the UART0 port.

■ Status Register (USR) Layout

Status register Address: ch0 000021H	15	14	13	12	11	10	9	8	Bit number
	RDRF	ORFE	PE	TDRE	RIE	TIE	RBF	TBF	USR0
Read/write	(R)	(R)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	
Initial value	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	

■ Status Register (USR) Contents

[Bit 15] RDRF (Receiver data register full)

This flag indicates the state of the UIDR0 (input data register). The flag is set when the receive data is loaded into UIDR0. Reading UIDR0 or writing "0" to RFC in the UMC0 register clears the flag. If RIE is active, a receive interrupt request is generated when RDRF is set.

0: No data in UIDR0

1: Data present in UIDR0

[Bit 14] ORFE (Over-run/framing error)

The flag is set when an overrun or framing error occurs in receiving. Writing "0" to RFC in the UMC0 register clears the flag. When this flag is set, the data in UIDR0 is invalid and the load from the receive shifter to UIDR0 is not performed. If RIE is active, a receive interrupt request is generated when ORFE is set.

0: No error

1: Error

Table 17.3-2 "UIDR State after Receive Completion" lists the UIDR0 states after receive completion by RDRF or ORFE.

Table 17.3-2 UIDR State after Receive Completion

RDRF	ORFE	UIDR0 Data State
0	0	Empty
0	1	Framing error
1	0	Valid data
1	1	Overrun error

The data in UIDR is invalid if an overrun or framing error has occurred. Next data can be received after clearing the flag(s).

[Bit 13] PE (Parity error)

The flag is set when a receive parity error occurs. Writing "0" to RFC in the UMC register clears the flag. When this flag is set, the data in UIDR0 is invalid and the load from the receive shifter to UIDR0 is not performed. If RIE is active, a receive interrupt request is generated when PE is set.

0: No parity error

1: Parity error

[Bit 12] TDRE (Transmitter data register empty)

This flag indicates the state of the UODR0 (output data register). Writing transmit data to the UODR0 register clears the flag. The flag is set when the data is loaded to the transmit shifter and the transmission is started. If TIE is active, a transmit interrupt request is generated when TDRE is set.

0: Data present in UODR0

1: No data in UODR0

[Bit 11] RIE (Receiver interrupt enable)

Enables receive interrupt requests.

0: Disable interrupts.

1: Enable interrupts.

[Bit 10] TIE (Transmitter interrupt enable)

Enables transmit interrupt requests. A transmit interrupt is generated immediately if transmit interrupts are enabled when TDRE is "1".

0: Disable interrupts.

1: Enable interrupts.

[Bit 9] RBF (Receiver busy flag)

This flag indicates that UART0 is receiving input data. The flag is set when the start bit is detected and cleared when the stop bit is detected.

0: Receiver idle

1: Receiver busy

[Bit 8] TBF (Transmitter busy flag)

This flag indicates that UART0 is transmitting input data. The flag is set when transmit data is written to the UODR0 register and cleared when transmission completes.

0: Transmitter idle

1: Transmitter busy

17.3.3 Input Data Register (UIDR) and Output Data Register (UODR)

UIDR (input data register) is the serial data input register. UODR (output data register) is the serial data output register.

The most significant two bits (D7 and D6) are ignored if the data length is 6 bits and the most significant bit (D7) is ignored if the data length is 7 bits. Write to UODR only when TDRE = "1" in the USR register. Read UIDR only when RDRF = "1" in the USR register.

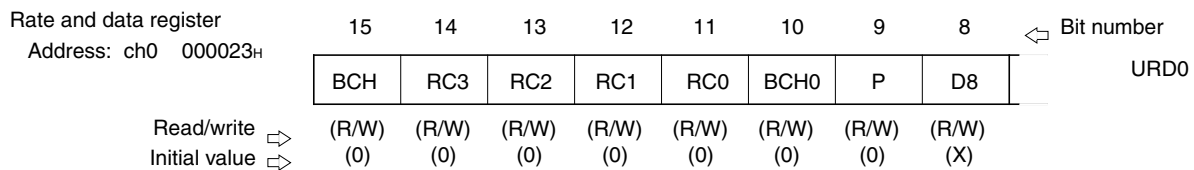
■ Input Data Register (UIDR) and Output Data Register (UODR)

Input data register/ Output data register Address: ch0 000022H		7	6	5	4	3	2	1	0	Bit number
		D7	D6	D5	D4	D3	D2	D1	D0	UIDR0(read) UODR0(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)	

17.3.4 Rate and Data Register (URD)

URD selects the data transfer speed (baud rate) for UART0. The register also holds the most significant bit (bit 8) of the data when the transmit data length is 9 bits. Set the baud rate and parity when UART0 is halted.

■ Layout of Rate and Data Register (URD)



■ Rate and Data Register (URD) Contents

[Bits 15, 10] BCH, BCH0 (Baud rate clock change)

Specifies the machine cycles for the baud rate clock (see Section 17.4 "UART0 Operation" for details).

Table 17.3-3 Clock Input Selection

BCH	BCH0	Divider ratio	Setting Example for Each Machine Cycle
0	0	-	- Prohibited setting -
0	1	Divide by 4	For a 16-MHz machine cycle: 16/4 = 4 MHz
1	0	Divide by 3	For a 12-MHz machine cycle: 12/3 = 4 MHz
1	1	Divide by 5	For a 10-MHz machine cycle: 10/5 = 2 MHz

Note:

Do not set BCH and BCH0 to "00".

[Bits 14 to 11] RC3, RC2, RC1, RC0 (Rate control)

Selects the clock input for the UART0 port (see Section 17.4 "UART0 Operation" for details).

Table 17.3-4 Clock Input Selection

RC3 to RC0	Clock Input
"0000" to "1011"	Dedicated baud rate generator
"1101"	16-bit Reload Timer 0
"1111"	External clock

Note:

Do not set the rate control bits to "1100" "1110".

[Bit 9] P

Sets even or odd parity when parity is active (PEN = "1").

0: Even parity

1: Odd parity

[Bit 8] D8

Holds the bit 8 of the transfer data in mode 2 or 3 (9-bit data length) and no parity. Treated as bit 8 of the UIDR0 register for reading. Treated as bit 8 of the UODR register for writing. The bit has no meaning in the other modes. Write to D8 only when TDRE = "1" in the USR0 register.

17.4 UART0 Operation

Table 17.4-1 "UART0 Operating Modes" lists the operating modes for UART0. Set the UMC register to switch between modes.

■ **UART0 Operation Modes**

Table 17.4-1 UART0 Operating Modes

Mode	Parity	Data Length	Clock Mode	Length of Stop Bits*
0	On	6	CLK asynchronous or CLK synchronous	1 bit or 2 bits
	Off	7		
1	On	7		
	Off	8		
2	Off	8+1		
3	On	8		
	Off	9		

*: The number of stop bits can only be set for transmission. The number of receive stop bits is always set to one. Do not set modes other than those listed above. UART0 does not operate if an invalid mode is set.

Note:

UART0 uses start-stop clock synchronous transfer. Therefore, a start and stop bit are added to the data even in clock synchronous transfer.

17.5 Baud Rate

When the dedicated baud rate generator is used, the following two types of baud rates are available:

- CLK synchronous baud rate
 - CLK asynchronous baud rate
-

■ CLK Synchronous Baud Rate

The five URD register bits: BCH, BCH0 and RC3, RC2, RC1 select the baud rate for CLK synchronous transfer.

First select the machine clock divider ratio using BCH and BCH0.

BCH	BCH0		
0	1	-->	Divide by 4 [For example, at 16 MHz: $16/4 = 4$ MHz]
1	0	-->	Divide by 3 [For example, at 12 MHz: $12/3 = 4$ MHz]
1	1	-->	Divide by 5 [For example, at 10 MHz: $10/5 = 2$ MHz]

Then, set the division ratio for the clock selected above in RC3, RC2, and RC1. The following three settings are available for CLK synchronous transfer. Other settings are prohibited.

RC3	RC2	RC1		
0	1	0	-->	Divide by 2 [For example, at 4 MHz: $4/2 = 2.0$ M (bps)]
0	1	1	-->	Divide by 4 [For example, at 4 MHz: $4/4 = 1.0$ M (bps)]
1	0	0	-->	Divide by 8 [For example, at 4 MHz: $4/8 = 0.5$ M (bps)] (At 2 MHz, the speed becomes half the above examples.)

■ CLK Asynchronous Baud Rate

The six URD register bits: BCH, BCH0 and RC3, RC2, RC1, RC0 select the baud rate for CLK asynchronous transfer.

First select the machine clock divider ratio using BCH and BCH0.

BCH	BCH0		
0	1	-->	Divide by 4 [For example, at 16 MHz: $16/4 = 4$ MHz]
1	0	-->	Divide by 3 [For example, at 12 MHz: $12/3 = 4$ MHz]
1	1	-->	Divide by 5 [For example, at 10 MHz: $10/5 = 2$ MHz]

Then, set the asynchronous transfer clock division ratio for the clock selected above in RC3, RC2, RC1, and RC0. The following settings are available.

Table 17.5-1 Baud Rate (Continued)

				CLK asynchronous ($\mu\text{s}/\text{Baud}$)			CLK asynchron ous divider ratio	CLK synchronous ($\mu\text{s}/\text{Baud}$)		
				16 MHz	12 MHz	10 MHz		16 MHz	12 MHz	10 MHz
RC 3	RC 2	RC 1	RC 0	BCH/ 0=01	BCH/ 0=10	BCH/ 0=11		BCH/ 0=01	BCH/ 0=10	BCH/ 0=11
0	1	1	1	104/ 9615	104/ 9615	208/ 4808	8 X 13	1 / 1M	1 / 1M	2 / 500K
1	0	0	0	192/ 5208	192/ 5208	-	8 X 12	-	-	-
1	0	0	1	208/ 4808	208/ 4808	416/ 2404	8 X 13	2 / 500K	2 / 500K	4 / 250K
1	0	1	0	-	-	-	8			
1	0	1	1	16/ 62500	16/ 62500	32/ 31250	8	-	-	-

17.6 Internal and External Clock

Setting RC3 to 0 to "1101" selects the clock signal from the 16-bit Reload Timer. Setting RC3 to 0 to "1111" selects the external clock. The external clock frequency has a maximum value of 2 MHz.

■ Internal and External Clock

The CLK asynchronous baud rate is the CLK synchronous baud rate divided by 8. Also, data transfer is possible if the CLK asynchronous baud rate is in the range -1% to +1% of the selected baud rate. Table 17.6-1 "Baud Rate and Reload Value" lists the baud rates when the internal timer is selected as the clock. The values in this table are calculated for a machine cycle of 7.3728 MHz. However, do not use the settings marked as "_" in the table.

$$\text{Baud rate} = \frac{\phi / X}{8 \times 2^{(n+1)}} \quad [\text{bps}]$$

(ϕ : Machine cycle
 X: Divider ratio for the count clock source for the internal timer
 n: Reload value (decimal))

Table 17.6-1 Baud Rate and Reload Value

Baud Rate	Reload Value	
	X = 2 ¹ (divide machine cycle by 2)	X = 2 ³ (divide machine cycle by 8)
76800	2	-
38400	5	-
19200	11	2
9600	23	5
4800	47	11
2400	95	23
1200	191	47
600	383	95
300	767	191

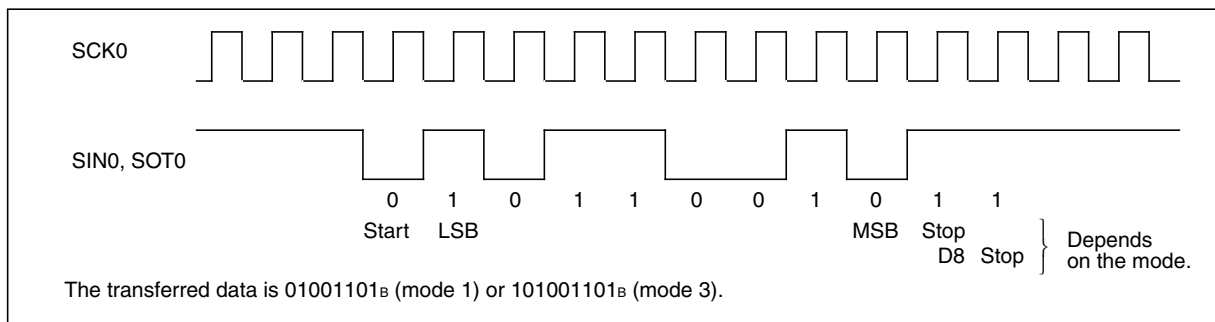
The values in the table are the reload values (decimal) for reload count operation of the 16-bit Reload Timer.

17.7 Transfer Data Format

UART0 only handles NRZ (non-return-to-zero) type data. Figure 17.7-1 "Transfer Data Format" shows the relationship between the transmit/receive clock and the data for CLK synchronous mode.

■ Transfer Data Format

Figure 17.7-1 Transfer Data Format



As shown in Figure 17.7-1 "Transfer Data Format", the transfer data always starts with the start bit (L level data), the specified number of data bits are transmitted with the LSB first, then transmission ends with the stop bit ("H" level data). Always input a clock if external clock operation is selected. When an internal clock (the dedicated baud rate generator or 16-bit Reload Timer) is selected, the clock is output continuously. When using CLK synchronous transfer, do not start data transfer until the selected baud rate clock has stabilized (for two baud rate clock cycles).

When using CLK asynchronous transfer, set the SCKE bit in the UMC0 register to "0" to disable clock output. The transfer data format of SINO and SOUT0 is the same as shown in Figure 17.7-1 "Transfer Data Format".

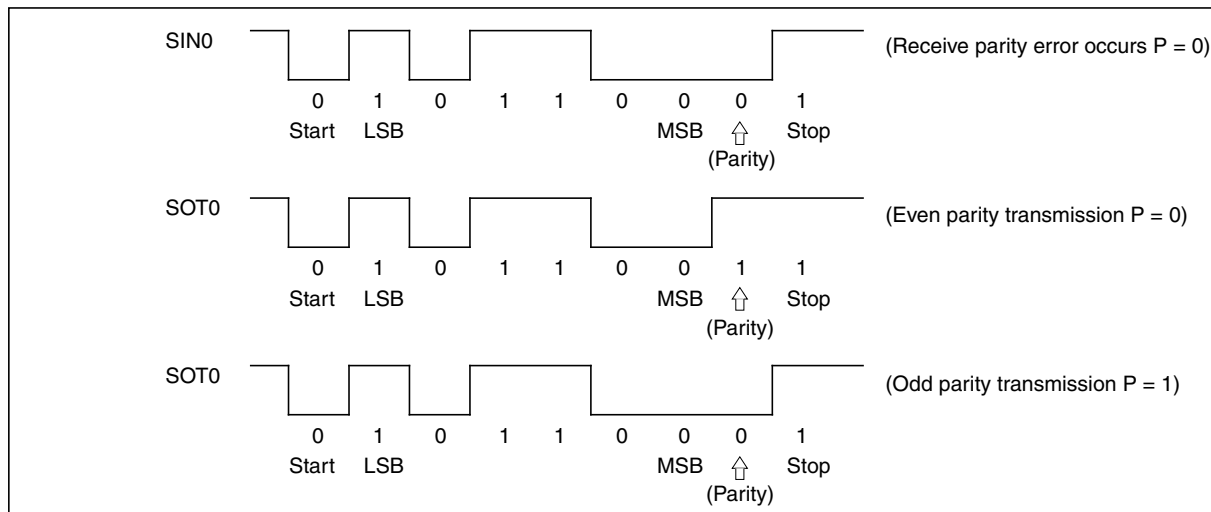
17.8 Parity Bit

The P bit in the URD0 register specifies whether to use even or odd parity when parity is enabled. The PEN bit in the UMC0 register enables parity.

■ Parity Bit

Inputting the data shown in Figure 17.8-1 "Serial Data with Parity Enabled" to SIN when even parity is set causes a receive parity error. Figure 17.8-1 "Serial Data with Parity Enabled" also shows the data transmitted when sending 001101_B with even parity and odd parity.

Figure 17.8-1 Serial Data with Parity Enabled



17.9 Interrupt Generation and Flag Set Timings

UART0 has two interrupt causes and six flags. The two interrupt causes are the receive and transmit interrupts. The six flags are RDRF, ORFE, PE, TDRE, RBF, and TBF. For reception, the RDRF, ORFE, and PE flags request an interrupt. For transmission, the TDRE flag requests an interrupt.

■ Set Timings of the Six Flags

○ RDRF flag

The RDRF flag is set when receive data is loaded into the UIDR register. The flag is cleared by writing "0" to RFC in the UMC0 register or by reading the UIDR0 register.

○ ORFE flag

The ORFE flag is an overrun or framing error flag. The flag is set when a receive error occurs and is cleared by writing "0" to RFC in the UMC0 register.

○ PE flag

The PE flag is a reception parity error flag. The flag is set when a receive parity error occurs and is cleared by writing "0" to RFC in the UMC0 register. Note that the parity detect function is not available in mode 2.

○ TDRE flag

The TDRE flag is set when the UODR0 register becomes empty and is available for writing. The flag is cleared by writing to the UODR0 register. The above four flags (RDRF, ORFE, PE, and TDRE) trigger transmit or receive interrupts.

○ RBF and TBF flags

The RBF and TBF flags indicate that reception or transmission is in progress. The RBF flag becomes active during reception, and the TBF flag becomes active during transmission.

17.9.1 Flag Set Timings for a Receive Operation (in Mode 0, 1, or 3)

The RDRF, ORFE, and PE flags are set and an interrupt request to the CPU generated when the final stop bit is detected indicating the end of reception transfer. The data in UIDR0 is invalid when either the ORFE or PE bit is active.

■ Flag set Timings for a Receive Operation (in Mode 0, 1, or 3)

Figure 17.9-1 "RDRF Set Timing (Mode 0, 1, or 3)", Figure 17.9-2 "ORFE Set Timing (Mode 0, 1, or 3)", and Figure 17.9-3 "PE Set Timing (Mode 0, 1, or 3)" show the set timings of the RDRF, ORFE, and PE flags respectively.

Figure 17.9-1 RDRF Set Timing (Mode 0, 1, or 3)

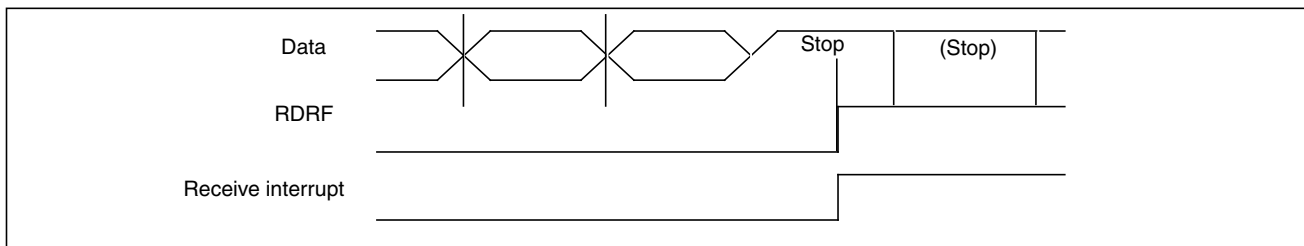


Figure 17.9-2 ORFE Set Timing (Mode 0, 1, or 3)

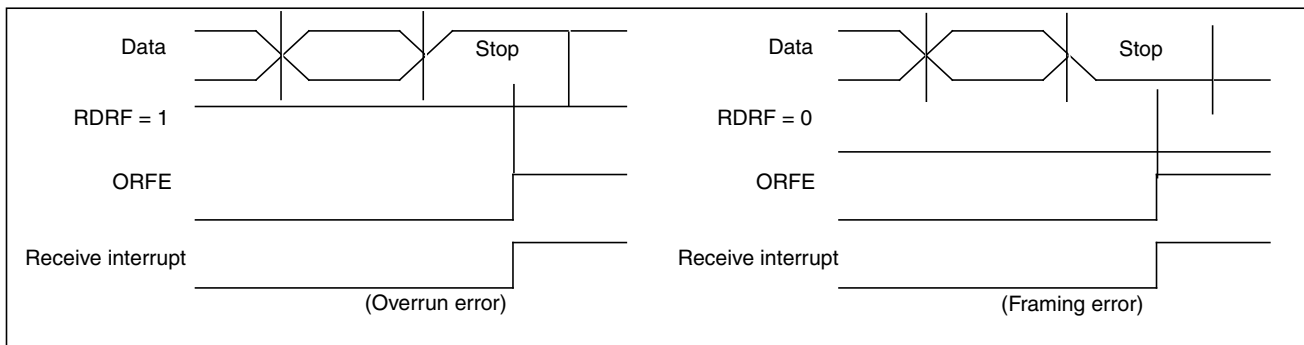
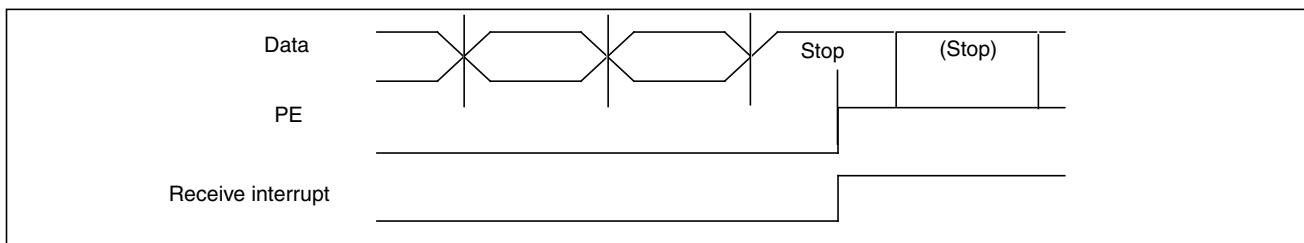


Figure 17.9-3 PE Set Timing (Mode 0, 1, or 3)



17.9.2 Flag Set Timings for a Receive Operation (in Mode 2)

The RDRF flag is set when the final stop bit is detected and reception transfer ends with the last data bit (D8) having the value "1".

The ORFE flag is set when the final stop bit is detected, irrespective of the value of the last data bit (D8). The data in UIDR0 is invalid when the ORFE bit is active.

The interrupt request to the CPU is generated when either of the flags are set (see Section 17.10 "UART0 Application Example" for details on using mode 2).

■ Flag Set Timings for a Receive Operation (in Mode 2)

Figure 17.9-4 RDRF Set Timing (Mode 2)

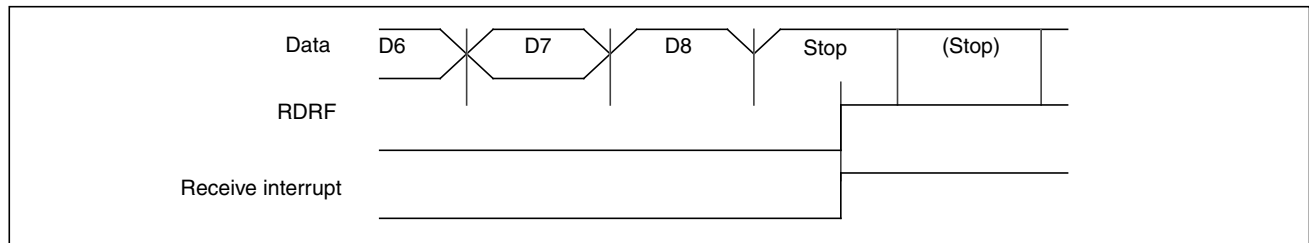
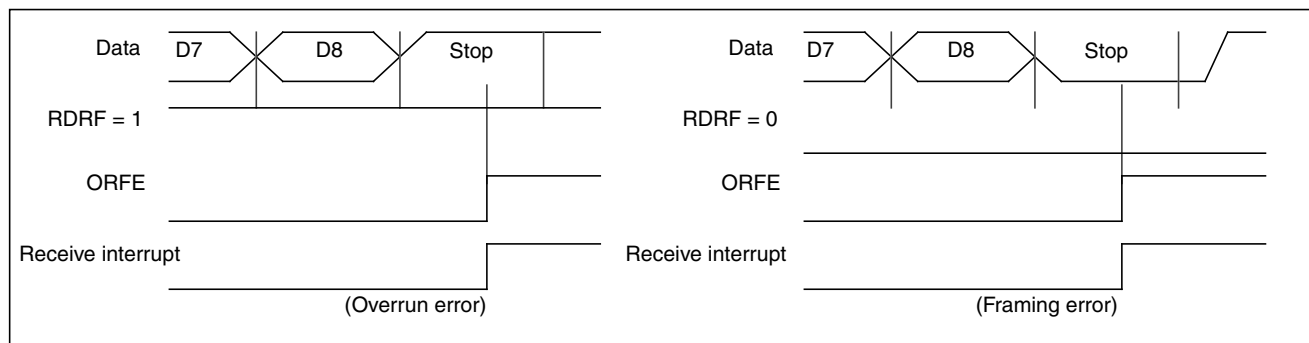


Figure 17.9-5 ORFE Set Timing (Mode 2)

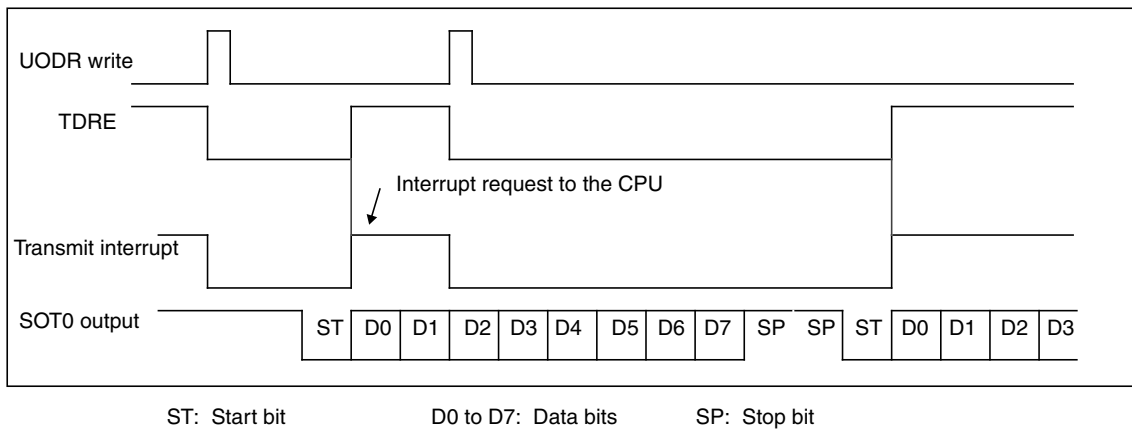


17.9.3 Flag Set Timings for a Transmit Operation

TDRE is set and an interrupt request to the CPU is generated when the data written in UODR0 register is transferred to the internal shift register and the next data can be written to UODR0.

■ Flag Set Timings for a Transmit Operation

Figure 17.9-6 TDRE Set Timing (Mode 0)



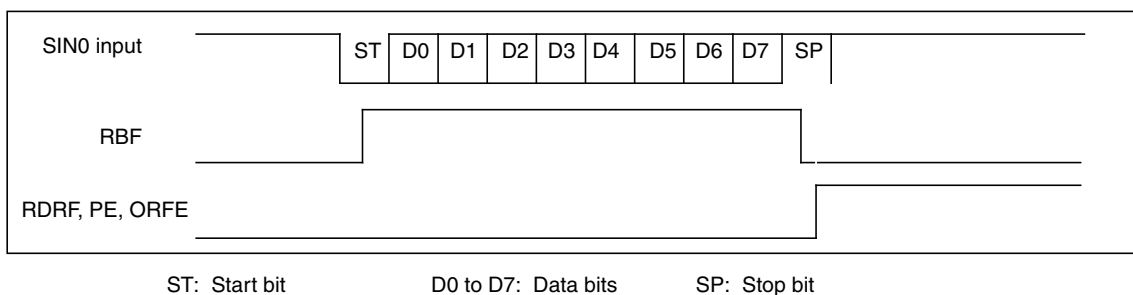
17.9.4 Status Flag During Transmit and Receive Operation

RBF is set when the start bit is detected and cleared when a stop bit is detected. The receive data in UIDR0 at the RBF clear timing is not yet valid. The data in UIDR0 becomes valid at the RDRF set timing.

■ Status Flag during Transmit and Receive Operation

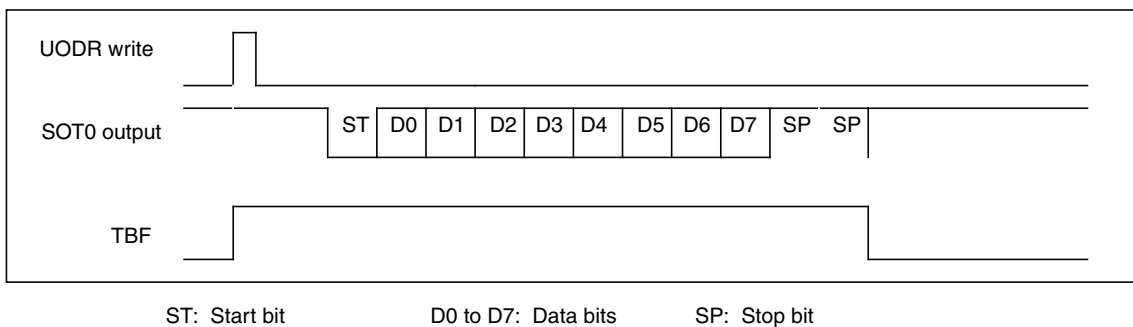
Figure 17.9-7 "RBF Set Timing (Mode 0)" shows the relationship between the RBF and receive interrupt flag timing.

Figure 17.9-7 RBF Set Timing (Mode 0)



Writing the transmission data to UODR0 sets TBF. TBF is cleared when transmission completes.

Figure 17.9-8 TBF Set Timing (Mode 0)



Note:

Receive operation starts after releasing a reset unless the SIN input pin is fixed at "1". Therefore, before setting the mode, write "0" to RFC in the UMC0 register to clear any receive flags that have been set.

Set the communication mode when the RBF and TBF flags in the USR0 register are "0". The data transmitted and received during mode setting cannot be guaranteed.

■ EI²OS (Extended intelligent I/O service)

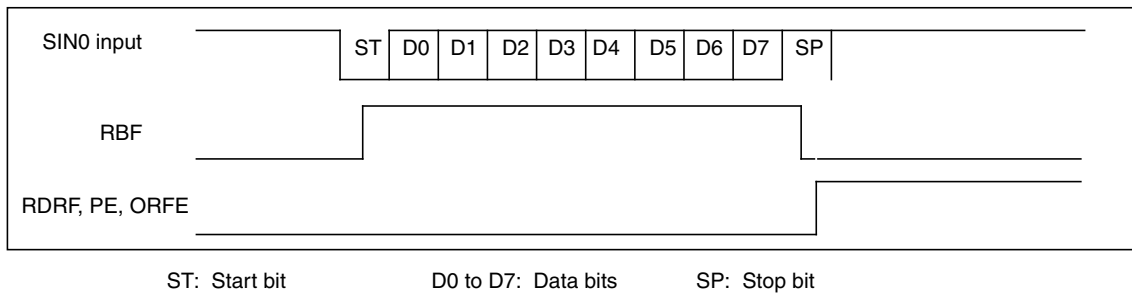
See the Section 3.7 "Extended intelligent I/O Service (EI²OS)" for details on EI²OS.

17.10 UART0 Application Example

Mode 2 is used when a number of slave CPUs are connected to a host CPU (see Figure 17.10-1 "RBF Set Timing (mode 0)".)

■ Application Example

Figure 17.10-1 RBF Set Timing (Mode 0)



As shown in Figure 17.10-2 "Example System Configuration Using Mode 2", communication starts with the host CPU transmitting address data. The ninth bit (D8) of the address data is set to "1". The address selects the slave CPU with which communication will be established. The selected slave CPU communicates with the host CPU using a protocol determined by the user. In normal data, D8 is set to "0". Unselected slave CPUs wait in standby until the next communication session starts. Figure 17.10-3 "Communication Flowchart for Mode 2 Operation" shows a flowchart of operation in this mode.

Because the parity check function is not available in this mode, set the PEN bit in the UMC0 register to "0".

Figure 17.10-2 Example System Configuration Using Mode 2

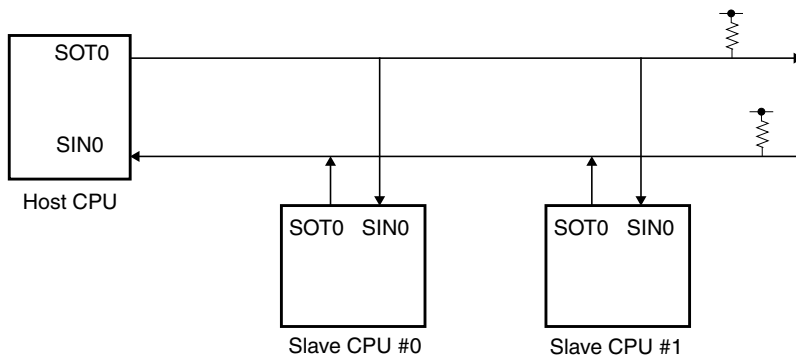
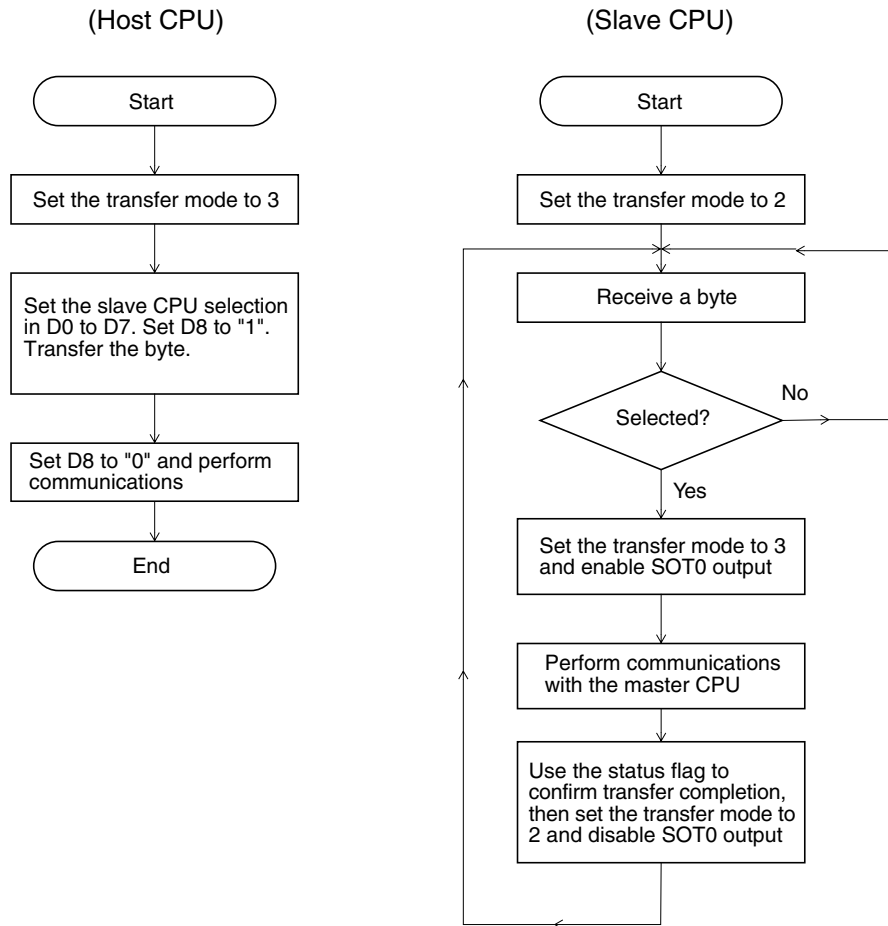


Figure 17.10-3 Communication Flowchart for Mode 2 Operation



CHAPTER 18 SERIAL I/O

This chapter explains the functions and operations of the serial I/O.

- 18.1 "Outline of Serial I/O"
- 18.2 "Serial I/O Registers"
- 18.3 "Serial I/O Prescaler (CDCR)"
- 18.4 "Serial I/O Operation"
- 18.5 "Negative Clock Operation"

18.1 Outline of Serial I/O

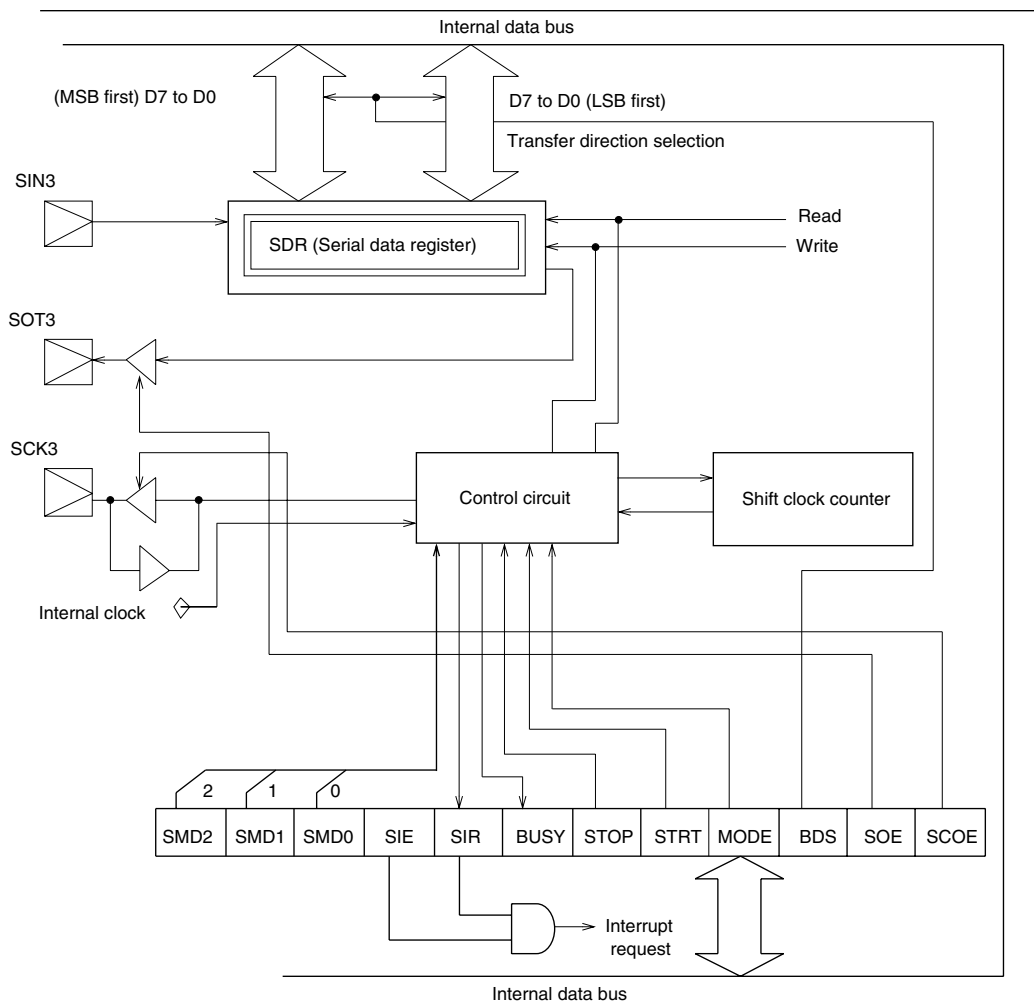
The serial I/O interface operates in two modes:

- **Internal shift clock mode:** Data is transferred in synchronization with the internal clock.
- **External shift clock mode:** Data is transferred in synchronization with the clock supplied via the external pin (SCK3). By manipulating the general-purpose port sharing the external pin (SCK3), data can also be transferred by a CPU instruction in this mode.

Serial I/O Block Diagram

This block is a serial I/O interface that allows data transfer using clock synchronization. The interface consists of a single eight-bit channel. Data can be transferred from the LSB or MSB.

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



18.2 Serial I/O Registers

The serial I/O has the following two registers:

- Serial mode control status register (SMCS)
 - Serial data register (SDR)
-

■ Serial I/O Registers

	15	14	13	12	11	10	9	8	
Address : 00002D _H	SMD2	SMD1	SMD0	SIE	SIR	BUSY	STOP	STRT	Serial mode control status register (SMCS)

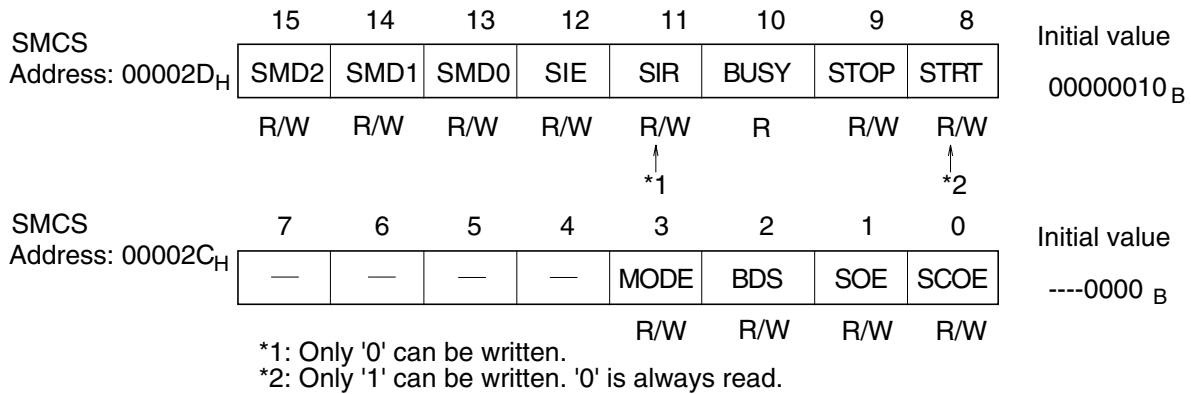
	7	6	5	4	3	2	1	0
Address : 00002C _H	—	—	—	—	MODE	BDS	SOE	SCOE

	7	6	5	4	3	2	1	0	
Address : 00002E _H	D7	D6	D5	D4	D3	D2	D1	D0	Serial data register (SDR)

18.2.1 Serial Mode Control Status Register (SMCS)

The serial mode control status register (SMCS) controls the serial I/O transfer mode.

■ Serial Mode Control Status Register (SMCS)



■ Bit functions of Serial Mode Control Status Register (SMCS)

[bit 3] Serial mode selection bit (MODE)

The serial mode selection bit is used to select the conditions to start the transfer operation from the stop state. This bit must not be updated during operation.

Table 18.2-1 Setting the Serial Mode Selection Bit

MODE	Operation
0	Transfer starts when STRT=1. [Default]
1	Transfer starts when the serial data register is read or written to.

This bit is initialized to a "0" upon a reset, and can be read or written to. To activate the intelligent I/O service, ensure that "1" is written to this bit.

[bit 2] Bit direction select bit (BDS)

When serial data is input or output, this bit determines from which bit data is to be transferred first, the least significant bit (LSB first) or the most significant bit (MSB first), as shown in Table 18.2-2 "Setting the Transfer Direction Selection Bit".

Table 18.2-2 Setting the Transfer Direction Selection Bit

0	LSB first [default]
1	MSB first

Note:

Specify the bit ordering before any data is written to SDR.

[bit 1] Serial output enable bit (SOE: Serial out enable)

This bit controls the output from the serial I/O output external pins (SOT3) as shown in Table 18.2-3 "Setting the Serial Output Enable Bit".

Table 18.2-3 Setting the Serial Output Enable Bit

0	General-purpose port pin [default]
1	Serial data output

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 0] Shift clock output enable bit (SCOE: SCK3 output enable)

This bit controls the output from the shift clock I/O output external pins (SCK3) as shown in Table 18.2-4 "Setting the Shift Clock Output Enable Bit".

Table 18.2-4 Setting the Shift Clock Output Enable Bit

0	General-purpose port pin, transfer for each instruction [default]
1	Shift clock output pin

Ensure that "0" is written to this bit when data is transferred for each instruction in external shift clock mode.

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[Bits 15, 14, and 13] Shift clock selection bits (SMD2, SMD1, SMD0: Serial shift clock mode)

These bits are used to select the serial shift clock mode, as shown in Table 18.2-5 "Setting the Serial Shift Clock Mode".

Table 18.2-5 Setting the Serial Shift Clock Mode

SMD2	SMD1	SMD0	$\phi=16\text{MHz}$ div=4	$\phi=8\text{MHz}$ div=4	$\phi=4\text{MHz}$ div=4
0	0	0	2 MHz	1 MHz	500 kHz
0	0	1	1 MHz	500 kHz	250 kHz
0	1	0	250 kHz	125 kHz	62.5 kHz
0	1	1	125 kHz	62.5 kHz	31.25 kHz
1	0	0	62.5 kHz	31.25 kHz	5.625 kHz
1	0	1	External shift clock mode		
1	1	0	Reserved		
1	1	1	Reserved		

div	M1	DIV3	DIV2	DIV1	DIV0	Recommended machine cycle
3	1	1	1	0	1	6 MHz
4	1	1	1	0	0	8 MHz
5	1	1	0	1	1	10 MHz
6	1	1	0	1	0	12 MHz
7	1	1	0	0	1	14 MHz
8	1	1	0	0	0	16 MHz

Setting of the Serial I/O prescaler (CDCR)

* For details, see 18.3 "Serial I/O Prescaler (CDCR)".

These bits are initialized to "000" upon a reset. These bits must not be updated during data transfer.

Five types of internal shift clock and an external shift clock are available. Do not set 110 or 111 in SMD2, SMD1, and SMD0 as these values are reserved.

Shift operation can be performed for each instruction by specifying SCOE =0 during clock selection and by using the ports that share the SCK3 pin.

[bit 12] Serial I/O interrupt enable bit (SIE: Serial I/O interrupt enable)

This bit controls the serial I/O interrupt request as shown in Table 18.2-6 "Setting the Interrupt Request Enable Bit".

Table 18.2-6 Setting the Interrupt Request Enable Bit

0	Serial I/O interrupt disabled [initial value]
1	Serial I/O interrupt enabled

This bit is initialized to "0" upon a reset. This bit is readable and writable.

[bit 11] Serial I/O interrupt request bit (SIR: Serial I/O interrupt request)

When serial data transfer is completed, "1" is set to this bit. If this bit is set while interrupts are enabled (SIE=1), an interrupt request is issued to the CPU. The clear condition varies with the MODE bit.

When "0" is written to the MODE bit, the SIR bit is cleared by writing "0". When "1" is written to the MODE bit, the SIR bit is cleared by reading or writing to SDR. When the system is reset or "1" is written to the STOP bit, the SIR bit is cleared regardless of the MODE bit value.

Writing "1" to the SIR bit has no effect. "1" is always read by a read operation of a read-modify-write instruction.

[bit 10] Transfer status bit (BUSY)

The transfer status bit indicates whether serial transfer is being executed.

Table 18.2-7 Setting the Transfer Status Bit

BUSY	Operating
0	Stopped, or standing by for serial data register R/W [default]
1	Serial transfer

This bit is initialized to "0" upon a reset. This is a read-only bit.

[bit 9] Stop bit (STOP)

The stop bit forcibly terminates serial transfer. When "1" is written to this bit, the transfer is stopped.

Table 18.2-8 Setting the Stop Bit

STOP	Operating
0	Normal operation
1	Transfer stop by STOP=1 [initial value]

This bit is initialized to "1" upon a reset. This bit is readable and writable.

[bit 8] Start bit (STRT: Start)

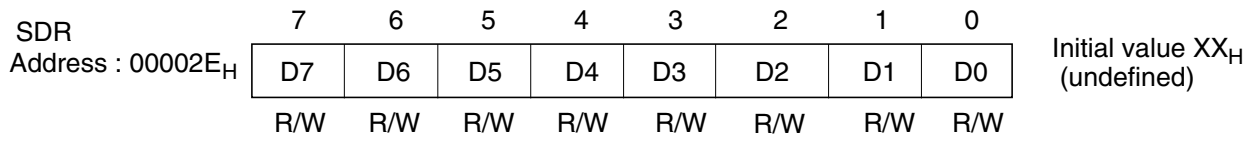
The start bit activates serial transfer. Writing "1" to this bit starts the data transfer when the MODE bit is set to 0. When the MODE bit is set to 1 and the STRT bit is set to 1, writing the data into serial data register starts the transfer.

Writing "1" is ignored while the system is performing serial transfer or standing by for a serial shift register read or write. Writing "0" has no effect. "0" is always read.

18.2.2 Serial Shift Data Register (SDR)

This serial data register stores the serial I/O transfer data. During transfer, the SDR must not be read or written to.

■ Serial Shift Data Register (SDR)

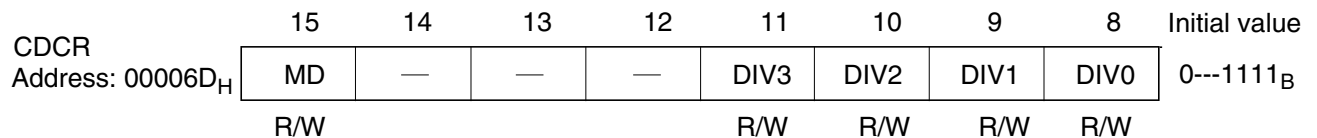


18.3 Serial I/O Prescaler (CDCR)

The Serial I/O Prescaler provides the shift clock for the Serial I/O.

The operation clock for the Serial I/O is obtained by dividing the machine clock. The Serial I/O is designed so that a constant baud rate can be obtained for a variety of machine clocks by the user of the communication prescaler. The CDCR register controls the machine clock division.

■ Serial I/O Prescaler (CDCR)



[bit 15] MD (Machine clock divide mode select):

This bit is used to control the operation of the communication prescaler.

0: The Serial I/O Prescaler is disabled.

1: The Serial I/O Prescaler is enabled.

[bits 11, 10, 9, and 8] DIV3 to DIV0 (Divide 3 to 0):

These bits are used to determine the machine clock division ratio.

Table 18.3-1 Machine Clock Division Ratio

DIV3 to 0	Division ratio
1101 _B	3
1100 _B	4
1011 _B	5
1010 _B	6
1001 _B	7
1000 _B	8

Note:

When the division ratio is changed, allow two cycles for the clock to stabilize before starting communication.

18.4 Serial I/O Operation

The extended serial I/O consists of the serial mode control status register (SMCS) and shift register (SDR), and is used for input and output of 8-bit serial data.

■ Serial I/O Operation

The bits in the shift register are serially output via the serial output pin (SOT3 pin) at the falling edge of the serial shift clock (external clock or internal clock). The bits are serially input to the shift register (SDR) via the serial input pin (SIN3 pin) at the rising edge of the serial shift clock. The shift direction (transfer from MSB or LSB) is specified by the direction specification bit (BDS) of the serial mode control status register (SMCS).

At the end of serial data transfer, this block is stopped or stands by for a read or write of the data register according to the MODE bit of the serial mode control status register (SMCS). To start transfer from the stop or standby state, follow the procedure below.

- **To resume operation from the stop state, write '0' to the STOP bit and '1' to the STRT bit. (The STOP and STRT bits can be set simultaneously.)**

- **To resume operation from the serial shift data register R/W standby state, read or write to the data register.**

18.4.1 Shift Clock

There are two modes of shift clock: internal or external shift clock. These two modes are selected by setting the SMCS. To switch the modes, ensure that serial I/O transfer is stopped. To check whether the serial I/O transfer is stopped, read the BUSY bit.

■ Internal Shift Clock Mode

In internal shift clock mode, data transfer is based on the internal clock. As a synchronization timing output, a shift clock of 50% duty ratio can be output from the SCK3 pin. Data is transferred at one bit per clock. The transfer speed is expressed as follows:

$$\text{Transfer speed (s)} = \frac{A \times \text{div}}{\text{Internal clock machine cycle (Hz)}}$$

"A" is the division ratio indicated by the SMD bits of SMCS. The value can be 2^1 , 2^2 , 2^4 , 2^5 , or 2^6 .

Table 18.4-1 Formulas for Calculation Baud Rate in Internal Shift Clock Mode

SMD2	SMD1	SMD0	$\phi/\text{div} = 4 \text{ MHz}$	$\phi/\text{div} = 2 \text{ MHz}$	$\phi/\text{div} = 1 \text{ MHz}$	Formula
0	0	0	2 MHz	1 MHz	500 kHz	$(\phi/\text{div})/2^1$
0	0	1	1 MHz	500 kHz	250 kHz	$(\phi/\text{div})/2^2$
0	1	0	250 kHz	125 kHz	62.5 kHz	$(\phi/\text{div})/2^4$
0	1	1	125 kHz	62.5 kHz	31.25 kHz	$(\phi/\text{div})/2^5$
1	0	0	62.5 kHz	31.2 kHz	15.625 kHz	$(\phi/\text{div})/2^6$

See Table 18.3-1 "Machine Clock Division Ratio" for the div value.

■ External Shift Clock Mode

In external shift clock mode, the data transfer is based on the external clock supplied via the SCK3 pin. Data is transferred at one bit per clock.

The transfer speed can be between DC and $1/(8 \text{ machine cycles})$. For example, the transfer speed can be up to 2 MHz when 1 machine cycle is equal to 62.5 ns. The external clock frequency has a maximum value of 2 MHz.

A data bit can also be transferred by software, which is enabled as described below.

Select external shift clock mode, and write "0" to the SCOE bit of SMCS. Then, write "1" to the direction register for the port sharing the SCK3 pin, and place the port in output mode. Then, when "1" and "0" are written to the data register (PDR) of the port, the port value output via the SCK3 pin is fetched as the external clock and transfer starts. Ensure that the shift clock starts from "H".

Note:

The SMCS or SDR must not be written to during serial I/O operation.

18.4.2 Serial I/O Operation

There are four serial I/O operation statuses:

- **STOP**
 - **Halt**
 - **SDR R/W standby**
 - **Transfer**
-

■ Serial I/O Operation

○ **STOP**

The STOP state is initiated upon RESET or when "1" is written to the STOP bit of SMCS. The shift counter is initialized, and "0" is written to SIR.

To resume operation from the STOP state, write "0" to STOP and "1" to STRT. (These two bits can be written to simultaneously.) Since the STOP bit overrides the STRT bit, transfer cannot be started by writing "1" to STRT while "1" is written to STOP.

○ **Halt**

When transfer is completed while the MODE bit is "0", "0" is set to BUSY and "1" is set to SIR of the SMCS, the counter is initialized, and the system stops. To resume operation from the stop state, write "1" to STRT.

○ **Serial data register R/W standby**

When transfer is completed while the MODE bit is "1", "0" is set to BUSY and "1" is set to SIR of the SMCS, and the system enters the serial data register R/W standby state. If the interrupt enable flag is set, an interrupt signal is output from this block.

To resume operation from R/W standby state, read or write to the serial data register. This sets the BUSY bit to "1" and starts data transfer.

○ **Transfer**

"1" is set to the BUSY bit and serial transfer is being performed. According to the MODE bit, the halt state or R/W standby state comes next.

Figure 18.4-1 "Extended I/O Serial Interface Operation Transitions" is diagrams of the operation transitions.

Figure 18.4-1 Extended I/O Serial Interface Operation Transitions

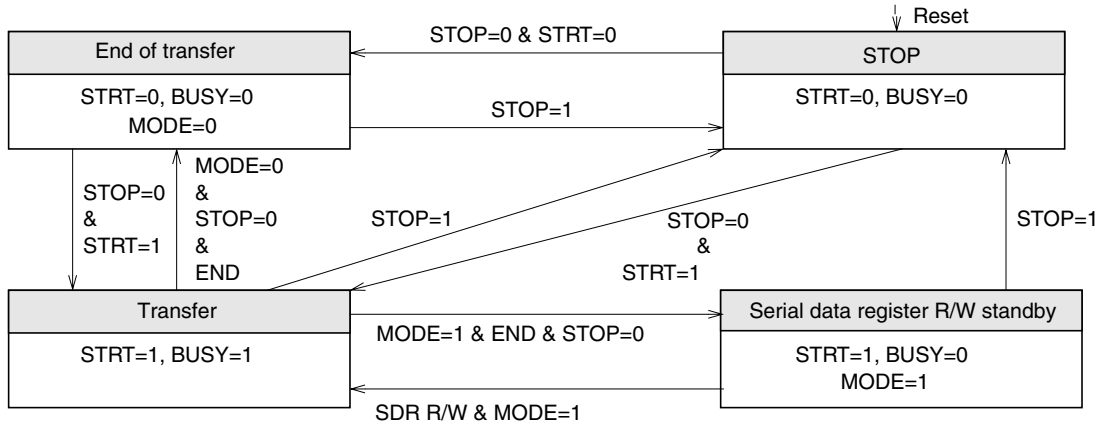
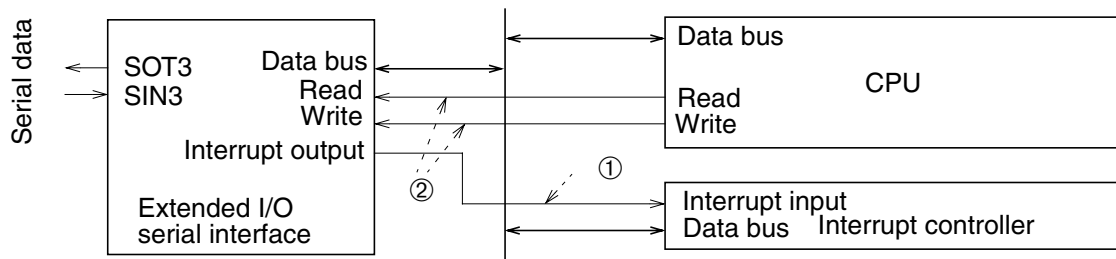


Figure 18.4-2 Serial Data Register Read/write



1. If "1" is written to MODE, transfer ends according to the shift clock counter. The read/write standby state starts when "1" is written to SIR. If "1" is written to the SIE bit, an interrupt signal is generated. No interrupt signal is generated when SIE is inactive or transfer has been terminated by writing "1" to STOP.
2. Reading or writing to the serial data register clears the interrupt request and starts serial transfer.

18.4.3 Shift Operation Start/Stop Timing

To start the shift operation, set the STOP bit to "0" and the STRT bit to "1" in SMCS. The system may stop the shift operation at the end of transfer or when "1" is set in the STOP bit.

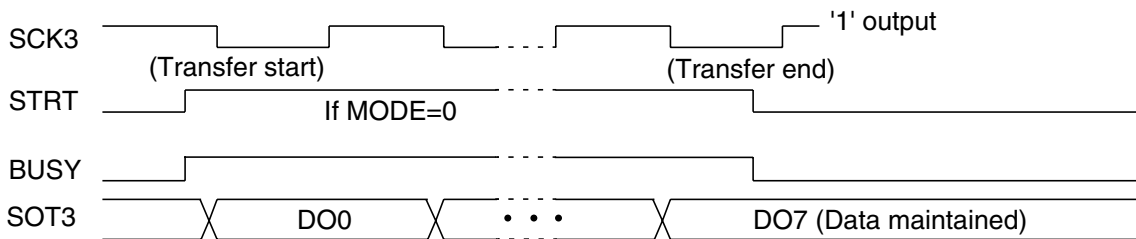
- Stop by STOP=1 -> The system stops with SIR=0 regardless of the MODE bit.
- Stop by end of transfer -> The system stops with SIR=1 regardless of the MODE bit.

Regardless of the MODE bit, the BUSY bit becomes "1" during serial transfer and becomes "0" during stop or R/W standby state. To check the transfer status, read this bit.

■ Shift Operation Start/Stop Timing

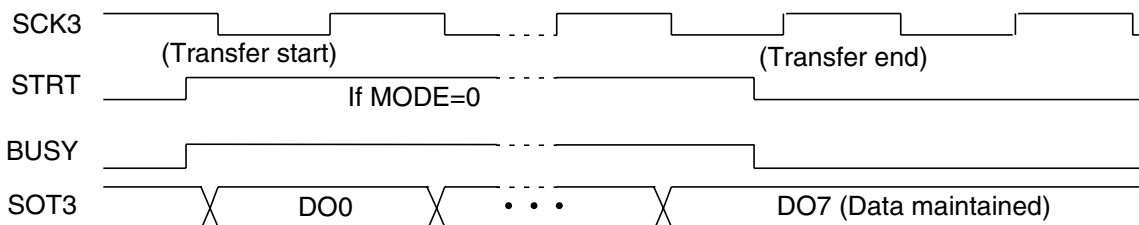
- Internal shift clock mode (LSB first)

Figure 18.4-3 Shift Operation Start/Stop Timing (Internal Clock)



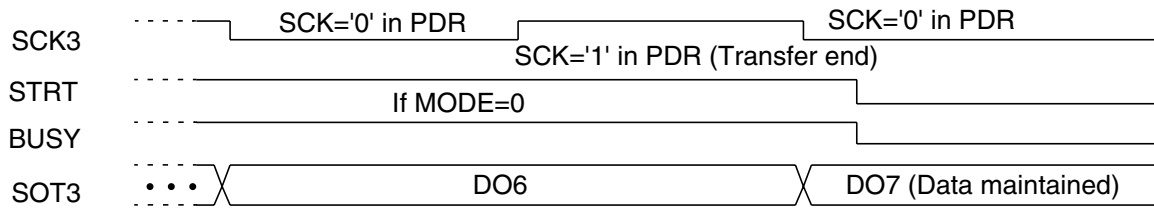
- External shift clock mode (LSB first)

Figure 18.4-4 Shift Operation Start/Stop Timing (External Clock)



○ External shift clock mode with instruction shift (LSB first)

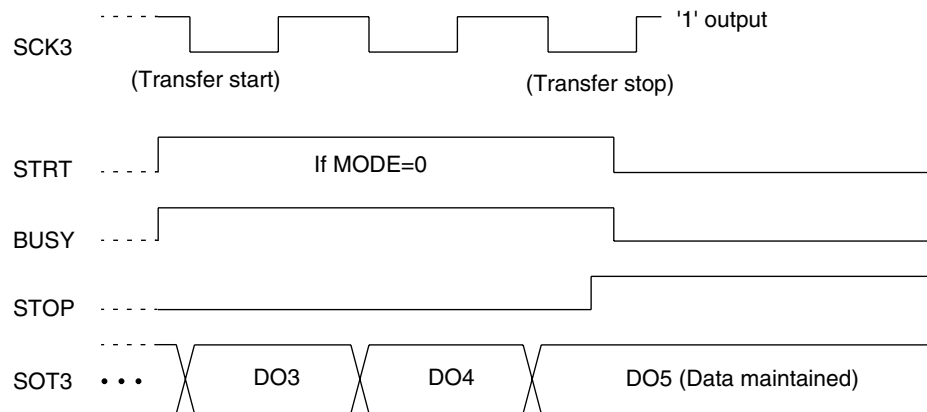
Figure 18.4-5 Shift Operation Start/Stop Timing (External Shift Clock Mode with Instruction Shift)



* For an instruction shift, 'H' is output when '1' is written to the bit corresponding to SCK of PDR, and 'L' is output when '0' is written. (When SCOE=0 in external shift clock mode)

○ Stop by STOP=1 (LSB first, internal clock)

Figure 18.4-6 Stop Timing when '1' is Written to the STOP Bit



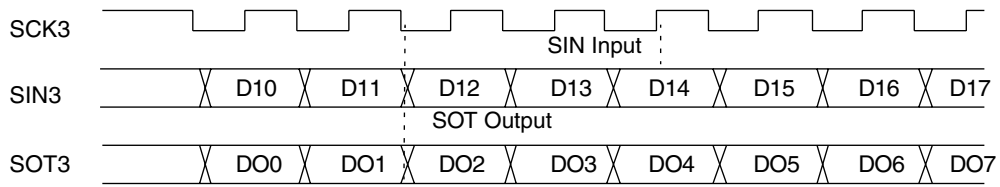
Note:

DO7 to DO0 indicate output data.

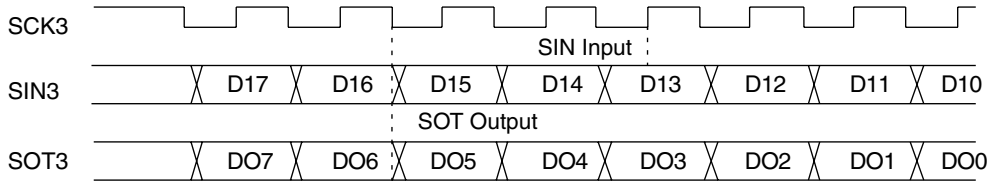
During serial data transfer, data is output from the serial output pin (SOT3) at the falling edge of the shift clock, and input from the serial input pin (SIN3) at the rising edge.

Figure 18.4-7 Serial Data I/O Shift Timing

○ LSB first (When the BDS bit is '0')



○ MSB first (When the BDS bit is '1')

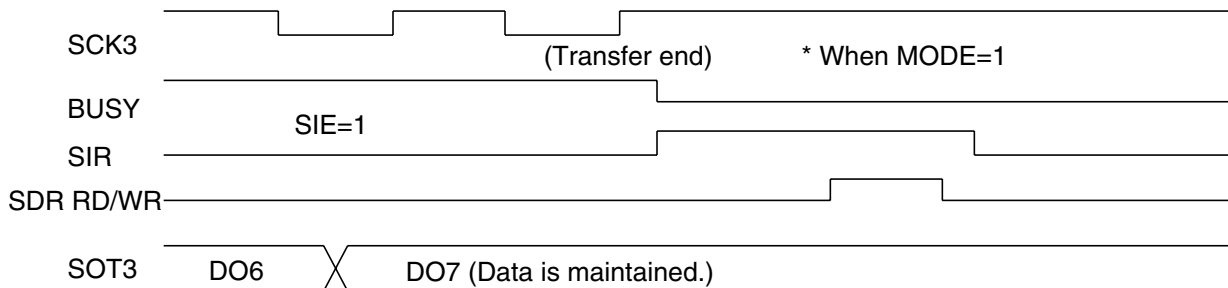


18.4.4 Interrupt Function of the Extended Serial I/O Interface

This block can issue an interrupt request to the CPU. At the end of data transfer, the SIR bit is set as an interrupt flag. When "1" is written to the interrupt enable bit (SIE bit) of SMCS, an interrupt request is issued to the CPU.

■ Interrupt Function of the Extended Serial I/O Interface

Figure 18.4-8 Interrupt Signal Output Timing of the Extended Serial I/O Interface



18.5 Negative Clock Operation

The MB90590 Series supports the negative clock operation of the Serial I/O. In this operation, the shift clock signal is simply negated by a inverter. Therefore the definition of the shift clock signal in the proceeding sections of the Serial I/O is inversed from the logic low level to logic high level, from the negative edge to the positive edge and vise-versa. This is the same for both the serial clock input and output.

■ Negative Clock Operation

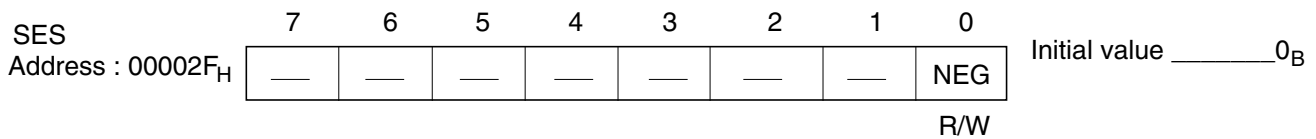


Table 18.5-1 Setting the NEG Bit

NEG	Operation
0	Normal operation [default]
1	The shift clock signal is inverted

CHAPTER 19 CAN CONTROLLER

This chapter explains the functions and operations of the CAN controller.

- 19.1 "Features of CAN Controller"
- 19.2 "Block Diagram of CAN Controller"
- 19.3 "List of Overall Control Registers"
- 19.4 "List of Message Buffers (ID Registers)"
- 19.5 "List of Message Buffers (DLC Registers and Data Registers)"
- 19.6 "Classifying the CAN Controller Registers"
- 19.7 "Transmission of CAN Controller"
- 19.8 "Reception of CAN Controller"
- 19.9 "Reception Flowchart of CAN Controller"
- 19.10 "How to Use the CAN Controller"
- 19.11 "Procedure for Transmission by Message Buffer (x)"
- 19.12 "Procedure for Reception by Message Buffer (x)"
- 19.13 "Setting Configuration of Multi-level Message Buffer"
- 19.14 "Precautions when Using CAN Controller"

19.1 Features of CAN Controller

The CAN controller is a module built into a 16-bit microcontroller (F²MC-16LX). The CAN (Controller Area Network) is the standard protocol for serial communication between automobile controllers and is widely used in industrial applications.

■ Features of CAN Controller

The CAN controller has the following features:

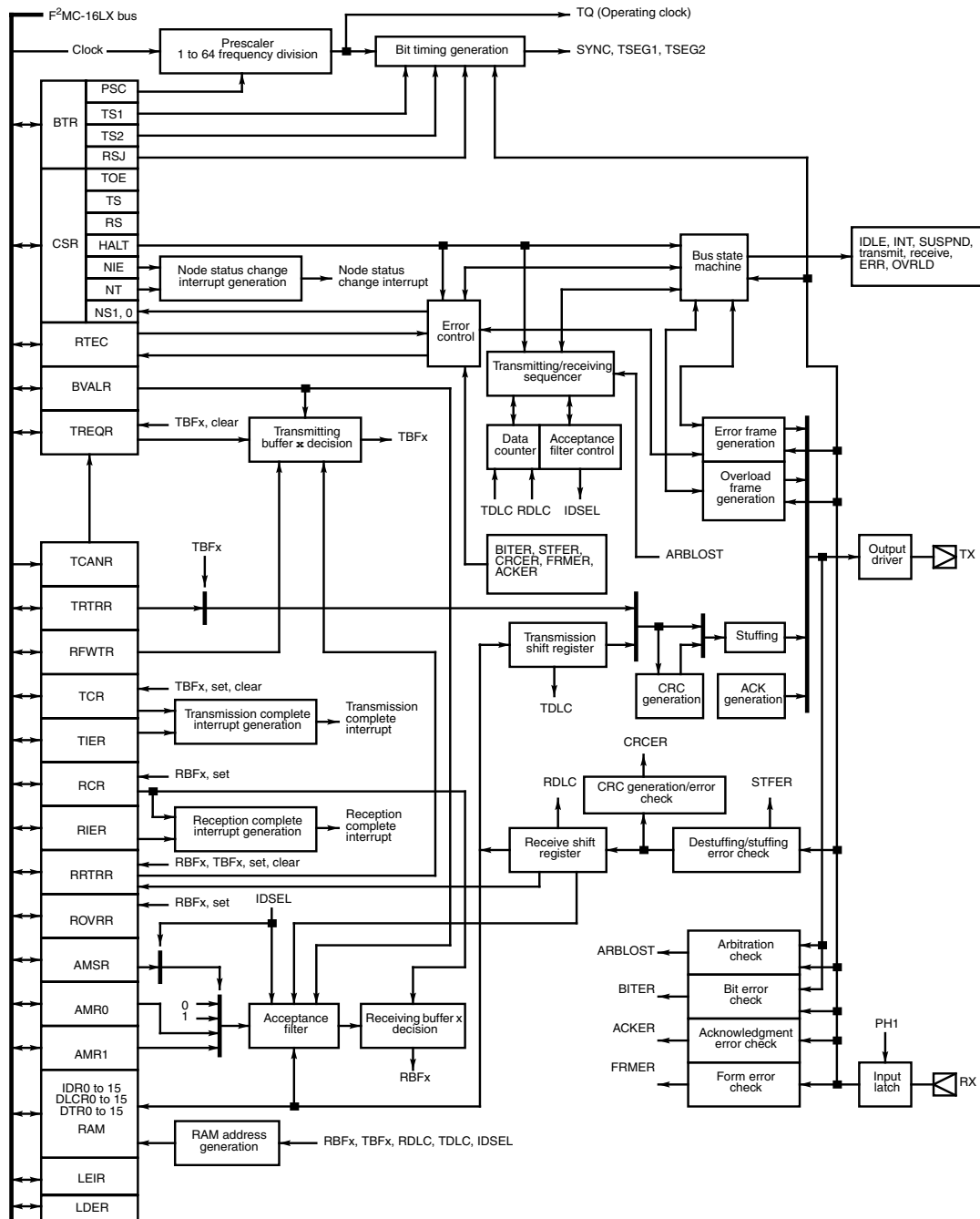
- **Conforms to CAN Specification Version 2.0 Part A and B**
Supports transmission/reception in standard frame and extended frame formats
- **Supports transmitting of data frames by receiving remote frames**
- **16 transmitting/receiving message buffers**
29-bit ID and 8-byte data
Multi-level message buffer configuration
- **Supports full-bit comparison, full-bit mask and partial bit mask filtering.**
Two acceptance mask registers in either standard frame format or extended frame formats
- **Bit rate programmable from 10 Kbps to 1 Mbps (A minimum 8 MHz machine clock is required if 1 Mbps is used)**

19.2 Block Diagram of CAN Controller

Figure 19.2-1 "Block Diagram of CAN Controller" shows a block diagram of the CAN controller.

■ Block Diagram of CAN Controller

Figure 19.2-1 Block Diagram of CAN Controller



19.3 List of Overall Control Registers

Table 19.3-1 "List of Overall Control Registers" lists overall control registers.

■ List of Overall Control Registers

Table 19.3-1 List of Overall Control Registers

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
000070 _H	000080 _H	Message buffer valid register	BVALR	R/W	00000000 00000000
000071 _H	000081 _H				
000072 _H	000082 _H	Transmit request register	TREQR	R/W	00000000 00000000
000073 _H	000083 _H				
000074 _H	000084 _H	Transmit cancel register	TCANR	W	00000000 00000000
000075 _H	000085 _H				
000076 _H	000086 _H	Transmit complete register	TCR	R/W	00000000 00000000
000077 _H	000087 _H				
000078 _H	000088 _H	Receive complete register	RCR	R/W	00000000 00000000
000079 _H	000089 _H				
00007A _H	00008A _H	Remote request receiving register	RRTRR	R/W	00000000 00000000
00007B _H	00008B _H				
00007C _H	00008C _H	Receive overrun register	ROVRR	R/W	00000000 00000000
00007D _H	00008D _H				
00007E _H	00008E _H	Receive interrupt enable register	RIER	R/W	00000000 00000000
00007F _H	00008F _H				
001C00 _H	001D00 _H	Control status register	CSR	R/W, R	00---000 0---001
001C01 _H	001D01 _H				
001C02 _H	001D02 _H	Last event indicator register	LEIR	R/W	----- 000-0000
001C03 _H	001D03 _H				
001C04 _H	001D04 _H	Receive/transmit error counter	RTEC	R	00000000 00000000
001C05 _H	001D05 _H				
001C06 _H	001D06 _H	Bit timing register	BTR	R/W	-1111111 11111111
001C07 _H	001D07 _H				

Table 19.3-1 List of Overall Control Registers (Continued)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001C08 _H	001D08 _H	IDE register	IDER	R/W	XXXXXXXX XXXXXXXX
001C09 _H	001D09 _H				
001C0A _H	001D0A _H	Transmit RTR register	TRTRR	R/W	00000000 00000000
001C0B _H	001D0B _H				
001C0C _H	001D0C _H	Remote frame receive waiting register	RFWTR	R/W	XXXXXXXX XXXXXXXX
001C0D _H	001D0D _H				
001C0E _H	001D0E _H	Transmit interrupt enable register	TIER	R/W	00000000 00000000
001C0F _H	001D0F _H				
001C10 _H	001D10 _H	Acceptance mask select register	AMSR	R/W	XXXXXXXX XXXXXXXX
001C11 _H	001D11 _H				XXXXXXXX XXXXXXXX
001C12 _H	001D12 _H				XXXXXXXX XXXXXXXX
001C13 _H	001D13 _H				XXXXXXXX XXXXXXXX
001C14 _H	001D14 _H	Acceptance mask register 0	AMR0	R/W	XXXXXXXX XXXXXXXX
001C15 _H	001D15 _H				XXXXXX--- XXXXXXXX
001C16 _H	001D16 _H				XXXXXX--- XXXXXXXX
001C17 _H	001D17 _H				XXXXXX--- XXXXXXXX
001C18 _H	001D18 _H	Acceptance mask register 1	AMR1	R/W	XXXXXXXX XXXXXXXX
001C19 _H	001D19 _H				XXXXXXXX XXXXXXXX
001C1A _H	001D1A _H				XXXXXX--- XXXXXXXX
001C1B _H	001D1B _H				XXXXXX--- XXXXXXXX

19.4 List of Message Buffers (ID Registers)

Table 19.4-1 "List of Message Buffers (ID Registers)" lists message buffers (ID registers).

■ List of Message Buffers (ID registers)

Table 19.4-1 List of Message Buffers (ID Registers)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001A00 _H to 001A1F _H	001B00 _H to 001B1F _H	General- purpose RAM	--	R/W	XXXXXXXX to XXXXXXXX
001A20 _H	001B20 _H	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX
001A21 _H	001B21 _H				
001A22 _H	001B22 _H				XXXXX---
001A23 _H	001B23 _H				XXXXXXXX
001A24 _H	001B24 _H	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX
001A25 _H	001B25 _H				
001A26 _H	001B26 _H				XXXXX---
001A27 _H	001B27 _H				XXXXXXXX
001A28 _H	001B28 _H	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX
001A29 _H	001B29 _H				
001A2A _H	001B2A _H				XXXXX---
001A2B _H	001B2B _H				XXXXXXXX
001A2C _H	001B2C _H	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX
001A2D _H	001B2D _H				
001A2E _H	001B2E _H				XXXXX---
001A2F _H	001B2F _H				XXXXXXXX
001A30 _H	001B30 _H	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX
001A31 _H	001B31 _H				
001A32 _H	001B32 _H				XXXXX---
001A33 _H	001B33 _H				XXXXXXXX

Table 19.4-1 List of Message Buffers (ID Registers) (Continued)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001A34 _H	001B34 _H	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX
001A35 _H	001B35 _H				
001A36 _H	001B36 _H				XXXXX--- XXXXXXXX
001A37 _H	001B37 _H				
001A38 _H	001B38 _H	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX
001A39 _H	001B39 _H				
001A3A _H	001B3A _H				XXXXX--- XXXXXXXX
001A3B _H	001B3B _H				
001A3C _H	001B3C _H	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX
001A3D _H	001B3D _H				
001A3E _H	001B3E _H				XXXXX--- XXXXXXXX
001A3F _H	001B3F _H				
001A40 _H	001B40 _H	ID register 8	IDR8	R/W	XXXXXXXX XXXXXXXX
001A41 _H	001B41 _H				
001A42 _H	001B42 _H				XXXXX--- XXXXXXXX
001A43 _H	001B43 _H				
001A44 _H	001B44 _H	ID register 9	IDR9	R/W	XXXXXXXX XXXXXXXX
001A45 _H	001B45 _H				
001A46 _H	001B46 _H				XXXXX--- XXXXXXXX
001A47 _H	001B47 _H				
001A48 _H	001B48 _H	ID register 10	IDR10	R/W	XXXXXXXX XXXXXXXX
001A49 _H	001B49 _H				
001A4A _H	001B4A _H				XXXXX--- XXXXXXXX
001A4B _H	001B4B _H				
001A4C _H	001B4C _H	ID register 11	IDR11	R/W	XXXXXXXX XXXXXXXX
001A4D _H	001B4D _H				
001A4E _H	001B4E _H				XXXXX--- XXXXXXXX
001A4F _H	001B4F _H				

Table 19.4-1 List of Message Buffers (ID Registers) (Continued)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001A50 _H	001B50 _H	ID register 12	IDR12	R/W	XXXXXXXX XXXXXXXX
001A51 _H	001B51 _H				
001A52 _H	001B52 _H				XXXXX--- XXXXXXXX
001A53 _H	001B53 _H				
001A54 _H	001B54 _H	ID register 13	IDR13	R/W	XXXXXXXX XXXXXXXX
001A55 _H	001B55 _H				
001A56 _H	001B56 _H				XXXXX--- XXXXXXXX
001A57 _H	001B57 _H				
001A58 _H	001B58 _H	ID register 14	IDR14	R/W	XXXXXXXX XXXXXXXX
001A59 _H	001B59 _H				
001A5A _H	001B5A _H				XXXXX--- XXXXXXXX
001A5B _H	001B5B _H				
001A5C _H	001B5C _H	ID register 15	IDR15	R/W	XXXXXXXX XXXXXXXX
001A5D _H	001B5D _H				
001A5E _H	001B5E _H				XXXXX--- XXXXXXXX
001A5F _H	001B5F _H				

19.5 List of Message Buffers (DLC Registers and Data Registers)

Table 19.5-1 "List of Message Buffers (DLC Registers and Data Registers)" lists message buffers (DLC registers), and Table 19.5-2 "List of Message Buffers (Data Registers)" lists message buffers (data registers).

■ List of Message Buffers (DLC Registers and Data Registers)

Table 19.5-1 List of Message Buffers (DLC Registers and Data Registers)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001A60 _H	001B60 _H	DLC register 0	DLCR0	R/W	----XXXX
001A61 _H	001B61 _H				
001A62 _H	001B62 _H	DLC register 1	DLCR1	R/W	----XXXX
001A63 _H	001B63 _H				
001A64 _H	001B64 _H	DLC register 2	DLCR2	R/W	----XXXX
001A65 _H	001B65 _H				
001A66 _H	001B66 _H	DLC register 3	DLCR3	R/W	----XXXX
001A67 _H	001B67 _H				
001A68 _H	001B68 _H	DLC register 4	DLCR4	R/W	----XXXX
001A69 _H	001B69 _H				
001A6A _H	001B6A _H	DLC register 5	DLCR5	R/W	----XXXX
001A6B _H	001B6B _H				
001A6C _H	001B6C _H	DLC register 6	DLCR6	R/W	----XXXX
001A6D _H	001B6D _H				
001A6E _H	001B6E _H	DLC register 7	DLCR7	R/W	----XXXX
001A6F _H	001B6F _H				
001A70 _H	001B70 _H	DLC register 8	DLCR8	R/W	----XXXX
001A71 _H	001B71 _H				
001A72 _H	001B72 _H	DLC register 9	DLCR9	R/W	----XXXX
001A73 _H	001B73 _H				
001A74 _H	001B74 _H	DLC register 10	DLCR10	R/W	----XXXX
001A75 _H	001B75 _H				

CHAPTER 19 CAN CONTROLLER

Table 19.5-1 List of Message Buffers (DLC Registers and Data Registers) (Continued)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001A76 _H	001B76 _H	DLC register 11	DLCR11	R/W	----XXXX
001A77 _H	001B77 _H				
001A78 _H	001B78 _H	DLC register 12	DLCR12	R/W	----XXXX
001A79 _H	001B79 _H				
001A7A _H	001B7A _H	DLC register 13	DLCR13	R/W	----XXXX
001A7B _H	001B7B _H				
001A7C _H	001B7C _H	DLC register 14	DLCR14	R/W	----XXXX
001A7D _H	001B7D _H				
001A7E _H	001B7E _H	DLC register 15	DLCR15	R/W	----XXXX
001A7F _H	001B7F _H				

■ List of Message Buffers (Data Registers)

Table 19.5-2 List of Message Buffers (Data Registers)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001A80 _H to 001A87 _H	001B80 _H to 001B87 _H	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX to XXXXXXXX
001A88 _H to 001A8F _H	001B88 _H to 001B8F _H	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX to XXXXXXXX
001A90 _H to 001A97 _H	001B90 _H to 001B97 _H	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX to XXXXXXXX
001A98 _H to 001A9F _H	001B98 _H to 001B9F _H	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX to XXXXXXXX
001AA0 _H to 001AA7 _H	001BA0 _H to 001BA7 _H	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX to XXXXXXXX
001AA8 _H to 001AAF _H	001BA8 _H to 001BAF _H	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX to XXXXXXXX
001AB0 _H to 001AB7 _H	001BB0 _H to 001BB7 _H	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX to XXXXXXXX
001AB8 _H to 001ABF _H	001BB8 _H to 001BBF _H	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX to XXXXXXXX
001AC0 _H to 001AC7 _H	001BC0 _H to 001BC7 _H	Data register 8 (8 bytes)	DTR8	R/W	XXXXXXXX to XXXXXXXX
001AC8 _H to 001ACF _H	001BC8 _H to 001BCF _H	Data register 9 (8 bytes)	DTR9	R/W	XXXXXXXX to XXXXXXXX
001AD0 _H to 001AD7 _H	001BD0 _H to 001BD7 _H	Data register 10 (8 bytes)	DTR10	R/W	XXXXXXXX to XXXXXXXX
001AD8 _H to 001ADF _H	001BD8 _H to 001BDF _H	Data register 11 (8 bytes)	DTR11	R/W	XXXXXXXX to XXXXXXXX
001AE0 _H to 001AE7 _H	001BE0 _H to 001BE7 _H	Data register 12 (8 bytes)	DTR12	R/W	XXXXXXXX to XXXXXXXX

CHAPTER 19 CAN CONTROLLER

Table 19.5-2 List of Message Buffers (Data Registers) (Continued)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
001AE8 _H to 001AEF _H	001BE8 _H to 001BEF _H	Data register 13 (8 bytes)	DTR13	R/W	XXXXXXXX to XXXXXXXX
001AF0 _H to 001AF7 _H	001BF0 _H to 001BF7 _H	Data register 14 (8 bytes)	DTR14	R/W	XXXXXXXX to XXXXXXXX
001AF8 _H to 001AFF _H	001BF8 _H to 001BFF _H	Data register 15 (8 bytes)	DTR15	R/W	XXXXXXXX to XXXXXXXX

19.6 Classifying the CAN Controller Registers

There are three types of CAN controller registers:

- Overall control registers
 - Message buffer control registers
 - Message buffers
-

■ Overall Control Registers

The overall control registers are the following four registers:

- Control status register (CSR)
- Last event indicator register (LEIR)
- Receive and transmit error counter (RTEC)
- Bit timing register (BTR)

■ Message Buffer Control Registers

The message buffer control registers are the following 14 registers:

- Message buffer valid register (BVALR)
- IDE register (IDER)
- Transmission request register (TREQR)
- Transmission RTR register (TRTRR)
- Remote frame receiving wait register (RFWTR)
- Transmission cancel register (TCANR)
- Transmission complete register (TCR)
- Transmission interrupt enable register (TIER)
- Reception complete register (RCR)
- Remote request receiving register (RRTRR)
- Receive overrun register (ROVRR)
- Reception interrupt enable register (RIER)
- Acceptance mask select register (AMSR)
- Acceptance mask registers 0 and 1 (AMR0 and AMR1)

■ Message Buffers

The message buffers are the following three registers:

- ID register x (x = 0 to 15) (IDRx)
- DLC register x (x = 0 to 15) (DLCRx)
- Data register x (x = 0 to 15) (DTRx)

19.6.1 Control Status Register (CSR)

Control status register (CSR) is prohibited from executing any bit manipulation instructions (Read-modify-write instructions).

■ Control Status Register (CSR)

	15	14	13	12	11	10	9	8
Address: 001C01 _H (CAN0) 001D01 _H (CAN1)	TS	RS	—	—	—	NT	NS1	NS0
Read/write:	(R)	(R)	(—)	(—)	(—)	(R/W)	(R)	(R)
Initial value:	(0)	(0)	(—)	(—)	(—)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 001C00 _H (CAN0) 001D00 _H (CAN1)	TOE	—	—	—	—	NIE	Reserved	HALT
Read/write:	(R/W)	(—)	(—)	(—)	(—)	(R/W)	(W)	(R/W)
Initial value:	(0)	(—)	(—)	(—)	(—)	(0)	(0)	(1)

[Bit 15] TS: Transmit status bit

This bit indicates whether a message is being transmitted.

0: Message not being transmitted

1: Message being transmitted

This bit is 0 even while error and overload frames are transmitted.

[Bit 14] RS: Receive status bit

This bit indicates whether a message is being received.

0: Message not being received

1: Message being received

While a message is on the bus, this bit becomes 1. Therefore, this bit is also 1 while a message is being transmitted. This bit does not necessarily indicate whether a receiving message passes through the acceptance filter.

As a result, when this bit is 0, it implies that the bus operation is stopped (HALT = 0); the bus is in the intermission/bus idle or an error/overload frame is on the bus.

[Bit 10] NT: Node status transition flag

If the node status is changed to increment, or from Bus Off to Error Active, this bit is set to 1.

In other words, the NT bit is set to 1 if the node status is changed from Error Active (00) to Warning (01), from Warning (01) to Error Passive (10), from Error Passive (10) to Bus Off (11), and from Bus Off (11) to Error Active (00). Numbers in parentheses indicate the values of NS1 and NS0 bits.

When the node status transition interrupt enable bit (NIE) is 1, an interrupt is generated. Writing 0 sets the NT bit to 0. Writing 1 to the NT bit is ignored. 1 is read when read-modify-write instruction is performed.

[Bits 9 to 8] NS1 and NS0: Node status bits 1 and 0

These bits indicate the current node status.

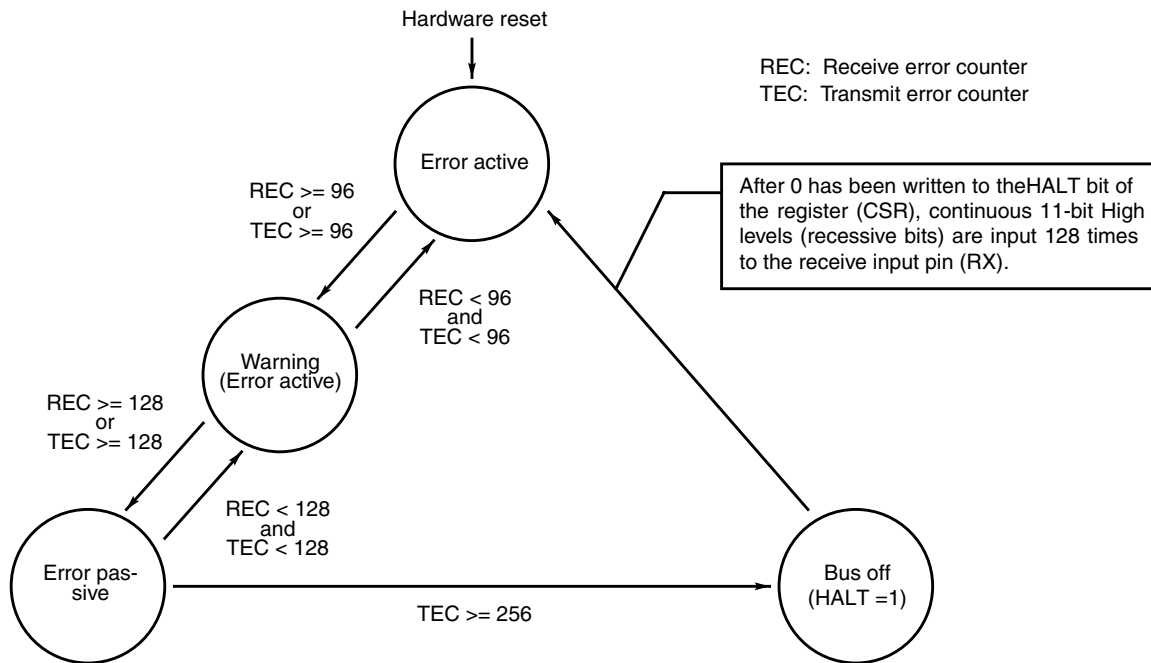
Table 19.6-1 Correspondence between NS1 and NS0 and Node Status

NS1	NS0	Node Status
0	0	Error active
0	1	Warning (error active)
1	0	Error passive
1	1	Bus off

Note:

Warning (error active) is included in the error active in CAN Specification 2.0B for the node status, however, indicates that the transmit error counter or receive error counter has exceeded 96. The node status change diagram is shown in Figure 19.6-1 "Node Status Transition Diagram".

Figure 19.6-1 Node Status Transition Diagram



[Bit 7] TOE: Transmit output enable bit

Writing 1 to this bit switches from a general-purpose port pin to a transmit pin of the CAN controller.

0: General-purpose port pin

1: Transmit pin of CAN controller

[Bit 2] NIE: Node status transition interrupt enable bit

This bit enables or disables a node status transition interrupt (when NT = 1).

0: Node status transition interrupt disabled

1: Node status transition interrupt enabled

[Bit 1] Reserved

This is a reserved bit. Do not write "1" to this bit.

[Bit 0] HALT: Bus operation stop bit

This bit sets or cancels bus operation stop, or displays its state.

19.6.2 Bus Operation Stop Bit (HALT = 1)

The bus operation stop bit sets or cancels stopping of bus operation, or indicates its status

■ Conditions for Setting Bus Operation Stop (HALT=1)

There are three conditions for setting bus operation stop (HALT = 1):

- After hardware reset
- When node status changed to bus off
- By writing 1 to HALT

Note:

The bus operation should be stopped by writing 1 to HALT before the F²MC-16LX is changed in low-power consumption mode (stop mode, clock mode, and hardware stand-by mode).

If transmission is in progress when 1 is written to HALT, the bus operation is stopped (HALT = 1) after transmission is terminated. If reception is in progress when 1 is written to HALT, the bus operation is stopped immediately (HALT = 1). If received messages are being stored in the message buffer (x), stop the bus operation (HALT = 1) after storing the messages.

To check whether the bus operation has stopped, always read the HALT bit.

■ Conditions for Canceling Bus Operation Stop (HALT = 0)

- By writing 0 to HALT

Note:

Canceling the bus operation stop after hardware reset or by writing 1 to HALT as above conditions is performed after 0 is written to HALT and continuous 11-bit High levels (recessive bits) have been input to the receive input pin (RX) (HALT = 0).

Canceling the bus operation stop when the node status is changed to bus off as above conditions is performed after 0 is written to HALT and continuous 11-bit High levels (recessive bits) have been input 128 times to the receive input pin (RX) (HALT = 0). Then, the values of both transmit and receive error counters reach 0 and the node status is changed to error active.

■ State during Bus Operation Stop (HALT = 1)

- The bus does not perform any operation, such as transmission and reception.
- The transmit output pin (TX) outputs a High level (recessive bit).
- The values of other registers and error counters are not changed.

Note:

The bit timing register (BTR) should be set during bus operation stop (HALT = 1).

19.6.3 Last Event Indicator Register (LEIR)

This register indicates the last event.

The NTE, TCE, and RCE bits are exclusive. When the corresponding bit of the last event is set to 1, other bits are set to 0s.

■ Last Event Indicator Register (LEIR)

	7	6	5	4	3	2	1	0
Address: 001C02 _H (CAN0) 001D02 _H (CAN1)	NTE	TCE	RCE	—	MBP3	MBP2	MBP1	MBP0
Read/write:	(R/W)	(R/W)	(R/W)	(—)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(—)	(0)	(0)	(0)	(0)

[Bit 7] NTE: Node status transition event bit

When this bit is 1, node status transition is the last event.

This bit is set to 1 at the same time the NT bit of the control status register (CSR) is set.

This bit is also set to 1 irrespective of the setting of the node status transition interrupt enable bit (NIE) of CSR.

Writing 0 to this bit sets the NTE bit to 0. Writing 1 to this bit is ignored.

1 is read when read-modify-write instruction is executed.

[Bit 6] TCE: Transmit completion event bit

When this bit is 1, it indicates that transmit completion is the last event.

This bit is set to 1 at the same time as any one of the bits of the transmit completion register (TCR). This bit is also set to 1, irrespective of the settings of the bits of the transmit interrupt enable register (TIER).

Writing 0 sets this bit to 0. Writing 1 to this bit is ignored.

1 is read when read-modify-write instruction is performed.

When this bit is set to 1, the MBP3 to MBP0 bits are used to indicate the message buffer number completing the transmit operation.

[Bit 5] RCE: Receive completion event bit

When this bit is 1, it indicates that receive completion is the last event.

This bit is set to 1 at the same time as any one of the bits of the receive complete register (RCR). This bit is also set to 1 irrespective of the settings of the bits of the receive interrupt enable register (RIER).

Writing 0 sets this bit to 0. Writing 1 to this bit is ignored.

1 is read when read-modify-write instruction is performed.

When this bit is set to 1, the MBP3 to MBP0 bits are used to indicate the message buffer number completing the receive operation.

[Bits 3 to 0] MBP3 to MBP0: Message buffer pointer bits

When the TCE or RCE bit is set to 1, these bits indicate the corresponding numbers of the message buffers (0 to 15). If the NTE bit is set to 1, these bits have no meaning.

Writing 0 sets these bits to 0s. Writing 1 to these bits is ignored.

1s are read when read-modify-write instruction is performed.

If LEIR is accessed within an CAN interrupt handler, the event causing the interrupt is not necessarily the same as indicated by LEIR. In the time from interrupt request to the LEIR access by the interrupt handler there may occur other CAN events.

19.6.4 Receive and Transmit Error Counters (RTEC)

The receive and transmit error counters indicate the counts for transmission errors and reception errors defined in the CAN specifications. These registers can only be read.

■ Receive and Transmit Error Counters (RTEC)

	15	14	13	12	11	10	9	8
Address: 001C05 _H (CAN0) 001D05 _H (CAN1)	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
Read/write:	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 001C04 _H (CAN0) 001D04 _H (CAN1)	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
Read/write:	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

[Bits 15 to 8] TEC7 to TEC0: Transmit error counter

These are transmit error counters.

TEC7 to TEC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Bus Off is indicated for the node status (NS1 and NS0 of control status register CSR = 11).

[Bits 7 to 0] REC7 to REC0: Receive error counter

These are receive error counters.

REC7 to REC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Error Passive is indicated for the node status (NS1 and NS0 of control status register CSR = 10).

19.6.5 Bit Timing Register (BTR)

Bit timing register (BTR) stores the prescaler and bit timing setting.

■ Bit Timing Register (BTR)

	15	14	13	12	11	10	9	8
Address: 001C07 _H (CAN0)	—	TS2.2	TS2.1	TS2.0	TS1.3	TS1.2	TS1.1	TS1.0
001D07 _H (CAN1)								
Read/write:	(—)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(—)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

	7	6	5	4	3	2	1	0
Address: 001C06 _H (CAN0)	RSJ1	RSJ0	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0
001D06 _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

Note:

This register should be set during bus operation stop (HALT = 1).

[Bits 14 to 12] TS2.2 to TS2.0: Time segment 2 setting bits 2 to 0

These bits define the number of the time quanta (TQ's) for the time segment 2 (TSEG2). The time segment 2 is equal to the phase buffer segment 2 (PHASE_SEG2) in the CAN specification.

[Bits 11 to 8] TS1.3 to TS1.0: Time segment 1 setting bits 3 to 0

These bits define the number of the time quanta (TQ's) for the time segment 1 (TSEG1). The time segment 1 is equal to the propagation segment (PROP_SEG) + phase buffer segment 1 (PHASE_SEG1) in the CAN specification.

[Bits 7 and 6] RSJ1 and RSJ0: Resynchronization jump width setting bits 1 and 0

These bits define the number of the time quanta (TQ's) for the resynchronization jump width.

[Bits 5 to 0] PSC5 to PSC0: Prescaler setting bits 5 to 0

These bits define the time quanta (TQ) of the CAN controller.

The bit time segments defined in the CAN specification, and the CAN controller are shown in Figure 19.6-2 "Bit Time Segment in CAN Specification" and Figure 19.6-3 "Bit Time Segment in CAN Controller" respectively.

Figure 19.6-2 Bit Time Segment in CAN Specification

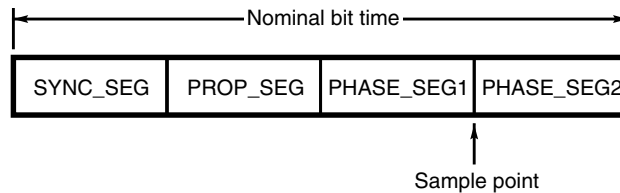
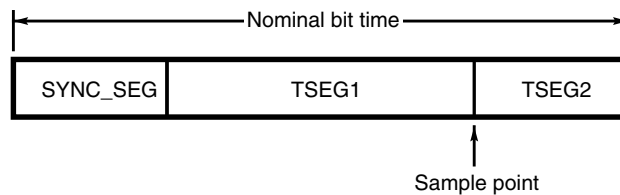


Figure 19.6-3 Bit Time Segment in CAN Controller



The relationship between PSC = PSC5 to PSC0, TSI = TS1.3 to TS1.0, TS2 = TS2.2 to TS1.0, and RSJ = RSJ1 and RSJ0 when the input clock (CLK), time quanta (TQ), bit time (BT), synchronous segment (SYNC_SEG), time segment 1 and 2 (TSEG1 and TSEG2), and resynchronization jump width [(RSJ1 and RSJ0) +1] frequency division is shown below.

The input clock is supplied with the machine clock.

$$\begin{aligned}
 TQ &= (PSC + 1) \times CLK \\
 BT &= SYNC_SEG + TSEG1 + TSEG2 \\
 &= (1 + (TS1 + 1) + (TS2 + 1)) \times TQ \\
 &= (3 + TS1 + TS2) \times TQ \\
 RSJW &= (RSJ + 1) \times TQ
 \end{aligned}$$

For correct operation, the following conditions should be met.

- Device with "G" suffix:
 - For $1 \leq \text{PSC} \leq 63$:
 - $\text{TSEG1} \geq 2\text{TQ}$
 - $\text{TSEG1} \geq \text{RSJW}$
 - $\text{TSEG2} \geq 2\text{TQ}$
 - $\text{TSEG2} \geq \text{RSJW}$
 - For $\text{PSC} = 0$:
 - $\text{TSEG1} \geq 5\text{TQ}$
 - $\text{TSEG2} \geq 2\text{TQ}$
 - $\text{TSEG2} \geq \text{RSJW}$

- Device without "G" suffix:
 - For $1 \leq \text{PSC} \leq 63$:
 - $\text{TSEG1} \geq \text{RSJW}$
 - $\text{TSEG2} \geq \text{RSJW} + 2\text{TQ}$
 - For $\text{PSC} = 0$:
 - $\text{TSEG1} \geq 5\text{TQ}$
 - $\text{TSEG2} \geq \text{RSJW} + 2\text{TQ}$

In order to meet the bit timing requirements defined in the CAN specification, additions have to be met, e.g. the propagation delay has to be considered.

19.6.6 Message Buffer Valid Register (BVALR)

Message buffer valid register (BVALR) stores the validity of the message buffers or displays their state.

■ Message Buffer Valid Register (BVALR)

	15	14	13	12	11	10	9	8
Address: 000071 _H (CAN0)	BVAL15	BVAL14	BVAL13	BVAL12	BVAL11	BVAL10	BVAL9	BVAL8
000081 _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: 000070 _H (CAN0)	BVAL7	BVAL6	BVAL5	BVAL4	BVAL3	BVAL2	BVAL1	BVAL0
000080 _H (CAN1)								
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)

0: Message buffer (x) invalid

1: Message buffer (x) valid

If the message buffer (x) is set to invalid, it will not transmit or receive messages.

If the buffer is set to invalid during transmission operating, it becomes invalid (BVALx = 0) after the transmission is completed or terminated by an error.

If the buffer is set to invalid during reception operating, it immediately becomes invalid (BVALx = 0). If received messages are stored in a message buffer (x), the message buffer (x) is invalid after storing the messages.

Note:

x indicates a message buffer number (x = 0 to 15).

When invaliding a message buffer (x) by writing 0 to a bit (BVALx), execution of a bit manipulation instruction is prohibited until the bit is set to 0.

To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while CAN Controller is participating in CAN communication (the read value of the CSR: HALT bit is 0 and CAN Controller is ready to receive or transmit messages), follow the cautions in Section 19.14 "Precautions when Using CAN Controller".

19.6.7 IDE register (IDER)

This register stores the frame format used by the message buffers (x) during transmission/reception.

■ IDE Register (IDER)

	15	14	13	12	11	10	9	8
Address: 001C09 _H (CAN0)	IDE15	IDE14	IDE13	IDE12	IDE11	IDE10	IDE9	IDE8
001D09 _H (CAN1)								
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)

	7	6	5	4	3	2	1	0
Address: 001C08 _H (CAN0)	IDE7	IDE6	IDE5	IDE4	IDE3	IDE2	IDE1	IDE0
001D08 _H (CAN1)								
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)

- 0: The standard frame format (ID11 bit) is used for the message buffer (x).
 1: The extended frame format (ID29 bit) is used for the message buffer (x).

Note:

This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) = 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while CAN Controller is participating in CAN communication (the read value of the CSR: HALT bit is 0 and CAN Controller is ready to receive or transmit messages), follow the cautions in Section 19.14 "Precautions when Using CAN Controller".

19.6.8 Transmission Request Register (TREQR)

Transmission request register (TREQR) stores transmission requests to the message buffers (x) or displays their state.

■ Transmission Request Register (TREQR)

	15	14	13	12	11	10	9	8
Address: 000073 _H (CAN0)	TREQ15	TREQ14	TREQ13	TREQ12	TREQ11	TREQ10	TREQ9	TREQ8
000083 _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 000072 _H (CAN0)	TREQ7	TREQ6	TREQ5	TREQ4	TREQ3	TREQ2	TREQ1	TREQ0
000082 _H (CAN1)								
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 1 is written to TREQ_x, transmission to the message buffer (x) starts. If RFWT_x of the remote frame receiving wait register (RFWTR)*1 is 0, transmission starts immediately. However, if RFWT_x = 1, transmission starts after waiting until a remote frame is received (RRTR_x of the remote request receiving register (RRTRR)*1 becomes 1). Transmission starts*2 immediately even when RFWT_x = 1, if RRTR_x is already 1 when 1 is written to TREQ_x.

*1: For RFWTR and TRTRR, see 19.6.9 "Transmission RTR Register (TRTRR)" and 19.6.10 "Remote Frame Receiving Wait Register (RFWTR)".

*2: For cancellation of transmission, see 19.6.11 "Transmission Cancel Register (TCANR)" and 19.6.12 "Transmission Complete Register (TCR)".

Writing 0 to TREQ_x is ignored.

0 is read when read-modify-write instruction is performed.

If clearing (to 0) at completion of the transmit operation and setting by writing 1 are concurrent, clearing is preferred.

If 1 is written to more than one bit, transmission is performed, starting with the lower-numbered message buffer (x).

TREQ_x is 1 while transmission is pending, and becomes 0 when transmission is completed or canceled.

19.6.9 Transmission RTR Register (TRTRR)

This register stores the RTR (Remote Transmission Request) bits for the message buffers (x).

■ Transmission RTR Register (TRTRR)

	15	14	13	12	11	10	9	8
Address: 001C0B _H (CAN0)	TRTR15	TRTR14	TRTR13	TRTR12	TRTR11	TRTR10	TRTR9	TRTR8
001D0B _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 001C0A _H (CAN0)	TRTR7	TRTR6	TRTR5	TRTR4	TRTR3	TRTR2	TRTR1	TRTR0
001D0A _H (CAN1)								
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)

0: Data frame

1: Remote frame

19.6.10 Remote Frame Receiving Wait Register (RFWTR)

Remote frame receiving wait register (RFWTR) stores the conditions for starting transmission when a request for data frame transmission is set (TREQx of the transmission request register (TREQR) is 1 and TRTRx of the transmitting RTR register (TRTRR) is 0).

- **0:** Transmission starts immediately
- **1:** Transmission starts after waiting until remote frame received (RRTRx of remote request receiving register (RRTRR) becomes 1)

■ Remote Frame Receiving Wait Register (RFWTR)

	15	14	13	12	11	10	9	8
Address: 001C0D _H (CAN0)	RFWT15	RFWT14	RFWT13	RFWT12	RFWT11	RFWT10	RFWT9	RFWT8
001D0D _H (CAN1)								
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)
	7	6	5	4	3	2	1	0
Address: 001C0C _H (CAN0)	RFWT7	RFWT6	RFWT5	RFWT4	RFWT3	RFWT2	RFWT1	RFWT0
001D0C _H (CAN1)								
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:

Transmission starts immediately if RRTRx is already 1 when a request for transmission is set.

For remote frame transmission, do not set RFWTx to 1.

19.6.11 Transmission Cancel Register (TCANR)

When 1 is written to TCANx, this register cancels a pending request for transmission to the message buffer (x).

At completion of cancellation, TREQx of the transmission request register (TREQR) becomes 0. Writing 0 to TCANx is ignored.

This is a write-only register and its read value is always 0.

■ Transmission Cancel Register (TCANR)

	15	14	13	12	11	10	9	8
Address: 000075 _H (CAN0)	TCAN15	TCAN14	TCAN13	TCAN12	TCAN11	TCAN10	TCAN9	TCAN8
000085 _H (CAN1)								
Read/write:	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 000074 _H (CAN0)	TCAN7	TCAN6	TCAN5	TCAN4	TCAN3	TCAN2	TCAN1	TCAN0
000084 _H (CAN1)								
Read/write:	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

19.6.12 Transmission Complete Register (TCR)

At completion of transmission by the message buffer (x), the corresponding TCx becomes 1.

If TIEx of the transmission complete interrupt enable register (TIER) is 1, an interrupt occurs.

■ Transmission Complete Register (TCR)

	15	14	13	12	11	10	9	8
Address: 000077 _H (CAN0)	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
000087 _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 000076 _H (CAN0)	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
000086 _H (CAN1)								
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)

○ Conditions for TCx = 0

- Write 0 to TCx.
- Write 1 to TREQx of the transmission request register (TREQR).

After the completion of transmission, write 0 to TCx to set it to 0. Writing 1 to TCx is ignored.

1 is read when read-modify-write instruction is performed.

Note:

If setting to 1 by completion of the transmit operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.13 Transmission Interrupt Enable Register (TIER)

This register enables or disables the transmission interrupt by the message buffer (x). The transmission interrupt is generated at transmission completion (when TCx of the transmission complete register (TCR) is 1).

■ Transmission Interrupt Enable Register (TIER)

	15	14	13	12	11	10	9	8
Address: 001C0F _H (CAN0)	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8
001D0F _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 001C0E _H (CAN0)	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
001D0E _H (CAN1)								
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)

0: Transmission interrupt disabled

1: Transmission interrupt enabled

19.6.14 Reception Complete Register (RCR)

At completion of storing received message in the message buffer (x), RCx becomes 1. If RIEx of the reception complete interrupt enable register (RIER) is 1, an interrupt occurs.

■ Reception Complete Register (RCR)

	15	14	13	12	11	10	9	8
Address: 000079 _H (CAN0)	RC15	RC14	RC13	RC12	RC11	RC10	RC9	RC8
000089 _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 000078 _H (CAN0)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
000088 _H (CAN1)								
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)

○ Conditions for RCx = 0

Write 0 to RCx.

After completion of handling received message, write 0 to RCx to set it to 0. Writing 1 to RCx is ignored.

1 is read when read-modify-write instruction is performed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.15 Remote Request Receiving Register (RRTRR)

After a remote frame is stored in the message buffer (x), RRTRx becomes 1 (at the same time as RCx setting to 1).

■ Remote Request Receiving Register (RRTRR)

	15	14	13	12	11	10	9	8
Address: 00007B _H (CAN0)	RRTR15	RRTR14	RRTR13	RRTR12	RRTR11	RRTR10	RRTR9	RRTR8
00008B _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: 00007A _H (CAN0)	RRTR7	RRTR6	RRTR5	RRTR4	RRTR3	RRTR2	RRTR1	RRTR0
00008A _H (CAN1)								
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)

○ Conditions for RRTRx = 0

- Write 0 to RRTRx.
- After a received data frame is stored in the message buffer (x) (at the same time as RCx setting to 1).
- Transmission by the message buffer (x) is completed (TCx of the transmission complete register (TCR) is 1).

Writing 1 to RRTRx is ignored.

1 is read when read-modify-write instruction is performed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.16 Receive Overrun Register (ROVRR)

If RCx of the reception complete register (RCR) is 1 when completing storing of a received message in the message buffer (x), ROVRx becomes 1, indicating that reception has overrun.

■ Receive Overrun Register (ROVRR)

	15	14	13	12	11	10	9	8
Address: 00007D _H (CAN0)	ROVR15	ROVR14	ROVR13	ROVR12	ROVR11	ROVR10	ROVR9	ROVR8
00008D _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 00007C _H (CAN0)	ROVR7	ROVR6	ROVR5	ROVR4	ROVR3	ROVR2	ROVR1	ROVR0
00008C _H (CAN1)								
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)

Writing 0 to ROVRx results in ROVRx = 0. Writing 1 to ROVRx is ignored. After checking that reception has overrun, write 0 to ROVRx to set it to 0.

1 is read when read-modify-write instruction is performed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.17 Reception Interrupt Enable Register (RIER)

Reception interrupt enable register (RIER) enables or disables the reception interrupt by the message buffer (x).

The reception interrupt is generated at reception completion (when RCx of the reception completion register (RCR) is 1).

■ Reception Interrupt Enable Register (RIER)

	15	14	13	12	11	10	9	8
Address: 00007F _H (CAN0)	RIE15	RIE14	RIE13	RIE12	RIE11	RIE10	RIE9	RIE8
00008F _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
	7	6	5	4	3	2	1	0
Address: 00007E _H (CAN0)	RIE7	RIE6	RIE5	RIE4	RIE3	RIE2	RIE1	RIE0
00008E _H (CAN1)								
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)

0: Reception interrupt disabled

1: Reception interrupt enabled

19.6.18 Acceptance Mask Select Register (AMSR)

This register selects masks (acceptance mask) for comparison between the received message ID's and the message buffer ID's.

■ Acceptance Mask Select Register (AMSR)

BYTE0	7	6	5	4	3	2	1	0
Address: 001C10 _H (CAN0)	AMS3.1	AMS3.0	AMS2.1	AMS2.0	AMS1.1	AMS1.0	AMS0.1	AMS0.0
001D10 _H (CAN1)								
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)
BYTE1	15	14	13	12	11	10	9	8
Address: 001C11 _H (CAN0)	AMS7.1	AMS7.0	AMS6.1	AMS6.0	AMS5.1	AMS5.0	AMS4.1	AMS4.0
001D11 _H (CAN1)								
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)
BYTE2	7	6	5	4	3	2	1	0
Address: 001C12 _H (CAN0)	AMS11.1	AMS11.0	AMS10.1	AMS10.0	AMS9.1	AMS9.0	AMS8.1	AMS8.0
001D12 _H (CAN1)								
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)
BYTE3	15	14	13	12	11	10	9	8
Address: 001C13 _H (CAN0)	AMS15.1	AMS15.0	AMS14.1	AMS14.0	AMS13.1	AMS13.0	AMS12.1	AMS12.0
001D13 _H (CAN1)								
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)

Table 19.6-2 Selection of Acceptance Mask

AMSx.1	AMSx.0	Acceptance Mask
0	0	Full-bit comparison
0	1	Full-bit mask
1	0	Acceptance mask register 0 (AMR0)
1	1	Acceptance mask register 1 (AMR1)

Note:

AMSx.1 and AMSx.0 should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored

To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while CAN Controller is participating in CAN communication (the read value of the CSR: HALT bit is 0 and CAN Controller is ready to receive or transmit messages), follow the cautions in Section 19.14 "Precautions when Using CAN Controller".

19.6.19 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)

There are two acceptance mask registers, AMR0 and AMR1, both of which are available either in the standard frame format or extended frame format. AM28 to AM18 (11 bits) are used for acceptance masks in the standard frame format and AM28 to AM0 (29 bits) are used for acceptance masks in the extended format.

■ Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)

AMR0 BYTE0	7	6	5	4	3	2	1	0
Address: 001C14 _H (CAN0) 001D14 _H (CAN1)	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21
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)
AMR0 BYTE1	15	14	13	12	11	10	9	8
Address: 001C15 _H (CAN0) 001D15 _H (CAN1)	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13
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)
AMR0 BYTE2	7	6	5	4	3	2	1	0
Address: 001C16 _H (CAN0) 001D16 _H (CAN1)	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5
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)
AMR0 BYTE3	15	14	13	12	11	10	9	8
Address: 001C17 _H (CAN0) 001D17 _H (CAN1)	AM4	AM3	AM2	AM1	AM0	—	—	—
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

AMR1 BYTE0	7	6	5	4	3	2	1	0
Address: 001C18 _H (CAN0)	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21
001D18 _H (CAN1)								
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)
AMR1 BYTE1	15	14	13	12	11	10	9	8
Address: 001C19 _H (CAN0)	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13
001D19 _H (CAN1)								
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)
AMR1 BYTE2	7	6	5	4	3	2	1	0
Address: 001C1A _H (CAN0)	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5
001D1A _H (CAN1)								
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)
AMR1 BYTE3	15	14	13	12	11	10	9	8
Address: 001C1B _H (CAN0)	AM4	AM3	AM2	AM1	AM0	—	—	—
001D1B _H (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

○ **0: Compare**

Compare the bit of the acceptance code (ID register IDR_x for comparing with the received message ID) corresponding to this bit with the bit of the received message ID. If there is no match, no message is received.

○ **1: Mask**

Mask the bit of the acceptance code ID register (IDR_x) corresponding to this bit. No comparison is made with the bit of the received message ID.

Note:

AMR0 and AMR1 should be set when all the message buffers (x) selecting AMR0 and AMR1 are invalid (BVAL_x of the message buffer valid register (BVALR) is 0). Setting when the buffers are valid (BVAL_x = 1) may cause unnecessary received messages to be stored.

To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while CAN Controller is participating in CAN communication (the read value of the CSR: HALT bit is 0 and CAN Controller is ready to receive or transmit messages), follow the cautions in Section 19.14 "Precautions when Using CAN Controller".

19.6.20 Message Buffers

There are 16 message buffers. Message buffer x ($x = 0$ to 15) consists of an ID register (IDRx), DLC register (DLCRx), and data register (DTRx).

■ Message Buffers

- The message buffer (x) is used both for transmission and reception.
- The lower-numbered message buffers are assigned higher priority.
 - At transmission, when a request for transmission is made to more than one message buffer, transmission is performed, starting with the lowest-numbered message buffer (See 19.7 "Transmission of CAN Controller").
 - At reception, when the received message ID passes through the acceptance filter (mechanism for comparing the acceptance-masked ID of received message and message buffer) of more than one message buffer, the received message is stored in the lowest-numbered message buffer (See 19.8 "Reception of CAN Controller").
- When the same acceptance filter is set in more than one message buffer, the message buffers can be used as a multi-level message buffer. This provides allowance for receiving time.
(See 19.12 "Procedure for Reception by Message Buffer (x)").

Note:

A write operation to message buffers and general-purpose RAM areas should be performed in words to even addresses only. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

When the BVALx bit of the message buffer valid register (BVALR) is 0 (Invalid), the message buffers x (IDRx, DLCRx, and DTRx) can be used as general-purpose RAM.

During the receive/transmit operation of the CAN controller, the CAN Controller write/read to/from the message buffers. If the CPU tries to write/read to/from the message buffers in this period, the CPU has to wait a maximum time of 64 machine cycles.

This is also true for the general-purpose RAM area (address 001A00_H to 001A1F_H and address 001B00_H to 001B1F_H).

19.6.21 ID Register x (x = 0 to 15) (IDRx)

ID Register x (x = 0 to 15) (IDRx) is the ID register for message buffer (x).

■ ID Register x (x = 0 to 15) (IDRx)

BYTE0		7	6	5	4	3	2	1	0
Address: 001A20 _H + 4 x (CAN0)		ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
001B20 _H + 4 x (CAN1)									
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)
BYTE1		15	14	13	12	11	10	9	8
Address: 001A21 _H + 4 x (CAN0)		ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
001B21 _H + 4 x (CAN1)									
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)
BYTE2		7	6	5	4	3	2	1	0
Address: 001A22 _H + 4 x (CAN0)		ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
001B22 _H + 4 x (CAN1)									
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)
BYTE3		15	14	13	12	11	10	9	8
Address: 001A23 _H + 4 x (CAN0)		ID4	ID3	ID2	ID1	ID0	—	—	—
001B23 _H + 4 x (CAN1)									
Read/write:		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:		(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

When using the message buffer (x) in the standard frame format (IDEx of the IDE register (IDER) = 0), use 11 bits of ID28 to ID18. When using the buffer in the extended frame format (IDEx = 1), use 29 bits of ID28 to ID0.

ID28 to ID0 have the following functions:

- Set acceptance code (ID for comparing with the received message ID).
- Set transmitted message ID.

Note:

In the standard frame format, setting 1s to all bits of ID28 to ID22 is prohibited).

- Store the received message ID.

Note:

All received message ID bits are stored (even if bits are masked). In the standard frame format, ID17 to ID0 stores image of old message left in the receive shift register.

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while CAN Controller is participating in CAN communication (the read value of the CSR: HALT bit is 0 and CAN Controller is ready to receive or transmit messages), follow the cautions in Section 19.14 "Precautions when Using CAN Controller".

19.6.22 DLC Register x (x = 0 to 15) (DLCRx)

DLC Register x (x = 0 to 15) (DLCRx) is the DLC register for message buffer x.

■ DLC Register x (x = 0 to 15) (DLCRx)

	7	6	5	4	3	2	1	0
Address: $001A60_H + 2 \times (\text{CAN0})$	—	—	—	—	DLC3	DLC2	DLC1	DLC0
$001B60_H + 2 \times (\text{CAN1})$	—	—	—	—	DLC3	DLC2	DLC1	DLC0
Read/write:	(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(—)	(—)	(—)	(—)	(X)	(X)	(X)	(X)

○ Transmission

- Set the data length (byte count) of a transmitted message when a data frame is transmitted (TRTRx of the transmitting RTR register (TRTRR) is 0).
- Set the data length (byte count) of a requested message when a remote frame is transmitted (TRTRx = 1).

Note:

Setting other than 0000 to 1000 (0 to 8 bytes) is prohibited.

○ Reception

- Store the data length (byte count) of a received message when a data frame is received (RRTRx of the remote frame request receiving register (RRTRR) is 0).
- Store the data length (byte count) of a requested message when a remote frame is received (RRTRx = 1).

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

19.6.23 Data Register x (x = 0 to 15) (DTRx)

Data register x (x = 0 to 15) (DTRx) is the data register for message buffer (x). This register is used only in transmitting and receiving a data frame but not in transmitting and receiving a remote frame.

■ Data Register x (x = 0 to 15) (DTRx)

BYTE0	7	6	5	4	3	2	1	0
Address: 001A80 _H + 8 x (CAN0) 001B80 _H + 8 x (CAN1)	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)
BYTE1	15	14	13	12	11	10	9	8
Address: 001A81 _H + 8 x (CAN0) 001B81 _H + 8 x (CAN1)	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)
BYTE2	7	6	5	4	3	2	1	0
Address: 001A82 _H + 8 x (CAN0) 001B82 _H + 8 x (CAN1)	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)
BYTE3	15	14	13	12	11	10	9	8
Address: 001A83 _H + 8 x (CAN0) 001B83 _H + 8 x (CAN1)	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)

BYTE4		7	6	5	4	3	2	1	0
Address:	001A84 _H + 8 x (CAN0)	D7	D6	D5	D4	D3	D2	D1	D0
	001B84 _H + 8 x (CAN1)								
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)

BYTE5		15	14	13	12	11	10	9	8
Address:	001A85 _H + 8 x (CAN0)	D7	D6	D5	D4	D3	D2	D1	D0
	001B85 _H + 8 x (CAN1)								
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)

BYTE6		7	6	5	4	3	2	1	0
Address:	001A86 _H + 8 x (CAN0)	D7	D6	D5	D4	D3	D2	D1	D0
	001B86 _H + 8 x (CAN1)								
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)

BYTE7		15	14	13	12	11	10	9	8
Address:	001A87 _H + 8 x (CAN0)	D7	D6	D5	D4	D3	D2	D1	D0
	001B87 _H + 8 x (CAN1)								
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)

○ **Sets transmitted message data (any of 0 to 8 bytes).**

Data is transmitted in the order of BYTE0, BYTE1, ..., BYTE7, starting with the MSB.

○ **Stores received message data.**

Data is stored in the order of BYTE0, BYTE1, ..., BYTE7, starting with the MSB.

Even if the received message data is less than 8 bytes, the remaining bytes of the data register (DTRx), to which data are stored, are undefined.

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

19.7 Transmission of CAN Controller

When 1 is written to TREQx of the transmission request register (TREQR), transmission by the message buffer (x) starts. At this time, TREQx becomes 1 and TCx of the transmission complete register (TCR) becomes 0.

■ Starting Transmission of the CAN Controller

If RFWTx of the remote frame receiving wait register (RFWTR) is 0, transmission starts immediately. If RFWTx is 1, transmission starts after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR) becomes 1).

If a request for transmission is made to more than one message buffer (more than one TREQx is 1), transmission is performed, starting with the lowest-numbered message buffer.

Message transmission to the CAN bus (by the transmit output pin TX) starts when the bus is idle.

If TRTRx of the transmission RTR register (TRTRR) is 0, a data frame is transmitted. If TRTRx is 1, a remote frame is transmitted.

If the message buffer competes with other CAN controllers on the CAN bus for transmission and arbitration fails, or if an error occurs during transmission, the message buffer waits until the bus is idle and repeats retransmission until it is successful.

■ Canceling a Transmission Request from the CAN Controller

○ Canceling by transmission cancel register (TCANR)

A transmission request for message buffer (x) having not executed transmission during transmission pending can be canceled by writing 1 to TCANx of the transmission cancel register (TCANR). At completion of cancellation, TREQx becomes 0.

○ Canceling by storing received message

The message buffer (x) having not executed transmission despite transmission request also performs reception.

If the message buffer (x) has not executed transmission despite a request for transmission of a data frame (TRTRx = 0 or TREQx = 1), the transmission request is canceled after storing received data frames passing through the acceptance filter (TREQx = 0).

Note:

A transmission request is not canceled by storing remote frames (TREQx = 1 remains unchanged).

If the message buffer (x) has not executed transmission despite a request for transmission of a remote frame (TRTRx = 1 or TREQx = 1), the transmission request is canceled after storing received remote frames passing through the acceptance filter (TREQx = 0).

Note:

The transmission request is canceled by storing either data frames or remote frames.

■ **Completing Transmission of the CAN Controller**

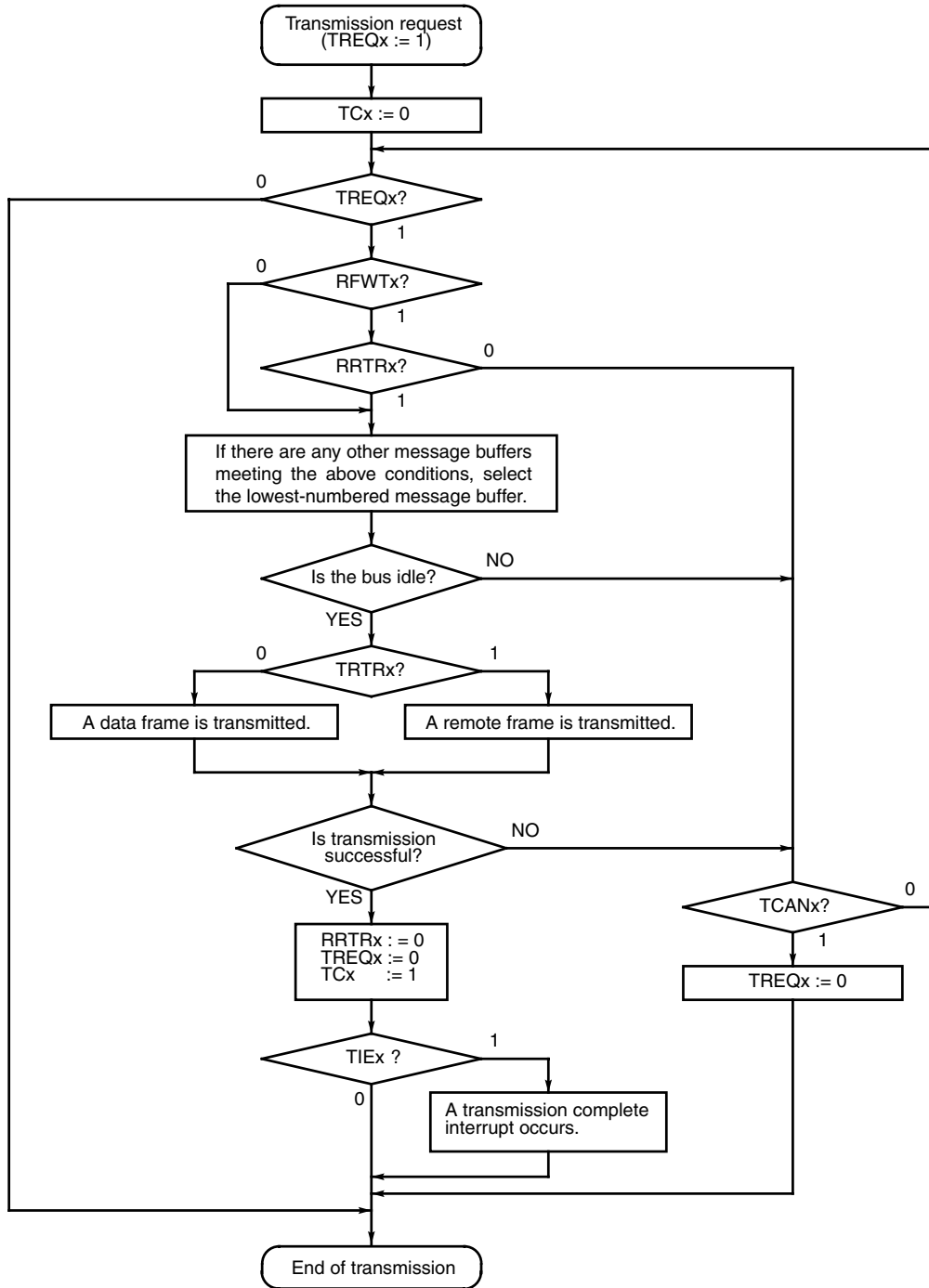
When transmission is successful, RRTRx becomes 0, TREQx becomes 0, and TCx of the transmission complete register (TCR) becomes 1.

If the transmission complete interrupt is enabled (TIEx of the transmission complete interrupt enable register (TIER) is 1), an interrupt occurs.

■ **Transmission Flowchart of the CAN Controller**

Figure 19.7-1 "Transmission Flowchart of the CAN Controller" shows a transmission flowchart of the CAN controller.

Figure 19.7-1 Transmission Flowchart of the CAN Controller



19.8 Reception of CAN Controller

Reception starts when the start of data frame or remote frame (SOF) is detected on the CAN bus.

■ Acceptance Filtering

The received message in the standard frame format is compared with the message buffer (x) set in the standard frame format (IDEx of the IDE register (IDER) is 0). The received message in the extended frame format is compared with the message buffer (x) set (IDEx is 1) in the extended frame format.

If all the bits set to Compare by the acceptance mask agree after comparison between the received message ID and acceptance code (ID register (IDRx) for comparing with the received message ID), the received message passes to the acceptance filter of the message buffer (x).

■ Storing Received Message

When the receive operation is successful, received messages are stored in a message buffer x including IDs passed through the acceptance filter.

When receiving data frames, received messages are stored in the ID register (IDRx), DLC register (DLCRx), and data register (DTRx).

Even if received message data is less than 8 bytes, some data is stored in the remaining bytes of the DTRx and its value is undefined.

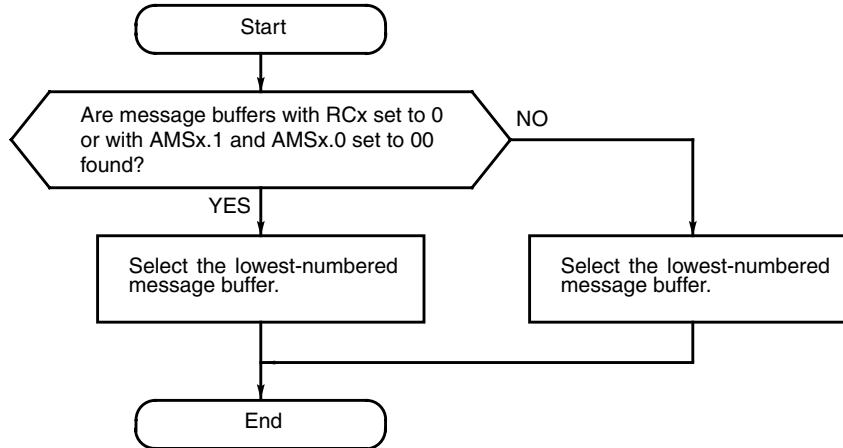
When receiving remote frames, received messages are stored only in the IDRx and DLCRx, and the DTRx remains unchanged.

If there is more than one message buffer including IDs passed through the acceptance filter, the message buffer x in which received messages are to be stored is determined according to the following rules.

- The order of priority of the message buffer x (x = 0 to 15) rises as its number lower; in other words, message buffer 0 is given the highest and the message buffer 15 is given the lowest priority.
- Basically, message buffers with the RCx bit of 0 in the receive completion register (RCR) are preferred in storing received messages.
- If the bits of the acceptance mask select register (AMSR) are set to All Bits Compare (for message buffers with the AMSx.1 and AMSx.0 bits set to 00), received messages are stored irrespective of the value of the RCx bit of the RCR.
- If there are message buffers with the RCx bit of the RCR set to 0, or with the bits of the AMSR set to All Bits Compare, received messages are stored in the lowest-number (highest-priority) message buffer x.
- If there are no message buffers above-mentioned, received messages are stored in a lower-number message buffer x.
- Message buffers should be arranged in ascending numeric order. The lowest message buffers should be with All Bits Compare, then AMR0 or AMR1 masks. And The highest message buffers should be with All Bits Mask.

Figure 19.8-1 "Flowchart Determining Message Buffer (x) where Received Messages Stored" shows a flowchart for determining the message buffer (x) where received messages are to be stored. It is recommended that message buffers be arranged in the following order: message buffers in which each AMSR bit is set to All Bits Compare, message buffers using AMR0 or AMR1, and message buffers in which each AMSR bit is set to All Bits Mask.

Figure 19.8-1 Flowchart Determining Message Buffer (x) where Received Messages Stored



■ **Receive Overrun**

When a message is stored in the message buffer with the corresponding RCx being already set to 1, it will result in receive overrun. In this case, the corresponding ROVRx bit in the receive overrun register ROVRR is set to 1.

■ **Processing for Reception of Data Frame and Remote Frame**

○ **Processing for reception of data frame**

RRTRx of the remote request receiving register (RRTRR) becomes 0.

TREQx of the transmission request register (TREQR) becomes 0 (immediately before storing the received message). A transmission request for message buffer (x) having not executed transmission will be canceled.

Note:

A request for transmission of either a data frame or remote frame is canceled.

○ **Processing for reception of remote frame**

RRTRx becomes 1.

If TRTRx of the transmitting RTR register (TRTRR) is 1, TREQx becomes 0. As a result, the request for transmitting remote frame to message buffer having not executed transmission will be canceled.

Note:

A request for data frame transmission is not canceled.

For cancellation of a transmission request, see Figure 19.7-1 "Transmission Flowchart of the CAN Controller".

■ Completing Reception

RCx of the reception complete register (RCR) becomes 1 after storing the received message.

If a reception interrupt is enabled (RIEx of the reception interrupt enable register (RIER) is 1), an interrupt occurs.

Note:

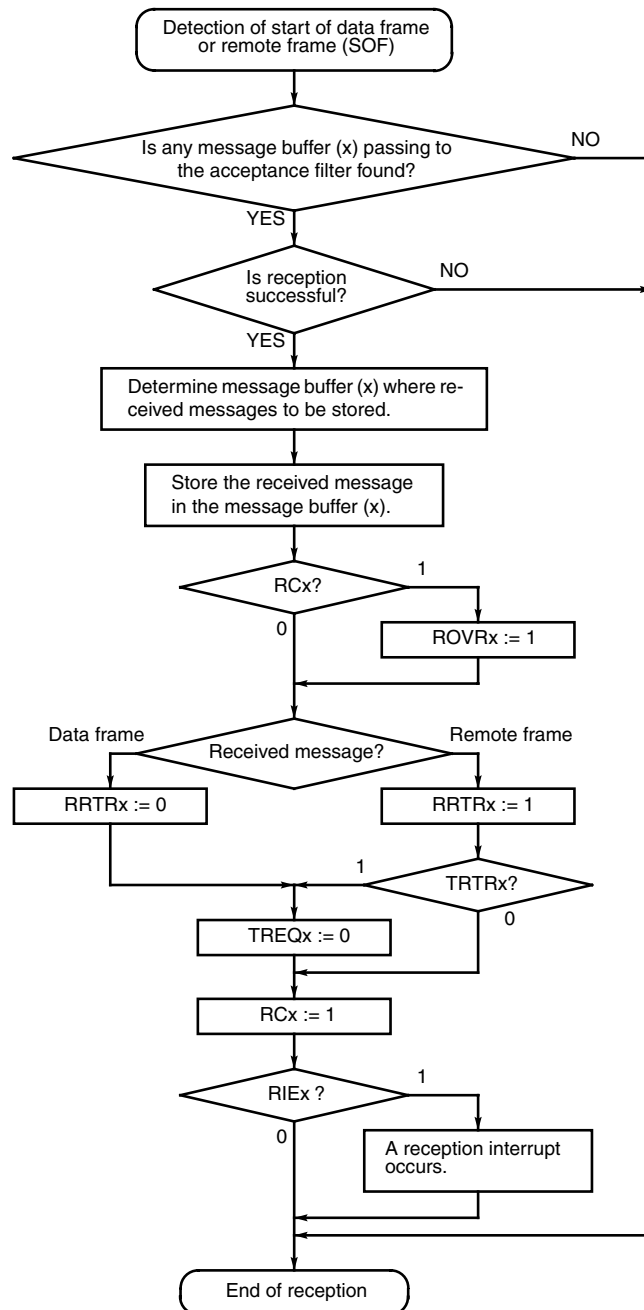
This CAN controller will not receive any messages transmitted by itself.

19.9 Reception Flowchart of CAN Controller

Figure 19.9-1 "Reception Flowchart of the CAN Controller" shows a reception flowchart of the CAN controller.

■ Reception Flowchart of the CAN Controller

Figure 19.9-1 Reception Flowchart of the CAN Controller



19.10 How to Use the CAN Controller

The following settings are required to use the CAN controller:

- Bit timing
 - Frame format
 - ID
 - Acceptance filter
 - Low-power consumption mode
-

■ Setting Bit Timing

The bit timing register (BTR) should be set during bus operation stop (when the bus operation stop bit (HALT) of the control status register (CSR) is 1).

After the setting completion, write 0 to HALT to cancel bus operation stop.

■ Setting Frame Format

Set the frame format used by the message buffer (x). When using the standard frame format, set IDE_x of the IDE register (IDER) to 0. When using the extended frame format, set IDE_x to 1.

This setting should be made when the message buffer (x) is invalid (BVAL_x of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVAL_x = 1) may cause unnecessary received messages to be stored.

■ Setting ID

Set the message buffer (x) ID to ID₂₈ to ID₀ of ID register (IDR_x). The message buffer (x) ID need not be set to ID₁₁ to ID₀ in the standard frame format. The message buffer (x) ID is used as a transmission message at transmission and is used as an acceptance code at reception.

This setting should be made when the message buffer (x) is invalid (BVAL_x of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVAL_x = 1) may cause unnecessary received messages to be stored.

■ Setting Acceptance Filter

The acceptance filter of the message buffer (x) is set by an acceptance code and acceptance mask set. It should be set when the acceptance message buffer (x) is invalid (BVAL_x of the message buffer enable register (BVALR) is 0). Setting when the buffer is valid (BVAL_x = 1) may cause unnecessary received messages to be stored.

Set the acceptance mask used in each message buffer (x) by the acceptance mask select register (AMSR). The acceptance mask registers (AMR₀ and AMR₁) should also be set if used (For the setting details, see 19.6.18 "Acceptance Mask Select Register (AMSR)" and 19.6.19 "Acceptance Mask Registers 0 and 1 (AMR₀ and AMR₁)").

The acceptance mask should be set so that a transmission request may not be canceled when unnecessary received messages are stored. For example, it should be set to a full-bit comparison if only one specific ID is used for the transmission.

■ Setting Low-power Consumption Mode

To set the F²MC-16LX in a low-power consumption mode (Stop, Watch, Hardware Standby, etc.), write 1 to the bus operation stop bit (HALT) of the control status register (CSR), and then check that the bus operation has stopped (HALT = 1).

19.11 Procedure for Transmission by Message Buffer (x)

After setting the bit timing, frame format, ID, and acceptance filter, set BVALx to 1 to activate the message buffer (x).

■ Procedure for Transmission by Message Buffer (x)

○ Setting transmit data length code

Set the transmit data length code (byte count) to DLC3 to DLC0 of the DLC register (DLCRx).

For data frame transmission (when TRTRx of the transmission RTR register (TRTRR) is 0), set the data length of the transmitted message.

For remote frame transmission (when TRTRx = 1), set the data length (byte count) of the requested message.

Note:

Setting other than 0000 to 1000 (0 to 8 bytes) is prohibited.

○ Setting transmit data (only for transmission of data frame)

For data frame transmission (when TRTRx of the transmission register (TRTRR) is 0), set data as the count of byte transmitted in the data register (DTRx).

Note:

Transmit data should be rewritten while the TREQx bit of the transmission request register (TREQR) set to 0. There is no need for setting the BVALx bit of the message buffer valid register (BVALR) to 0. Setting the BVALx bit to 0 may cause incoming remote frame to be lost.

○ Setting transmission RTR register

For data frame transmission, set TRTRx of the transmission RTR register (TRTRR) to 0.

For remote frame transmission, set TRTRx to 1.

○ Setting conditions for starting transmission (only for transmission of data frame)

Set RFWTx of the remote frame receiving wait register (RFWTR) to 0 to start transmission immediately after a request for data frame transmission is set (TREQx of the transmission request register (TREQR) is 1 and TRTRx of the transmission RTR register (TRTRR) is 0).

Set RFWTx to 1 to start transmission after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR) becomes 1) after a request for data frame transmission is set (TREQx = 1 and TRTRx = 0).

Note:

Remote frame transmission can not be made, if RFWTx is set to 1.

○ **Setting transmission complete interrupt**

When generating a transmission complete interrupt, set TIE_x of the transmission complete interrupt enable register (TIER) to 1.

When not generating a transmission complete interrupt, set TIE_x to 0.

○ **Setting transmission request**

For a transmission request, set TREQ_x of the transmission request register (TREQR) to 1.

○ **Canceling transmission request**

When canceling a pending request for transmission to the message buffer (x), write 1 to TCAN_x of the transmission cancel register (TCANR).

Check TREQ_x. For TREQ_x = 0, transmission cancellation is terminated or transmission is completed. Check TC_x of the transmission complete register (TCR). For TC_x = 0, transmission cancellation is terminated. For TC_x = 1, transmission is completed.

○ **Processing for completion of transmission**

If transmission is successful, TC_x of the transmission complete register (TCR) becomes 1.

If the transmission complete interrupt is enabled (TIE_x of the transmission complete interrupt enable register (TIER) is 1), an interrupt occurs.

After checking the transmission completion, write 0 to TC_x to set it to 0. This cancels the transmission complete interrupt.

In the following cases, the pending transmission request is canceled by receiving and storing a message.

- Request for data frame transmission by reception of data frame
- Request for remote frame transmission by reception of data frame
- Request for remote frame transmission by reception of remote frame

Request for data frame transmission is not canceled by receiving and storing a remote frame. ID and DLC, however, are changed by the ID and DLC of the received remote frame. Note that the ID and DLC of data frame to be transmitted become the value of received remote frame.

19.12 Procedure for Reception by Message Buffer (x)

After setting the bit timing, frame format, ID, and acceptance filter, make the settings described below.

■ Procedure for Reception by Message Buffer (x)

○ Setting reception interrupt

To enable reception interrupt, set RIE_x of the reception interrupt enable register (RIER) to 1.

To disable reception interrupt, set RIE_x to 0.

○ Starting reception

When starting reception after setting, set BVAL_x of the message buffer valid register (BVALR) to 1 to make the message buffer (x) valid.

○ Processing for reception completion

If reception is successful after passing to the acceptance filter, the received message is stored in the message buffer (x) and RC_x of the reception complete register (RCR) becomes 1. For data frame reception, RRTR_x of the remote request receiving register (RRTRR) becomes 0. For remote frame reception, RRTR_x becomes 1.

If a reception interrupt is enabled (RIE_x of the reception interrupt enable register (RIER) is 1), an interrupt occurs.

After checking the reception completion (RC_x = 1), process the received message.

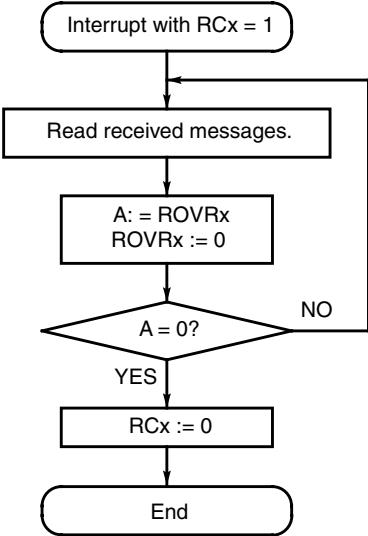
After completion of processing the received message, check ROVR_x of the reception overrun register (ROVRR).

If ROVR_x = 0, the processed received message is valid. Write 0 to RCR_x to set it to 0 (the reception complete interrupt is also canceled) to terminate reception.

If ROVR_x = 1, a reception overrun occurred and the next message may have overwritten the processed message. In this case, received messages should be processed again after setting the ROVR_x bit to 0 by writing 0 to it.

Figure 19.12-1 "Example of Receive Interrupt Handling" shows an example of receive interrupt handling.

Figure 19.12-1 Example of Receive Interrupt Handling



19.13 Setting Configuration of Multi-level Message Buffer

If the receptions are performed frequently, or if several different ID's of messages are received, in other words, if there is insufficient time for handling messages, more than one message buffer can be combined into a multi-level message buffer to provide allowance for processing time of the received message by CPU.

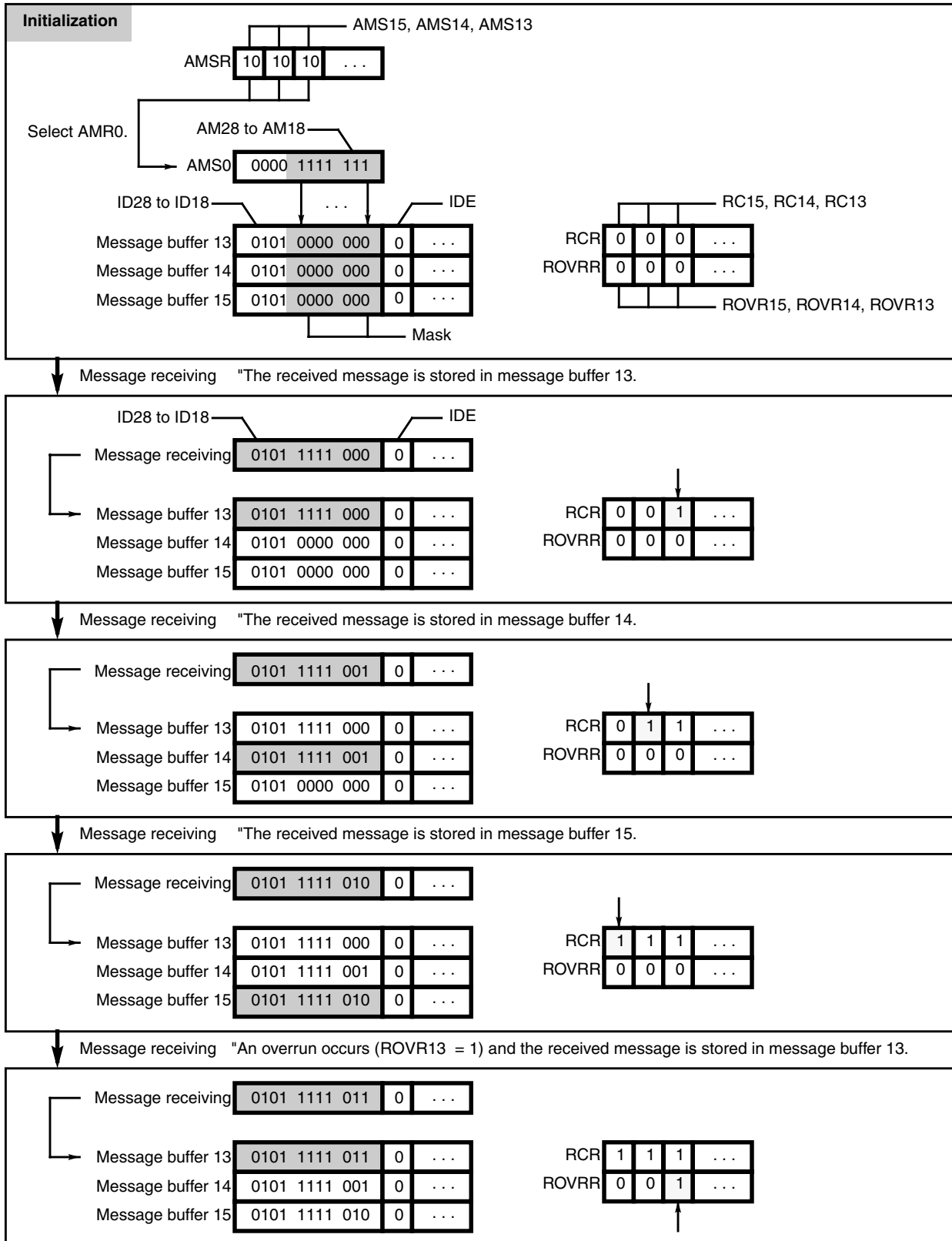
■ Setting Configuration of Multi-level Message Buffer

To provide a multi-level message buffer, the same acceptance filter must be set in the combined message buffers.

If the bits of the acceptance mask select register (AMSR) are set to All Bits Compare ((AM_{Sx.1}, AM_{Sx.0}) = (0, 0)), multi-level message configuration of message buffers is not allowed. This is because All Bits Compare causes received messages to be stored irrespective of the value of the RC_x bit of the receive completion register (RCR), so received messages are always stored in lower-numbered (lower-priority) message buffers even if All Bits Compare and identical acceptance code (ID register (IDR_x)) are specified for more than one message buffer. Therefore, All Bits Compare and identical acceptance code should not be specified for more than one message buffer.

Figure 19.13-1 "Examples of Operation of Multi-level Message Buffer" shows operational examples of multi-level message buffers.

Figure 19.13-1 Examples of Operation of Multi-level Message Buffer



Note:

Four messages are received with the same acceptance filter set in message buffers 13, 14 and 15.

19.14 Precautions when Using CAN Controller

Use of the CAN Controller requires the following cautions.

■ Caution for Disabling Message Buffers by BVAL bits

The use of BVAL bits may affect malfunction of CAN Controller when messages buffers are set disabled while CAN Controller is participating in CAN communication (read value of HALT bit is 0 and CAN Controller is ready to receive or transmit messages). This section shows the work around of this malfunction.

○ Condition

When following two conditions occur at the same time, CAN Controller will not perform to receive or transmit messages normally.

- CAN Controller is participating in the CAN communication. (i.e. The read value of HALT bit is 0 and CAN Controller is ready to receive or transmit messages)
- Message buffers are read or written when the message buffers are disabled by BVAL bits.

○ Work around

Operation for re-configuring receiving message buffers

While CAN Controller is participating in CAN communication (the read value of HALT bit is 0 and CAN Controller is ready to receive or transmit messages), it is necessary to following one from the two operations described below to re-configure message buffers by ID, AMS and AMR0/1 register-settings.

- Use of HALT bit
 - Write 1 to HALT bit and read it back for checking the result is 1. Then change the settings for ID/AMS/AMR0/1 registers.
- No Use of Message Buffer 0
 - Don't use the message buffer 0. In other words, disable message buffer (BVAL0=0), prohibit receive interrupt (RIE0=0) and do not request transmission (TREQ0=0).

Operation for processing received message

Don't use the receiving prohibition by BVAL bit to avoid over-written of next message. Use the ROVR bit for checking if over-write has been performed. For details, refer to sections 19.6.16 "Receive Overrun Register (ROVRR)" and 19.12 "Procedure for Reception by Message Buffer (x)".

Operation for suppressing transmission request

Don't use BVAL bit for suppressing transmission request, use TCAN bit instead of it.

Operation for composing transmission message

For composing a transmission message, it is necessary to disable the message buffer by BVAL bit to change contents of ID and IDE registers. In this case, BVAL bit should reset (BVAL=0) after checking if TREQ bit is 0 or after completion of the previous message transmission (TC=1).

CHAPTER 20 STEPPING MOTOR CONTROLLER

This chapter explains the functions and operations of the stepping motor controller.

20.1 "Outline of Stepping Motor Controller"

20.2 "Stepping Motor Controller Registers"

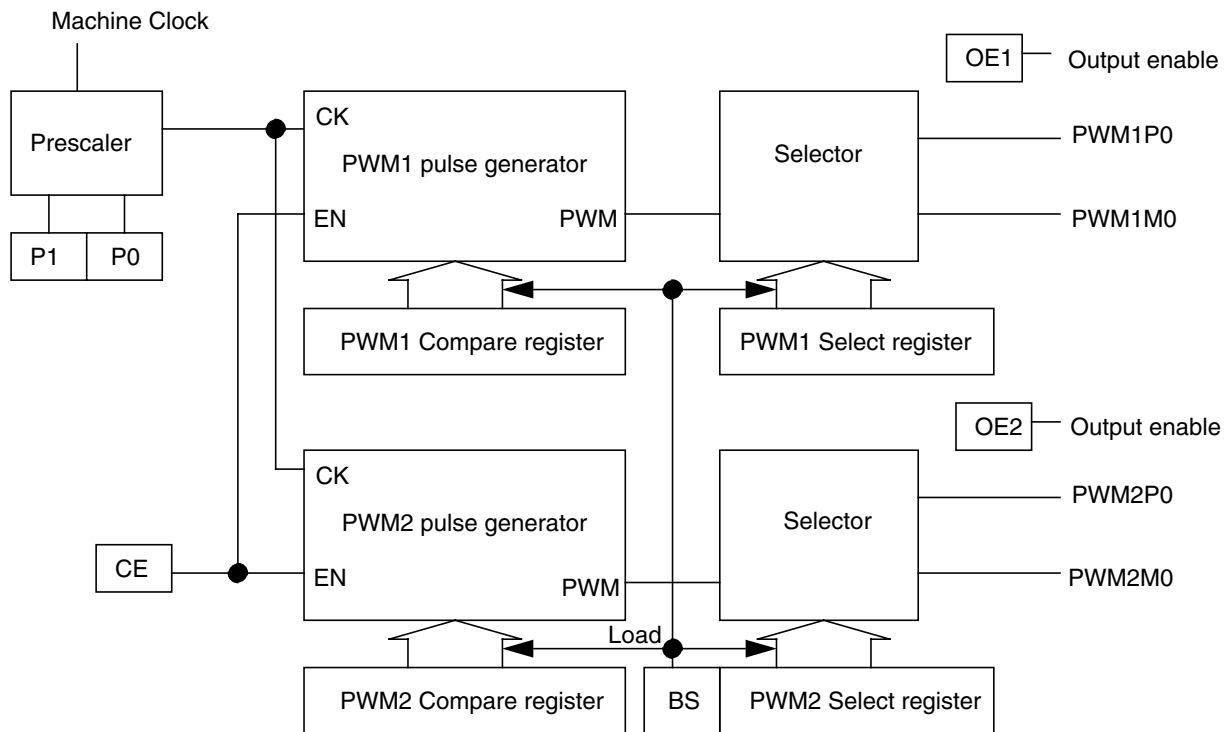
20.1 Outline of Stepping Motor Controller

The Stepping Motor Controller consists of two PWM Pulse Generators, four motor drivers and Selector Logic. The four motor drivers have high output drive capabilities and they can be directly connected to the four ends of two motor coils. The combination of the PWM Pulse Generators and Selector Logic is designed to control the rotation of the motor. A Synchronization mechanism assures the synchronous operations of the two PWMs. The following sections describe the Stepping Motor Controller 0 only. The other controllers have the same functions. The register addresses are found in the I/O map.

■ Block Diagram of Stepping Motor Controller

Figure 20.1-1 "Block Diagram of Stepping Motor Controller" shows a block diagram of the stepping motor controller.

Figure 20.1-1 Block Diagram of Stepping Motor Controller

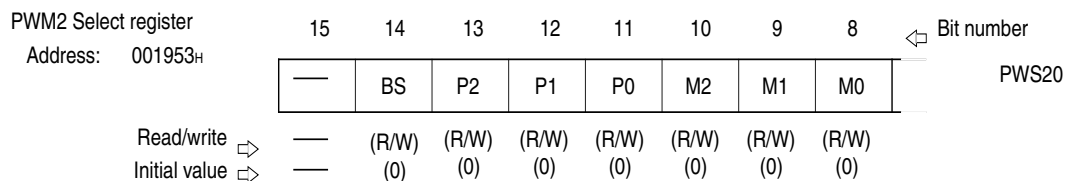
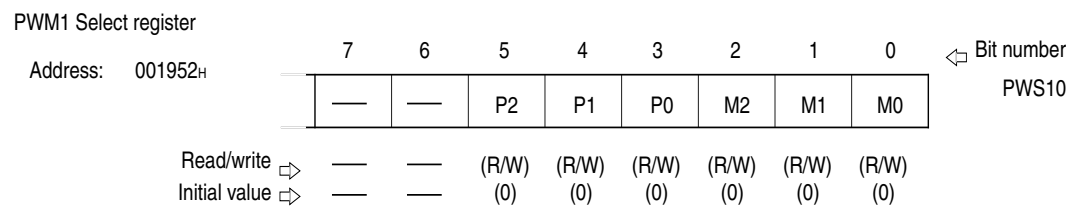
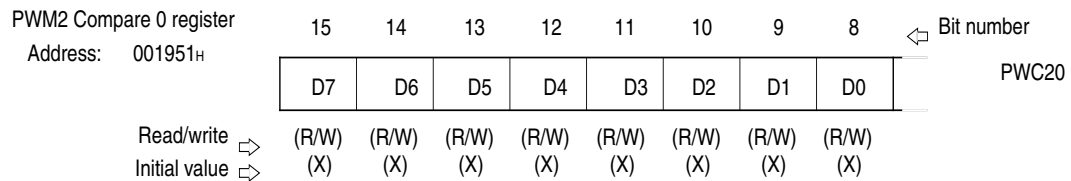
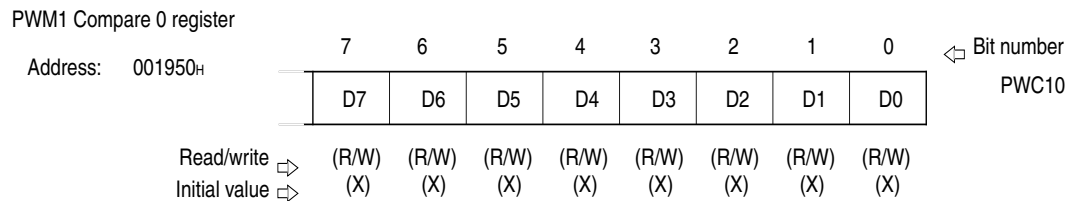
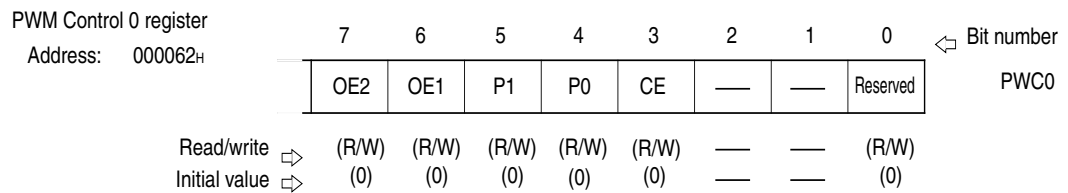


20.2 Stepping Motor Controller Registers

The stepping motor controller has the following five types of registers:

- PWM control 0 register (PWMC0)
- PWM1 compare 0 register (PWC10)
- PWM2 compare 0 register (PWC20)
- PWM1 select register (PWS10)
- PWM2 select register (PWS20)

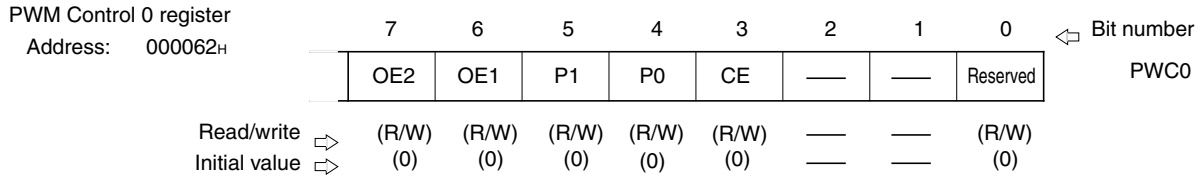
■ Stepping Motor Controller Registers



20.2.1 PWM Control 0 register

The PWM control 0 register starts and stops the stepping motor controller, controls interrupts, and sets the external output pins.

■ PWM Control 0 Register



[bits 7] OE2: Output enable bit

When this bit is set to "1", the external pins are assigned as PWM2P0 and PWM2M0 outputs. Otherwise they can be used as general purpose IO.

[bits 6] OE1: Output enable bit

When this bit is set to "1", the external pins are assigned as PWM1P0 and PWM1M0 outputs. Otherwise they can be used as general purpose IO.

[bits 5 to 4] P1 to P0: Operation clock select bits

These bits specify the clock input signal for the PWM pulse generators.

P1	P0	Clock input
0	0	Machine clock
0	1	1/2 Machine clock
1	0	1/4 Machine clock
1	1	1/8 Machine clock

[bits 3] CE: Count enable bit

This bit enables the operation of the PWM pulse generators. When it is set to "1", the PWM pulse generators start their operation. Note that the PWM2 pulse generator starts the operation one machine clock cycle after the PWM1 pulse generators is started. This is to help reduce the switching noise from the output drivers.

[bits 0] Reserved bit

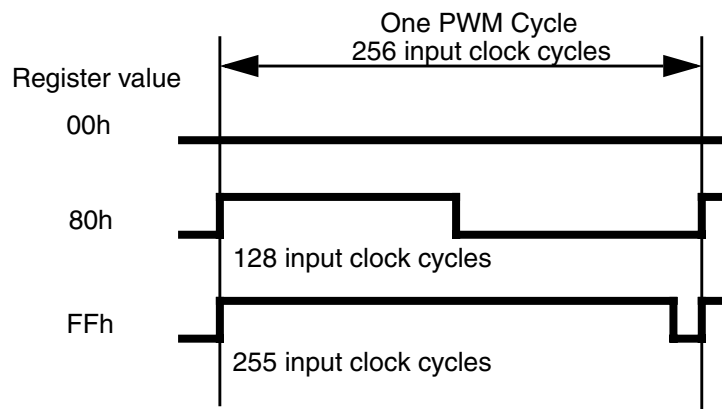
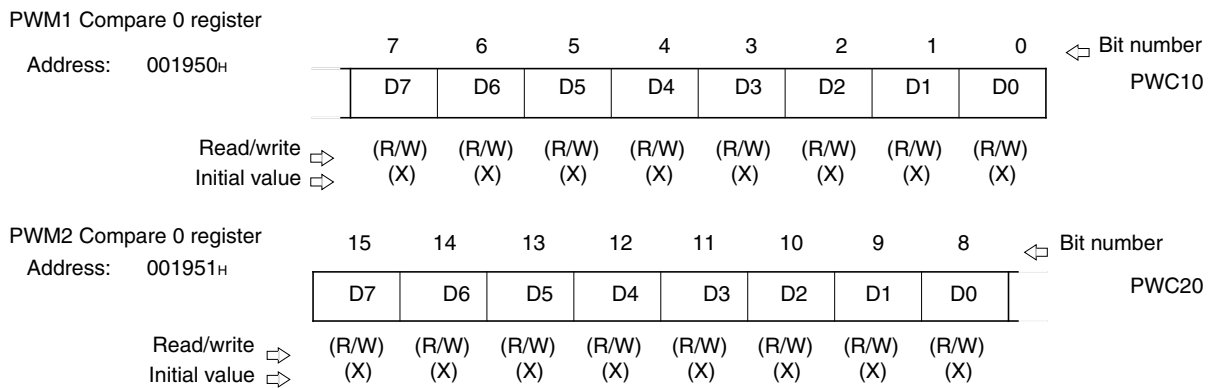
This is a reserved bit. Always write "0" to this bit.

20.2.2 PWM1&2 Compare Registers

The contents of the two 8-bit compare registers determine the widths of PWM pulses. The stored value of "00H" represents the PWM duty of 0% and "FF_H" represents the duty of 99.6%.

■ PWM1&2 Compare Registers

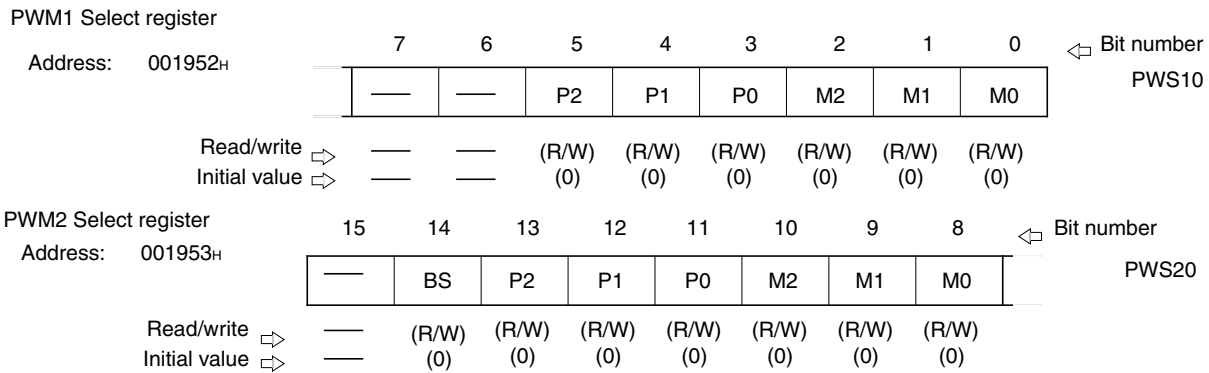
PWM1&2 compare registers are accessible at any time, however the modified values are reflected to the pulse width at the end of the current PWM cycle after the BS bit of the PWM2 Select register is set to "1".



20.2.3 PWM1&2 Select Registers

The PWM1 and PWM2 select registers select 0, 1, the PWM pulse, or high impedance for the external pin output of the stepping motor controller.

■ PWM1&2 Select Registers



[bits 14] BS: Update bit

This bit is prepared to synchronize the settings for the PWM outputs. Any modifications in the two compare registers and two select registers are not reflected to the output signals until this bit is set.

When this bit is set to "1", the PWM pulse generators and selectors load the register contents at the end of the current PWM cycle. The BS bit is reset to "0" automatically at the beginning of the next PWM cycle. If the BS bit is set to "1" by software at the same time as this automatic reset, the BS bit is set to "1" (or remains unchanged) and the automatic reset is cancelled.

[bits 13 to 11] P2 to P0: Output Select bits

These bits selects the output signal at PWM2P0.

[bits 10 to 8] M2 to M0: Output Select bits

These bits selects the output signal at PWM2M0.

[bits 5 to 3] P2 to P0: Output Select bits

These bits selects the output signal at PWM1P0.

[bits 2 to 0] M2 to M0: Output Select bits

These bits selects the output signal at PWM1M0.

The following table shows the relationship between the output levels and select bits.

P2	P1	P0	PWMnP0	M2	M1	M0	PWMnM0
0	0	0	L	0	0	0	L
0	0	1	H	0	0	1	H
0	1	X	PWM pulses	0	1	X	PWM pulses
1	X	X	High impedance	1	X	X	High impedance

CHAPTER 21 SOUND GENERATOR

This chapter explains the functions and operations of the sound generator.

21.1 "Outline of Sound Generator"

21.2 "Sound Generator Registers"

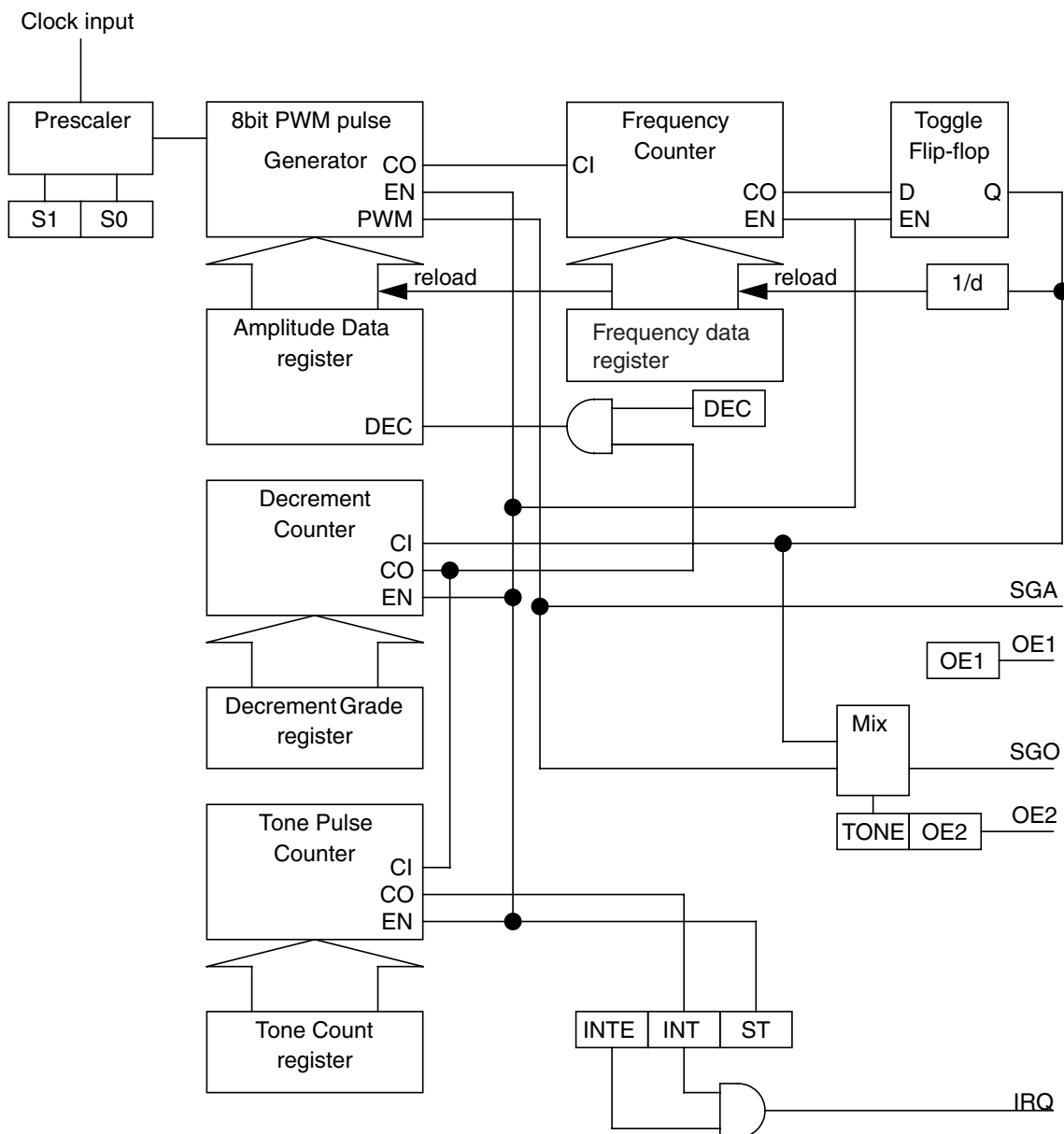
21.1 Outline of Sound Generator

The Sound Generator consists of the Sound Control register, Frequency Data register, Amplitude Data register, Decrement Grade register, Tone Count register, PWM pulse generator, Frequency counter, Decrement counter and Tone Pulse counter.

■ Block Diagram of Sound Generator

Figure 21.1-1 "Block Diagram of Sound Generator" shows a block diagram of the sound generator.

Figure 21.1-1 Block Diagram of Sound Generator

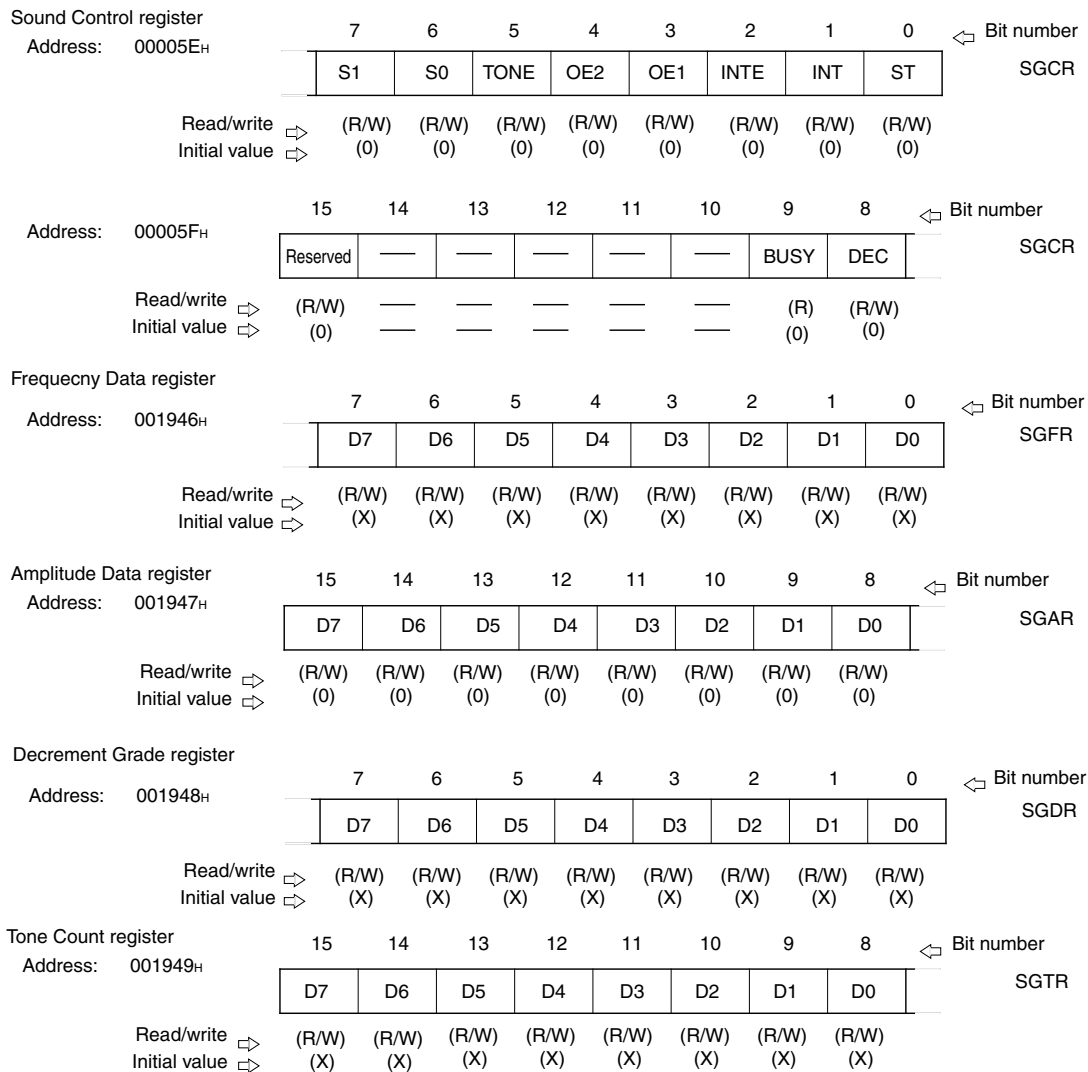


21.2 Sound Generator Registers

The sound generator has the following types of registers:

- Sound control register (SGCR)
- Frequency data register (SGFR)
- Amplitude data register (SGAR)
- Decrement grade register (SGDR)
- Tone count register (SGTR)

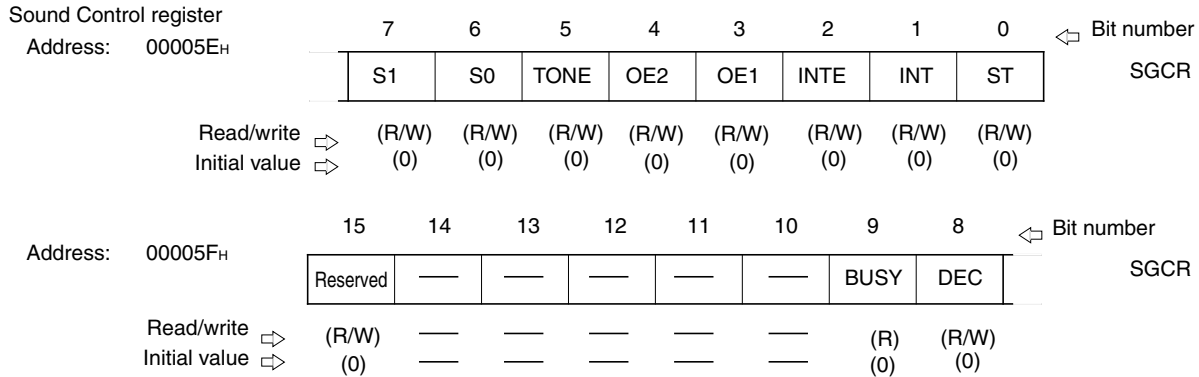
■ Sound Generator Registers



21.2.1 Sound Control Register

The sound control register controls the operation status of the sound generator by controlling interrupts and setting the external output pins.

■ Sound Control Register



[bits 15] Reserved bit

This is a reserved bit. Always write "0" to this bit.

[bits 9] BUSY: Busy bit

This bit indicates whether the Sound Generator is in operation. This bit is set to "1" upon the ST bit is set to "1". It is reset to "0" when the ST bit is reset to "0" and the operation is completed at the end of one tone cycle. Any write instructions performed on this bit has no effect.

[bits 8] DEC: Auto-decrement enable bit

The DEC bit is prepared for an automatic de-gradation of the sound in conjunction with the Decrement Grade register.

If this bit is set to "1", the stored value in the Amplitude Data register is decremented by 1(one), every time when the Decrement counter counts the number of tone pulses from the toggle flip-flop specified by the Decrement Grade register.

[bits 7 to 6] S1 to S0: Operation clock select bits

These bits specify the clock input signal for the Sound Generator.

S1	S0	Clock input
0	0	Machine clock
0	1	1/2 Machine clock
1	0	1/4 Machine clock
1	1	1/8 Machine clock

[bits 5] TONE: Tone output bit

When this bit is set to "1", the SGO signal becomes a simple square-waveform (tone pulses) from the toggle flip-flop. Otherwise it is the mixed (AND logic) signal of the tone and PWM pulses.

[bits 4] OE2: Sound output enable bit

When this bit is set to "1", the external pin is assigned as the SGO output. Otherwise the pin can be used as a general purpose IO. To enable the SGO output, the corresponding bit of the Port Direction register should also be set to "1".

[bits 3] OE1: Amplitude output enable bit

When this bit is set to "1", the external pin is assigned as the SGA output. Otherwise the pin can be used as a general purpose IO. To enable the SGA output, the corresponding bit of the Port Direction register should also be set to "1".

The SGA signal is the PWM pulses from the PWM pulse generator representing the amplitude of the sound.

[bits 2] INTE: Interrupt enable bit

This bit enables the interrupt signal of the Sound Generator. When this bit is "1" and the INT bit is set to "1", the Sound Generator signals an interrupt.

[bits 1] INT: Interrupt bit

This bit is set to "1" when the Tone Pulse counter counts the number of the tone pulses specified by the Tone Count register and Decrement Grade register.

This bit is reset to "0" by writing "0". Writing "1" has no effect and Read-Modify-Write instructions always result in reading "1".

[bits 0] ST: Start bit

This bit is for starting the operation of the Sound Generator. While this bit is "1", the Sound Generator perform its operation.

When this bit is reset to "0", the Sound Generator stops its operation at the end of the current tone cycle. The BUSY bit indicates whether the Sound Generator is fully stopped.

When this bit is changed from "0" to "1", the value of Frequency Data register, Amplitude Data register, Decrement Grade register, and Tone Count register is loaded into each counter.

21.2.2 Frequency Data register

The Frequency Data register stores the reload value for the Frequency counter. The stored value represents the frequency of the sound (or the tone signal from the toggle flip-flop). The register value is reloaded into the counter at Frequency counter underflow and PWM pulse generator underflow.

The following figure shows the relationship between the tone signal and the register value.

■ Frequency Data Register

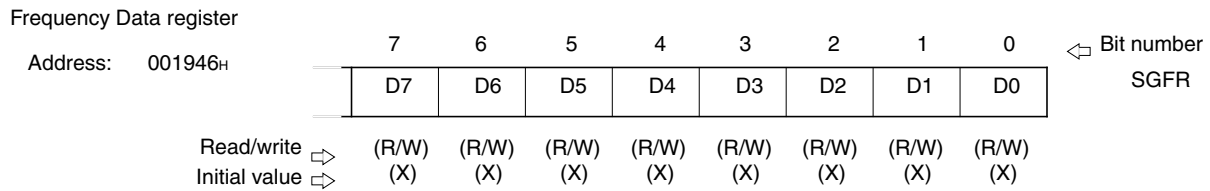
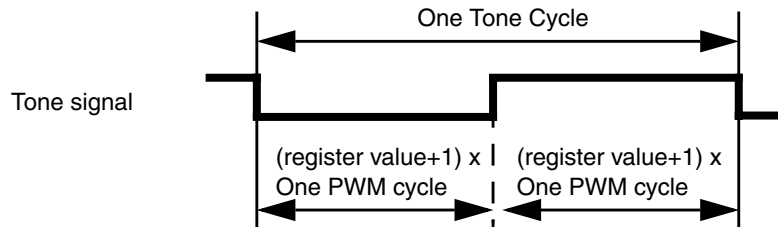


Figure 21.2-1 "Relationship between Tone Signal and Register Value" shows the relationship between a tone signal and a register value.

Figure 21.2-1 Relationship between Tone Signal and Register Value



It should be noted that modifications of the register value while operation may alter the duty cycle of 50% depending on the timing of the modification.

21.2.3 Amplitude Data Register

The Amplitude Data register stores the reload value for the PWM pulse generator. The register value represents the amplitude of the sound. The register value is reloaded into the PWM pulse generator at falling edge of tone signal.

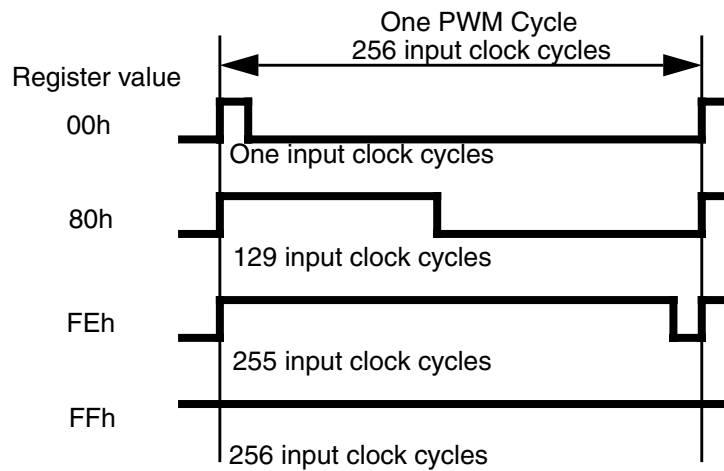
■ Amplitude Data Register

Amplitude Data register Address: 001947 _H	15	14	13	12	11	10	9	8	Bit number
	D7	D6	D5	D4	D3	D2	D1	D0	SGAR
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 the DEC bit is "1" and the Decrement counter reaches its reload value, this register value is decremented by 1(one). And when the register value reaches "00", further decrements are not performed. However the sound generator continues its operation until the ST bit is cleared.

Figure 21.2-2 "Relationship between Register Value and PWM Pulse" shows the relationship between the register value and the PWM pulse.

Figure 21.2-2 Relationship between Register Value and PWM Pulse



When the register value is set to "FF", the PWM signal is always "1".

21.2.4 Decrement Grade Register

The Decrement Grade register stores the reload value for the Decrement counter. They are prepared to automatically decrement the stored value in the Amplitude Data register. The register value is reloaded into the counter at Decrement counter underflow and falling edge of tone signal.

■ Decrement Grade Register

Decrement Grade register

Address: 001948H

	7	6	5	4	3	2	1	0	Bit number SGDR
	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)	

When the DEC bit is "1" and the Decrement counter counts the number of tone pulses up to the reload value, the stored value in the Amplitude Data register is decremented by 1(one) at the end of the tone cycle.

This operation realizes automatic de-gradation of the sound with fewer number of CPU interventions.

It should be noted that the number of the tone pulses specified by this register equals to "register value +1". When the Decrement Grade register is set to "00", the decrement operation is performed every tone cycle.

21.2.5 Tone Count Register

The Tone Count register stores the reload value for the Tone Pulse counter. The Tone Pulse counter accumulates the number of tone pulses (or number of decrement operations) and when it reaches the reload value it sets the INT bit. They are intended to reduce the frequency of interrupts. The register value is reloaded into the counter at Tone Pulse counter underflow, Decrement counter underflow, and falling edge of tone signal.

■ Tone Count Register

Tone Count register		15	14	13	12	11	10	9	8	Bit number
Address:	001949 _H									SGTR
		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)	

The count input of the Tone Pulse counter is connected to the carry-out signal from the Decrement counter. And when the Tone count register is set to "00", the Tone Pulse counter sets the INT bit every carry-out from the Decrement counter. Thus the number of accumulated tone pulses is;

$$((\text{Decrement Grade register}) + 1) \times ((\text{Tone Count register}) + 1)$$

i.e. When the both registers are set to "00", the INT bit is set every tone cycle.

CHAPTER 22 ADDRESS MATCH DETECTION FUNCTION

This chapter explains the address match detection function and operation.

22.1 "Outline of the Address Match Detection Function"

22.2 "Registers of the Address Match Detection Function"

22.3 "Operation of the Address Match Detection Function"

22.4 "Example of the Address Match Detection Function"

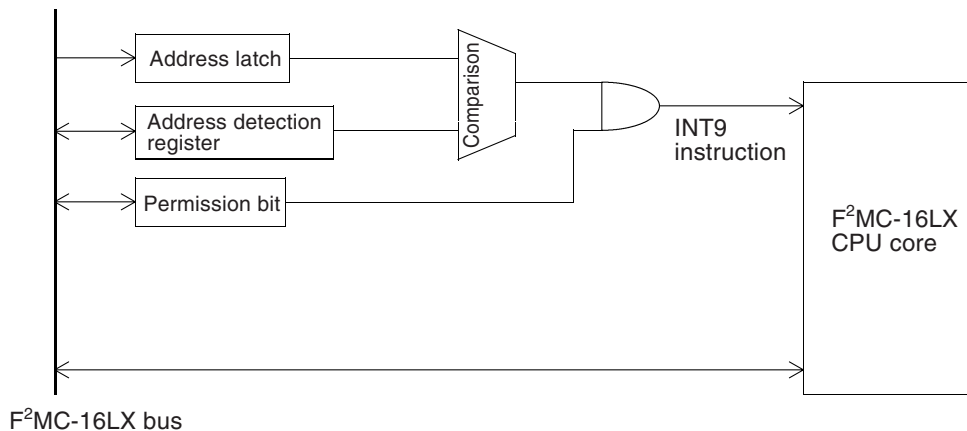
22.1 Outline of the Address Match Detection Function

When an address matches the value set in the address detection register, the instruction code to be read by the CPU is replaced with the INT9 instruction code (01H). Consequently, the CPU executes the INT9 instruction when executing a specified instruction. The address match detection function can be achieved using the INT9 interrupt routine for processing.

There are two address detection registers, each with an interrupt permission bit. When an address matches the value set in the address detection register and the interrupt permission bit is 1, the instruction code to be read by the CPU is replaced with the INT9 instruction code.

■ Block Diagram of the Address Match Detection Function

Figure 22.1-1 Block Diagram of the Address Match Detection Function



22.2 Registers of the Address Match Detection Function

The two types of registers for the address match detection function are as follows:

- Program address detection registers (PADR0 and PADR1)
- Program address detection control status register (PACSR)

■ Program Address Detection Registers (PADR0 and PADR1)

The program address detection registers 0 and 1 (PADR0 and PADR1) compare the address with the value written in each register. If they match when the interrupt permission bit corresponding to ADCSR is 1, the CPU is requested to issue the INT9 instruction.

When the corresponding interrupt bit is 0, nothing occurs.

Figure 22.2-1 Program Address Detection Registers (PADR0 and PADR1)

Program address detection registers	byte	byte	byte	Access	Initial value
PADR0 1FF2H/1FF1H/1FF0H				R/W	Not defined
PADR1 1FF5H/1FF4H/1FF3H				R/W	Not defined

Table 22.2-1 "Correspondence between PADR0 and PADR1 Registers and PACSR" lists the correspondence between the program address detection registers (PADR0 and PADR1) and PACSR.

Table 22.2-1 Correspondence between PADR0 and PADR1 Registers and PACSR

Address detection register	Interrupt permission bit
PADR0	AD0E
PADR1	AD1E

■ Program Address Detection Control Status Register (PACSR)

The program address detection control status register (PACSR) controls the operation of the address detection function.

Figure 22.2-2 Program Address Detection Control Status Register (PACSR)

Program address detection control status register	7	6	5	4	3	2	1	0	Bit No.
Address: 009EH	Reserved	Reserved	Reserved	Reserved	AD1E	Reserved	AD0E	Reserved	PACSR
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

[Bits 7 to 4] Reserved bits

Bits 7 to 4 are reserved. Set these bits to 0 before setting PACSR.

[Bit 3] AD1E (Address detect register 1 enable)

The AD1E bit is the operation permission bit of ASIE ADR1.

When this bit is 1, the address is compared with the PADR1 register. If they match, the INT9 instruction is issued.

CHAPTER 22 ADDRESS MATCH DETECTION FUNCTION

[Bit 2] Reserved bit

Bit 2 is reserved. Set this bit to 0 before setting PACSR.

[Bit 1] AD0E (Address Detect register 0 Enable)

The AD0E bit is the operation permission bit of ADR0.

When this bit is 1, the address is compared with the PADR0 register. If they match, the INT9 instruction is issued.

[Bit 0] Reserved bit

Bit 0 is reserved. Set this bit to 0 before setting PACSR.

22.3 Operation of the Address Match Detection Function

If the program counter specifies the same address as the address match detection register, the INT9 instruction is executed. The address match detection function can be achieved by processing the INT9 instruction routine.

■ Operation of the Address Match Detection Function

There are two address detection registers with a compare enable bit. When the value set in the address detection register and the value of the program counter match and the compare enable bit is set to 1, the CPU executes the INT9 instruction.

Note:

If the value of the address detection register and the value of the program counter match, the contents of internal data bus is changed to 01_H. Consequently, the INT9 instruction is executed. Before changing the contents of the address detection register, always set the compare enable bit to 0. While the compare enable bit is set to 1, changing the contents of the address detection register may result in a malfunction.

22.4 Example of the Address Match Detection Function

Figure 22.4-1 "System Configuration Example of the Address Match Detection Function" shows a system configuration example of the address match detection function. Table 22.4-1 "EEPROM Memory Map" lists the EEPROM memory map.

■ System Configuration Example of the Address Match Detection Function

Figure 22.4-1 System Configuration Example of the Address Match Detection Function

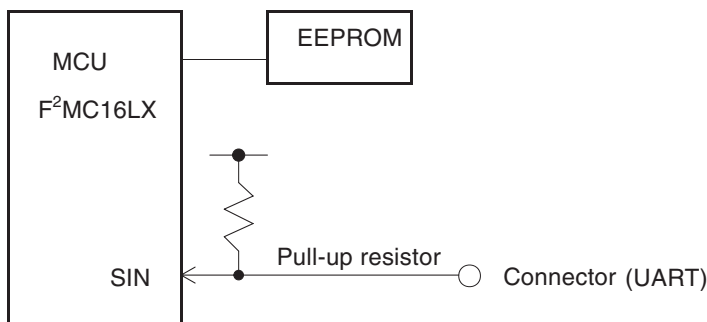


Table 22.4-1 EEPROM Memory Map

Address	Description
0000 _H	Number of bytes of patch program No.0 (If 0, no program error exists.)
0001 _H	Program address No.0 bits 7 to 0
0002 _H	Program address No.0 bits 15 to 8
0003 _H	Program address No.0 bits 24 to 16
0004 _H	Number of bytes of patch program No.1 (If 0, no program error exists.)
0005 _H	Program address No.1 bits 7 to 0
0006 _H	Program address No.1 bits 15 to 8
0007 _H	Program address No.1 bits 24 to 16
0010 _H or higher	Main body of patch program No. 0

○ **Initial status**

EEPROM is set to all 0s.

○ **When a program error occurs:**

The main body of the patch program and program address are transferred to the MCU through the connector (UART). The MCU writes the information to EEPROM.

○ **Reset sequence**

The MCU reads the value of EEPROM after reset. If the number of bytes of the patch program is not 0, the main body of the patch program is read from EEPROM and written to RAM. The MCU then uses either PADR0 or PADR1 to set the patch address and sets the compare enable bit. If the relocatable patch program is required, the first address of the patched program can be written to the RAM area. In this case, the INT9 routine accesses this user-defined RAM area and jumps to the patched program.

○ **INT9 interrupt**

The interrupt routine can know the address where the interrupt occurs by checking the value of the stack program counter. The information that has been placed on the stack during the interrupt is discarded.

■ **Example of program patch processing**

Figure 22.4-2 Example of program patch processing

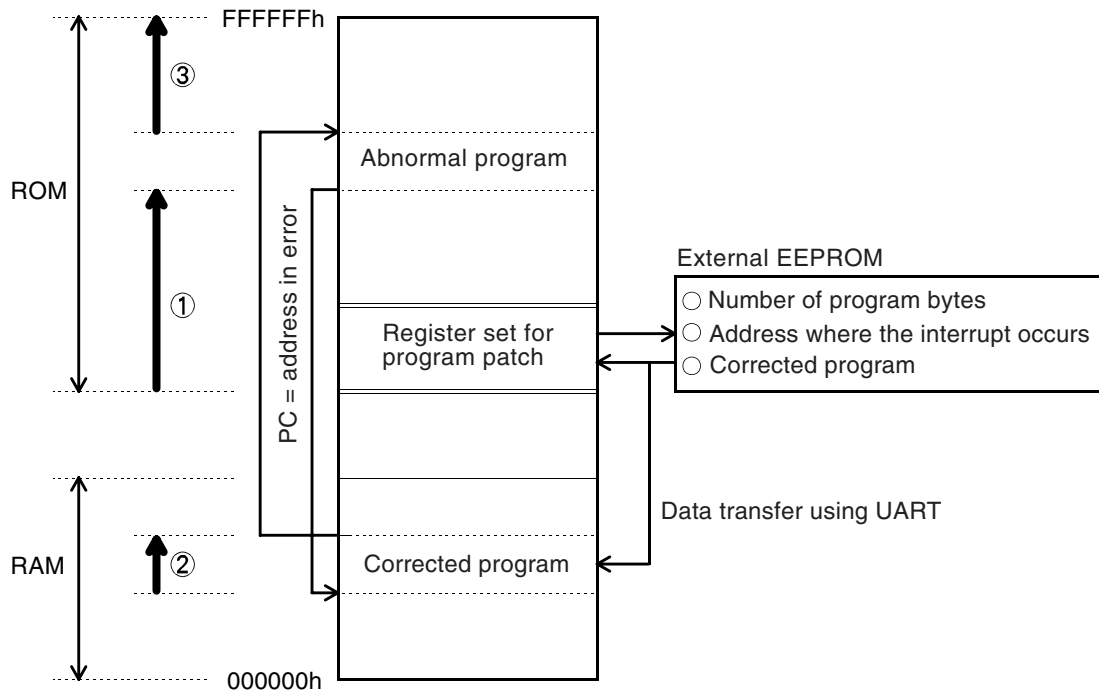
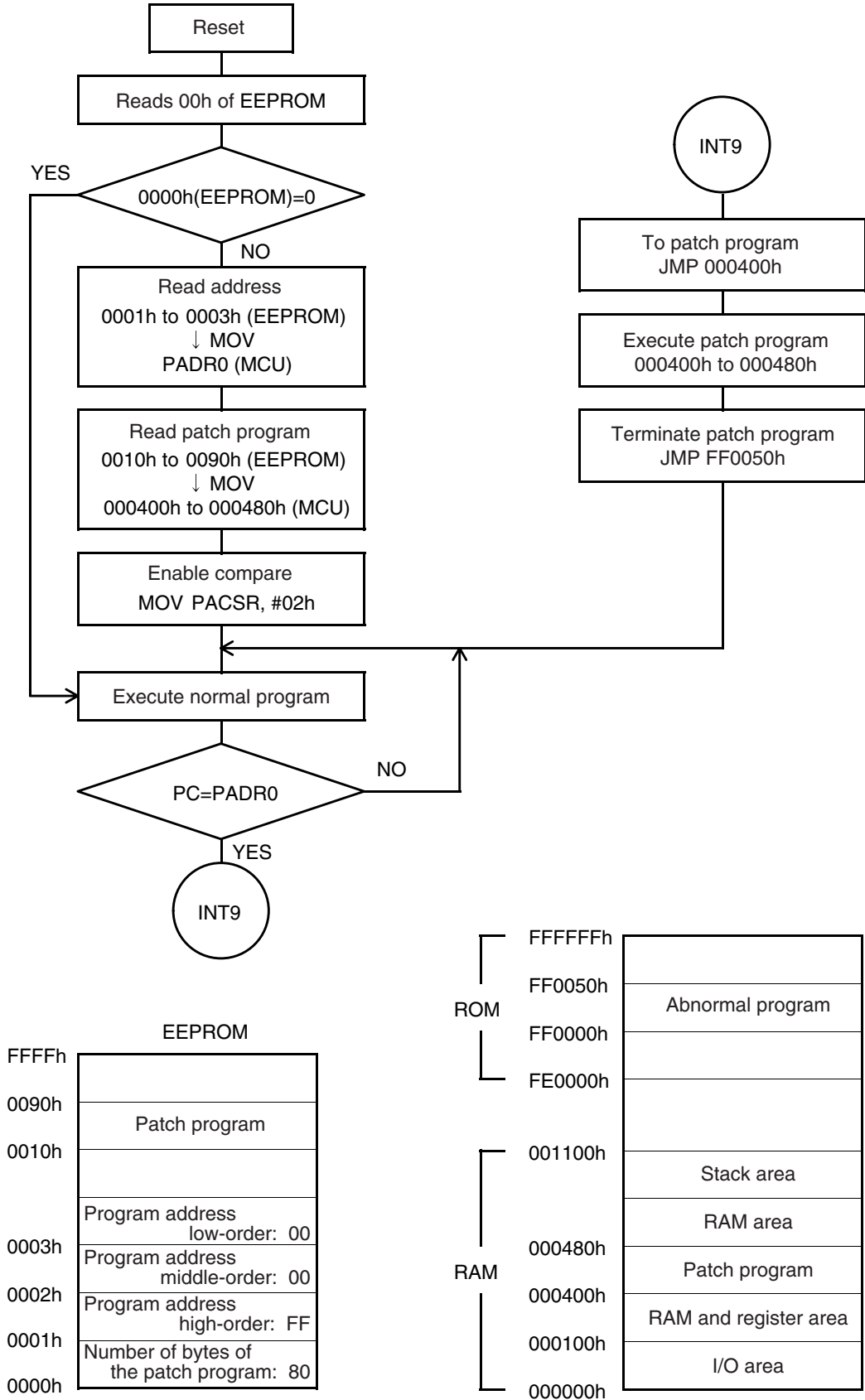


Figure 22.4-3 Flow of program patch processing



CHAPTER 23 ROM MIRRORING MODULE

This chapter explains the ROM mirroring module.

23.1 "Outline of ROM Mirroring Module"

23.2 "ROM Mirroring Register (ROMM)"

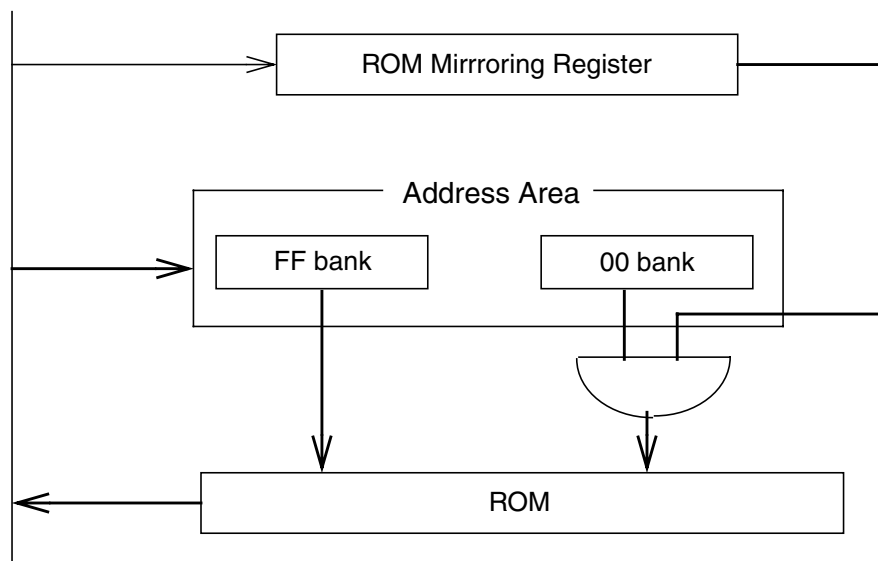
23.1 Outline of ROM Mirroring Module

The ROM Mirroring module switches whether to mirror the image of the FF bank of the ROM to the 00 bank.

■ Block Diagram of ROM Mirroring Module

Figure 23.1-1 Block Diagram of ROM Mirroring Module

F²MC-16LX BUS



23.2 ROM Mirroring Register (ROMM)

Do not access the ROM mirroring register (ROMM) when addresses 004000_H to 00FFFF_H are being accessed.

■ ROM Mirroring Register (ROMM)

	15	14	13	12	11	10	9	8	Bit number
Address : 0006F _H	—	—	—	—	—	—	—	MI	ROMM
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(1)	

[bit 8]: MI

The image of the ROM data in the FF bank can also be found in the 00 bank when "1" is written to this bit. However, this memory mapping will not be done when this bit is written to "0". This bit is write only.

Note:

Only FF4000 to FFFFFFFF is mirrored to 004000 to 00FFFF when ROM mirroring function is activated. Therefore, addresses FF0000 to FF3FFF will not be mirrored to 00 bank.

CHAPTER 24 2M/3M-BIT FLASH MEMORY

This chapter explains the functions and operation of the 2M/3M-bit flash memory. The following three methods are available for writing data to and erasing data from the flash memory:

- Parallel programmer
- Serial programmer
- Executing programs to write/erase data

This chapter explains "Executing programs to write/erase data".

- 24.1 "Outline of 2M/3M-bit Flash Memory"
- 24.2 "Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory"
- 24.3 "Write/Erase Modes"
- 24.4 "Flash Memory Control Status Register (FMCS)"
- 24.5 "Starting the Flash Memory Automatic Algorithm"
- 24.6 "Confirming the Automatic Algorithm Execution State"
- 24.7 "Detailed Explanation of Writing to and Erasing Flash Memory"
- 24.8 "Notes on Using 2M/3M-bit Flash Memory"
- 24.9 "Reset Vector Address in Flash Memory"
- 24.10 "Example of Programming 2M/3M-bit Flash Memory"

24.1 Overview of 2M/3M-bit Flash Memory

The 2M/3M-bit flash memory is mapped to the FC (F9) to FF bank in the CPU memory map. The functions of the flash memory interface circuit enable read-access and program-access from the CPU in the same way as mask ROM. Instructions from the CPU can be used via the flash memory interface circuit to write data to and erase data from the flash memory. Internal CPU control therefore enables rewriting of the flash memory while it is mounted. As a result, improvements in programs and data can be performed efficiently.

■ 2M/3M-bit Flash Memory Features

- Use of automatic program algorithm (Embedded Algorithm: Equivalent to MBM29LV200)
- Erase pause/restart functions provided
- Detection of completion of writing/erasing using data polling or toggle bit functions
- Detection of completion of writing/erasing using CPU interrupts
- Sector erase function (any combination of sectors)
- Minimum of 10,000 write/erase operations
- Flash memory read cycle time (minimum): 2 machine cycles

Embedded Algorithm is a trademark of Advanced Micro Device, Inc.

Note:

The manufacturer code and device code do not have the reading function. These codes cannot be accessed by the command.

■ Writing to/Erasing Flash Memory

The flash memory cannot be written to and read at the same time. That is, when data is written to or erased data from the flash memory, the program in the flash memory must first be copied to RAM. The entire process is then executed in RAM so that data is simply written to the flash memory. This eliminates the need for the program to access the flash memory from the flash memory itself.

■ Flash Memory Register

○ Flash Memory Control Status Register (FMCS)

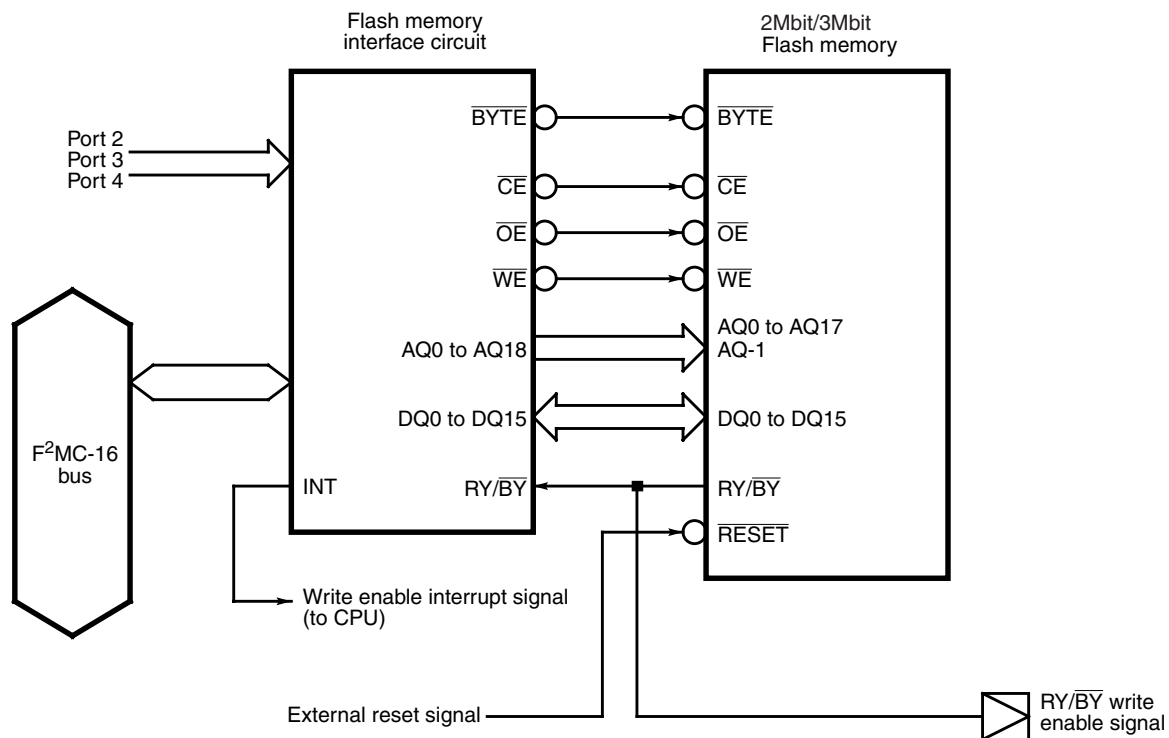
	7	6	5	4	3	2	1	0	← Bit No.
Address: 0000AE _H	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)	

24.2 Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory

Figure 24.2-1 "Block Diagram of the Entire Flash Memory" shows a block diagram of the entire flash memory with the flash memory interface circuit included. Figure 24.2-2 "Sector Configuration of the 2M/3M-bit Flash Memory" shows the sector configuration of the flash memory.

■ Block Diagram of the Entire Flash Memory

Figure 24.2-1 Block Diagram of the Entire Flash Memory



■ Sector Configuration of the 2M/3M-bit Flash Memory

Figure 24.2-2 "Sector Configuration of the 2M/3M-bit Flash Memory" shows the sector configuration of the 2M/3M-bit flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

Figure 24.2-2 Sector Configuration of the 2M/3M-bit Flash Memory

MB90F594A/MB90F594G (2M-bit flash memory)

MB90F591A/MB90F591G (3M-bit flash memory)

	Programmer address*	CPU address		Programmer address*	CPU address
SA6 (16 Kbytes)	7FFF _H	FFFF _H	SA11 (16 Kbytes)	7FFF _H	FFFF _H
SA5 (8 Kbytes)	7BFF _H	FFBFFF _H	SA10 (8 Kbytes)	7BFF _H	FFBFFF _H
SA4 (8 Kbytes)	79FF _H	FF9FFF _H	SA8 (8 Kbytes)	79FF _H	FF9FFF _H
SA3 (32 Kbytes)	77FF _H	FF7FFF _H	SA8 (32 Kbytes)	77FF _H	FF7FFF _H
SA2 (64 Kbytes)	6FFF _H	FEFFFF _H	SA7 (64 Kbytes)	6FFF _H	FEFFFF _H
SA1 (64 Kbytes)	5FFF _H	FDFFF _H	SA6 (64 Kbytes)	5FFF _H	FDFFF _H
SA0 (64 Kbytes)	4FFF _H	FCFFF _H	Unused	4FFF _H	FCFFF _H
	4000 _H	FC000 _H		3FFF _H	FBFFF _H
			SA5 (16 Kbytes)	3BFF _H	FBBFFF _H
			SA4 (8 Kbytes)	39FF _H	FB9FFF _H
			SA3 (8 Kbytes)	37FF _H	FB7FFF _H
			SA2 (32 Kbytes)	2FFF _H	FAFFF _H
			SA1 (64 Kbytes)	1FFF _H	F9FFF _H
			SA0 (64 Kbytes)	0FFF _H	F8FFF _H
			Unused	0000 _H	F8000 _H

*: The programmer address is equivalent to the CPU address when data is written to the flash memo using a parallel programmer. When a general programmer is used for writing/erasing, this address is used for writing/erasing.

24.3 Write/Erase Modes

The flash memory can be accessed in two different ways: Flash memory mode and alternative mode. Flash memory mode enables data to be directly written to or erased from the external pins. Alternative mode enables data to be written to or erased from the CPU via the internal bus. Use the mode external pins to select the mode.

■ Flash Memory Mode

The CPU stops when the mode pins are set to 111 while the reset signal is asserted. The flash memory interface circuit is connected directly to ports 0, 2, 3, and 4, enabling direct control from the external pins. This mode makes the MCU seem like a standard flash memory to the external pins, and write/erase can be performed using a flash memory programmer.

In flash memory mode, all operations supported by the flash memory automatic algorithm can be used.

■ Alternative Mode

The flash memory is located in the FC (F9) to FF banks in the CPU memory space, and like ordinary mask ROM, can be read-accessed and program-accessed from the CPU via the flash memory interface circuit.

Since writing/erasing the flash memory is performed by instructions from the CPU via the flash memory interface circuit, this mode allows rewriting even when the MCU is soldered on the target board.

Sector protect operations cannot be performed in these modes.

■ Flash Memory Control Signals

Table 24.3-1 "Flash Memory Control Signals" lists the flash memory control signals in flash memory mode.

There is almost a one-to-one correspondence between the flash memory control signals and the external pins of the MBM29LV200. The V_{ID} (12 V) pins required by the sector protect operations are MD0, MD1, and MD2 instead of A9, \overline{RESET} , and \overline{OE} for the MBM29LV200.

In flash memory mode, the external data bus signal width is limited to 8 bits, enabling only one-byte access. The DQ15 to DQ8 pins are not supported. The \overline{BYTE} pin should always be set to 0.

Table 24.3-1 Flash Memory Control Signals

MB90F594A/MB90F594G/MB90F591A/MB90F591G			MBM29LV200
Pin number	Normal 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
16	P36	\overline{BYTE}	\overline{BYTE}
17	P37	$\overline{RY/BY}$	$\overline{RY/BY}$
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	OE (V_{ID})
85 to 92	P00 to P07	DQ0 to DQ7	DQ0 to DQ7
77	\overline{RST}	\overline{RESET}	\overline{RESET}
Not supported			DQ8 to DQ15

24.4 Flash Memory Control Status Register (FMCS)

The flash memory control status register (FMCS), together with the flash memory interface circuit, is used to write data to and erase data from the flash memory.

■ Flash Memory Control Status Register (FMCS)

	7	6	5	4	3	2	1	0	← Bit No.
Address: 0000AE _H	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)	

○ Explanation of bits

[Bit 7] INTE (interrupt enable)

This bit generates an interrupt to the CPU when flash memory write/erase terminates.

An interrupt to the CPU is generated when the INTE and RDYINT bits are 1. No interrupt is generated when the INTE bit is 0.

- 0: Disables interrupts when write/erase terminates.
- 1: Enables interrupts when write/erase terminates.

[Bit 6] RDYINT (ready interrupt)

This bit indicates the operating state of the flash memory.

This bit is set to 1 when flash memory write/erase terminates. Data cannot be written to or erased from the flash memory while this bit is 0 after a flash memory write/erase. Flash memory write/erase is enabled when write/erase terminates and this bit is set to 1.

Writing 0 clears this bit to 0. Writing 1 is ignored. This bit is set to 1 at the termination timing of the flash memory automatic algorithm (see Section 24.5 "Starting the Flash Memory Automatic Algorithm"). When the read-modify-write (RMW) instruction is used, 1 is always read.

- 0: Write/erase is being executed.
- 1: Write/erase has terminated (interrupt request generated).

[Bit 5] WE (write enable)

This bit enables writing to the flash memory area.

When this bit is 1, writing after the command sequence (see Section 24.5 "Starting the Flash Memory Automatic Algorithm") is issued to the FC (F9) to FF bank writes to the flash memory area. When this bit is 0, the write/erase signal is not generated. This bit is used when the flash memory Write/Erase command is started.

If write/erase is not performed, it is recommended that this bit be set to 0 to prevent data from being mistakenly written to the flash memory.

- 0: Disables flash memory write/erase.
- 1: Enables flash memory write/erase.

[Bit 4] RDY (ready)

This bit enables flash memory write/erase.

Flash memory write/erase is disabled while this bit is 0. However, Suspend commands, such as the Read/Reset command and Sector Erase Suspend command, can be accepted even if this bit is 0.

- 0: Write/erase is being executed.
- 1: Write/erase has terminated (next data write/erase enabled).

[Bits 3 and 1] Reserved bits

These bits are reserved for testing. During regular use, they should always be set to 0.

[Bits 2 and 0] LPM1 and LPM0 (low power mode)

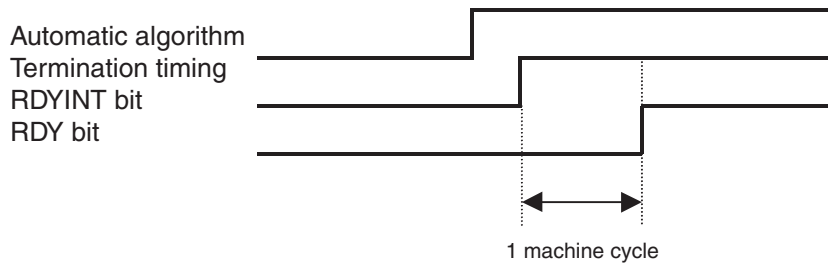
These bits control the current consumed by the flash memory when the flash memory is accessed. Since the access time to the flash memory from the CPU is largely dependent on this setting, select a setting value based on the operating frequency of the CPU.

- 01: Low power consumption mode (Operates at an internal operating frequency up to 4 MHz.)
- 10: Low power consumption mode (Operates at an internal operating frequency up to 8 MHz.)
- 11: Low power consumption mode (Operates at an internal operating frequency up to 10 MHz.)
- 00: Regular power consumption mode (Operates at an internal operating frequency up to 16 MHz.)

For the MB90F591A and the MB90F591G, these bits must be set to 00. For settings other than 00, there will be no access to the Flash Memory.

Note:

The RDYINT and RDY bits cannot be changed at the same time. Create a program so that decisions are made using one or the other of these bits.



24.5 Starting the Flash Memory Automatic Algorithm

Four types of commands are available for starting the flash memory automatic algorithm: Read/Reset, Write, and Chip Erase. Control of suspend and restart is enabled for sector erase.

■ Command Sequence Table

Table 24.5-1 "Command Sequence Table" lists the commands used for flash memory write/erase. All of the data written to the command register is in bytes, but use word access to write. The data of the high-order bytes at this time is ignored.

Table 24.5-1 Command Sequence Table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset (*1)	1	FxXXXX	XXF0	-	-	-	-	-	-	-	-	-	-
Read/Reset (*1)	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	-	-	-	-
Write program	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector Erase Suspend		Entering address FxXXXX data (xxB0H) suspends erasing during sector erase.											
Sector Erase Restart		Entering address FxXXXX data (xx30H) restarts erasing after erasing is suspended during sector erase.											
Auto-select	3	FxAAA	XXAA	Fx5554	XX55	FxAAAA	XX90	-	-	-	-	-	-

Note:

- The addresses Fx in the table mean FF, FE, FD, and FC for 2M-bit Flash Memory and FF, FE, FD, FB, FA and F9 for 3M-bit Flash Memory. Use these addresses as the access target bank values for operations.
- The addresses in the table are the values in the CPU memory map. All addresses and data are represented using hexadecimal notation. However, the letter X is an optional value.
- RA: Read address
- PA: Write address. Only even addresses can be specified.
- SA: Sector address. See Section 24.2 "Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory".
- RD: Read data
- PD: Write data. Only word data can be specified.

*1: Both of the two types of Read/Reset commands can reset the flash memory to read mode.

The Auto-select command shown in Table 24.5-1 "Command Sequence Table" is used to know the state of sector protection. When using the Auto-select command, set the address as follows.

Table 24.5-2 Address Setting at Auto-select

	AQ13 to AQ17 (,AQ18)	AQ7	AQ2	AQ1	AQ0	DQ7 to DQ0
Sector protection	Sector Address	L	H	L	L	CODE*

*: When the sector address is protected, the output is "01H".
When the sector address is not protected, the output is "00H".

24.6 Confirming the Automatic Algorithm Execution State

Because the write/erase flow of the flash memory is controlled using the automatic algorithm, the flash memory has hardware for posting its internal operating state and completion of operation. This automatic algorithm enables confirmation of the operating state of the built-in flash memory using the following hardware sequences.

■ Hardware Sequence Flags

The hardware sequence flags are configured from the five-bit output of DQ7, DQ6, DQ5, DQ3 and DQ2. The functions of these bits are those of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit exceeded flag (DQ5), sector erase timer flag (DQ3) and toggle bit-2 flag (DQ2). The hardware sequence flags can therefore be used to confirm that writing or chip sector erase has been completed or that erase code write is valid.

The hardware sequence flags can be accessed by read-accessing the addresses of the target sectors in the flash memory after setting of the command sequence (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm". Table 24.6-1 "Bit Assignments of Hardware Sequence Flags" lists the bit assignments of the hardware sequence flags.

Table 24.6-1 Bit Assignments of Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	DQ2	-	-

To determine whether automatic writing or chip sector erase is being executed, the hardware sequence flags can be checked or the status can be determined from the RDY bit of the flash memory control register (FMCS) that indicates whether writing has been completed. After writing/erasing has terminated, the state returns to the read/reset state. When creating a program, use one of the flags to confirm that automatic writing/erasing has terminated. Then, perform the next processing operation, such as data read. In addition, the hardware sequence flags can be used to confirm whether the second or subsequent sector erase code write is valid. The following sections describe each hardware sequence flag separately. Table 24.6-2 "Hardware Sequence Flag Functions" lists the functions of the hardware sequence flags.

Table 24.6-2 Hardware Sequence Flag Functions

State		DQ7	DQ6	DQ5	DQ3	DQ2
State change for normal operation	Write --> Write completed (write address specified)	$\overline{DQ7}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3	1 --> DATA:2
	Chip/sector erase --> Erase completed	0 --> 1	Toggle --> Stop	0 --> 1	1	Toggle --> Stop
	Sector erase wait --> Erase started	0	Toggle	0	0 --> 1	Toggle
	Erase --> Sector erase suspended (sector being erased)	0 --> 1	Toggle --> 1	0	1 --> 0	Toggle
	Sector erase suspend --> Erase restarted (sector being erased)	1 --> 0	1 --> Toggle	0	0 --> 1	Toggle
	Sector erase suspended (sector not being erased)	DATA:7	DATA:6	DATA:5	DATA:3	DATA:2
Abnormal operation	Write	$\overline{DQ7}$	Toggle	1	0	1
	Chip/sector erase	0	Toggle	1	1	*1

*1: If the DQ5 outputs "1" (exceed the timing limit), successive reads from a writing or erasing sector cause DQ2 to toggle. DQ2 does not toggle when the successive reads are executed from other sectors.

24.6.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) uses the data polling function to post that the automatic algorithm is being executed or has terminated

■ Data Polling Flag (DQ7)

Table 24.6-3 "Data Polling Flag State Transitions (State Change for Normal Operation)" and Table 24.6-4 "Data Polling Flag State Transitions (State Change for Abnormal Operation)" list the state transitions of the data polling flag.

Table 24.6-3 Data Polling Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ7	$\overline{DQ7}$ -->	0 --> 1	0	0 --> 1	1 --> 0	DATA:7

Table 24.6-4 Data Polling Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ7	$\overline{DQ7}$	0

○ Write

Read-access during execution of the automatic write algorithm causes the flash memory to output the opposite data of bit 7 last written, regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output bit 7 of the read value of the address specified by the address signal.

○ Chip/sector erase

For a sector erase, read-access during execution of the chip erase/sector erase algorithm causes the flash memory to output 0 from the sector currently being erased. For a chip erase, read-access causes the flash memory to output 0 regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output 1 in the same way.

○ Sector erase suspend

Read-access during a sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 7 (DATA: 7) of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased. Referencing this flag together with the toggle bit flag (DQ6) enables a decision to be made on whether the flash memory is in the erase suspended state and which sector is being erased.

Note:

When the automatic algorithm is being started, read-access to the specified address is ignored. Since termination of the data polling flag (DQ7) can be accepted for a data read and other bits output, data read after the automatic algorithm has terminated should be performed after read-access has confirmed that data polling has terminated.

24.6.2 Toggle Bit Flag (DQ6)

Like the data polling flag, the toggle bit flag (DQ6) uses the toggle bit function to post that the automatic algorithm is being executed or has terminated.

■ Toggle Bit Flag (DQ6)

Table 24.6-5 "Toggle Bit Flag State Transitions (State Change for Normal Operation)" and Table 24.6-6 "Toggle Bit Flag State Transitions (State Change for Abnormal Operation)" list the state transitions of the toggle bit flag.

Table 24.6-5 Toggle Bit Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ6	Toggle --> DATA:6	Toggle --> Stop	Toggle	Toggle --> 1	1 --> Toggle	DATA:6

Table 24.6-6 Toggle Bit Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ6	Toggle	Toggle

○ Write/chip sector erase

Continuous read-access during execution of the automatic write algorithm and chip/sector erase algorithm causes the flash memory to toggle the 1 or 0 state for every read cycle, regardless of the value at the address specified by the address signal. Continuous read-access at the end of the automatic write algorithm and chip/sector erase algorithm causes the flash memory to stop toggling bit 6 and output bit 6 (DATA: 6) of the read value of the address specified by the address signal.

○ Sector erase suspend

Read-access during a sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 6 (DATA: 6) of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

Note:

For a write, if the sector where data is to be written is rewrite-protected, the toggle bit terminates the toggle operation after approximately 2μs without any data being rewritten. For an erase, if all of the selected sectors are write-protected, the toggle bit performs toggling for approximately 100μs and then returns to the read/reset state without any data being rewritten.

24.6.3 Timing Limit Exceeded Flag (DQ5)

The timing limit exceeded flag (DQ5) is used to post that execution of the automatic algorithm has exceeded the time (internal pulse count) prescribed in the flash memory.

■ Timing Limit Exceeded Flag (DQ5)

Table 24.6-7 "Timing Limit Exceeded Flag State Transitions (State Change for Normal Operation)" and Table 24.6-8 "Timing Limit Exceeded Bit Flag State Transitions (State Change for Abnormal Operation)" list the state transitions of the timing limit exceeded flag.

Table 24.6-7 Timing Limit Exceeded Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ5	0 --> DATA:5	0 --> 1	0	0	0	DATA:5

Table 24.6-8 Timing Limit Exceeded Bit Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ5	1	1

○ Write/chip sector erase

Read-access after write or chip/sector erase automatic algorithm activation causes the flash memory to output 0 if the time is within the prescribed time (time required for write/erase) or to output 1 if the prescribed time has been exceeded. Because this is done regardless of whether the automatic algorithm is being executed or has terminated, it is possible to determine whether write/erase was successful or unsuccessful. That is, when this flag outputs 1, writing can be determined to have been unsuccessful if the automatic algorithm is still being executed by the data polling function or toggle bit function.

For example, writing 1 to a flash memory address where 0 has been written will cause the fail state to occur. In this case, the flash memory will lock and execution of the automatic algorithm will not terminate. As a result, valid data will not be output from the data polling flag (DQ7). In addition, the toggle bit flag (DQ6) will exceed the time limit without stopping the toggle operation and the timing limit exceeded flag (DQ5) will output 1. Note that this state indicates that the flash memory is not faulty, but has been used correctly. When this state occurs, execute the Reset command.

24.6.4 Sector Erase Timer Flag (DQ3)

The sector erase timer flag (DQ3) is used to post whether the automatic algorithm is being executed during the sector erase wait period after the Sector Erase command has been started.

■ Sector Erase Timer Flag (DQ3)

Table 24.6-9 "Sector Erase Timer Flag State Transitions (State Change for Normal Operation)" and Table 24.6-10 "Sector Erase Timer Flag State Transitions (State Change for Abnormal Operation)" list the state transitions of the sector erase timer flag.

Table 24.6-9 Sector Erase Timer Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ3	0 --> DATA:3	1	0 --> 1	1 --> 0	0 --> 1	DATA:3

Table 24.6-10 Sector Erase Timer Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ3	0	1

○ Sector erase

Read-access after the Sector Erase command has been started causes the flash memory to output 0 if the automatic algorithm is being executed during the sector erase wait period, regardless of the value at the address specified by the address signal of the sector that issued the command. The flash memory outputs 1 if the sector erase wait period has been exceeded.

If the data polling function or toggle bit function indicates that the erase algorithm is being executed, internally controlled erase has already started if this flag is 1. Continuous write of the sector erase codes or commands other than the Sector Erase Suspend command will be ignored until erase is terminated.

If this flag is 0, the flash memory will accept write of additional sector erase codes. To confirm this, it is recommended that the state of this flag be checked before continuing to write sector erase codes. If this flag is 1 after the second state check, it is possible that additional sector erase codes may not be accepted.

○ **Sector erase**

Read-access during execution of sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 3 (DATA: 3) of the read value of the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

24.6.5 Toggle Bit-2 Flag (DQ2)

The toggle bit-2 flag (DQ2) is a flag that uses the toggle bit function to indicate that the sector is in the erase-suspended state.

■ Toggle Bit-2 Flag (DQ2)

Table 24.6-11 "Toggle Bit-2 Flag State Transitions (State Change for Normal Operation)" and Table 24.6-12 "Toggle Bit-2 Flag State Transitions (State Change for Abnormal Operation)" list the state transitions of the toggle bit flag.

Table 24.6-11 Toggle Bit-2 Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ2	1 --> DATA:2	Toggle --> Stop	Toggle	Toggle	Toggle	DATA:2

Table 24.6-12 Toggle Bit-2 Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ2	1	*1

*1: If the DQ5 outputs "1" (exceed the timing limit), successive reads from a writing or erasing sector cause DQ2 to toggle. DQ2 does not toggle when the successive reads are executed from other sectors.

○ During a sector erase operation

If successive reads are executed during the execution of the chip sector erase algorithm, a flash memory toggles to output "1" and "0" to addresses alternately at every read access regardless of the location indicated by the addresses. If successive reads are executed after the chip sector erase algorithm is completed, the flash memory stops the toggle operation of the bit 2 and outputs the read value of the bit 2 (DATA: 2) to the location indicated by the address.

○ **While a sector erase operation is suspended**

If successive reads are executed while a sector erase operation is suspended, and if the address indicates the sector to be erased, the flash memory toggles to alternately output "1" and "0". If the address indicates the sector is not to be erased, the flash memory outputs the read value of the bit 2 (DATA: 2) to the location indicated by the address.

In the erase-suspend-program mode, successive reads from the non-erase suspended sector causes the flash memory to output "1".

Both DQ2 and DQ6 are used for detecting an erase-suspended sector (DQ2 toggles, but DQ6 does not).

DQ2 is also used for detecting an erasing sector. While erasing a sector, if a read access is executed from the erasing sector, DQ2 toggles.

Reference:

If all sectors selected for erasing are write-protected, the toggle bit-2 toggles for about 100 μ s, and then returns to the read/reset mode without writing the data.

24.7 Detailed Explanation of Writing to and Erasing Flash Memory

This section describes each operation procedure of flash memory Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, and Sector Erase Restart when a command that starts the automatic algorithm is issued.

■ Detailed Explanation of Flash Memory Write/Erase

The flash memory executes the automatic algorithm by issuing a command sequence (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") for a write cycle to the bus to perform Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, or Sector Erase Restart operations. Each bus write cycle must be performed continuously. In addition, whether the automatic algorithm has terminated can be determined using the data polling or other function. At normal termination, the flash memory is returned to the read/reset state.

Each operation of the flash memory is described in the following order:

- Setting the read/reset state
- Writing data
- Erasing all data (erasing chips)
- Erasing optional data (erasing sectors)
- Suspending sector erase
- Restarting sector erase

24.7.1 Setting The Read/Reset State

This section describes the procedure for issuing the Read/Reset command to set the flash memory to the read/reset state.

■ Setting the Flash Memory to the Read/Reset State

The flash memory can be set to the read/reset state by sending the Read/Reset command in the command sequence table (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Read/Reset command has two types of command sequences that execute the first and third bus operations. However, there are no essential differences between these command sequences.

The read/reset state is the initial state of the flash memory. When the power is turned on and when a command terminates normally, the flash memory is set to the read/reset state. In the read/reset state, other commands wait for input.

In the read/reset state, data is read by regular read-access. As with the mask ROM, program access from the CPU is enabled. The Read/Reset command is not required to read data by a regular read. The Read/Reset command is mainly used to initialize the automatic algorithm in such cases as when a command does not terminate normally.

24.7.2 Writing Data

This section describes the procedure for issuing the Write command to write data to the flash memory.

■ Writing Data to the Flash Memory

The data write automatic algorithm of the flash memory can be started by sending the Write command in the command sequence table (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory. When data write to the target address is completed in the fourth cycle, the automatic algorithm and automatic write are started.

○ Specifying addresses

Only even addresses can be specified as the write addresses specified in a write data cycle. Odd addresses cannot be written correctly. That is, writing to even addresses must be done in units of word data.

Writing can be done in any order of addresses or even if the sector boundary is exceeded. However, the Write command writes only data of one word for each execution.

○ Notes on writing data

Writing cannot return data 0 to data 1. When data 1 is written to data 0, the data polling algorithm (DQ7) or toggle operation (DQ6) does not terminate and the flash memory elements are determined to be faulty. If the time prescribed for writing is thus exceeded, the timing limit exceeded flag (DQ5) is determined to be an error. Otherwise, the data is viewed as if dummy data 1 had been written. However, when data is read in the read/reset state, the data remains 0. Data 0 can be set to data 1 only by erase operations.

All commands are ignored during execution of the automatic write algorithm. If a hardware reset is started during writing, the data of the written addresses will be unpredictable.

■ Writing to the Flash Memory

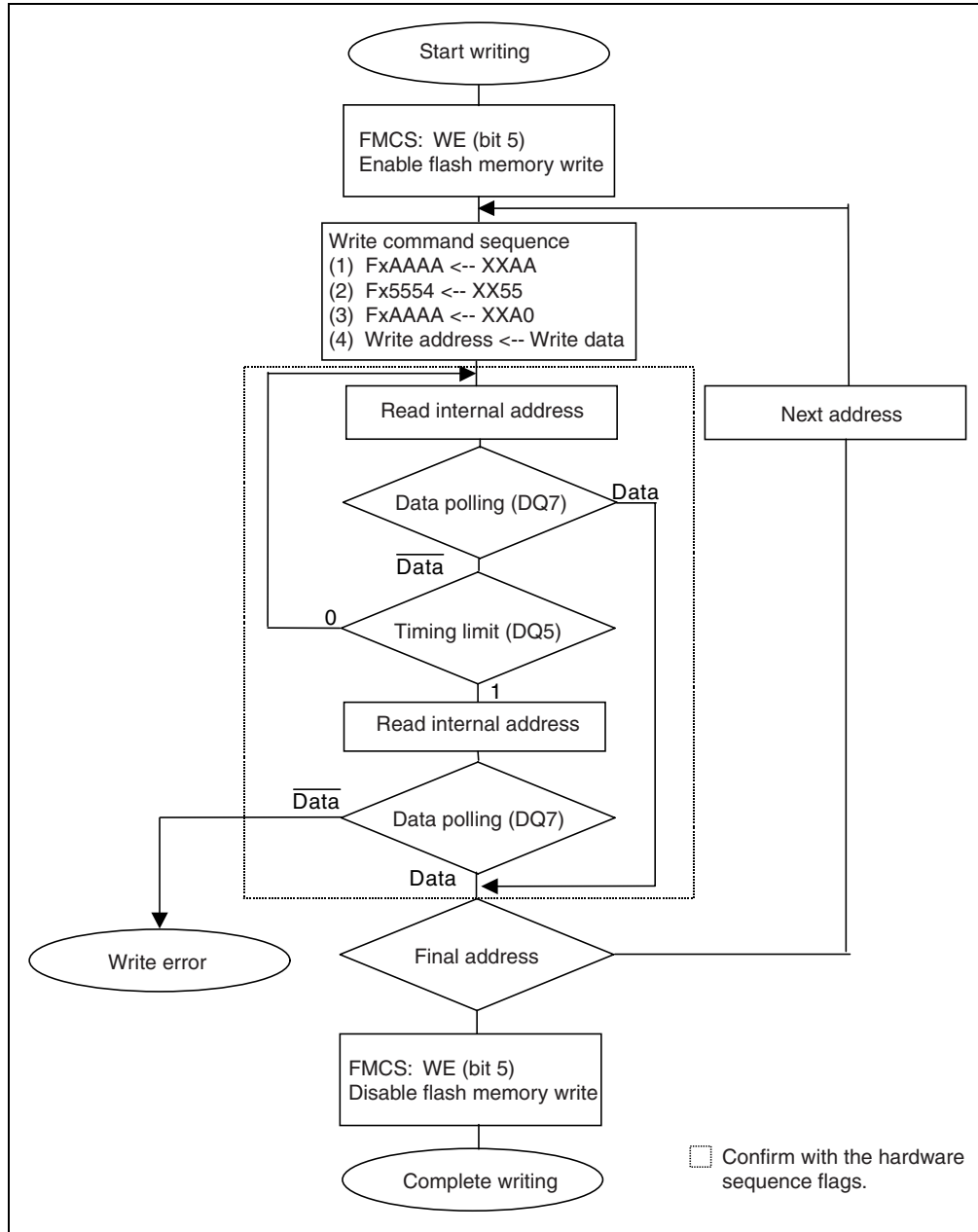
Figure 24.7-1 "Example of the Flash Memory Write Procedure" is an example of the procedure for writing to the flash memory. The hardware sequence flags (see Section 24.6 "Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Here, the data polling flag (DQ7) is used to confirm that writing has terminated.

The data read to check the flag is read from the address written to last.

The data polling flag (DQ7) changes at the same time that the timing limit exceeded flag (DQ5) changes. For example, even if the timing limit exceeded flag (DQ5) is 1, the data polling flag bit (DQ7) must be rechecked.

Also for the toggle bit flag (DQ6), the toggle operation stops at the same time that the timing limit exceeded flag bit (DQ5) changes to 1. The toggle bit flag (DQ6) must therefore be rechecked.

Figure 24.7-1 Example of the Flash Memory Write Procedure



24.7.3 Erasing All Data (Erasing Chips)

This section describes the procedure for issuing the Chip Erase command to erase all data in the flash memory.

■ Erasing all Data in the Flash Memory (Erasing Chips)

All data can be erased from the flash memory by sending the Chip Erase command in the command sequence table (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Chip Erase command is executed in six bus operations. When writing of the sixth cycle is completed, the chip erase operation is started. For chip erase, the user need not write to the flash memory before erasing. During execution of the automatic erase algorithm, the flash memory writes 0 for verification before all of the cells are erased automatically.

24.7.4 Erasing Optional Data (Erasing Sectors)

This section describes the procedure for issuing the Sector Erase command to erase optional data (erase sector) in the flash memory. Individual sectors can be erased. Multiple sectors can also be specified at one time.

■ Erasing Optional Data (Erasing Sectors) in the Flash Memory

Optional sectors in the flash memory can be erased by sending the Sector Erase command in the command sequence table (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

○ Specifying sectors

The Sector Erase command is executed in six bus operations. Sector erase wait of 50 μ s is started by writing the sector erase code (30h) to an accessible even-numbered address in the target sector in the sixth cycle. To erase multiple sectors, write the erase code (30h) to the addresses in the target sectors after the above processing operation.

○ Notes on specifying multiple sectors

Erase is started when the sector erase wait period of 50 μ s terminates after the final sector erase code has been written. That is, to erase multiple sectors at one time, an erase code (sixth cycle of the command sequence) must be written within 50 μ s of writing of the address of a sector and the address of the next sector must be written within 50 μ s of writing of the previous erase code. Otherwise, the address and erase code may not be accepted. The sector erase timer (hardware sequence flag DQ3) can be used to check whether writing of the subsequent sector erase code is valid. At this time, specify so that the address used for reading the sector erase timer indicates the sector to be erased.

■ Erasing Sectors in the Flash Memory

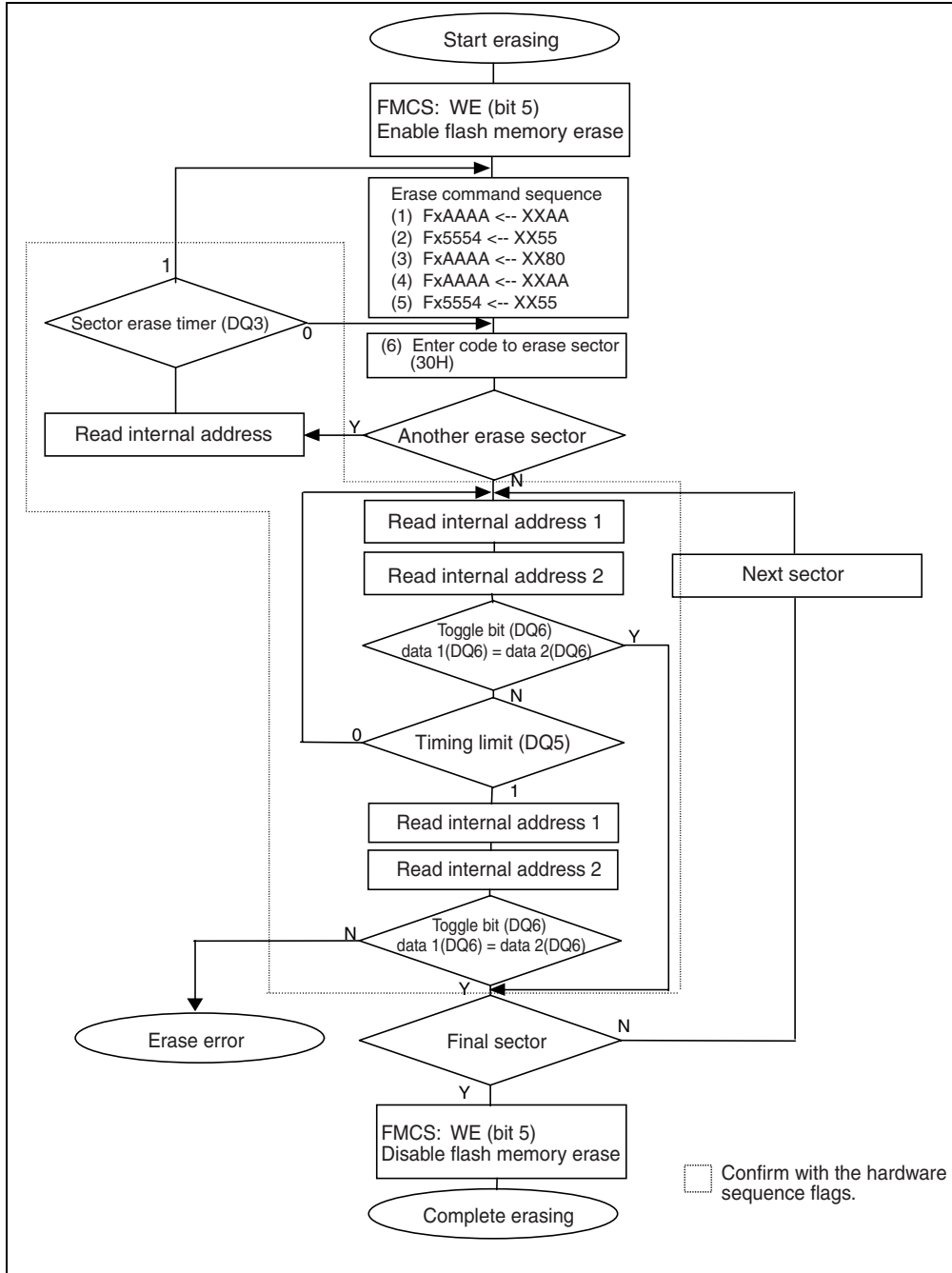
The hardware sequence flags (see Section 24.6 "Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 24.7-2 "Example of the Flash Memory Sector Erase Procedure" is an example of the procedure for erasing sectors in the flash memory. Here, the toggle bit flag (DQ6) is used to confirm that erasing has terminated.

The data that is read to check the flag is read from the sector to be erased.

The toggle bit flag (DQ6) stops the toggle operation at the same time that the timing limit exceeded flag (DQ5) is changed to 1. For example, even if the timing limit exceeded flag (DQ5) is 1, the toggle bit flag (DQ6) must be rechecked.

The data polling flag (DQ7) also changes at the same time that the timing limit exceeded flag bit (DQ5) changes. As a result, the data polling flag (DQ7) must be rechecked.

Figure 24.7-2 Example of the Flash Memory Sector Erase Procedure



24.7.5 Suspending Sector Erase

This section describes the procedure for issuing the Sector Erase Suspend command to suspend erasing of flash memory sectors. Data can be read from sectors that are not being erased.

■ Suspending Erasing of Flash Memory Sectors

Erasing of flash memory sectors can be suspended by sending the Sector Erase Suspend command in the command sequence table (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Sector Erase Suspend command suspends the sector erase operation being executed and enables data to be read from sectors that are not being erased. In this state, only reading is enabled; data cannot be written. This command is valid only during sector erase operations that include the erase wait time. The command will be ignored during chip erase or write operations.

This command is implemented by writing the erase suspend code (B0h). At this time, specify an optional address in the flash memory for the address. An Erase Suspend command issued again during erasing of sectors will be ignored.

Entering the Sector Erase Suspend command during the sector erase wait period will immediately terminate sector erase wait, cancel the erase operation, and set the erase stop state. Entering the Erase Suspend command during the erase operation after the sector erase wait period has terminated will set the erase suspend state after a maximum period of 15 μ s has elapsed.

24.7.6 Restarting Sector Erase

This section describes the procedure for issuing the Sector Erase Restart command to restart suspended erasing of flash memory sectors.

■ Restarting Erasing of Flash Memory Sectors

Suspended erasing of flash memory sectors can be restarted by sending the Sector Erase Restart command in the command sequence table (see Table 24.5-1 "Command Sequence Table" in Section 24.5 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Sector Erase Restart command is used to restart erasing of sectors from the sector erase suspend state set using the Sector Erase Suspend command. The Sector Erase Restart command is implemented by writing the erase restart code (30h). At this time, specify an optional address in the flash memory area for the address.

If a Sector Erase Restart command is issued during sector erase, the command will be ignored.

24.8 Notes on using 2M-bit Flash Memory

This section contains notes on using 2M-bit flash memory.

■ Notes on using flash memory

○ Input of a hardware reset (RST)

To input a hardware reset when the automatic algorithm has not been started and reading is in progress, a minimum low-level width of 500 ns must be maintained. In this case, a maximum of 500 ns is required until data can be read from the flash memory after a hardware reset has been activated.

Similarly, to input a hardware reset when the automatic algorithm has been activated and writing or erasing is in progress, a minimum low-level width of 50 ns must be maintained. In this case, 20 μ s are required until data can be read after the operation for initializing the flash memory has terminated.

A hardware reset during writing the data being written to be undefined. A hardware reset during erasing may make the sector being erased unusable.

○ Canceling of a software reset, watchdog timer reset, and hardware standby

When the flash memory is being written to or erased with CPU access and if reset conditions occur while the automatic algorithm is active, the CPU may run out of control. This occurs because these reset conditions cause the automatic algorithm to continue without initializing the flash memory unit, possibly preventing the flash memory unit from entering the read state when the CPU starts the sequence after the reset has been deasserted. These reset conditions must be disabled during writing to or erasing of the flash memory.

○ Program access to flash memory

When the automatic algorithm is operating, read access to the flash memory is disabled. With the memory access mode of the CPU set to internal ROM mode, writing or erasing must be started after the program area is switched to another area such as RAM. In this case, when sectors (SA6/SA11) containing interrupt vectors are erased, writing or erasing interrupt processing cannot be executed. For the same reason, all interrupt sources other than the flash memory are disabled while the automatic algorithm is operating.

Also, while the automatic algorithm is being executed, all interrupt sources except flash memory are disabled.

○ Hold function

When the CPU accepts a hold request, the Write signal \overline{WE} of the flash memory unit may be skewed, causing erroneous writing or erasing due to an erroneous write. When the acceptance of a hold request is enabled (HDE bit of EPCR set to 1), ensure that the WE bit of the control status register (FMCS) is 0.

○ Extended intelligent I/O service (EI²OS)

Because write and erase interrupts issued to the CPU from the flash memory interface circuit cannot be accepted by the EI²OS, they should not be used.

○ **Applying V_{ID}**

Applying V_{ID} required for the sector protect operation should always be started and terminated when the supply voltage is on.

24.9 Reset Vector Address in Flash Memory

The MB90F594A, MB90F594G, MB90F591A, and MB90F591G supports a hard-wired reset vector.

When the addresses FFFFDC_H to FFFFDF_H are accessed for reading data in internal vector mode, the values that have been determined by the hard-wired logic in advance are read. However, in flash memory mode, as mentioned in the previous chapter, all addresses can be accessed.

Consequently, it is meaningless to write data to these addresses. Especially when programming flash memory from the CPU (that is, not in flash memory mode), do not read these addresses for software polling. Otherwise, the flash memory returns a fixed reset vector instead of the hardware sequence flag value.

■ Reset vector address in flash memory

The following table shows the reset vector and mode data values determined in advance.

Reset vector	FFA000 _H
Mode data	00 _H

Note:

Because of the hard-wired reset vector, it is not necessary to specify the reset vector in the software. However it is recommended to specify the same vector and the same mode data in the program, this will prevent the Mask ROM device to behave differently from the Flash device when the same program is used.

24.10 Example of Programming 2M/3M-bit Flash Memory

This section presents a programming example of 2M/3M-bit flash memory.

■ Programming example of 2M/3M-bit flash memory

Flash Memory Sample Program

```

NAME    FLASHWE
TITLE   FLASHWE
;-----
;2M/3M-bit-FLASH test program
;
;1: Transmits the program (address: FFBC00H, sector: SA6) from FLASH to RAM
;   (address: 001500H).
;2: Executes the program on RAM.
;3: Writes the PDR1 value to FLASH (address: FI0000H, sector: SA1).
;4: Reads the written value (address: FD0000H, sector: SA1) and outputs it to PDR2.
;5: Erases the written sector (SA1).
;6: Checks and outputs erase data.
;Conditions
; - Number of bytes transmitted to RAM: 100H (256B)
; - Write/erase termination judgment
;   Judgment according to DQ5 (timing limit excess flag)
;   Judgment according to DQ6 (toggle bit flag)
;   Judgment according to RDY (FMCS)
; - Error handling
;   Hi output to P00 to P07
;   Reset command issuance
;-----
;
RESOUS  IOSEG  ABS=00          ;"RESOUS" I/O segment definition
        ORG    0000H
PDR0    RB     1
PDR1    RB     1
PDR2    RB     1
PDR3    RB     1
        ORG    0010H
DDR0    RB     1
DDR1    RB     1
DDR2    RB     1
DDR3    RB     1
        ORG    00A1H
CKSCR   RB     1
        ORG    00AEH
FMCS    RB     1
        ORG    006FH
ROMM    RB     1
RESOUS  ENDS
;
SSTA    SSEG
        RW     0127H
STA_T   RW     1
SSTA    ENDS
;
DATA    DSEG  ABS=0FFH      ;FLASH command address
        ORG    5554H
COMADR2 RW     1
        ORG    0AAAAH
COMADR1 RW     1
DATA    ENDS

```

```

;////////////////////////////////////
;Main program (FFA000H)
;////////////////////////////////////
CODE    CSEG
START:
;
;   Initialization
;
MOV     CKSCR,#0BAH    ;3-multiple setting
MOV     RP,#0
MOV     A,#!STA_T
MOV     SSB,A
MOVW    A,#STA_T
MOVW    SP,A
MOV     ROMM,#00H     ;Mirror OFF
MOV     PDR0,#00H     ;For error check
MOV     DDR0,#0FFH
MOV     PDR1,#00H     ;Port for data input
MOV     DDR1,#00H
MOV     PDR2,#00H     ;Port for data output
MOV     DDR2,#0FFH
;
;   Transfer of "FLASH write erase program (FFBC00H)" to RAM (1500H address)
;
MOVW    A,#1500H      ;Transfer destination RAM area
MOVW    A,#0BC00H     ;Transfer source address (program position)
MOVW    RW0,#100H     ;Number of bytes to be transferred
MOV     ADB,PCB      ;Transfer of 100H from FFBC00H to 001500H
CALLP   001500H      ;Jump to the address containing the transferred
;                          program
;
;   Data output
;
OUT     MOV     A,#0FDH
        MOV     ADB,A
        MOVW    RW2,#0000H
        MOVW    A,@RW2+00
        MOV     PDR2,A
END     JMP     *
CODE    ENDS
;////////////////////////////////////
;FLASH write erase program (SA6)
;////////////////////////////////////
RAMPRG  CSEG    ABS=0FFH
        ORG     0BC00H
;
;   Initialization
;
MOVW    RW0,#0500H    ;RW0:RAM space for input data acquisition
;                          From 00:0500
MOVW    RW2,#0000H    ;RW2:Flash memory write address
;                          From FD:0000
MOV     A,#00H        ;DTB modification
MOV     DTB,A         ;Bank specification for @RW0
MOV     A,#0FDH       ;ADB modification 1
MOV     ADB,A         ;Bank specification for write mode specification
;                          address
;
MOV     PDR3,#00H     ;Switch initialization
MOV     DDR3,#00H
;
WAIT1   BBC     PDR3:0,WAIT1 ;PDR3: 0(write start at high level)
;
;////////////////////////////////////
;Write (SA1)
;////////////////////////////////////
MOV     A,PDR1
MOVW    @RW0+00,A     ;PDR1 data allocation to RAM
MOV     FMCS,#20H     ;Write mode setting
MOVW    ADB:COMADR1,#00AAH ;Flash write command 1

```



```
;//////////////////////////////////////////
;Error
;//////////////////////////////////////////
ERROR    MOV     FMCS,#00H           ;FLASH mode release
          MOV     PDR0,#0FFH        ;Error handling check
          MOV     ADB:COMADR1,#0F0H ;Reset command (read is enabled)
          RETP     ;Return to the main program
RAMPRG   ENDS
;//////////////////////////////////////////
VECT     CSEG    ABS=0FFH
          ORG     0FFDCH
          DSL     START
          DB      00H
VECT     ENDS
;
```


CHAPTER 25 EXAMPLES OF MB90F594A/MB90F594G/ MB90F591A/MB90F591G SERIAL PROGRAMMING CONNECTION

**This chapter provides examples of F²MC-16LX MB90F594A/MB90F594G/MB90F591A/
MB90F591G serial programming connection.**

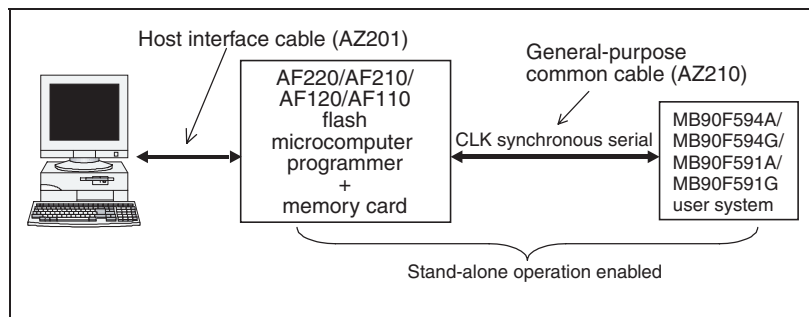
- 25.1 "Basic Configuration of F²MC-16LX MB90F594A/MB90F594G/
MB90F591A/MB90F591G Serial Programming Connection"
- 25.2 "Example of Serial Programming Connection (User Power Supply Used)"
- 25.3 "Example of Serial Programming Connection (Power Supplied from the
Programmer)"
- 25.4 "Example of Minimum Connection to the Flash Microcomputer
Programmer (User Power Supply Used)"
- 25.5 "Example of Minimum Connection to the Flash Microcomputer
Programmer (Power Supplied from the Programmer)"

25.1 Basic Configuration of MB90F594A/MB90F594G/ MB90F591A/MB90F591G Serial Programming Connection

The MB90F594A/MB90F594G/MB90F591A/MB90F591G supports flash ROM serial onboard programming (Fujitsu standard). This section describes the specifications.

■ Basic Configuration of MB90F594A/MB90F594G/MB90F591A/MB90F591G Serial Programming Connection

The AF220/AF210/AF120/AF110 flash microcomputer programmer from Yokogawa Digital Computer Corporation is used for Fujitsu standard serial onboard programming.



* The MB90F591G is under development.

Note:

Ask the company representative from Yokogawa Digital Computer Corporation for details about the functions and operations of the AF220/AF210/AF120/AF110 flash microcomputer programmer, general-purpose common cable for connection (AZ210), and connectors.

Table 25.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming

Pin	Function	Additional information
MD2, MD1 MD0	Mode pins	Controls programming mode from the flash microcomputer programmer.
X0, X1	Oscillation pins	In programming mode, the CPU internal operation clock signal is one multiple of the PLL clock signal frequency. Therefore, the oscillation clock frequency becomes the internal operation clock signal.
P00, P01	programming activation pins	-
$\overline{\text{RST}}$	Reset pin	-
SIN3	Serial data input pin	Serial input-output is used.
SOT3	Serial data output pin	
SCK3	Serial clock signal input pin	

Table 25.1-1 Pins Used for Fujitsu Standard Serial Onboard Programming (Continued)

Pin	Function	Additional information
C	C pin	This external capacitor pin is used to stabilize the power supply. Connect a ceramic capacitor of approximately 0.1 μ F to the outside.
V _{CC}	Power voltage supply pin	If the programming voltage (5 V \pm 10%) is supplied from the user system, the flash microcomputer programmer need not be connected. Connect so that the power supply of the user side is not short-circuited.
V _{SS}	GND pin	Common to the ground of the flash microcomputer programmer.
$\overline{\text{HST}}$	Hardware standby pin	Input high level during serial programming mode.

Even if the P00, SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.

Sections 25.2 "Example of Serial Programming Connection (User Power Supply Used)" to 25.5 "Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)" present examples the following four types of serial programming connection. See each Section as required.

- Serial programming connection (user power supply used)
- Serial programming connection (power supplied from the programmer)
- Minimum connection to the flash microcomputer programmer (user power supply used)
- Minimum connection to the flash microcomputer programmer (power supplied from the programmer)

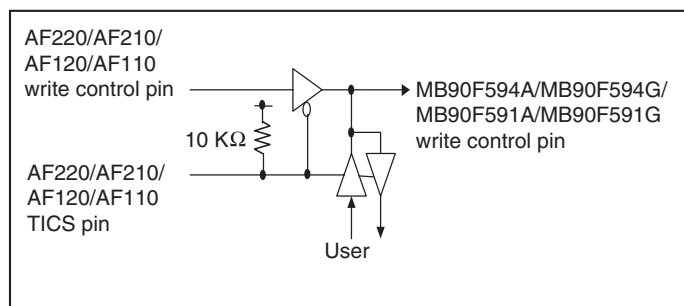


Table 25.1-2 System configuration of flash microcomputer programmers (manufactured by Yokogawa Digital Computer Corporation)

Model		Function
Main unit	AF220/AC4P	Ethernet interface built-in model and 100 to 220 V AC power adapter
	AF210/AC4P	Standard model and 100 to 220 V AC power adapter
	AF120/AC4P	Single-key Ethernet interface built-in model and 100 to 220 V AC power adapter
	AF110/AC4P	Single-key model and 100 to 220 V AC power adapter
AZ221		PC/AT RS232C cable for programmer
AZ210		Standard target probe (a) with a 1 m cable
FF201		Fujitsu F ² MC-16LX flash microcomputer control module
AZ290		Remote controller
/P2		2 MB PC card (optional) for flash memory sizes up to 128 KB
/P4		4 MB PC card (optional) for flash memory sizes up to 512 KB

Inquiries: Yokogawa Digital Computer Corporation
 Telephone number: (81)-42-333-6224

Note:

Although the AF200 flash microcomputer programmer is no longer manufactured, the programmer still can be used in combination with the FF201 control module.

Examples of serial programming connection are given in Sections 25.2 "Example of Serial Programming Connection (User Power Supply Used)" and 25.3 "Example of Serial Programming Connection (Power Supplied from the Programmer)".

■ Oscillating Clock Frequency and Serial Clock Input Frequency

The equation listed below can be used to calculate the serial clock frequencies that can be used for the MB90F594A, MB90F594G, MB90F591A, and MB90F591G. Set an appropriate serial clock input frequency in the flash microcomputer programmer according to the oscillating clock frequency in use.

Serial clock frequency that can be used = 0.125 x oscillating clock frequency

Table 25.1-3 Examples of serial clock frequencies that can be used

Oscillating clock frequency	Maximum serial clock frequency that can be used for microcomputers	Maximum serial clock frequency that can be used for the AF220, AF210, AF120, and AF110	Maximum serial clock frequency that can be used for the 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

25.2 Example of Serial Programming Connection (User Power Supply Used)

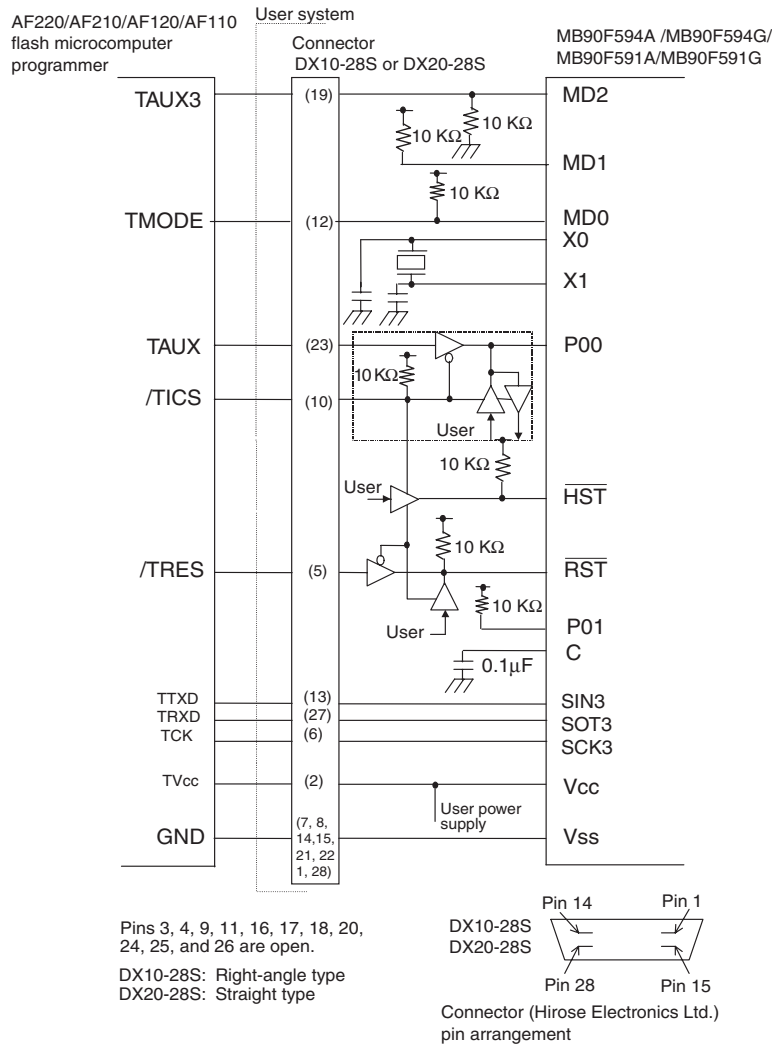
Figure 25.2-1 "Example of Serial Programming Connection for MB90F594A/MB90F594G/MB90F591A Internal Vector Modes (User Power Supply Used)" is an example of a serial programming connection for internal vector modes (single-chip mode) when the user power supply is used.

The value 1 and 0 are input to mode pins MD2 and MD0 from TAUX3 and TMODE of the AF220/AF210/AF120/AF110 programmer.

Serial reprogramming mode: MD2, MD1, MD0 = 110.

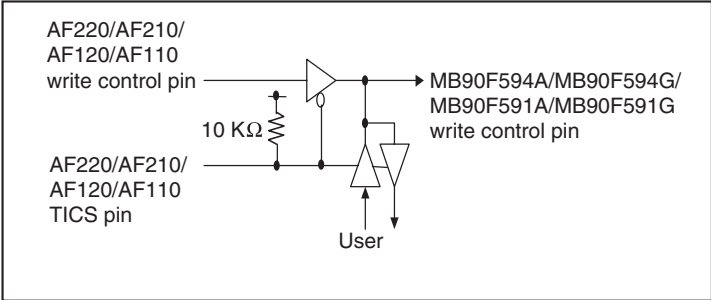
■ Example of Serial Programming Connection (User Power Supply Used)

Figure 25.2-1 Example of Serial Programming Connection for MB90F594A/MB90F594G/MB90F591A/MB90F591G Internal Vector Modes (User Power Supply Used)

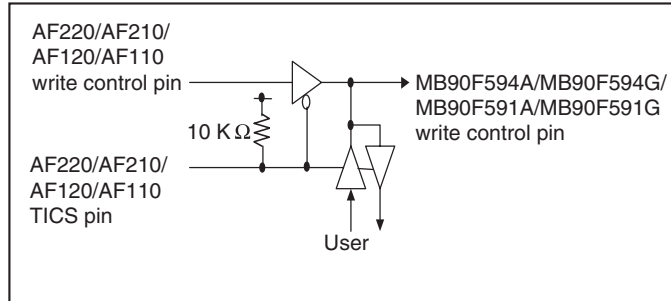


CHAPTER 25 EXAMPLES OF MB90F594A/MB90F594G/MB90F591A/MB90F591G SERIAL PROGRAMMING CONNECTION

- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required in the same way that it is for P00. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.
- Connect the AF220/AF210/AF120/AF110 while the user power is off.



- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required in the same way that it is for P00. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.
- Connect the AF220/AF210/AF120/AF110 while the user power is off.
- When the programming power is supplied from the AF220/AF210/AF120/AF110, be careful not to short-circuit the user power supply.



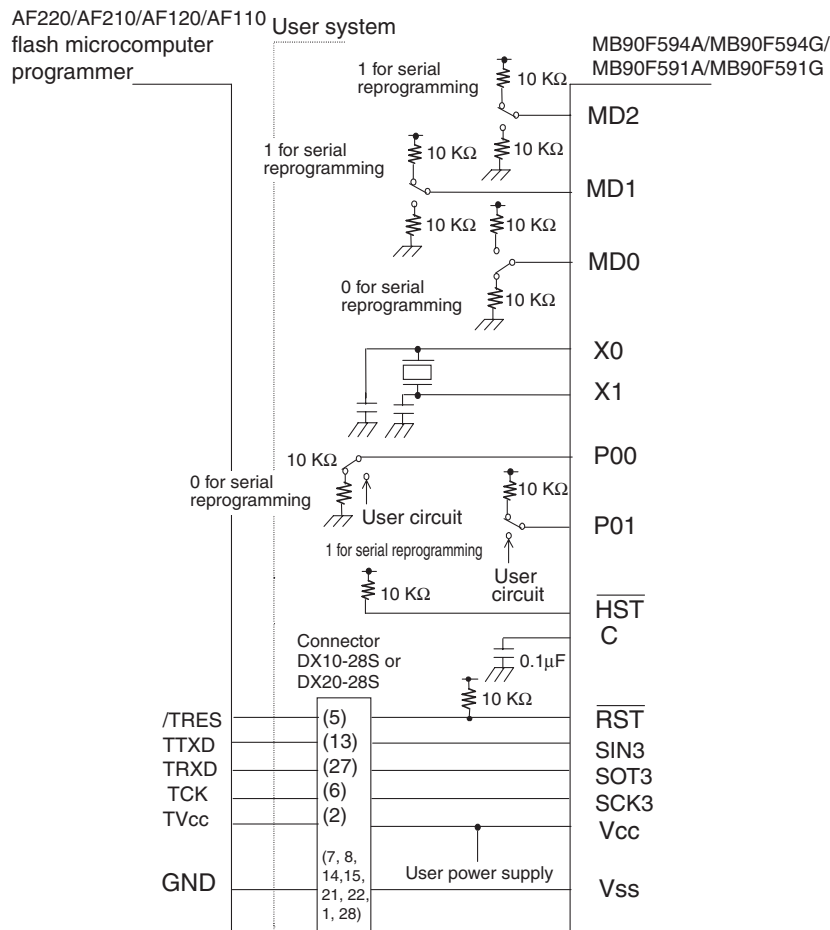
25.4 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)

Figure 25.4-1 "Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)" is an example of the minimum connection to the flash microcomputer programmer when the user power supply is used. Serial reprogramming mode: MD2, MD1, MD0 = 110.

■ Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)

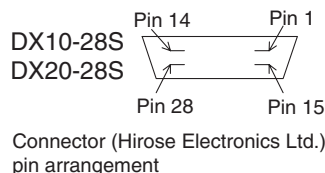
For a flash memory write, the MD2, MD1, MD0, and P00 pins and flash microcomputer programmer need not be connected if the pins are set as described below.

Figure 25.4-1 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)



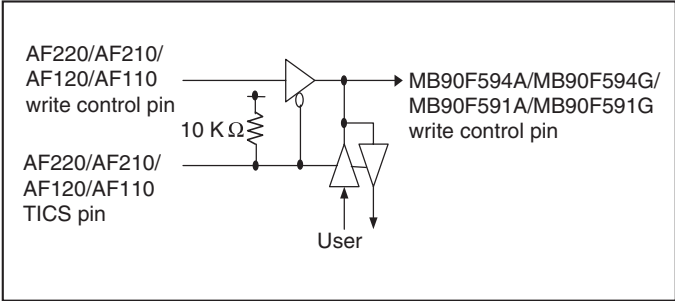
Pins 3, 4, 9, 10, 11, 12, 16, 17, 18, 19, 20, 23, 24, 25, and 26 are open.

DX10-28S: Right-angle type
DX20-28S: Straight type



CHAPTER 25 EXAMPLES OF MB90F594A/MB90F594G/MB90F591A/MB90F591G SERIAL PROGRAMMING CONNECTION

- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.
- Connect the AF220/AF210/AF120/AF110 while the user power is off.



25.5 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)

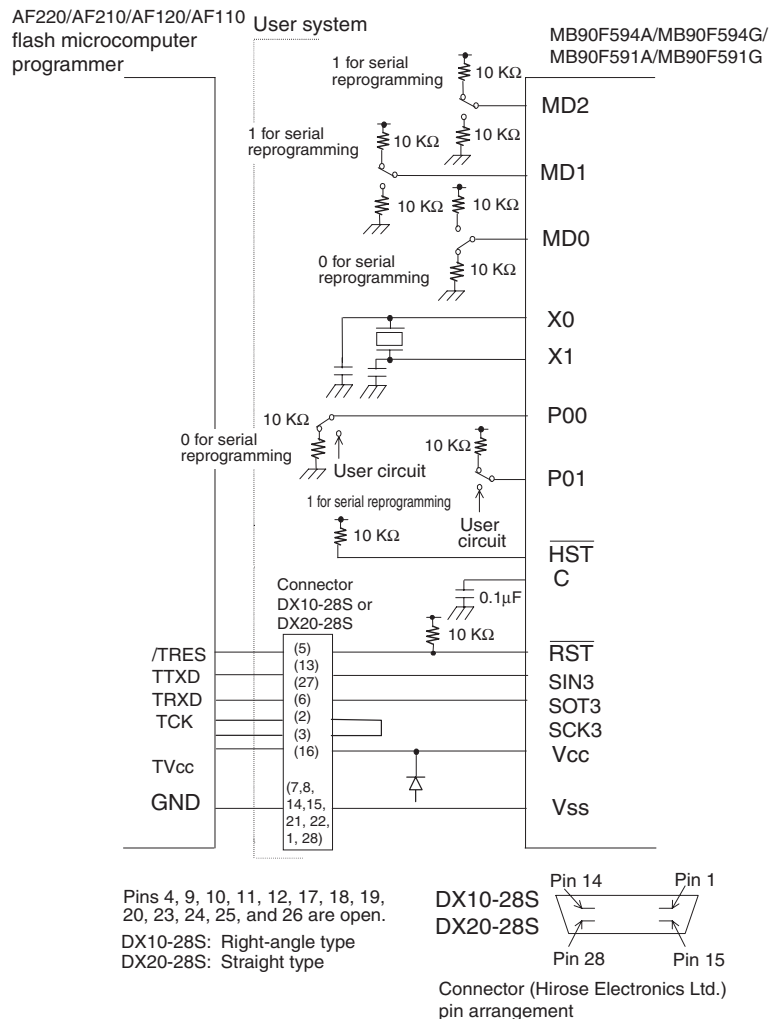
Figure 25.5-1 "Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)" is an example of the minimum connection to the flash microcomputer programmer when power is supplied from the Programmer.

Serial reprogramming mode: MD2, MD1, MD0 = 110.

■ Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)

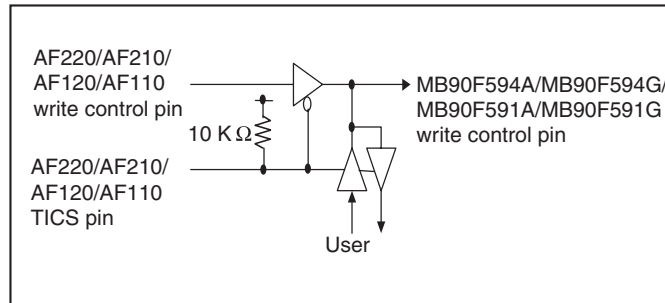
For a flash memory write, the MD2, MD1, MD0, and P00 pins and flash microcomputer programmer need not be connected if the pins are set as described below.

Figure 25.5-1 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)



CHAPTER 25 EXAMPLES OF MB90F594A/MB90F594G/MB90F591A/MB90F591G SERIAL PROGRAMMING CONNECTION

- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.
- Connect the AF220/AF210/AF120/AF110 while the user power is off.
- When the programming power is supplied from the AF220/AF210/AF120/AF110, be careful not to short-circuit the user power supply.



APPENDIX

The appendixes provide I/O maps, instructions, and other information.

APPENDIX A "I/O Maps"

APPENDIX B "Instructions"

APPENDIX C "Timing Diagrams in Flash Memory Mode"

APPENDIX D "List of MB90590 Interrupt Vectors"

APPENDIX A I/O Maps

Table A-1 "I/O Map" lists addresses to be assigned to the registers in the peripheral blocks.

■ I/O Maps

Table A-1 I/O Map

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	XXXXXXXX _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	---XXXX _B
08 _H	Port 8 Data Register	PDR8	R/W	Port 8	XXXXXXXX _B
09 _H	Port 9 Data Register	PDR9	R/W	Port 9	XXXXXXXX _B
0A to 0F _H	Use prohibited				
10 _H	Port 0 Direction Register	DDR0	R/W	Port 0	0000000 _B
11 _H	Port 1 Direction Register	DDR1	R/W	Port 1	0000000 _B
12 _H	Port 2 Direction Register	DDR2	R/W	Port 2	0000000 _B
13 _H	Port 3 Direction Register	DDR3	R/W	Port 3	0000000 _B
14 _H	Port 4 Direction Register	DDR4	R/W	Port 4	0000000 _B
15 _H	Port 5 Direction Register	DDR5	R/W	Port 5	0000000 _B
16 _H	Port 6 Direction Register	DDR6	R/W	Port 6	0000000 _B
17 _H	Port 7 Direction Register	DDR7	R/W	Port 7	0000000 _B
18 _H	Port 8 Direction Register	DDR8	R/W	Port 8	0000000 _B
19 _H	Port 9 Direction Register	DDR9	R/W	Port 9	--00000 _B
1A _H	Use prohibited				
1B _H	Analog Input Enable Register	ADER	R/W	Port 6, A/D	1111111 _B
1C to 1F _H	Use prohibited				

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
20 _H	Serial Mode Control Register 0	UMC0	R/W	UART0	00000100 _B
21 _H	Status Register 0	USR0	R/W		00010000 _B
22 _H	Input/Output Data Register 0	UIDR0/ UODR0	R/W		XXXXXXXX _B
23 _H	Rate/Data Register 0	URD0	R/W		0000000X _B
24 _H	Serial Mode Control Register 1	UMC1	R/W	UART1	00000100 _B
25 _H	Status Register 1	USR1	R/W		00010000 _B
26 _H	Input/Output Data Register 1	UIDR1/ UODR1	R/W		XXXXXXXX _B
27 _H	Rate/Data Register 1	URD1	R/W		0000000X _B
28 _H	Serial Mode Control Register 2	UMC2	R/W	UART2	00000100 _B
29 _H	Status Register 2	USR2	R/W		00010000 _B
2A _H	Input/Output Data Register 2	UIDR2/ UODR2	R/W		XXXXXXXX _B
2B _H	Rate/Data Register 2	URD2	R/W		0000000X _B
2C _H	Serial Mode Control Register	SMCS	R/W	Serial I/O	----0000 _B
2D _H	Serial Mode Control Register	SMCS	R/W		00000010 _B
2E _H	Serial Data Register	SDR	R/W		XXXXXXXX _B
2F _H	Edge Selector Register	SES	R/W		-----0 _B
30 _H	External Interrupt Enable Register	ENIR	R/W	External interrupt	00000000 _B
31 _H	External Interrupt Request Register	EIRR	R/W		XXXXXXXX _B
32 _H	External Interrupt Level Register	ELVR	R/W		00000000 _B
33 _H	External Interrupt Level Register	ELVR	R/W		00000000 _B
34 _H	A/D Control Status Register 0	ADCS0	R/W	A/D converter	00000000 _B
35 _H	A/D Control Status Register 1	ADCS1	R/W		00000000 _B
36 _H	A/D Data Register 0	ADCR0	R		XXXXXXXX _B
37 _H	A/D Data Register 1	ADCR1	R/W		000010XX _B
38 _H	PPG0 Operation Mode Control Register	PPGC0	R/W	16-bit programmable pulse generator 0/1	0-000--1 _B
39 _H	PPG1 Operation Mode Control Register	PPGC1	R/W		0-000001 _B
3A _H	PPG0, 1 Output Pin Control Register	PPC01	R/W		000000-- _B
3B _H	Use prohibited				

APPENDIX

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
3C _H	PPG2 Operation Mode Control Register	PPGC2	R/W	16-bit programmable pulse generator 2/3	0-000--1 _B
3D _H	PPG3 Operation Mode Control Register	PPGC3	R/W		0-000001 _B
3E _H	PPG2, 3 Output Pin Control Register	PPG23	R/W		000000-- _B
3F _H	Use prohibited				
40 _H	PPG4 Operation Mode Control Register	PPGC4	R/W	16-bit programmable pulse generator 4/5	0-000--1 _B
41 _H	PPG5 Operation Mode Control Register	PPGC5	R/W		0-000001 _B
42 _H	PPG4, 5 Output Pin Control Register	PPG45	R/W		000000-- _B
43 _H	Use prohibited				
44 _H	PPG6 Operation Mode Control Register	PPGC6	R/W	16-bit programmable pulse generator 6/7	0-000--1 _B
45 _H	PPG7 Operation Mode Control Register	PPGC7	R/W		0-000001 _B
46 _H	PPG6, 7 Output Pin Control Register	PPG67	R/W		000000-- _B
47 _H	Use prohibited				
48 _H	PPG8 Operation Mode Control Register	PPGC8	R/W	16-bit programmable pulse generator 8/9	0-000--1 _B
49 _H	PPG9 Operation Mode Control Register	PPGC9	R/W		0-000001 _B
4A _H	PPG8, 9 Output Pin Control Register	PPG89	R/W		000000-- _B
4B _H	Use prohibited				
4C _H	PPGA Operation Mode Control Register	PPGCA	R/W	16-bit programmable pulse generator A/B	0-000--1 _B
4D _H	PPGB Operation Mode Control Register	PPGCB	R/W		0-000001 _B
4E _H	PPGA, B Output Pin Control Register	PPGAB	R/W		00000000 _B
4F _H	Use prohibited				
50 _H	Timer Control Status Register 0	TMCSR0	R/W	16-bit reload timer 0	00000000 _B
51 _H	Timer Control Status Register 0	TMCSR0	R/W		----0000 _B

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
52 _H	Timer Control Status Register 1	TMCSR1	R/W	16-bit reload timer 1	00000000 _B
53 _H	Timer Control Status Register 1	TMCSR1	R/W		----0000 _B
54 _H	Input Capture Control Status Register 0/1	ICS01	R/W	Input capture 0/1	00000000 _B
55 _H	Input Capture Control Status Register 2/3	ICS23	R/W	Input capture 2/3	00000000 _B
56 _H	Input Capture Control Status Register 4/5	ICS45	R/W	Input capture 4/5	00000000 _B
57 _H	Use prohibited				
58 _H	Output Compare Control Status Register 0	OCS0	R/W	Output compare 0/1	0000--00 _B
59 _H	Output Compare Control Status Register 1	OCS1	R/W		---00000 _B
5A _H	Output Compare Control Status Register 2	OCS2	R/W	Output compare 2/3	0000--00 _B
5B _H	Output Compare Control Status Register 3	OCS3	R/W		---00000 _B
5C _H	Output Compare Control Status Register 4	OCS4	R/W	Output compare 4/5	0000--00 _B
5D _H	Output Compare Control Status Register 5	OCS5	R/W		---00000 _B
5E _H	Sound Control Register	SGCR	R/W	Sound generator	00000000 _B
5F _H	Sound Control Register	SGCR	R/W		0-----0 _B
60 _H	Timer Control Register	WTCR	R/W	Watch timer	000--000 _B
61 _H	Timer Control Register	WTCR	R/W		00000000 _B
62 _H	PWM Control Register 0	PWC0	R/W	Stepping motor controller 0	00000--0 _B
63 _H	Use prohibited				
64 _H	PWM Control Register 1	PWC1	R/W	Stepping motor controller 1	00000--0 _B
65 _H	Use prohibited				
66 _H	PWM Control Register 2	PWC2	R/W	Stepping motor controller 2	00000--0 _B
67 _H	Use prohibited				

APPENDIX

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
68 _H	PWM Control Register 3	PWC3	R/W	Stepping motor controller 3	00000--0 _B
69 _H	Use prohibited				
6A to 6C _H	Use prohibited				
6D _H	Serial I/O Prescaler Register	CDCR	R/W	Prescaler (Serial I/O)	0---1111 _B
6E _H	Timer Control Status Register	TCCS	R/W	I/O timer	00000000 _B
6F _H	ROM Mirror Function Select Register	ROMM	W	ROM mirror	XXXXXXXX1 _B
70 to 8F _H	Reserved for CAN interface 0/1. See the "CAN Controller Hardware Manual".				
90 to 9D _H	Use prohibited				
9E _H	Program Address Detection Control Status Register	PACSR	R/W	Address Match Detection Function	00000000 _B
9F _H	Delayed Interrupt/Release Register	DIRR	R/W	Delayed interrupt	-----0 _B
A0 _H	Low-power Mode Control Register	LPMCR	R/W	Low-power control circuit	00011000 _B
A1 _H	Clock Selection Register	CKSCR	R/W		11111100 _B
A2 to A7 _H	Use prohibited				
A8 _H	Watch-dog Timer Control Register	WDTC	R/W	Watch-dog timer	XXXXX111 _B
A9 _H	Timebase Timer Control Register	TBTC	R/W	Timebase timer	1--00100 _B
AA to AD _H	Use prohibited				
AE _H	Flash Memory Control Status Register (only for MB90F594A/MB90F594G/MB90591A/MB90F591G. Use prohibited for other controllers.)	FMCS	R/W	Flash memory	000X0000 _B
AF _H	Use prohibited				

Table A-1 I/O Map (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
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 to FF _H	Use prohibited				

Table A-2 I/O Map (19XX Address)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1900 _H	Reload Register L	PRLLO	R/W	16-bit programmable pulse generator 0/1	XXXXXXXX _B
1901 _H	Reload Register H	PRLH0	R/W		XXXXXXXX _B
1902 _H	Reload Register L	PRLLO	R/W		XXXXXXXX _B
1903 _H	Reload Register H	PRLH1	R/W		XXXXXXXX _B
1904 _H	Reload Register L	PRLLO	R/W	16-bit programmable pulse generator 2/3	XXXXXXXX _B
1905 _H	Reload Register H	PRLH2	R/W		XXXXXXXX _B
1906 _H	Reload Register L	PRLLO	R/W		XXXXXXXX _B
1907 _H	Reload Register H	PRLH3	R/W		XXXXXXXX _B

APPENDIX

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1908 _H	Reload Register L	PRL4	R/W	16-bit programmable pulse generator 4/5	XXXXXXXX _B
1909 _H	Reload Register H	PRLH4	R/W		XXXXXXXX _B
190A _H	Reload Register L	PRL5	R/W		XXXXXXXX _B
190B _H	Reload Register H	PRLH5	R/W		XXXXXXXX _B
190C _H	Reload Register L	PRL6	R/W	16-bit programmable pulse generator 6/7	XXXXXXXX _B
190D _H	Reload Register H	PRLH6	R/W		XXXXXXXX _B
190E _H	Reload Register L	PRL7	R/W		XXXXXXXX _B
190F _H	Reload Register H	PRLH7	R/W		XXXXXXXX _B
1910 _H	Reload Register L	PRL8	R/W	16-bit programmable pulse generator 8/9	XXXXXXXX _B
1911 _H	Reload Register H	PRLH8	R/W		XXXXXXXX _B
1912 _H	Reload Register L	PRL9	R/W		XXXXXXXX _B
1913 _H	Reload Register H	PRLH9	R/W		XXXXXXXX _B
1914 _H	Reload Register L	PRLA	R/W	16-bit programmable pulse generator A/B	XXXXXXXX _B
1915 _H	Reload Register H	PRLHA	R/W		XXXXXXXX _B
1916 _H	Reload Register L	PRLB	R/W		XXXXXXXX _B
1917 _H	Reload Register H	PRLHB	R/W		XXXXXXXX _B
1918 to 191F _H	Use prohibited				
1920 _H	Input Capture Register 0	IPCP0	R	Input capture 0/1	XXXXXXXX _B
1921 _H	Input Capture Register 0	IPCP0	R		XXXXXXXX _B
1922 _H	Input Capture Register 1	IPCP1	R		XXXXXXXX _B
1923 _H	Input Capture Register 1	IPCP1	R		XXXXXXXX _B
1924 _H	Input Capture Register 2	IPCP2	R	Input capture 2/3	XXXXXXXX _B
1925 _H	Input Capture Register 2	IPCP2	R		XXXXXXXX _B
1926 _H	Input Capture Register 3	IPCP3	R		XXXXXXXX _B
1927 _H	Input Capture Register 3	IPCP3	R		XXXXXXXX _B
1928 _H	Input Capture Register 4	IPCP4	R	Input capture 4/5	XXXXXXXX _B
1929 _H	Input Capture Register 4	IPCP4	R		XXXXXXXX _B
192A _H	Input Capture Register 5	IPCP5	R		XXXXXXXX _B
192B _H	Input Capture Register 5	IPCP5	R		XXXXXXXX _B
192D to 192F _H	Use prohibited				

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1930 _H	Output Compare Register 0	OCCP0	R/W	Output compare 0/1	XXXXXXXX _B
1931 _H	Output Compare Register 0	OCCP0	R/W		XXXXXXXX _B
1932 _H	Output Compare Register 1	OCCP1	R/W		XXXXXXXX _B
1933 _H	Output Compare Register 1	OCCP1	R/W		XXXXXXXX _B
1934 _H	Output Compare Register 2	OCCP2	R/W	Output compare 2/3	XXXXXXXX _B
1935 _H	Output Compare Register 2	OCCP2	R/W		XXXXXXXX _B
1936 _H	Output Compare Register 3	OCCP3	R/W		XXXXXXXX _B
1937 _H	Output Compare Register 3	OCCP3	R/W		XXXXXXXX _B
1938 _H	Output Compare Register 4	OCCP4	R/W	Output compare 4/5	XXXXXXXX _B
1939 _H	Output Compare Register 4	OCCP4	R/W		XXXXXXXX _B
193A _H	Output Compare Register 5	OCCP5	R/W		XXXXXXXX _B
193B _H	Output Compare Register 5	OCCP5	R/W		XXXXXXXX _B
193D to 193F _H	Use prohibited				
1940 _H	Timer Register 0/reload Register 0	TMR0/ TMRLR0	R/W	16-bit reload timer 0	XXXXXXXX _B
1941 _H	Timer Register 0/Reload Register 0	TMR0/ TMRLR0	R/W		XXXXXXXX _B
1942 _H	Timer Register 1/Reload Register 1	TMR1/ TMRLR1	R/W	16-bit reload timer 1	XXXXXXXX _B
1943 _H	Timer Register 1/Reload Register 1	TMR1/ TMRLR1	R/W		XXXXXXXX _B
1944 _H	Timer Data Register	TCDT	R/W	I/O timer	0000000 _B
1945 _H	Timer Data Register	TCDT	R/W		0000000 _B
1946 _H	Frequency Data Register	SGFR	R/W	Sound generator	XXXXXXXX _B
1947 _H	Amplitude Data Register	SGAR	R/W		XXXXXXXX _B
1948 _H	Decrement Grade Register	SGDR	R/W		XXXXXXXX _B
1949 _H	Tone Count Register	SGTR	R/W		XXXXXXXX _B
194A _H	Sub-second Data Register	WTBR	R/W	Watch timer	XXXXXXXX _B
194B _H	Sub-second Data Register	WTBR	R/W		XXXXXXXX _B
194C _H	Sub-second Data Register	WTBR	R/W		---XXXX _B
194D _H	Second Data Register	WTSR	R/W		--00000 _B
194E _H	Minute Data Register	WTMR	R/W		--00000 _B
194F _H	Hour Data Register	WTHR	R/W		---00000 _B

APPENDIX

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1950 _H	PWM1 Compare Register 0	PWC10	R/W	Stepping motor controller 0	XXXXXXXX _B
1951 _H	PWM2 Compare Register 0	PWC20	R/W		XXXXXXXX _B
1952 _H	PWM1 Select Register 0	PWS10	R/W		--000000 _B
1953 _H	PWM2 Select Register 0	PWS20	R/W		-0000000 _B
1954 _H	PWM1 Compare 1	PWC11	R/W	Stepping motor controller 1	XXXXXXXX _B
1955 _H	PWM2 Compare 1	PWC21	R/W		XXXXXXXX _B
1956 _H	PWM1 Select Register 1	PWS11	R/W		--000000 _B
1957 _H	PWM2 Select Register 1	PWS21	R/W		-0000000 _B
1958 _H	PWM1 Compare Register 2	PWC12	R/W	Stepping motor controller 2	XXXXXXXX _B
1959 _H	PWM2 Compare Register 2	PWC22	R/W		XXXXXXXX _B
195A _H	PWM1 Select Register 2	PWS12	R/W		--000000 _B
195B _H	PWM2 Select Register 2	PWS22	R/W		-0000000 _B
195C _H	PWM1 Compare Register 3	PWC13	R/W	Stepping motor controller 3	XXXXXXXX _B
195D _H	PWM2 Compare Register 3	PWC23	R/W		XXXXXXXX _B
195E _H	PWM1 Select Register 3	PWS13	R/W		--000000 _B
195F _H	PWM2 Select Register 3	PWS23	R/W		-0000000 _B
1960 to 19FF _H	Used prohibited				
1A00 to 1AFF _H	Reserved for CAN interface 0. See the "CAN Controller Hardware Manual".				
1B00 to 1BFF _H	Reserved for CAN interface 1. See the "CAN Controller Hardware Manual".				
1C00 to 1CFF _H	Reserved for CAN interface 0. See the "CAN Controller Hardware Manual".				
1D00 to 1DFF _H	Reserved for CAN interface 1. See the "CAN Controller Hardware Manual".				
1E00 to 1EFF _H	Use prohibited				

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1EF0 _H	Program Address Detection Register 0 (low-order)	PADR0	R/W	Address match detection function	XXXXXXXX _B
1EF1 _H	Program Address Detection Register 0 (middle-order)				XXXXXXXX _B
1EF2 _H	Program Address Detection Register 0 (high-order)				XXXXXXXX _B
1EF3 _H	Program Address Detection Register 1 (low-order)	PADR1	R/W		XXXXXXXX _B
1EF4 _H	Program Address Detection Register 1 (middle-order)				XXXXXXXX _B
1EF5 _H	Program Address Detection Register 1 (high-order)				XXXXXXXX _B
1EF6 to 1FFF _H	Use prohibited				

- Initial value "?" indicates an unused bit, and "X" indicates an undefined value.
- The addresses between 0000_H and 00FF_H, which are not listed, have been reserved for the main functions of the MCU. The result of read access to these reserved addresses is "X". Write access to these addresses is not allowed.

○ **Explanation of write and read**

R/W: Both read and write enabled

R: Only read enabled

W: Only write enabled

○ **Explanation of initial values**

0: The initial value of this bit is "0".

1: The initial value of this bit is "1".

X: The initial value of this bit is undefined.

-: This bit is not used, and the initial value is undefined.

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 "Effective address field" 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 Register indirect with index PC indirect with 16-bit displacement Direct address	DTB
1D	@RW1+RW7				DTB
1E	@PC+disp16				PCB
1F	addr16				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 border="1"><tr><td>2</td><td>2</td><td>3</td><td>3</td><td>4</td><td>4</td><td>5</td><td>5</td></tr></table>	2	2	3	3	4	4	5	5
2	2	3	3	4	4	5	5			
After execution	A	<table border="1"><tr><td>4</td><td>4</td><td>5</td><td>5</td><td>1</td><td>2</td><td>1</td><td>2</td></tr></table> (Some instructions transfer AL to AH.)	4	4	5	5	1	2	1	2
4	4	5	5	1	2	1	2			

○ Register direct addressing

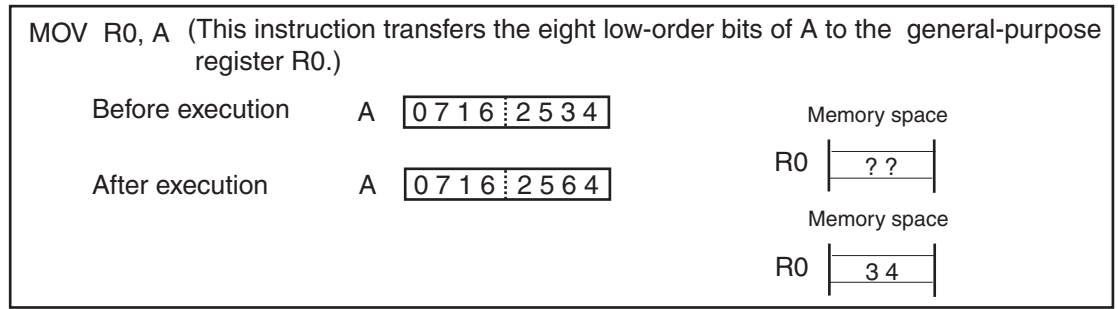
Specify a register explicitly as an operand. Table B.3-1 "Direct addressing registers" lists the registers that can be specified. Figure B.3-2 "Example of register direct addressing" shows an example of register direct addressing.

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 *1
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*1: 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.

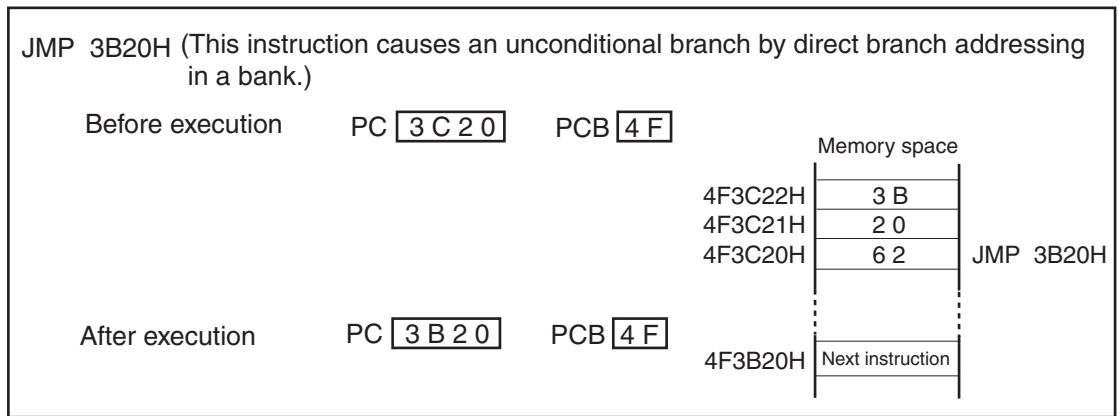
Figure B.3-2 Example of register direct addressing



○ **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. Bits 23 to 16 of the address are specified by the program bank register (PCB).

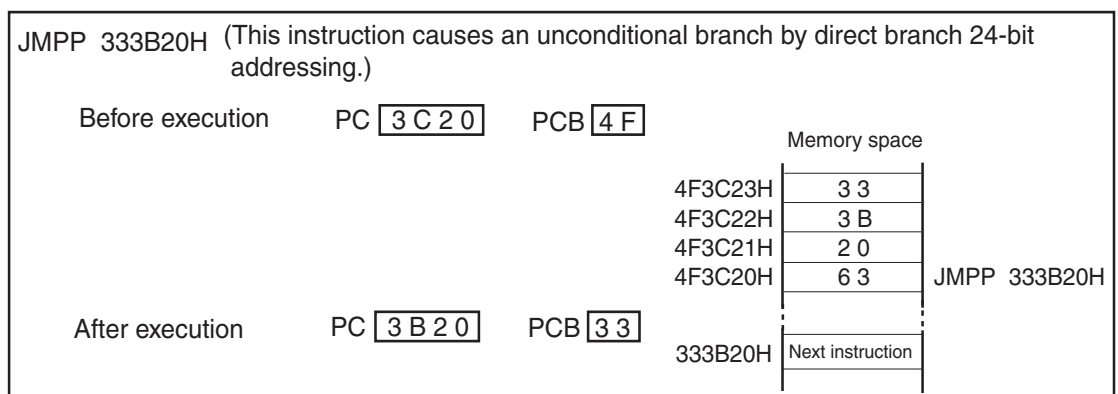
Figure B.3-3 Example of direct branch addressing (addr16)



○ **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.

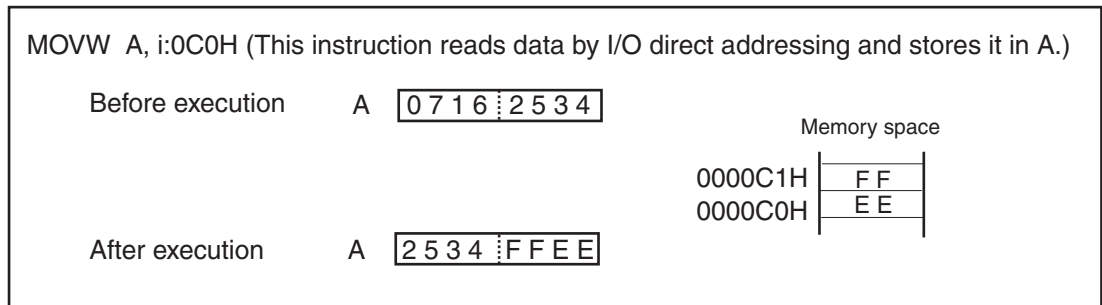
Figure B.3-4 Example of direct branch addressing (addr24)



○ 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 000000H to 0000FFH 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.

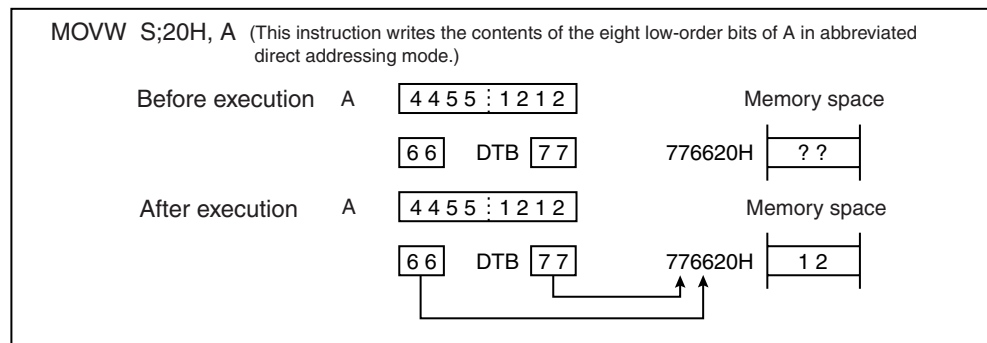
Figure B.3-5 Example of I/O direct addressing (io)



○ 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).

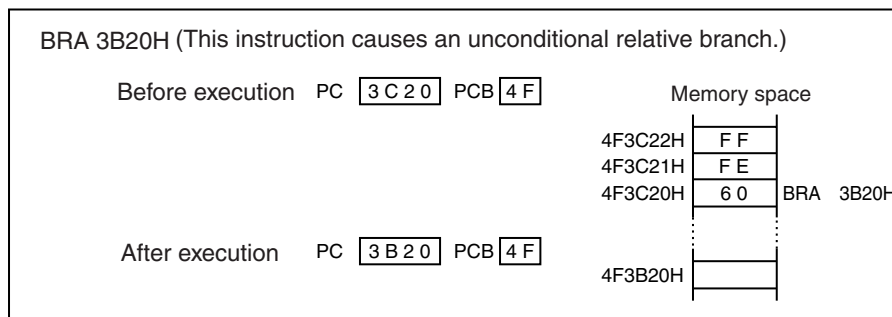
Figure B.3-6 Example of abbreviated direct addressing (dir)



○ 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.

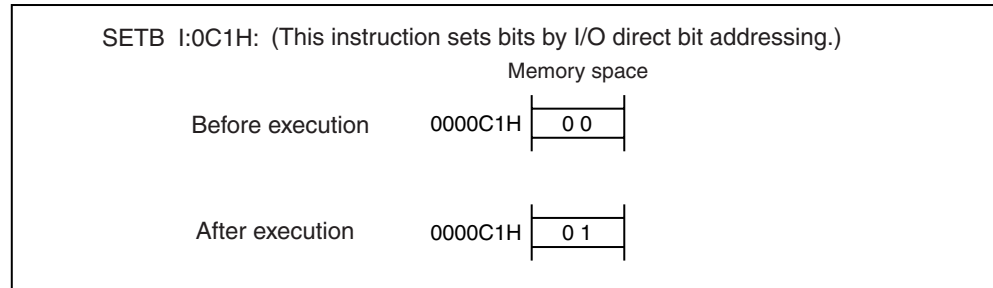
Figure B.3-7 Example of direct addressing (addr16)



○ **I/O direct bit addressing (io:bp)**

Specify bits in physical addresses 000000H to 0000FFH 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).

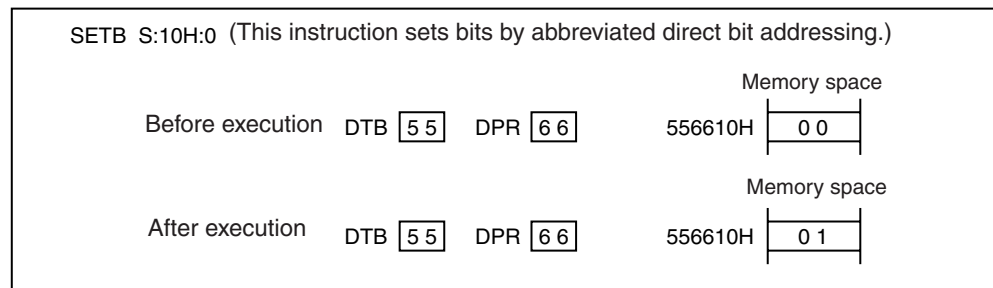
Figure B.3-8 Example of I/O direct bit addressing (io:bp)



○ **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).

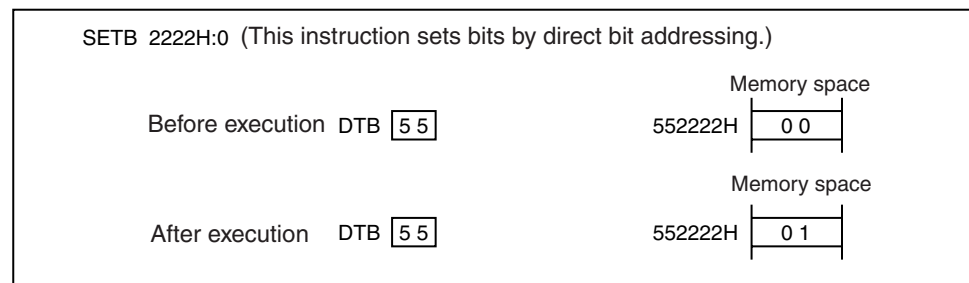
Figure B.3-9 Example of abbreviated direct bit addressing (dir:bp)



○ **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)

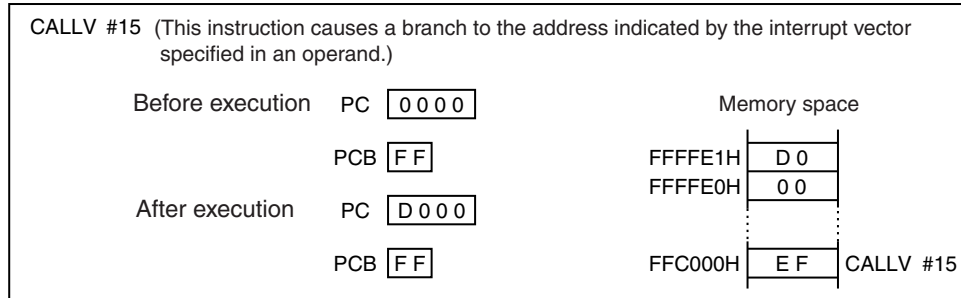


Table B.3-2 CALLV vector list

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 bank register (PCB) is FF_H, the vector area overlaps the vector area of IN #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2 "CALLV vector list").

B.4 Indirect Addressing

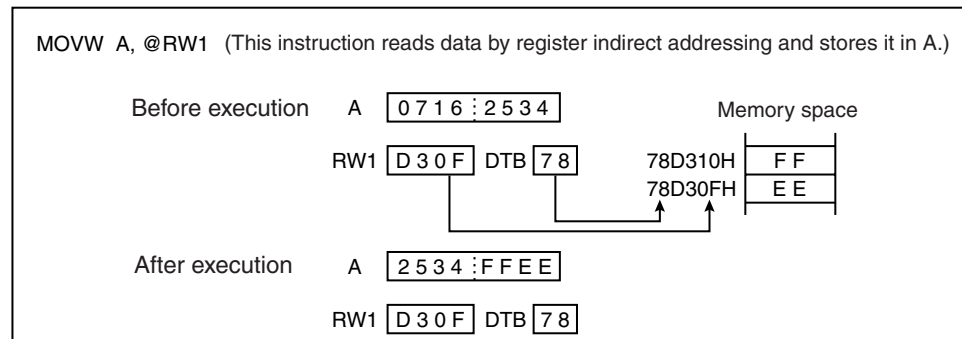
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ Indirect Addressing

○ Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

Figure B.4-1 Example of register indirect addressing (@RWj j = 0 to 3)

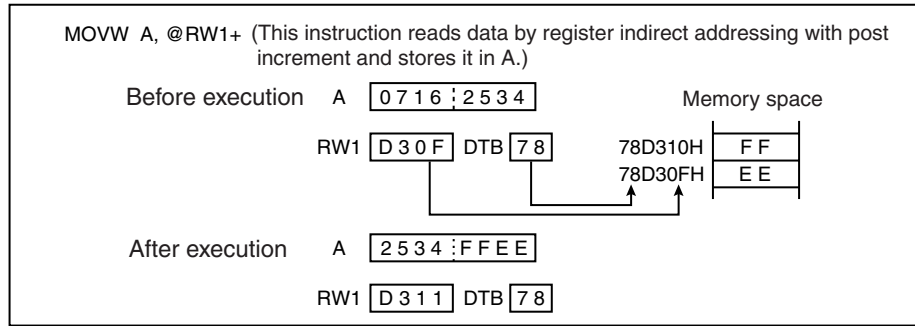


○ Register indirect addressing with post increment (@RWj+ j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.

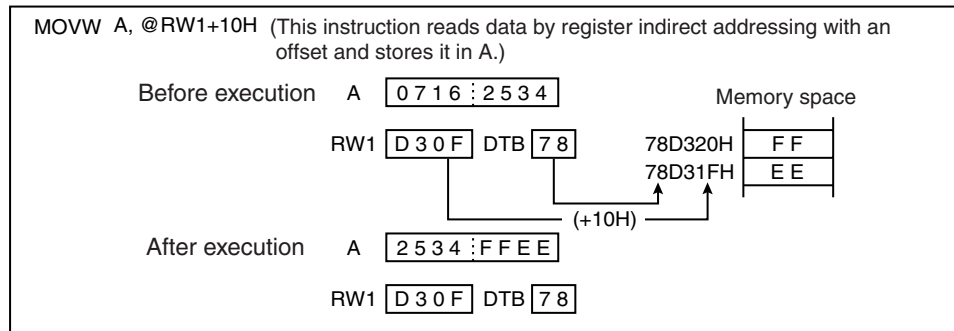
Figure B.4-2 Example of register indirect addressing with post increment (@RWj+ j = 0 to 3)



○ **Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)**

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

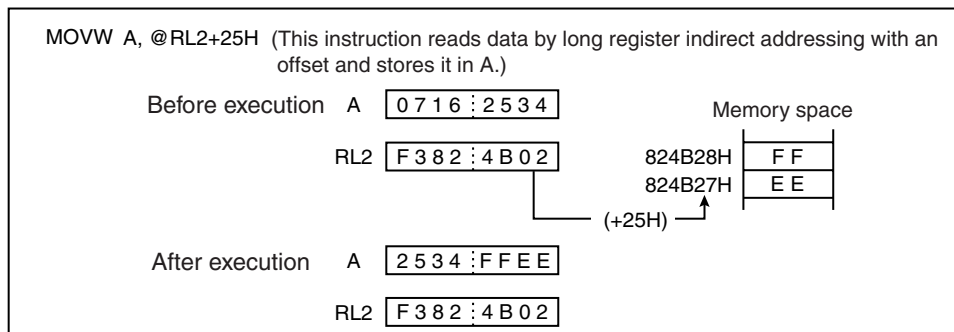
Figure B.4-3 Example of register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)



○ **Long register indirect addressing with offset (@RLi + 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)

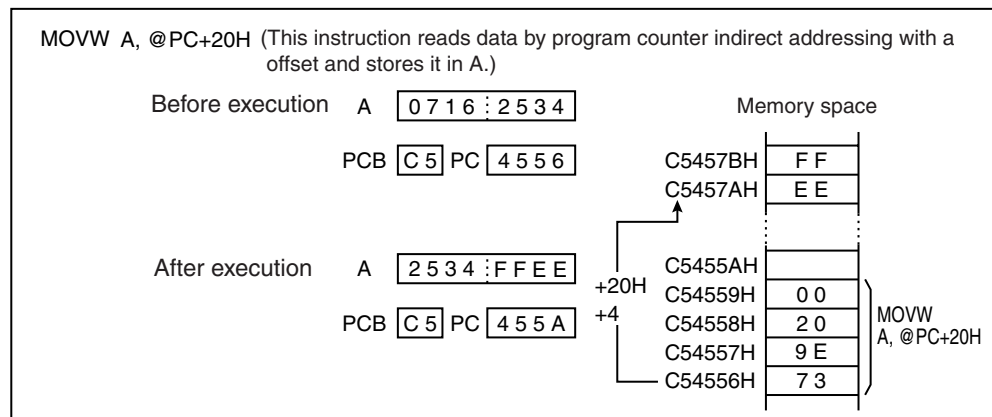


○ Program counter indirect addressing with offset (@PC + disp16)

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 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

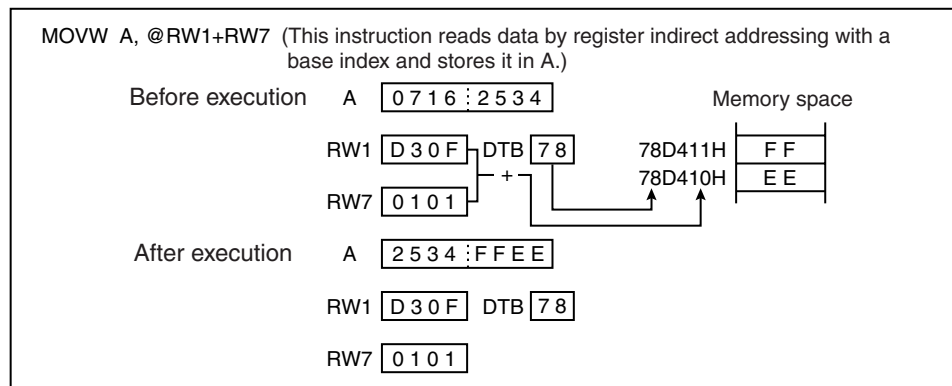
Figure B.4-5 Example of program counter indirect addressing with offset (@PC + disp16)



○ 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).

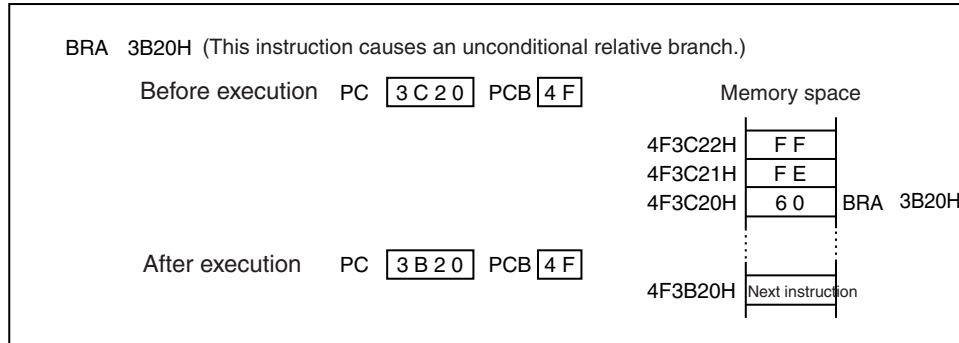
Figure B.4-6 Example of register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)



○ 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 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

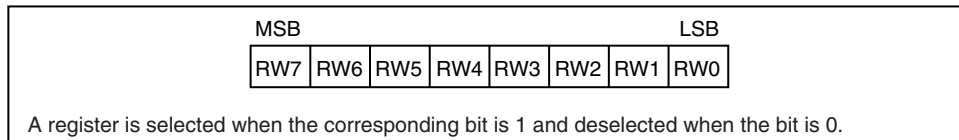
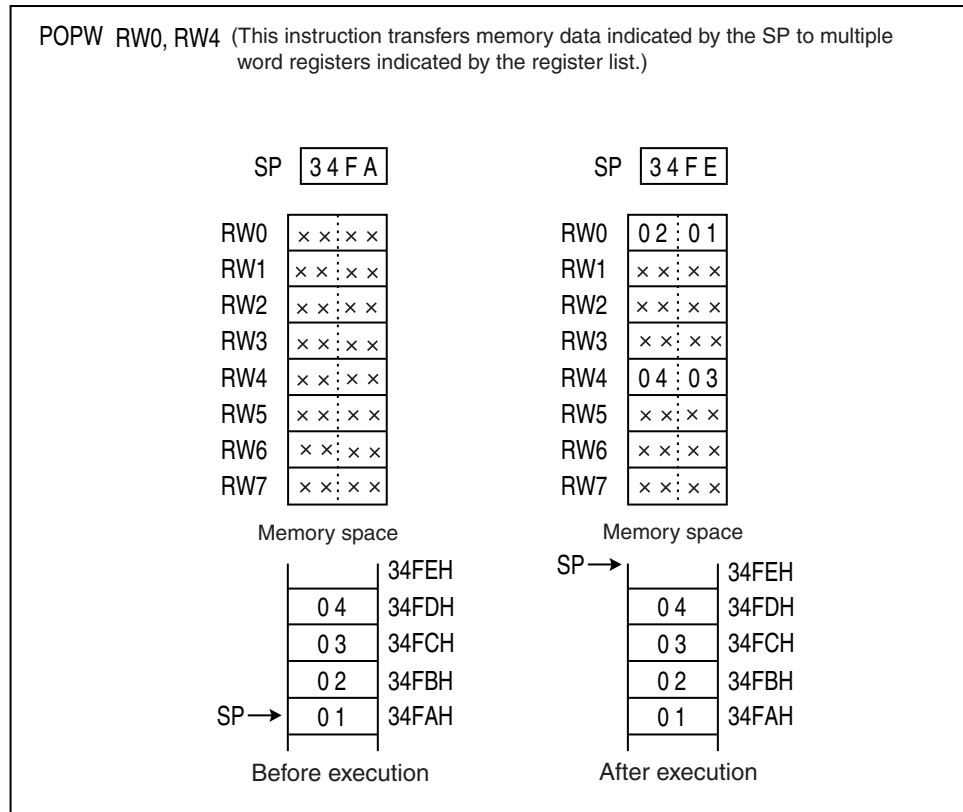


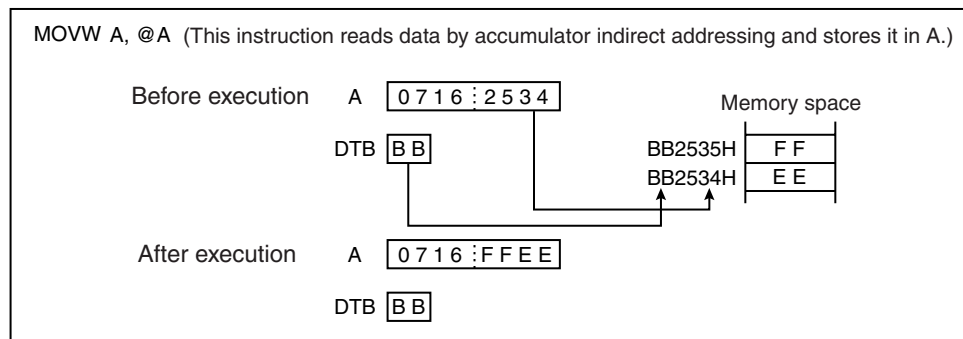
Figure B.4-9 Example of register list (rlist)



○ **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).

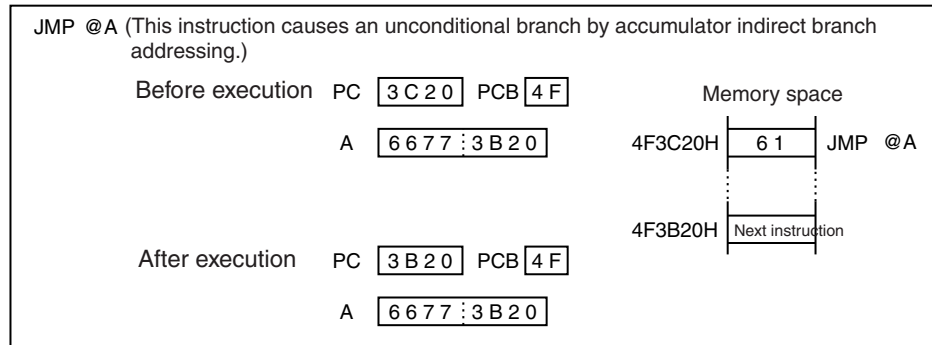
Figure B.4-10 Example of accumulator indirect addressing (@A)



○ **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 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.

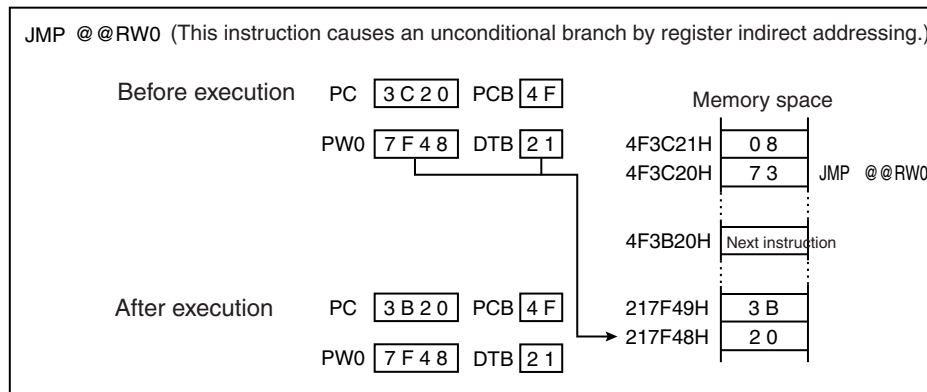
Figure B.4-11 Example of accumulator indirect branch addressing (@A)



○ **Indirect specification branch addressing (@ear)**

The address of the branch destination is the word data at the address indicated by ear.

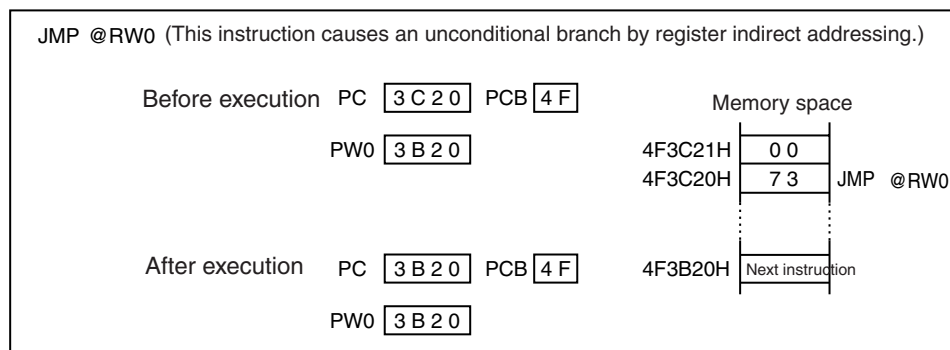
Figure B.4-12 Example of indirect specification branch addressing (@ear)



○ **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 "Execution cycle counts in each addressing mode" lists execution cycle counts and Table B.5-2 "Cycle count correction values for counting execution cycles" and Table B.5-3 "Cycle count correction values for counting instruction fetch cycles" summarize correction value data.

Table B.5-1 Execution cycle counts in each addressing mode

Code	Operand	(a) (*1)	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

*1: (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 ^(*1)		(c) word ^(*1)		(d) long ^(*1)	
	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

*1: (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	-

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.
- 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 "Effective address field" 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 ^(*1)
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 "Description of items in the instruction list" describes the items used in the F²MC-16LX Instruction List, and Table B.7-2 "Explanation on symbols in the instruction list" 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

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 "Effective address field" 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 bits 15 to 08 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.

Table B.7-1 Description of items in the instruction list (Continued)

Item	Description
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change Z: Set upon instruction execution. X: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	
RMW	Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory). *: Read Modify Write instruction -: Not Read Modify Write instruction Note: Cannot be used for an address that has different meanings between read and write operations.

Table B.7-2 Explanation on symbols in the instruction list

Symbol	Explanation
A	The bit length used varies depending on the 32-bit accumulator instruction. Byte: Low-order 8 bits of byte AL Word: 16 bits of word AL Long word: 32 bits of AL and AH
AH	16 high-order bits of A
AL	16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	Program bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB

Table B.7-2 Explanation on symbols in the instruction list (Continued)

Symbol	Explanation
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Abbreviated direct addressing
addr16	Direct addressing
addr24	Physical direct addressing
ad24 0-15	Bits 0 to 15 of addr24
ad24 16-23	Bits 16 to 23 of addr24
io	I/O area (000000H to 0000FFH)
#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 "41 Transfer instructions (byte)" to Table B.8-18 "10 String instructions" list the instructions used by the F²MC-16LX.

■ F²MC-16LX Instruction List

Table B.8-1 41 Transfer instructions (byte)

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
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 / MOV @A,T	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 x (b)	byte (A) <--> (eam)	Z	-	-	-	-	-	-	-	-	-
XCH Ri,ear	2	7	4	0	byte (Ri) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCH Ri,eam	2+	9 + (a)	2	2 x (b)	byte (Ri) <--> (eam)	-	-	-	-	-	-	-	-	-	-

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-2 38 Transfer instructions (byte)

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
MOVW A,dir	2	3	0	(c)	word (A) <-- (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	4	0	(c)	word (A) <-- (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	3	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	2	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 ear,Rwi	2+	4 + (a)	1	(c)	word (RWi) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVW eam,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 / MOVW @A,T	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 x (c)	word (A) <--> (eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi, ear	2	7	4	0	word (RWi) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi, eam	2+	9 + (a)	2	2 x (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 (ear1) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVL eam,A	2+	5 + (a)	0	(d)	long(eam1) <-- (A)	-	-	-	-	-	*	*	-	-	-

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

APPENDIX

Table B.8-3 42 Addition/subtraction instructions (byte, word, long word)

Mnemonic	#	~	RG	B	Operation	L	A	I	S	T	N	Z	V	C	R	M	W
ADD A,#imm8	2	2	0	0	byte (A) <-- (A) + imm8	Z	-	-	-	-	*	*	*	*	-	-	-
ADD A,dir	2	5	0	(b)	byte (A) <-- (A) + (dir)	Z	-	-	-	-	*	*	*	*	-	-	-
ADD A,ear	2	3	1	0	byte (A) <-- (A) + (ear)	Z	-	-	-	-	*	*	*	*	-	-	-
ADD A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) + (eam)	Z	-	-	-	-	*	*	*	*	-	-	-
ADD ear,A	2	3	2	0	byte (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-	-	-
ADD eam,A	2+	5 + (a)	0	2 x (b)	byte (eam) <-- (eam) + (A)	Z	-	-	-	-	*	*	*	*	-	-	-
ADDC A	1	2	0	0	byte (A) <-- (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-	-	-
ADDC A,ear	2	3	1	0	byte (A) <-- (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-	-	-
ADDC A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-	-	-
ADDDC A	1	3	0	0	byte (A) <-- (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-	-	-
SUB A,#imm8	2	2	0	0	byte (A) <-- (A) - imm8	Z	-	-	-	-	*	*	*	*	-	-	-
SUB A,dir	2	5	0	(b)	byte (A) <-- (A) - (dir)	Z	-	-	-	-	*	*	*	*	-	-	-
SUB A,ear	2	3	1	0	byte (A) <-- (A) - (ear)	Z	-	-	-	-	*	*	*	*	-	-	-
SUB A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) - (eam)	Z	-	-	-	-	*	*	*	*	-	-	-
SUB ear,A	2	3	2	0	byte (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-	-	-
SUB eam,A	2+	5 + (a)	0	2 x (b)	byte (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	-	-	-
SUBC A	1	2	0	0	byte (A) <-- (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-	-	-
SUBC A,ear	2	3	1	0	byte (A) <-- (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-	-	-
SUBC A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-	-	-
SUBDC A	1	3	0	0	byte (A) <-- (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-	-	-
ADDW A	1	2	0	0	word (A) <-- (AH) + (AL)	-	-	-	-	-	*	*	*	*	-	-	-
ADDW A,ear	2	3	1	0	word (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-	-	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-	-	-
ADDW A,#imm16	3	2	0	0	word (A) <-- (A) + imm16	-	-	-	-	-	*	*	*	*	-	-	-
ADDW ear,A	2	3	2	0	word (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-	-	-
ADDW eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) + (A)	-	-	-	-	-	*	*	*	*	-	-	-
ADDCW A,ear	2	3	1	0	word (A) <-- (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-	-	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-	-	-
SUBW A	1	2	0	0	word (A) <-- (AH) - (AL)	-	-	-	-	-	*	*	*	*	-	-	-
SUBW A,ear	2	3	1	0	word (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-	-	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-	-	-
SUBW A,#imm16	3	2	0	0	word (A) <-- (A) - imm16	-	-	-	-	-	*	*	*	*	-	-	-
SUBW ear,A	2	3	2	0	word (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-	-	-
SUBW eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	-	-	-
SUBCW A,ear	2	3	1	0	word (A) <-- (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-	-	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-	-	-
ADDL A,ear	2	6	2	0	long (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-	-	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-	-	-
ADDL A,#imm32	5	4	0	0	long (A) <-- (A) + imm32	-	-	-	-	-	*	*	*	*	-	-	-
SUBL A,ear	2	6	2	0	long (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-	-	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-	-	-
SUBL A,#imm32	5	4	0	0	long (A) <-- (A) - imm32	-	-	-	-	-	*	*	*	*	-	-	-

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" 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	L H	A H	I	S	T	N	Z	V	C	R M W
INC	ear	2	3	2	0	byte (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2 x (d)	long (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2 x (d)	long (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-5 11 Compare instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

APPENDIX

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 x (b): Normal
- *7: (c): Division by 0 or overflow 2 x (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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) 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 x (b): Normal
- *7: (c): Division by 0 or overflow, 2 x (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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

APPENDIX

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 x (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 x (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 x (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 x (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 x (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 x (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 x (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 x (c)	word (eam) <-- not (eam)	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-9 6 Logic 2 instructions (long word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	6	2	0	long (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	long (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	long (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-10 6 Sign inversion instructions (byte, word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	0	byte (A) <-- 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	byte (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	word (A) <-- 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	word (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 x (c)	word (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-11 1 Normalization instruction (long word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A,R0	2	*1	1	0	long (A) <-- Shifts to the position where '1' is set for the first time. byte (RD) <-- Shift count at that time	-	-	-	-	-	-	*	-	-	-

*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

APPENDIX

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) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	A	2	2	0	0	byte (A) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	ear	2	3	2	0	byte (ear) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	*
ROLC	ear	2	3	2	0	byte (ear) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- With left rotation 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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-13 31 Branch 1 instructions

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
BZ/BEQ rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/BNE rel	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/BHS rel	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT rel	2	*1	0	0	Branch on (V) nor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE rel	2	*1	0	0	Branch on (V) nor (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 addr16 *5	2+	7+(a)	0	2 x (c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
CALL @eam *4	3	6	0	(c)	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
CALLV #vct4 *5	1	7	0	2 x (c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP @ear *6	2	10	2	2 x (c)	word (PC) <-- (ear)0-15, (PCB) <-- (ear)16-23	-	-	-	-	-	-	-	-	-	-
CALLP @eam *6	2+	11+(a)	0	*2	word (PC) <-- (eam)0-15, (PCB) <-- (eam)16-23	-	-	-	-	-	-	-	-	-	-
CALLP addr24 *7	4	10	0	2 x (c)	word (PC) <-- addr0-15, (PCB) <-- addr16-23	-	-	-	-	-	-	-	-	-	-

*1: 4 when a branch is made; otherwise, 3
 *2: 3 x (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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" 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	R	M	W
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	Branch on byte (ear) = (ear) - 1, (ear) not equal to 0	-	-	-	-	-	*	*	*	*	-	-	-
DBNZ eam,rel	3+	*6	2	2 x (b)	Branch on byte (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	*	-	-	*
DWBNZ ear,rel	3	*5	2	0	Branch on word (ear) = (ear) - 1, (ear) not equal to 0	-	-	-	-	-	*	*	*	*	-	-	-
DWBNZ eam,rel	3+	*6	2	2 x (c)	Branch on word (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	*	-	-	*
INT #vct8	2	20	0	8 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-	-	-
INT addr16	3	16	0	6 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-	-	-
INTP addr24	4	17	0	6 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-	-	-
INT9	1	20	0	8 x (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 x (b) + 2 x (c) when jumping to the next interruption request; 6 x (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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" 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	L H	A H	I	S	T	N	Z	V	C	R M W
PUSHW A	1	4	0	(c)	word (SP) <-- (SP) - 2, ((SP)) <-- (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	4	0	(c)	word (SP) <-- (SP) - 2, ((SP)) <-- (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	4	0	(c)	word (SP) <-- (SP) - 2, ((SP)) <-- (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*5	*4	(SP) <-- (SP) - 2n, ((SP)) <-- (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	0	(c)	word (A) <-- ((SP)), (SP) <-- (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	0	(c)	word (AH) <-- ((SP)), (SP) <-- (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	4	0	(c)	word (PS) <-- ((SP)), (SP) <-- (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*5	*4	(rlst) <-- ((SP)), (SP) <-- (SP)	-	-	-	-	-	-	-	-	-	-
JCTX @A	1	14	0	6 x (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	0	byte (CCR) <-- (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	0	byte (CCR) <-- (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	0	byte (RP) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	0	byte (ILM) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	1	0	word (RWi) <-- ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	1	0	word (RWi) <-- eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	1	0	0	word (A) <-- ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	0	word (A) <-- eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	0	word (SP) <-- ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	0	word (SP) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	0	byte (A) <-- (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	0	byte (brg2) <-- (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 x (POP count) + 2 x (POP last register number), 7 when RLST = 0 (no transfer register)

*3: 29 + 3 x (PUSH count) - 3 x (PUSH last register number), 8 when RLST = 0 (no transfer register)

*4: (POP count) x (c) or (PUSH count) x (c)

*5: (POP count) or (PUSH count)

Note:

See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

APPENDIX

Table B.8-16 21 Bit operand instructions

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
MOVB A,dir:bp	3	5	0	(b)	byte (A) <-- (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,addr16:bp	4	5	0	(b)	byte (A) <-- (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,io:bp	3	4	0	(b)	byte (A) <-- (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB dir:bp,A	3	7	0	2 x (b)	bit (dir:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	7	0	2 x (b)	bit (addr16:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	6	0	2 x (b)	bit (io:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	7	0	2 x (b)	bit (dir:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	7	0	2 x (b)	bit (addr16:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	7	0	2 x (b)	bit (io:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	7	0	2 x (b)	bit (dir:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	7	0	2 x (b)	bit (addr16:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	7	0	2 x (b)	bit (io:bp)b <-- 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	*1	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*3	0	2 x (b)	Branch on (addr16:bp) b = 1, bit = 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 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

Table B.8-17 6 Accumulator operation instructions (byte, word)

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
SWAP	1	3	0	0	byte (A)0-7 <--> (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW / XCHW A,T	1	2	0	0	word (AH) <--> (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	L H	A H	I	S	T	N	Z	V	C	R M W
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	*5	*4	byte search @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*5	*4	byte search @AH- <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*5	*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	*5	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*5	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*5	*6	word fill @AH+ <-- AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, 4 + 7 x (RW0) when the counter expires, or 7n + 5 when a match occurs
 *2: 5 when RW0 is 0; otherwise, 4 + 8 x (RW0)
 *3: (b) x (RW0) + (b) x (RW0) When the source and destination access different areas, calculate the (b) item individually.
 *4: (b) x n
 *5: 2 x (RW0)
 *6: (c) x (RW0) + (c) x (RW0) When the source and destination access different areas, calculate the (c) item individually.
 *7: (c) x n

Note:

m: RW0 value (counter value), n: Loop count

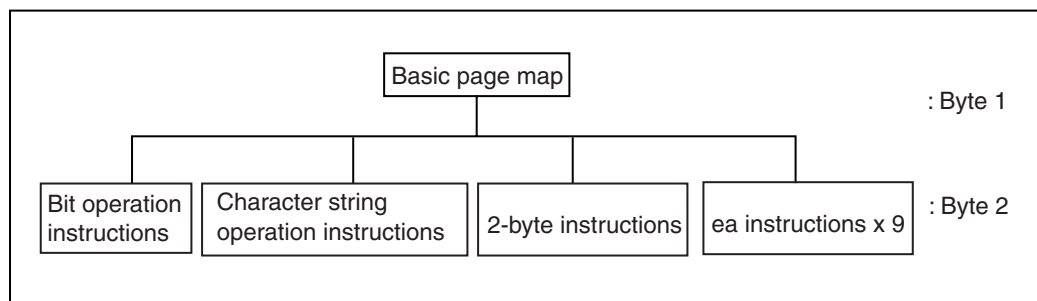
See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

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 "Basic page map" to Table B.9-21 "XCHW RWi, ea instruction (first byte = 7FH)" summarize the F²MC-16LX instruction map.

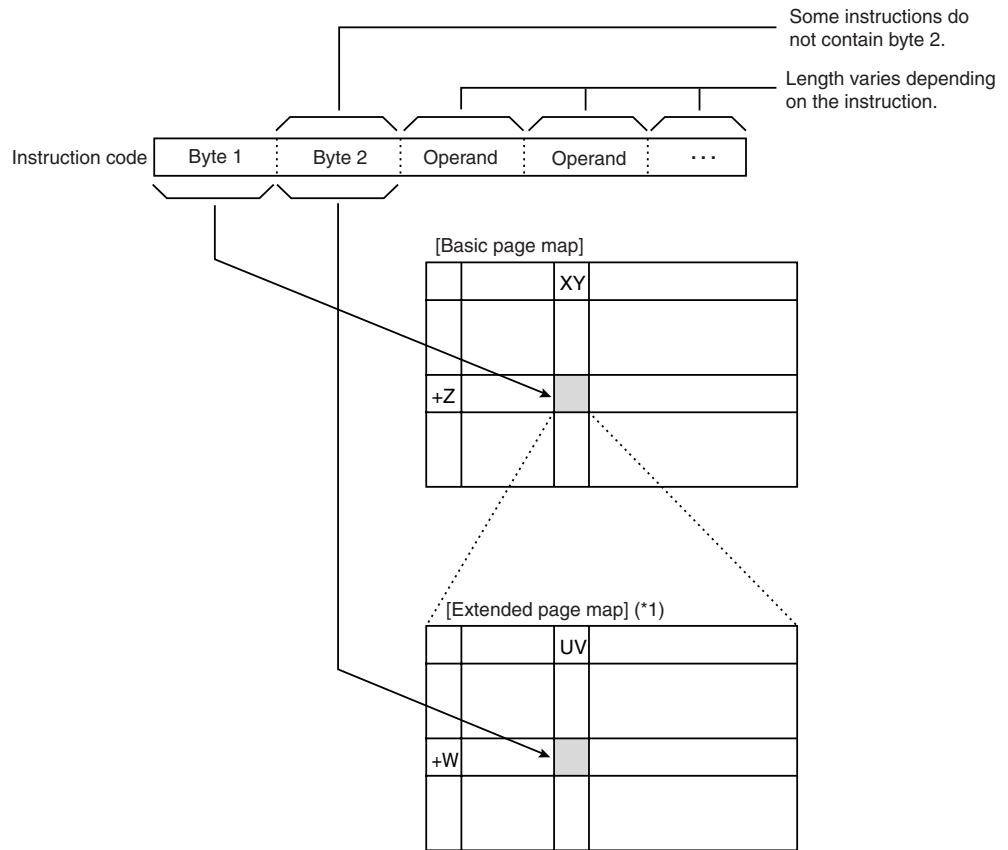
■ Structure of Instruction Map

Figure B.9-1 Structure of instruction map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 "Correspondence between actual instruction code and instruction map" shows the correspondence between an actual instruction code and instruction map.

Figure B.9-2 Correspondence between actual instruction code and instruction map



*1 The extended page map is a generic name of maps for bit operation instructions, character string operation instructions, 2-byte instructions, and ea instructions. Actually, there are multiple extended page maps for each type of instructions.

An example of an instruction code is shown in Table B.9-1 "Example of an instruction code".

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, #8rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic page map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	NOP	CMR	ADD A,dir	ADD A,#8	MOV A,dir	MOV A,io	BRA rel	ea instruction 1	MOV A,Ri	MOV Ri,A	MOV Ri,#8	MOVX A,Ri	MOVX A, @Ri+dB	MOVN A,#4	CALLV #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	JMP 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 #8	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+dB	MOVW @R W+dB,A			BT rel
+ 9	UNLINK	SUBL A,#32	SUBW A	SUBW A,#16	MOVW dir,A	MOVW io,ea	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	CMPW A,#32	CMPW A	CMPW 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 A	ORW A	ORW A,#16	PUSHW AH	POPW AH		MOVW ea,RWi								BGT rel
+ E	ASRW A	SWAPW A	XORW A	XORW A,#16	PUSHW PS	POPW PS	Character string operation instruction	XCH Ri,ea								BLS rel
+ F	LSRW A	ADDSP #16	MULW A	NOTW A	PUSHW rlist	POPW rlist	2-byte instruction	XCHW RWi,ea								BHI rel

Table B.9-3 Bit operation instruction map (first byte = 6C_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVB A,io:bp		MOVB io:bp,A		CLRB io:bp		SETB io:bp		BBC io .bp,rel		BBS io .bp,rel		WBTS io:bp		WBTC io:bp	
+ 1																
+ 2																
+ 3																
+ 4																
+ 5																
+ 6																
+ 7																
+ 8	MOVB A,dir:bp	MOVB A,a ddr16:bp	MOVB dir:bp,A	MOVB addr16:bp,A	CLRB dir:bp	CLRB a addr16:bp	SETB dir:bp	SETB a ddr16:bp	BBC dir:bp,rel	BBC ad16 .bp,rel	BBS dir:bp,rel	BBS ad16 .bp,rel				SBBS a ddr16:bp
+ 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	MOVSWI	MOVSWD					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB	FILSWI PCB		
+1	PCB,DTB								DTB	DTB	DTB	DTB	DTB		DTB	
+2	PCB,ADB								ADB	ADB	ADB	ADB	ADB		ADB	
+3	PCB,SPB								SPB	SPB	SPB	SPB	SPB		SPB	
+4	DTB,PCB															
+5	DTB,DTB															
+6	DTB,ADB															
+7	DTB,SPB															
+8	ADB,PCB															
+9	ADB,DTB															
+A	ADB,ADB															
+B	ADB,SPB															
+C	SPB,PCB															
+D	SPB,DTB															
+E	SPB,ADB															
+F	SPB,SPB															

Table B.9-5 2-byte instruction map (first byte = 6FH)

	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	MOVX A,@A	MOVX A, @RL3+d8	MOV @RL3 +d8,A	MOV A, @RL3+d8											
+7	ROL A	RORC A														
+8							MUL A									
+9							MULW A									
+A							DIVU A									
+B																
+C	LSLW A,R0	LSLL A,R0	LSL A,R0	MOVW @RL 2+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 @RL 3+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
+ 0	ADDL A, A,RL0 @RW0-d8	SUBL A, A,RL0 @RW0-d8	SUBL A, A,RL0 @RW0-d8	SUBL A, A,RL0 @RW0-d8	RW0, #16,rel	CWENE, I, @RW0-d8, #16,rel	CMP A, A,RL0 @RW0-d8	CMP A, A,RL0 @RW0-d8	ANDL A, A,RL0 @RW0-d8	ANDL A, A,RL0 @RW0-d8	ORL A, A,RL0 @RW0-d8	ORL A, A,RL0 @RW0-d8	XORL A, A,RL0 @RW0-d8	XORL A, A,RL0 @RW0-d8	R0, #8,rel	CWENE, I, @RW0-d8, #8,rel
+ 1	ADDL A, A,RL0 @RW1-d8	SUBL A, A,RL0 @RW1-d8	SUBL A, A,RL0 @RW1-d8	SUBL A, A,RL0 @RW1-d8	RW1, #16,rel	CWENE, I, @RW1-d8, #16,rel	CMP A, A,RL0 @RW1-d8	CMP A, A,RL0 @RW1-d8	ANDL A, A,RL0 @RW1-d8	ANDL A, A,RL0 @RW1-d8	ORL A, A,RL0 @RW1-d8	ORL A, A,RL0 @RW1-d8	XORL A, A,RL0 @RW1-d8	XORL A, A,RL0 @RW1-d8	R1, #8,rel	CWENE, I, @RW1-d8, #8,rel
+ 2	ADDL A, A,RL1 @RW2-d8	SUBL A, A,RL1 @RW2-d8	SUBL A, A,RL1 @RW2-d8	SUBL A, A,RL1 @RW2-d8	RW2, #16,rel	CWENE, I, @RW2-d8, #16,rel	CMP A, A,RL1 @RW2-d8	CMP A, A,RL1 @RW2-d8	ANDL A, A,RL1 @RW2-d8	ANDL A, A,RL1 @RW2-d8	ORL A, A,RL1 @RW2-d8	ORL A, A,RL1 @RW2-d8	XORL A, A,RL1 @RW2-d8	XORL A, A,RL1 @RW2-d8	R2, #8,rel	CWENE, I, @RW2-d8, #8,rel
+ 3	ADDL A, A,RL1 @RW3-d8	SUBL A, A,RL1 @RW3-d8	SUBL A, A,RL1 @RW3-d8	SUBL A, A,RL1 @RW3-d8	RW3, #16,rel	CWENE, I, @RW3-d8, #16,rel	CMP A, A,RL1 @RW3-d8	CMP A, A,RL1 @RW3-d8	ANDL A, A,RL1 @RW3-d8	ANDL A, A,RL1 @RW3-d8	ORL A, A,RL1 @RW3-d8	ORL A, A,RL1 @RW3-d8	XORL A, A,RL1 @RW3-d8	XORL A, A,RL1 @RW3-d8	R3, #8,rel	CWENE, I, @RW3-d8, #8,rel
+ 4	ADDL A, A,RL2 @RW4-d8	SUBL A, A,RL2 @RW4-d8	SUBL A, A,RL2 @RW4-d8	SUBL A, A,RL2 @RW4-d8	RW4, #16,rel	CWENE, I, @RW4-d8, #16,rel	CMP A, A,RL2 @RW4-d8	CMP A, A,RL2 @RW4-d8	ANDL A, A,RL2 @RW4-d8	ANDL A, A,RL2 @RW4-d8	ORL A, A,RL2 @RW4-d8	ORL A, A,RL2 @RW4-d8	XORL A, A,RL2 @RW4-d8	XORL A, A,RL2 @RW4-d8	R4, #8,rel	CWENE, I, @RW4-d8, #8,rel
+ 5	ADDL A, A,RL2 @RW5-d8	SUBL A, A,RL2 @RW5-d8	SUBL A, A,RL2 @RW5-d8	SUBL A, A,RL2 @RW5-d8	RW5, #16,rel	CWENE, I, @RW5-d8, #16,rel	CMP A, A,RL2 @RW5-d8	CMP A, A,RL2 @RW5-d8	ANDL A, A,RL2 @RW5-d8	ANDL A, A,RL2 @RW5-d8	ORL A, A,RL2 @RW5-d8	ORL A, A,RL2 @RW5-d8	XORL A, A,RL2 @RW5-d8	XORL A, A,RL2 @RW5-d8	R5, #8,rel	CWENE, I, @RW5-d8, #8,rel
+ 6	ADDL A, A,RL3 @RW6-d8	SUBL A, A,RL3 @RW6-d8	SUBL A, A,RL3 @RW6-d8	SUBL A, A,RL3 @RW6-d8	RW6, #16,rel	CWENE, I, @RW6-d8, #16,rel	CMP A, A,RL3 @RW6-d8	CMP A, A,RL3 @RW6-d8	ANDL A, A,RL3 @RW6-d8	ANDL A, A,RL3 @RW6-d8	ORL A, A,RL3 @RW6-d8	ORL A, A,RL3 @RW6-d8	XORL A, A,RL3 @RW6-d8	XORL A, A,RL3 @RW6-d8	R6, #8,rel	CWENE, I, @RW6-d8, #8,rel
+ 7	ADDL A, A,RL3 @RW7-d8	SUBL A, A,RL3 @RW7-d8	SUBL A, A,RL3 @RW7-d8	SUBL A, A,RL3 @RW7-d8	RW7, #16,rel	CWENE, I, @RW7-d8, #16,rel	CMP A, A,RL3 @RW7-d8	CMP A, A,RL3 @RW7-d8	ANDL A, A,RL3 @RW7-d8	ANDL A, A,RL3 @RW7-d8	ORL A, A,RL3 @RW7-d8	ORL A, A,RL3 @RW7-d8	XORL A, A,RL3 @RW7-d8	XORL A, A,RL3 @RW7-d8	R7, #8,rel	CWENE, I, @RW7-d8, #8,rel
+ 8	ADDL A, A,@RW0 @RW0-d16	SUBL A, A,@RW0 @RW0-d16	SUBL A, A,@RW0 @RW0-d16	SUBL A, A,@RW0 @RW0-d16	@RW0, #16,rel	CWENE, I, @RW0-d16, #16,rel	CMP A, A,@RW0 @RW0-d16	CMP A, A,@RW0 @RW0-d16	ANDL A, A,@RW0 @RW0-d16	ANDL A, A,@RW0 @RW0-d16	ORL A, A,@RW0 @RW0-d16	ORL A, A,@RW0 @RW0-d16	XORL A, A,@RW0 @RW0-d16	XORL A, A,@RW0 @RW0-d16	@RW0, #8,rel	CWENE, I, @RW0-d16, #8,rel
+ 9	ADDL A, A,@RW1 @RW1-d16	SUBL A, A,@RW1 @RW1-d16	SUBL A, A,@RW1 @RW1-d16	SUBL A, A,@RW1 @RW1-d16	@RW1, #16,rel	CWENE, I, @RW1-d16, #16,rel	CMP A, A,@RW1 @RW1-d16	CMP A, A,@RW1 @RW1-d16	ANDL A, A,@RW1 @RW1-d16	ANDL A, A,@RW1 @RW1-d16	ORL A, A,@RW1 @RW1-d16	ORL A, A,@RW1 @RW1-d16	XORL A, A,@RW1 @RW1-d16	XORL A, A,@RW1 @RW1-d16	@RW1, #8,rel	CWENE, I, @RW1-d16, #8,rel
+ A	ADDL A, A,@RW2 @RW2-d16	SUBL A, A,@RW2 @RW2-d16	SUBL A, A,@RW2 @RW2-d16	SUBL A, A,@RW2 @RW2-d16	@RW2, #16,rel	CWENE, I, @RW2-d16, #16,rel	CMP A, A,@RW2 @RW2-d16	CMP A, A,@RW2 @RW2-d16	ANDL A, A,@RW2 @RW2-d16	ANDL A, A,@RW2 @RW2-d16	ORL A, A,@RW2 @RW2-d16	ORL A, A,@RW2 @RW2-d16	XORL A, A,@RW2 @RW2-d16	XORL A, A,@RW2 @RW2-d16	@RW2, #8,rel	CWENE, I, @RW2-d16, #8,rel
+ B	ADDL A, A,@RW3 @RW3-d16	SUBL A, A,@RW3 @RW3-d16	SUBL A, A,@RW3 @RW3-d16	SUBL A, A,@RW3 @RW3-d16	@RW3, #16,rel	CWENE, I, @RW3-d16, #16,rel	CMP A, A,@RW3 @RW3-d16	CMP A, A,@RW3 @RW3-d16	ANDL A, A,@RW3 @RW3-d16	ANDL A, A,@RW3 @RW3-d16	ORL A, A,@RW3 @RW3-d16	ORL A, A,@RW3 @RW3-d16	XORL A, A,@RW3 @RW3-d16	XORL A, A,@RW3 @RW3-d16	@RW3, #8,rel	CWENE, I, @RW3-d16, #8,rel
+ C	ADDL A, A,@RW0+ @RW0-RW7	SUBL A, A,@RW0+ @RW0-RW7	SUBL A, A,@RW0+ @RW0-RW7	SUBL A, A,@RW0+ @RW0-RW7	Use prohibited	CWENE, I, @RW0-RW7, prohibited	CMP A, A,@RW0+ @RW0-RW7	CMP A, A,@RW0+ @RW0-RW7	ANDL A, A,@RW0+ @RW0-RW7	ANDL A, A,@RW0+ @RW0-RW7	ORL A, A,@RW0+ @RW0-RW7	ORL A, A,@RW0+ @RW0-RW7	XORL A, A,@RW0+ @RW0-RW7	XORL A, A,@RW0+ @RW0-RW7	Use prohibited	CWENE, I, @RW0-RW7, prohibited
+ D	ADDL A, A,@RW1+ @RW1-RW7	SUBL A, A,@RW1+ @RW1-RW7	SUBL A, A,@RW1+ @RW1-RW7	SUBL A, A,@RW1+ @RW1-RW7	Use prohibited	CWENE, I, @RW1-RW7, prohibited	CMP A, A,@RW1+ @RW1-RW7	CMP A, A,@RW1+ @RW1-RW7	ANDL A, A,@RW1+ @RW1-RW7	ANDL A, A,@RW1+ @RW1-RW7	ORL A, A,@RW1+ @RW1-RW7	ORL A, A,@RW1+ @RW1-RW7	XORL A, A,@RW1+ @RW1-RW7	XORL A, A,@RW1+ @RW1-RW7	Use prohibited	CWENE, I, @RW1-RW7, prohibited
+ E	ADDL A, A,@RW2+ @PC-d16	SUBL A, A,@RW2+ @PC-d16	SUBL A, A,@RW2+ @PC-d16	SUBL A, A,@RW2+ @PC-d16	Use prohibited	CWENE, I, @PC-d16, prohibited	CMP A, A,@RW2+ @PC-d16	CMP A, A,@RW2+ @PC-d16	ANDL A, A,@RW2+ @PC-d16	ANDL A, A,@RW2+ @PC-d16	ORL A, A,@RW2+ @PC-d16	ORL A, A,@RW2+ @PC-d16	XORL A, A,@RW2+ @PC-d16	XORL A, A,@RW2+ @PC-d16	Use prohibited	CWENE, I, @PC-d16, prohibited
+ F	ADDL A, A,@RW3+ addr16	SUBL A, A,@RW3+ addr16	SUBL A, A,@RW3+ addr16	SUBL A, A,@RW3+ addr16	Use prohibited	CWENE, I, addr16, #16,rel	CMP A, A,@RW3+ addr16	CMP A, A,@RW3+ addr16	ANDL A, A,@RW3+ addr16	ANDL A, A,@RW3+ addr16	ORL A, A,@RW3+ addr16	ORL A, A,@RW3+ addr16	XORL A, A,@RW3+ addr16	XORL A, A,@RW3+ addr16	Use prohibited	CWENE, I, addr16, #8,rel

Table B.9-7 ea instruction 2 (first byte = 71H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMPP @RL0	JMPP @RL0	CALLP @RL0	CALLP @RL0	INCL RL0	INCL @RW0+d8	DECL RL0	DECL @RW0+d8	MOVL ARL0	MOVL A, @RW0+d8	MOVL RL0,A	MOVL @R W0+d8,A	MOV R0,#8	MOV @R W0+d8,#8	MOVEA ARW0	MOVEA A, @RW0+d8
+ 1	JMPP @RL0	JMPP @RL0	CALLP @RL0	CALLP @RL0	INCL RL0	INCL @RW1+d8	DECL RL0	DECL @RW1+d8	MOVL ARL0	MOVL A, @RW1+d8	MOVL RL0,A	MOVL @R W1+d8,A	MOV R1,#8	MOV @R W1+d8,#8	MOVEA ARW1	MOVEA A, @RW1+d8
+ 2	JMPP @RL1	JMPP @RL1	CALLP @RL1	CALLP @RL1	INCL RL1	INCL @RW2+d8	DECL RL1	DECL @RW2+d8	MOVL ARL1	MOVL A, @RW2+d8	MOVL RL1,A	MOVL @R W2+d8,A	MOV R2,#8	MOV @R W2+d8,#8	MOVEA ARW2	MOVEA A, @RW2+d8
+ 3	JMPP @RL1	JMPP @RL1	CALLP @RL1	CALLP @RL1	INCL RL1	INCL @RW3+d8	DECL RL1	DECL @RW3+d8	MOVL ARL1	MOVL A, @RW3+d8	MOVL RL1,A	MOVL @R W3+d8,A	MOV R3,#8	MOV @R W3+d8,#8	MOVEA ARW3	MOVEA A, @RW3+d8
+ 4	JMPP @RL2	JMPP @RL2	CALLP @RL2	CALLP @RL2	INCL RL2	INCL @RW4+d8	DECL RL2	DECL @RW4+d8	MOVL ARL2	MOVL A, @RW4+d8	MOVL RL2,A	MOVL @R W4+d8,A	MOV R4,#8	MOV @R W4+d8,#8	MOVEA ARW4	MOVEA A, @RW4+d8
+ 5	JMPP @RL2	JMPP @RL2	CALLP @RL2	CALLP @RL2	INCL RL2	INCL @RW5+d8	DECL RL2	DECL @RW5+d8	MOVL ARL2	MOVL A, @RW5+d8	MOVL RL2,A	MOVL @R W5+d8,A	MOV R5,#8	MOV @R W5+d8,#8	MOVEA ARW5	MOVEA A, @RW5+d8
+ 6	JMPP @RL3	JMPP @RL3	CALLP @RL3	CALLP @RL3	INCL RL3	INCL @RW6+d8	DECL RL3	DECL @RW6+d8	MOVL ARL3	MOVL A, @RW6+d8	MOVL RL3,A	MOVL @R W6+d8,A	MOV R6,#8	MOV @R W6+d8,#8	MOVEA ARW6	MOVEA A, @RW6+d8
+ 7	JMPP @RL3	JMPP @RL3	CALLP @RL3	CALLP @RL3	INCL RL3	INCL @RW7+d8	DECL RL3	DECL @RW7+d8	MOVL ARL3	MOVL A, @RW7+d8	MOVL RL3,A	MOVL @R W7+d8,A	MOV R7,#8	MOV @R W7+d8,#8	MOVEA ARW7	MOVEA A, @RW7+d8
+ 8	JMPP @RW0	JMPP @RW0	CALLP @RW0	CALLP @RW0	INCL @RW0	INCL @RW0+d16	DECL @RW0	DECL @RW0+d16	MOVL A,@RW0	MOVL A, @RW0+d16	MOVL @RW0,A	MOVL @R W0+d16,A	MOV @RW0,#8	MOV @RW 0+d16,#8	MOVEA A,@RW0	MOVEA A, @RW0+d16
+ 9	JMPP @RW1	JMPP @RW1	CALLP @RW1	CALLP @RW1	INCL @RW1	INCL @RW1+d16	DECL @RW1	DECL @RW1+d16	MOVL A,@RW1	MOVL A, @RW1+d16	MOVL @RW1,A	MOVL @R W1+d16,A	MOV @RW1,#8	MOV @RW 1+d16,#8	MOVEA A,@RW1	MOVEA A, @RW1+d16
+ A	JMPP @RW2	JMPP @RW2	CALLP @RW2	CALLP @RW2	INCL @RW2	INCL @RW2+d16	DECL @RW2	DECL @RW2+d16	MOVL A,@RW2	MOVL A, @RW2+d16	MOVL @RW2,A	MOVL @R W2+d16,A	MOV @RW2,#8	MOV @RW 2+d16,#8	MOVEA A,@RW2	MOVEA A, @RW2+d16
+ B	JMPP @RW3	JMPP @RW3	CALLP @RW3	CALLP @RW3	INCL @RW3	INCL @RW3+d16	DECL @RW3	DECL @RW3+d16	MOVL A,@RW3	MOVL A, @RW3+d16	MOVL @RW3,A	MOVL @R W3+d16,A	MOV @RW3,#8	MOV @RW 3+d16,#8	MOVEA A,@RW3	MOVEA A, @RW3+d16
+ C	JMPP @RW0+	JMPP @RW0+	CALLP @RW0+	CALLP @RW0+	INCL @RW0+	INCL @RW0+RW7	DECL @RW0+	DECL @RW0+RW7	MOVL A,@RW0+	MOVL A, @RW0+RW7	MOVL @RW0+,A	MOVL @R W0+RW7,A	MOV @RW0+,#8	MOV @RW 0+RW7,#8	MOVEA A,@RW0+	MOVEA A, @RW0+RW7
+ D	JMPP @RW1+	JMPP @RW1+	CALLP @RW1+	CALLP @RW1+	INCL @RW1+	INCL @RW1+RW7	DECL @RW1+	DECL @RW1+RW7	MOVL A,@RW1+	MOVL A, @RW1+RW7	MOVL @RW1+,A	MOVL @R W1+RW7,A	MOV @RW1+,#8	MOV @RW 1+RW7,#8	MOVEA A,@RW1+	MOVEA A, @RW1+RW7
+ E	JMPP @RW2+	JMPP @RW2+	CALLP @RW2+	CALLP @RW2+	INCL @RW2+	INCL @RW2+d16	DECL @RW2+	DECL @RW2+d16	MOVL A,@RW2+	MOVL A, @RW2+d16	MOVL @RW2+,A	MOVL @P C+d16,A	MOV @RW2+,#8	MOV @P C+d16,#8	MOVEA A,@RW2+	MOVEA A, @PC+d16
+ F	JMPP @RW3+	JMPP @RW3+	CALLP @RW3+	CALLP @RW3+	INCL @RW3+	INCL addr16	DECL @RW3+	DECL addr16	MOVL A,@RW3+	MOVL A, addr16	MOVL @RW3+,A	MOVL addr16,A	MOV @RW3+,#8	MOV addr16,#8	MOVEA A,@RW3+	MOVEA A, 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	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW0+d8	XCH	XCH A, @RW0+d8
+ 1	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW1+d8	XCH	XCH A, @RW1+d8
+ 2	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW2+d8	XCH	XCH A, @RW2+d8
+ 3	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW3+d8	XCH	XCH A, @RW3+d8
+ 4	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW4+d8	XCH	XCH A, @RW4+d8
+ 5	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW5+d8	XCH	XCH A, @RW5+d8
+ 6	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW6+d8	XCH	XCH A, @RW6+d8
+ 7	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW7+d8	XCH	XCH A, @RW7+d8
+ 8	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW0+d16	XCH	XCH A, @RW0+d16
+ 9	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW1+d16	XCH	XCH A, @RW1+d16
+ A	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW2+d16	XCH	XCH A, @RW2+d16
+ B	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW3+d16	XCH	XCH A, @RW3+d16
+ C	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW0+RW7	XCH	XCH A, @RW0+RW7
+ D	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @R	MOVX	MOVX A, @RW1+RW7	XCH	XCH A, @RW1+RW7
+ E	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV @P	MOVX	MOVX A, @PC+d16	XCH	XCH A, @PC+d16
+ F	ROL	ROL	RORC	RORC	INC	INC	DEC	DEC	MOV	MOV	MOV	MOV	MOVX	MOVX A, @RW3+	XCH	XCH A, @RW3+

Table B.9-9 ea instruction 4 (first byte = 73H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMP @RW0	JMP @RW0-d8	CALL @RW0	CALL @RW0-d8	INCW RW0	DECW RW0	DECW RW0	DECW @RW0-d8	MOVW A, RW0	MOVW A, @RW0-d8	MOVW RW0, A	MOVW @RW0-d8, A	MOVW RW0, #16	MOVW @RW0-d8, #16	XCHW A, RW0	XCHW A, RW0-d8
+ 1	JMP @RW1	JMP @RW1-d8	CALL @RW1	CALL @RW1-d8	INCW RW1	DECW RW1	DECW RW1	DECW @RW1-d8	MOVW A, RW1	MOVW A, @RW1-d8	MOVW RW1, A	MOVW @RW1-d8, A	MOVW RW1, #16	MOVW @RW1-d8, #16	XCHW A, RW1	XCHW A, RW1-d8
+ 2	JMP @RW2	JMP @RW2-d8	CALL @RW2	CALL @RW2-d8	INCW RW2	DECW RW2	DECW RW2	DECW @RW2-d8	MOVW A, RW2	MOVW A, @RW2-d8	MOVW RW2, A	MOVW @RW2-d8, A	MOVW RW2, #16	MOVW @RW2-d8, #16	XCHW A, RW2	XCHW A, RW2-d8
+ 3	JMP @RW3	JMP @RW3-d8	CALL @RW3	CALL @RW3-d8	INCW RW3	DECW RW3	DECW RW3	DECW @RW3-d8	MOVW A, RW3	MOVW A, @RW3-d8	MOVW RW3, A	MOVW @RW3-d8, A	MOVW RW3, #16	MOVW @RW3-d8, #16	XCHW A, RW3	XCHW A, RW3-d8
+ 4	JMP @RW4	JMP @RW4-d8	CALL @RW4	CALL @RW4-d8	INCW RW4	DECW RW4	DECW RW4	DECW @RW4-d8	MOVW A, RW4	MOVW A, @RW4-d8	MOVW RW4, A	MOVW @RW4-d8, A	MOVW RW4, #16	MOVW @RW4-d8, #16	XCHW A, RW4	XCHW A, RW4-d8
+ 5	JMP @RW5	JMP @RW5-d8	CALL @RW5	CALL @RW5-d8	INCW RW5	DECW RW5	DECW RW5	DECW @RW5-d8	MOVW A, RW5	MOVW A, @RW5-d8	MOVW RW5, A	MOVW @RW5-d8, A	MOVW RW5, #16	MOVW @RW5-d8, #16	XCHW A, RW5	XCHW A, RW5-d8
+ 6	JMP @RW6	JMP @RW6-d8	CALL @RW6	CALL @RW6-d8	INCW RW6	DECW RW6	DECW RW6	DECW @RW6-d8	MOVW A, RW6	MOVW A, @RW6-d8	MOVW RW6, A	MOVW @RW6-d8, A	MOVW RW6, #16	MOVW @RW6-d8, #16	XCHW A, RW6	XCHW A, RW6-d8
+ 7	JMP @RW7	JMP @RW7-d8	CALL @RW7	CALL @RW7-d8	INCW RW7	DECW RW7	DECW RW7	DECW @RW7-d8	MOVW A, RW7	MOVW A, @RW7-d8	MOVW RW7, A	MOVW @RW7-d8, A	MOVW RW7, #16	MOVW @RW7-d8, #16	XCHW A, RW7	XCHW A, RW7-d8
+ 8	JMP @RW0	JMP @RW0-d16	CALL @RW0	CALL @RW0-d16	INCW RW0	DECW RW0	DECW RW0	DECW @RW0-d16	MOVW A, RW0	MOVW A, @RW0-d16	MOVW RW0, A	MOVW @RW0-d16, A	MOVW RW0, #16	MOVW @RW0-d16, #16	XCHW A, RW0	XCHW A, RW0-d16
+ 9	JMP @RW1	JMP @RW1-d16	CALL @RW1	CALL @RW1-d16	INCW RW1	DECW RW1	DECW RW1	DECW @RW1-d16	MOVW A, RW1	MOVW A, @RW1-d16	MOVW RW1, A	MOVW @RW1-d16, A	MOVW RW1, #16	MOVW @RW1-d16, #16	XCHW A, RW1	XCHW A, RW1-d16
+ A	JMP @RW2	JMP @RW2-d16	CALL @RW2	CALL @RW2-d16	INCW RW2	DECW RW2	DECW RW2	DECW @RW2-d16	MOVW A, RW2	MOVW A, @RW2-d16	MOVW RW2, A	MOVW @RW2-d16, A	MOVW RW2, #16	MOVW @RW2-d16, #16	XCHW A, RW2	XCHW A, RW2-d16
+ B	JMP @RW3	JMP @RW3-d16	CALL @RW3	CALL @RW3-d16	INCW RW3	DECW RW3	DECW RW3	DECW @RW3-d16	MOVW A, RW3	MOVW A, @RW3-d16	MOVW RW3, A	MOVW @RW3-d16, A	MOVW RW3, #16	MOVW @RW3-d16, #16	XCHW A, RW3	XCHW A, RW3-d16
+ C	JMP @RW0+	JMP @RW0-RW7	CALL @RW0+	CALL @RW0-RW7	INCW RW0+	DECW RW0+	DECW RW0+	DECW @RW0-RW7	MOVW A, RW0+	MOVW A, @RW0-RW7	MOVW RW0+, A	MOVW @RW0-RW7, A	MOVW RW0+, #16	MOVW @RW0-RW7, #16	XCHW A, RW0+	XCHW A, RW0-RW7
+ D	JMP @RW1+	JMP @RW1-RW7	CALL @RW1+	CALL @RW1-RW7	INCW RW1+	DECW RW1+	DECW RW1+	DECW @RW1-RW7	MOVW A, RW1+	MOVW A, @RW1-RW7	MOVW RW1+, A	MOVW @RW1-RW7, A	MOVW RW1+, #16	MOVW @RW1-RW7, #16	XCHW A, RW1+	XCHW A, RW1-RW7
+ E	JMP @PC-d16	JMP @PC-d16	CALL @PC-d16	CALL @PC-d16	INCW @PC-d16	DECW @PC-d16	DECW @PC-d16	DECW @PC-d16	MOVW A, @PC-d16	MOVW A, @PC-d16	MOVW @PC-d16, A	MOVW @PC-d16, A	MOVW @PC-d16, #16	MOVW @PC-d16, #16	XCHW A, @PC-d16	XCHW A, @PC-d16
+ F	JMP @RW3+	JMP @RW3+	CALL @RW3+	CALL @RW3+	INCW @RW3+	DECW @RW3+	DECW @RW3+	DECW @RW3+	MOVW A, @RW3+	MOVW A, @RW3+	MOVW @RW3+, A	MOVW @RW3+, A	MOVW @RW3+, #16	MOVW @RW3+, #16	XCHW A, @RW3+	addr16

Table B.9-10 ea instruction 5 (first byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	E0	C0	D0	E0	F0
+ 0	ADD A.R0	ADD A, @RW0-d8	SUB A.R0	SUB A, @RW0-d8	ADD A.R0	ADD A, @RW0-d8	CMP A.R0	CMP A, @RW0-d8	AND A.R0	AND A, @RW0-d8	OR A.R0	OR A, @RW0-d8	XOR A.R0	XOR A, @RW0-d8	DBNZ R0,r	DBNZ @ RW0-d8,r
+ 1	ADD A.R1	ADD A, @RW1-d8	SUB A.R1	SUB A, @RW1-d8	ADD A.R1	ADD A, @RW1-d8	CMP A.R1	CMP A, @RW1-d8	AND A.R1	AND A, @RW1-d8	OR A.R1	OR A, @RW1-d8	XOR A.R1	XOR A, @RW1-d8	DBNZ R1,r	DBNZ @ RW1-d8,r
+ 2	ADD A.R2	ADD A, @RW2-d8	SUB A.R2	SUB A, @RW2-d8	ADD A.R2	ADD A, @RW2-d8	CMP A.R2	CMP A, @RW2-d8	AND A.R2	AND A, @RW2-d8	OR A.R2	OR A, @RW2-d8	XOR A.R2	XOR A, @RW2-d8	DBNZ R2,r	DBNZ @ RW2-d8,r
+ 3	ADD A.R3	ADD A, @RW3-d8	SUB A.R3	SUB A, @RW3-d8	ADD A.R3	ADD A, @RW3-d8	CMP A.R3	CMP A, @RW3-d8	AND A.R3	AND A, @RW3-d8	OR A.R3	OR A, @RW3-d8	XOR A.R3	XOR A, @RW3-d8	DBNZ R3,r	DBNZ @ RW3-d8,r
+ 4	ADD A.R4	ADD A, @RW4-d8	SUB A.R4	SUB A, @RW4-d8	ADD A.R4	ADD A, @RW4-d8	CMP A.R4	CMP A, @RW4-d8	AND A.R4	AND A, @RW4-d8	OR A.R4	OR A, @RW4-d8	XOR A.R4	XOR A, @RW4-d8	DBNZ R4,r	DBNZ @ RW4-d8,r
+ 5	ADD A.R5	ADD A, @RW5-d8	SUB A.R5	SUB A, @RW5-d8	ADD A.R5	ADD A, @RW5-d8	CMP A.R5	CMP A, @RW5-d8	AND A.R5	AND A, @RW5-d8	OR A.R5	OR A, @RW5-d8	XOR A.R5	XOR A, @RW5-d8	DBNZ R5,r	DBNZ @ RW5-d8,r
+ 6	ADD A.R6	ADD A, @RW6-d8	SUB A.R6	SUB A, @RW6-d8	ADD A.R6	ADD A, @RW6-d8	CMP A.R6	CMP A, @RW6-d8	AND A.R6	AND A, @RW6-d8	OR A.R6	OR A, @RW6-d8	XOR A.R6	XOR A, @RW6-d8	DBNZ R6,r	DBNZ @ RW6-d8,r
+ 7	ADD A.R7	ADD A, @RW7-d8	SUB A.R7	SUB A, @RW7-d8	ADD A.R7	ADD A, @RW7-d8	CMP A.R7	CMP A, @RW7-d8	AND A.R7	AND A, @RW7-d8	OR A.R7	OR A, @RW7-d8	XOR A.R7	XOR A, @RW7-d8	DBNZ R7,r	DBNZ @ RW7-d8,r
+ 8	ADD A.@RW0	ADD A, @RW0-d16	SUB A.@RW0	SUB A, @RW0-d16	ADD A.@RW0	ADD A, @RW0-d16	CMP A.@RW0	CMP A, @RW0-d16	AND A.@RW0	AND A, @RW0-d16	OR A.@RW0	OR A, @RW0-d16	XOR A.@RW0	XOR A, @RW0-d16	DBNZ @RW0,r	DBNZ @R W0-d16,r
+ 9	ADD A.@RW1	ADD A, @RW1-d16	SUB A.@RW1	SUB A, @RW1-d16	ADD A.@RW1	ADD A, @RW1-d16	CMP A.@RW1	CMP A, @RW1-d16	AND A.@RW1	AND A, @RW1-d16	OR A.@RW1	OR A, @RW1-d16	XOR A.@RW1	XOR A, @RW1-d16	DBNZ @RW1,r	DBNZ @R W1-d16,r
+ A	ADD A.@RW2	ADD A, @RW2-d16	SUB A.@RW2	SUB A, @RW2-d16	ADD A.@RW2	ADD A, @RW2-d16	CMP A.@RW2	CMP A, @RW2-d16	AND A.@RW2	AND A, @RW2-d16	OR A.@RW2	OR A, @RW2-d16	XOR A.@RW2	XOR A, @RW2-d16	DBNZ @RW2,r	DBNZ @R W2-d16,r
+ B	ADD A.@RW3	ADD A, @RW3-d16	SUB A.@RW3	SUB A, @RW3-d16	ADD A.@RW3	ADD A, @RW3-d16	CMP A.@RW3	CMP A, @RW3-d16	AND A.@RW3	AND A, @RW3-d16	OR A.@RW3	OR A, @RW3-d16	XOR A.@RW3	XOR A, @RW3-d16	DBNZ @RW3,r	DBNZ @R W3-d16,r
+ C	ADD A.@RW0+	ADD A, @RW0+RW7	SUB A.@RW0+	SUB A, @RW0+RW7	ADD A.@RW0+	ADD A, @RW0+RW7	CMP A.@RW0+	CMP A, @RW0+RW7	AND A.@RW0+	AND A, @RW0+RW7	OR A.@RW0+	OR A, @RW0+RW7	XOR A.@RW0+	XOR A, @RW0+RW7	DBNZ @RW0+ r	DBNZ @R W0+RW7,r
+ D	ADD A.@RW1+	ADD A, @RW1+RW7	SUB A.@RW1+	SUB A, @RW1+RW7	ADD A.@RW1+	ADD A, @RW1+RW7	CMP A.@RW1+	CMP A, @RW1+RW7	AND A.@RW1+	AND A, @RW1+RW7	OR A.@RW1+	OR A, @RW1+RW7	XOR A.@RW1+	XOR A, @RW1+RW7	DBNZ @RW1+ r	DBNZ @R W1+RW7,r
+ E	ADD A.@RW2+	ADD A, @RW2+PC-d16	SUB A.@RW2+	SUB A, @RW2+PC-d16	ADD A.@RW2+	ADD A, @RW2+PC-d16	CMP A.@RW2+	CMP A, @RW2+PC-d16	AND A.@RW2+	AND A, @RW2+PC-d16	OR A.@RW2+	OR A, @RW2+PC-d16	XOR A.@RW2+	XOR A, @RW2+PC-d16	DBNZ @RW2+ r	DBNZ @ PC-d16,r
+ F	ADD A.@RW3+	ADD A, @RW3+addr16	SUB A.@RW3+	SUB A, @RW3+addr16	ADD A.@RW3+	ADD A, @RW3+addr16	CMP A.@RW3+	CMP A, @RW3+addr16	AND A.@RW3+	AND A, @RW3+addr16	OR A.@RW3+	OR A, @RW3+addr16	XOR A.@RW3+	XOR A, @RW3+addr16	DBNZ @RW3+ r	DBNZ addr16,r

Table B.9-11 ea instruction 6 (first byte = 75H)

	00	10	20	30	40	50	60	70	80	90	A0	E0	D0	C0	F0
+ 0	ADD R0,A	ADD @R, W0+d8,A	SUB R0,A	SUB @R, W0+d8,A	SUBC A, A,R0	SUBC A, @RWO+d8	NEG R0	NEG @RWO+d8	AND R0,A	AND @R, W0+d8,A	OR R0,A	OR @R, W0+d8,A	XOR R0,A	XOR @R, W0+d8,A	NOT R0
+ 1	ADD R1,A	ADD @R, W1+d8,A	SUB R1,A	SUB @R, W1+d8,A	SUBC A, A,R1	SUBC A, @RW1+d8	NEG R1	NEG @RW1+d8	AND R1,A	AND @R, W1+d8,A	OR R1,A	OR @R, W1+d8,A	XOR R1,A	XOR @R, W1+d8,A	NOT R1
+ 2	ADD R2,A	ADD @R, W2+d8,A	SUB R2,A	SUB @R, W2+d8,A	SUBC A, A,R2	SUBC A, @RW2+d8	NEG R2	NEG @RW2+d8	AND R2,A	AND @R, W2+d8,A	OR R2,A	OR @R, W2+d8,A	XOR R2,A	XOR @R, W2+d8,A	NOT R2
+ 3	ADD R3,A	ADD @R, W3+d8,A	SUB R3,A	SUB @R, W3+d8,A	SUBC A, A,R3	SUBC A, @RW3+d8	NEG R3	NEG @RW3+d8	AND R3,A	AND @R, W3+d8,A	OR R3,A	OR @R, W3+d8,A	XOR R3,A	XOR @R, W3+d8,A	NOT R3
+ 4	ADD R4,A	ADD @R, W4+d8,A	SUB R4,A	SUB @R, W4+d8,A	SUBC A, A,R4	SUBC A, @RW4+d8	NEG R4	NEG @RW4+d8	AND R4,A	AND @R, W4+d8,A	OR R4,A	OR @R, W4+d8,A	XOR R4,A	XOR @R, W4+d8,A	NOT R4
+ 5	ADD R5,A	ADD @R, W5+d8,A	SUB R5,A	SUB @R, W5+d8,A	SUBC A, A,R5	SUBC A, @RW5+d8	NEG R5	NEG @RW5+d8	AND R5,A	AND @R, W5+d8,A	OR R5,A	OR @R, W5+d8,A	XOR R5,A	XOR @R, W5+d8,A	NOT R5
+ 6	ADD R6,A	ADD @R, W6+d8,A	SUB R6,A	SUB @R, W6+d8,A	SUBC A, A,R6	SUBC A, @RW6+d8	NEG R6	NEG @RW6+d8	AND R6,A	AND @R, W6+d8,A	OR R6,A	OR @R, W6+d8,A	XOR R6,A	XOR @R, W6+d8,A	NOT R6
+ 7	ADD R7,A	ADD @R, W7+d8,A	SUB R7,A	SUB @R, W7+d8,A	SUBC A, A,R7	SUBC A, @RW7+d8	NEG R7	NEG @RW7+d8	AND R7,A	AND @R, W7+d8,A	OR R7,A	OR @R, W7+d8,A	XOR R7,A	XOR @R, W7+d8,A	NOT R7
+ 8	ADD @RW0,A	ADD @R, W0+d16,A	SUB @RW0,A	SUB @R, W0+d16,A	SUBC A, A,@RW0	SUBC A, @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0,A	AND @R, W0+d16,A	OR @RW0,A	OR @R, W0+d16,A	XOR @RW0,A	XOR @R, W0+d16,A	NOT @RW0
+ 9	ADD @RW1,A	ADD @R, W1+d16,A	SUB @RW1,A	SUB @R, W1+d16,A	SUBC A, A,@RW1	SUBC A, @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1,A	AND @R, W1+d16,A	OR @RW1,A	OR @R, W1+d16,A	XOR @RW1,A	XOR @R, W1+d16,A	NOT @RW1
+ A	ADD @RW2,A	ADD @R, W2+d16,A	SUB @RW2,A	SUB @R, W2+d16,A	SUBC A, A,@RW2	SUBC A, @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2,A	AND @R, W2+d16,A	OR @RW2,A	OR @R, W2+d16,A	XOR @RW2,A	XOR @R, W2+d16,A	NOT @RW2
+ B	ADD @RW3,A	ADD @R, W3+d16,A	SUB @RW3,A	SUB @R, W3+d16,A	SUBC A, A,@RW3	SUBC A, @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3,A	AND @R, W3+d16,A	OR @RW3,A	OR @R, W3+d16,A	XOR @RW3,A	XOR @R, W3+d16,A	NOT @RW3
+ C	ADD @RW0+,A	ADD @R, W0+RW7,A	SUB @RW0+,A	SUB @R, W0+RW7,A	SUBC A, A,@RW0+	SUBC A, @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+,A	AND @R, W0+RW7,A	OR @RW0+,A	OR @R, W0+RW7,A	XOR @RW0+,A	XOR @R, W0+RW7,A	NOT @RW0+
+ D	ADD @RW1+,A	ADD @R, W1+RW7,A	SUB @RW1+,A	SUB @R, W1+RW7,A	SUBC A, A,@RW1+	SUBC A, @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+,A	AND @R, W1+RW7,A	OR @RW1+,A	OR @R, W1+RW7,A	XOR @RW1+,A	XOR @R, W1+RW7,A	NOT @RW1+
+ E	ADD @RW2+,A	ADD @R, C+d16,A	SUB @RW2+,A	SUB @R, C+d16,A	SUBC A, A,@RW2+	SUBC A, @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+,A	AND @R, C+d16,A	OR @RW2+,A	OR @R, C+d16,A	XOR @RW2+,A	XOR @R, C+d16,A	NOT @RW2+
+ F	ADD @RW3+,A	ADD @R, addr16,A	SUB @RW3+,A	SUB @R, addr16,A	SUBC A, A,@RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+,A	AND @R, addr16,A	OR @RW3+,A	OR @R, addr16,A	XOR @RW3+,A	XOR @R, addr16,A	NOT @RW3+

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-d8,r	DWBZ @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-d8,r	DWBZ @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-d8,r	DWBZ @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-d8,r	DWBZ @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-d8,r	DWBZ @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-d8,r	DWBZ @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-d8,r	DWBZ @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-d8,r	DWBZ @RW7-d8,r
+ 8	ADDW A, @RW0	SUBW A, @RW0	SUBW A, @RW0	SUBW A, @RW0	ADDCW A, @RW0	ADDCW A, @RW0	CMPW A, @RW0	CMPW A, @RW0	ANDW A, @RW0	ANDW A, @RW0	ORW A, @RW0	ORW A, @RW0	XORW A, @RW0	XORW A, @RW0	DWBZ @RW0-d16,r	DWBZ @RW0-d16,r
+ 9	ADDW A, @RW1	SUBW A, @RW1	SUBW A, @RW1	SUBW A, @RW1	ADDCW A, @RW1	ADDCW A, @RW1	CMPW A, @RW1	CMPW A, @RW1	ANDW A, @RW1	ANDW A, @RW1	ORW A, @RW1	ORW A, @RW1	XORW A, @RW1	XORW A, @RW1	DWBZ @RW1-d16,r	DWBZ @RW1-d16,r
+ A	ADDW A, @RW2	SUBW A, @RW2	SUBW A, @RW2	SUBW A, @RW2	ADDCW A, @RW2	ADDCW A, @RW2	CMPW A, @RW2	CMPW A, @RW2	ANDW A, @RW2	ANDW A, @RW2	ORW A, @RW2	ORW A, @RW2	XORW A, @RW2	XORW A, @RW2	DWBZ @RW2-d16,r	DWBZ @RW2-d16,r
+ B	ADDW A, @RW3	SUBW A, @RW3	SUBW A, @RW3	SUBW A, @RW3	ADDCW A, @RW3	ADDCW A, @RW3	CMPW A, @RW3	CMPW A, @RW3	ANDW A, @RW3	ANDW A, @RW3	ORW A, @RW3	ORW A, @RW3	XORW A, @RW3	XORW A, @RW3	DWBZ @RW3-d16,r	DWBZ @RW3-d16,r
+ C	ADDW A, @RW0+	SUBW A, @RW0+	SUBW A, @RW0+	SUBW A, @RW0+	ADDCW A, @RW0+	ADDCW A, @RW0+	CMPW A, @RW0+	CMPW A, @RW0+	ANDW A, @RW0+	ANDW A, @RW0+	ORW A, @RW0+	ORW A, @RW0+	XORW A, @RW0+	XORW A, @RW0+	DWBZ @RW0-d16,r	DWBZ @RW0-d16,r
+ D	ADDW A, @RW1+	SUBW A, @RW1+	SUBW A, @RW1+	SUBW A, @RW1+	ADDCW A, @RW1+	ADDCW A, @RW1+	CMPW A, @RW1+	CMPW A, @RW1+	ANDW A, @RW1+	ANDW A, @RW1+	ORW A, @RW1+	ORW A, @RW1+	XORW A, @RW1+	XORW A, @RW1+	DWBZ @RW1-d16,r	DWBZ @RW1-d16,r
+ E	ADDW A, @RW2+	SUBW A, @RW2+	SUBW A, @RW2+	SUBW A, @RW2+	ADDCW A, @RW2+	ADDCW A, @RW2+	CMPW A, @RW2+	CMPW A, @RW2+	ANDW A, @RW2+	ANDW A, @RW2+	ORW A, @RW2+	ORW A, @RW2+	XORW A, @RW2+	XORW A, @RW2+	DWBZ @RW2-d16,r	DWBZ @RW2-d16,r
+ F	ADDW A, @RW3+	SUBW A, @RW3+	SUBW A, @RW3+	SUBW A, @RW3+	ADDCW A, @RW3+	ADDCW A, @RW3+	CMPW A, @RW3+	CMPW A, @RW3+	ANDW A, @RW3+	ANDW A, @RW3+	ORW A, @RW3+	ORW A, @RW3+	XORW A, @RW3+	XORW A, @RW3+	DWBZ @RW3-d16,r	DWBZ @RW3-d16,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 @R0 RW0.A	ADDW @R W0+d8.A	SUBW @R W0+d8.A	SUBW @R W0+d8.A	SUBW @R W0+d8.A	SUBW @R W0+d8.A	NEGW @R W0+d8.A	NEGW @R W0+d8.A	ANDW @R W0+d8.A	ANDW @R W0+d8.A	ORW @R W0.A	ORW @R W0+d8.A	XORW @R W0.A	XORW @R W0+d8.A	NOTW @R W0	NOTW @R W0+d8
+ 1	ADDW @R1 RW1.A	ADDW @R W1+d8.A	SUBW @R W1+d8.A	SUBW @R W1+d8.A	SUBW @R W1+d8.A	SUBW @R W1+d8.A	NEGW @R W1+d8.A	NEGW @R W1+d8.A	ANDW @R W1+d8.A	ANDW @R W1+d8.A	ORW @R RW1.A	ORW @R W1+d8.A	XORW @R RW1.A	XORW @R W1+d8.A	NOTW @R RW1	NOTW @R RW1+d8
+ 2	ADDW @R2 RW2.A	ADDW @R W2+d8.A	SUBW @R W2+d8.A	SUBW @R W2+d8.A	SUBW @R W2+d8.A	SUBW @R W2+d8.A	NEGW @R W2	NEGW @R W2+d8.A	ANDW @R W2.A	ANDW @R W2+d8.A	ORW @R RW2.A	ORW @R W2+d8.A	XORW @R RW2.A	XORW @R W2+d8.A	NOTW @R RW2	NOTW @R RW2+d8
+ 3	ADDW @R3 RW3.A	ADDW @R W3+d8.A	SUBW @R W3+d8.A	SUBW @R W3+d8.A	SUBW @R W3+d8.A	SUBW @R W3+d8.A	NEGW @R RW3	NEGW @R W3+d8.A	ANDW @R RW3.A	ANDW @R W3+d8.A	ORW @R RW3.A	ORW @R W3+d8.A	XORW @R RW3.A	XORW @R W3+d8.A	NOTW @R RW3	NOTW @R RW3+d8
+ 4	ADDW @R4 RW4.A	ADDW @R W4+d8.A	SUBW @R W4+d8.A	SUBW @R W4+d8.A	SUBW @R W4+d8.A	SUBW @R W4+d8.A	NEGW @R RW4	NEGW @R W4+d8.A	ANDW @R RW4.A	ANDW @R W4+d8.A	ORW @R RW4.A	ORW @R W4+d8.A	XORW @R RW4.A	XORW @R W4+d8.A	NOTW @R RW4	NOTW @R RW4+d8
+ 5	ADDW @R5 RW5.A	ADDW @R W5+d8.A	SUBW @R W5+d8.A	SUBW @R W5+d8.A	SUBW @R W5+d8.A	SUBW @R W5+d8.A	NEGW @R RW5	NEGW @R W5+d8.A	ANDW @R RW5.A	ANDW @R W5+d8.A	ORW @R RW5.A	ORW @R W5+d8.A	XORW @R RW5.A	XORW @R W5+d8.A	NOTW @R RW5	NOTW @R RW5+d8
+ 6	ADDW @R6 RW6.A	ADDW @R W6+d8.A	SUBW @R W6+d8.A	SUBW @R W6+d8.A	SUBW @R W6+d8.A	SUBW @R W6+d8.A	NEGW @R RW6	NEGW @R W6+d8.A	ANDW @R RW6.A	ANDW @R W6+d8.A	ORW @R RW6.A	ORW @R W6+d8.A	XORW @R RW6.A	XORW @R W6+d8.A	NOTW @R RW6	NOTW @R RW6+d8
+ 7	ADDW @R7 RW7.A	ADDW @R W7+d8.A	SUBW @R W7+d8.A	SUBW @R W7+d8.A	SUBW @R W7+d8.A	SUBW @R W7+d8.A	NEGW @R RW7	NEGW @R W7+d8.A	ANDW @R RW7.A	ANDW @R W7+d8.A	ORW @R RW7.A	ORW @R W7+d8.A	XORW @R RW7.A	XORW @R W7+d8.A	NOTW @R RW7	NOTW @R RW7+d8
+ 8	ADDW @R0 @RW0.A	ADDW @R W0+d16.A	SUBW @R W0+d16.A	SUBW @R W0+d16.A	SUBW @R W0+d16.A	SUBW @R W0+d16.A	NEGW @R @RW0	NEGW @R @RW0+d16	ANDW @R @RW0.A	ANDW @R W0+d16.A	ORW @R @RW0.A	ORW @R W0+d16.A	XORW @R @RW0.A	XORW @R W0+d16.A	NOTW @R @RW0	NOTW @R @RW0+d16
+ 9	ADDW @R1 @RW1.A	ADDW @R W1+d16.A	SUBW @R W1+d16.A	SUBW @R W1+d16.A	SUBW @R W1+d16.A	SUBW @R W1+d16.A	NEGW @R @RW1	NEGW @R @RW1+d16	ANDW @R @RW1.A	ANDW @R W1+d16.A	ORW @R @RW1.A	ORW @R W1+d16.A	XORW @R @RW1.A	XORW @R W1+d16.A	NOTW @R @RW1	NOTW @R @RW1+d16
+ A	ADDW @R2 @RW2.A	ADDW @R W2+d16.A	SUBW @R W2+d16.A	SUBW @R W2+d16.A	SUBW @R W2+d16.A	SUBW @R W2+d16.A	NEGW @R @RW2	NEGW @R @RW2+d16	ANDW @R @RW2.A	ANDW @R W2+d16.A	ORW @R @RW2.A	ORW @R W2+d16.A	XORW @R @RW2.A	XORW @R W2+d16.A	NOTW @R @RW2	NOTW @R @RW2+d16
+ B	ADDW @R3 @RW3.A	ADDW @R W3+d16.A	SUBW @R W3+d16.A	SUBW @R W3+d16.A	SUBW @R W3+d16.A	SUBW @R W3+d16.A	NEGW @R @RW3	NEGW @R @RW3+d16	ANDW @R @RW3.A	ANDW @R W3+d16.A	ORW @R @RW3.A	ORW @R W3+d16.A	XORW @R @RW3.A	XORW @R W3+d16.A	NOTW @R @RW3	NOTW @R @RW3+d16
+ C	ADDW @R0+ @RW0+.A	ADDW @R W0+RW7.A	SUBW @R W0+RW7.A	SUBW @R W0+RW7.A	SUBW @R W0+RW7.A	SUBW @R W0+RW7.A	NEGW @R @RW0+	NEGW @R @RW0+RW7	ANDW @R @RW0+.A	ANDW @R W0+RW7.A	ORW @R @RW0+.A	ORW @R W0+RW7.A	XORW @R @RW0+.A	XORW @R W0+RW7.A	NOTW @R @RW0+	NOTW @R @RW0+RW7
+ D	ADDW @R1+ @RW1+.A	ADDW @R W1+RW7.A	SUBW @R W1+RW7.A	SUBW @R W1+RW7.A	SUBW @R W1+RW7.A	SUBW @R W1+RW7.A	NEGW @R @RW1+	NEGW @R @RW1+RW7	ANDW @R @RW1+.A	ANDW @R W1+RW7.A	ORW @R @RW1+.A	ORW @R W1+RW7.A	XORW @R @RW1+.A	XORW @R W1+RW7.A	NOTW @R @RW1+	NOTW @R @RW1+RW7
+ E	ADDW @R2+ @RW2+.A	ADDW @P C+d16.A	SUBW @P C+d16.A	SUBW @P C+d16.A	SUBW @P C+d16.A	SUBW @P C+d16.A	NEGW @R @RW2+	NEGW @R @PC+d16	ANDW @R @RW2+.A	ANDW @P C+d16.A	ORW @R @RW2+.A	ORW @P C+d16.A	XORW @R @RW2+.A	XORW @P C+d16.A	NOTW @R @RW2+	NOTW @R @PC+d16
+ F	ADDW @R3+ @RW3+.A	ADDW @R addr16.A	SUBW @R addr16.A	SUBW @R addr16.A	SUBW @R addr16.A	SUBW @R addr16.A	NEGW @R @RW3+	NEGW @R addr16	ANDW @R @RW3+.A	ANDW @R addr16.A	ORW @R @RW3+.A	ORW @R addr16.A	XORW @R @RW3+.A	XORW @R addr16.A	NOTW @R @RW3+	NOTW @R addr16

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	MULU A @RW0-d8	MULUW A.RW0	MULUW A @RW0-d8	MUL A.R0	MUL A @RW0-d8	MULW A.RW0	MULW A @RW0-d8	DIVU A.R0	DIVU A @RW0-d8	DIVUW A.RW0	DIVUW A @RW0-d8	DIV A.R0	DIV A @RW0-d8	DIVW A.RW0	DIVW A @RW0-d8
+ 1	MULU A.R1	MULU A @RW1-d8	MULUW A.RW1	MULUW A @RW1-d8	MUL A.R1	MUL A @RW1-d8	MULW A.RW1	MULW A @RW1-d8	DIVU A.R1	DIVU A @RW1-d8	DIVUW A.RW1	DIVUW A @RW1-d8	DIV A.R1	DIV A @RW1-d8	DIVW A.RW1	DIVW A @RW1-d8
+ 2	MULU A.R2	MULU A @RW2-d8	MULUW A.RW2	MULUW A @RW2-d8	MUL A.R2	MUL A @RW2-d8	MULW A.RW2	MULW A @RW2-d8	DIVU A.R2	DIVU A @RW2-d8	DIVUW A.RW2	DIVUW A @RW2-d8	DIV A.R2	DIV A @RW2-d8	DIVW A.RW2	DIVW A @RW2-d8
+ 3	MULU A.R3	MULU A @RW3-d8	MULUW A.RW3	MULUW A @RW3-d8	MUL A.R3	MUL A @RW3-d8	MULW A.RW3	MULW A @RW3-d8	DIVU A.R3	DIVU A @RW3-d8	DIVUW A.RW3	DIVUW A @RW3-d8	DIV A.R3	DIV A @RW3-d8	DIVW A.RW3	DIVW A @RW3-d8
+ 4	MULU A.R4	MULU A @RW4-d8	MULUW A.RW4	MULUW A @RW4-d8	MUL A.R4	MUL A @RW4-d8	MULW A.RW4	MULW A @RW4-d8	DIVU A.R4	DIVU A @RW4-d8	DIVUW A.RW4	DIVUW A @RW4-d8	DIV A.R4	DIV A @RW4-d8	DIVW A.RW4	DIVW A @RW4-d8
+ 5	MULU A.R5	MULU A @RW5-d8	MULUW A.RW5	MULUW A @RW5-d8	MUL A.R5	MUL A @RW5-d8	MULW A.RW5	MULW A @RW5-d8	DIVU A.R5	DIVU A @RW5-d8	DIVUW A.RW5	DIVUW A @RW5-d8	DIV A.R5	DIV A @RW5-d8	DIVW A.RW5	DIVW A @RW5-d8
+ 6	MULU A.R6	MULU A @RW6-d8	MULUW A.RW6	MULUW A @RW6-d8	MUL A.R6	MUL A @RW6-d8	MULW A.RW6	MULW A @RW6-d8	DIVU A.R6	DIVU A @RW6-d8	DIVUW A.RW6	DIVUW A @RW6-d8	DIV A.R6	DIV A @RW6-d8	DIVW A.RW6	DIVW A @RW6-d8
+ 7	MULU A.R7	MULU A @RW7-d8	MULUW A.RW7	MULUW A @RW7-d8	MUL A.R7	MUL A @RW7-d8	MULW A.RW7	MULW A @RW7-d8	DIVU A.R7	DIVU A @RW7-d8	DIVUW A.RW7	DIVUW A @RW7-d8	DIV A.R7	DIV A @RW7-d8	DIVW A.RW7	DIVW A @RW7-d8
+ 8	MULU A.@RW0	MULU A @RW0-d16	MULUW A.@RW0	MULUW A @RW0-d16	MUL A.@RW0	MUL A @RW0-d16	MULW A.@RW0	MULW A @RW0-d16	DIVU A.@RW0	DIVU A @RW0-d16	DIVUW A.@RW0	DIVUW A @RW0-d16	DIV A.@RW0	DIV A @RW0-d16	DIVW A.@RW0	DIVW A @RW0-d16
+ 9	MULU A.@RW1	MULU A @RW1-d16	MULUW A.@RW1	MULUW A @RW1-d16	MUL A.@RW1	MUL A @RW1-d16	MULW A.@RW1	MULW A @RW1-d16	DIVU A.@RW1	DIVU A @RW1-d16	DIVUW A.@RW1	DIVUW A @RW1-d16	DIV A.@RW1	DIV A @RW1-d16	DIVW A.@RW1	DIVW A @RW1-d16
+ A	MULU A.@RW2	MULU A @RW2-d16	MULUW A.@RW2	MULUW A @RW2-d16	MUL A.@RW2	MUL A @RW2-d16	MULW A.@RW2	MULW A @RW2-d16	DIVU A.@RW2	DIVU A @RW2-d16	DIVUW A.@RW2	DIVUW A @RW2-d16	DIV A.@RW2	DIV A @RW2-d16	DIVW A.@RW2	DIVW A @RW2-d16
+ B	MULU A.@RW3	MULU A @RW3-d16	MULUW A.@RW3	MULUW A @RW3-d16	MUL A.@RW3	MUL A @RW3-d16	MULW A.@RW3	MULW A @RW3-d16	DIVU A.@RW3	DIVU A @RW3-d16	DIVUW A.@RW3	DIVUW A @RW3-d16	DIV A.@RW3	DIV A @RW3-d16	DIVW A.@RW3	DIVW A @RW3-d16
+ C	MULU A.@RW0+	MULU A @RW0-RW7	MULUW A.@RW0+	MULUW A @RW0-RW7	MUL A.@RW0+	MUL A @RW0-RW7	MULW A.@RW0+	MULW A @RW0-RW7	DIVU A.@RW0+	DIVU A @RW0-RW7	DIVUW A.@RW0+	DIVUW A @RW0-RW7	DIV A.@RW0+	DIV A @RW0-RW7	DIVW A.@RW0+	DIVW A @RW0-RW7
+ D	MULU A.@RW1+	MULU A @RW1-RW7	MULUW A.@RW1+	MULUW A @RW1-RW7	MUL A.@RW1+	MUL A @RW1-RW7	MULW A.@RW1+	MULW A @RW1-RW7	DIVU A.@RW1+	DIVU A @RW1-RW7	DIVUW A.@RW1+	DIVUW A @RW1-RW7	DIV A.@RW1+	DIV A @RW1-RW7	DIVW A.@RW1+	DIVW A @RW1-RW7
+ E	MULU A.@RW2+	MULU A @PC-d16	MULUW A.@RW2+	MULUW A @PC-d16	MUL A.@RW2+	MUL A @PC-d16	MULW A.@RW2+	MULW A @PC-d16	DIVU A.@RW2+	DIVU A @PC-d16	DIVUW A.@RW2+	DIVUW A @PC-d16	DIV A.@RW2+	DIV A @PC-d16	DIVW A.@RW2+	DIVW A @PC-d16
+ F	MULU A.@RW3+	MULU A addr16	MULUW A.@RW3+	MULUW A addr16	MUL A.@RW3+	MUL A addr16	MULW A.@RW3+	MULW A addr16	DIVU A.@RW3+	DIVU A addr16	DIVUW A.@RW3+	DIVUW A addr16	DIV A.@RW3+	DIV A addr16	DIVW A.@RW3+	DIVW A addr16

Table B.9-15 MOVEA RWi, ea instruction (first byte = 79H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVEA R0,R0	MOVEA @R0-d8	MOVEA R1,R0	MOVEA @R0-d8	MOVEA R2,R0	MOVEA @R0-d8	MOVEA R3,R0	MOVEA @R0-d8	MOVEA R4,R0	MOVEA @R0-d8	MOVEA R5,R0	MOVEA @R0-d8	MOVEA R6,R0	MOVEA @R0-d8	MOVEA R7,R0	MOVEA @R0-d8
+ 1	MOVEA R0,R1	MOVEA @R1-d8	MOVEA R1,R1	MOVEA @R1-d8	MOVEA R2,R1	MOVEA @R1-d8	MOVEA R3,R1	MOVEA @R1-d8	MOVEA R4,R1	MOVEA @R1-d8	MOVEA R5,R1	MOVEA @R1-d8	MOVEA R6,R1	MOVEA @R1-d8	MOVEA R7,R1	MOVEA @R1-d8
+ 2	MOVEA R0,R2	MOVEA @R2-d8	MOVEA R1,R2	MOVEA @R2-d8	MOVEA R2,R2	MOVEA @R2-d8	MOVEA R3,R2	MOVEA @R2-d8	MOVEA R4,R2	MOVEA @R2-d8	MOVEA R5,R2	MOVEA @R2-d8	MOVEA R6,R2	MOVEA @R2-d8	MOVEA R7,R2	MOVEA @R2-d8
+ 3	MOVEA R0,R3	MOVEA @R3-d8	MOVEA R1,R3	MOVEA @R3-d8	MOVEA R2,R3	MOVEA @R3-d8	MOVEA R3,R3	MOVEA @R3-d8	MOVEA R4,R3	MOVEA @R3-d8	MOVEA R5,R3	MOVEA @R3-d8	MOVEA R6,R3	MOVEA @R3-d8	MOVEA R7,R3	MOVEA @R3-d8
+ 4	MOVEA R0,R4	MOVEA @R4-d8	MOVEA R1,R4	MOVEA @R4-d8	MOVEA R2,R4	MOVEA @R4-d8	MOVEA R3,R4	MOVEA @R4-d8	MOVEA R4,R4	MOVEA @R4-d8	MOVEA R5,R4	MOVEA @R4-d8	MOVEA R6,R4	MOVEA @R4-d8	MOVEA R7,R4	MOVEA @R4-d8
+ 5	MOVEA R0,R5	MOVEA @R5-d8	MOVEA R1,R5	MOVEA @R5-d8	MOVEA R2,R5	MOVEA @R5-d8	MOVEA R3,R5	MOVEA @R5-d8	MOVEA R4,R5	MOVEA @R5-d8	MOVEA R5,R5	MOVEA @R5-d8	MOVEA R6,R5	MOVEA @R5-d8	MOVEA R7,R5	MOVEA @R5-d8
+ 6	MOVEA R0,R6	MOVEA @R6-d8	MOVEA R1,R6	MOVEA @R6-d8	MOVEA R2,R6	MOVEA @R6-d8	MOVEA R3,R6	MOVEA @R6-d8	MOVEA R4,R6	MOVEA @R6-d8	MOVEA R5,R6	MOVEA @R6-d8	MOVEA R6,R6	MOVEA @R6-d8	MOVEA R7,R6	MOVEA @R6-d8
+ 7	MOVEA R0,R7	MOVEA @R7-d8	MOVEA R1,R7	MOVEA @R7-d8	MOVEA R2,R7	MOVEA @R7-d8	MOVEA R3,R7	MOVEA @R7-d8	MOVEA R4,R7	MOVEA @R7-d8	MOVEA R5,R7	MOVEA @R7-d8	MOVEA R6,R7	MOVEA @R7-d8	MOVEA R7,R7	MOVEA @R7-d8
+ 8	MOVEA R0,R0	MOVEA @R0-d16	MOVEA R1,R0	MOVEA @R0-d16	MOVEA R2,R0	MOVEA @R0-d16	MOVEA R3,R0	MOVEA @R0-d16	MOVEA R4,R0	MOVEA @R0-d16	MOVEA R5,R0	MOVEA @R0-d16	MOVEA R6,R0	MOVEA @R0-d16	MOVEA R7,R0	MOVEA @R0-d16
+ 9	MOVEA R0,R1	MOVEA @R1-d16	MOVEA R1,R1	MOVEA @R1-d16	MOVEA R2,R1	MOVEA @R1-d16	MOVEA R3,R1	MOVEA @R1-d16	MOVEA R4,R1	MOVEA @R1-d16	MOVEA R5,R1	MOVEA @R1-d16	MOVEA R6,R1	MOVEA @R1-d16	MOVEA R7,R1	MOVEA @R1-d16
+ A	MOVEA R0,R2	MOVEA @R2-d16	MOVEA R1,R2	MOVEA @R2-d16	MOVEA R2,R2	MOVEA @R2-d16	MOVEA R3,R2	MOVEA @R2-d16	MOVEA R4,R2	MOVEA @R2-d16	MOVEA R5,R2	MOVEA @R2-d16	MOVEA R6,R2	MOVEA @R2-d16	MOVEA R7,R2	MOVEA @R2-d16
+ B	MOVEA R0,R3	MOVEA @R3-d16	MOVEA R1,R3	MOVEA @R3-d16	MOVEA R2,R3	MOVEA @R3-d16	MOVEA R3,R3	MOVEA @R3-d16	MOVEA R4,R3	MOVEA @R3-d16	MOVEA R5,R3	MOVEA @R3-d16	MOVEA R6,R3	MOVEA @R3-d16	MOVEA R7,R3	MOVEA @R3-d16
+ C	MOVEA R0,R4	MOVEA @R4-d16	MOVEA R1,R4	MOVEA @R4-d16	MOVEA R2,R4	MOVEA @R4-d16	MOVEA R3,R4	MOVEA @R4-d16	MOVEA R4,R4	MOVEA @R4-d16	MOVEA R5,R4	MOVEA @R4-d16	MOVEA R6,R4	MOVEA @R4-d16	MOVEA R7,R4	MOVEA @R4-d16
+ D	MOVEA R0,R5	MOVEA @R5-d16	MOVEA R1,R5	MOVEA @R5-d16	MOVEA R2,R5	MOVEA @R5-d16	MOVEA R3,R5	MOVEA @R5-d16	MOVEA R4,R5	MOVEA @R5-d16	MOVEA R5,R5	MOVEA @R5-d16	MOVEA R6,R5	MOVEA @R5-d16	MOVEA R7,R5	MOVEA @R5-d16
+ E	MOVEA R0,R6	MOVEA @R6-d16	MOVEA R1,R6	MOVEA @R6-d16	MOVEA R2,R6	MOVEA @R6-d16	MOVEA R3,R6	MOVEA @R6-d16	MOVEA R4,R6	MOVEA @R6-d16	MOVEA R5,R6	MOVEA @R6-d16	MOVEA R6,R6	MOVEA @R6-d16	MOVEA R7,R6	MOVEA @R6-d16
+ F	MOVEA R0,R3+	MOVEA @R3+ .addr16	MOVEA R1,R3+	MOVEA @R3+ .addr16	MOVEA R2,R3+	MOVEA @R3+ .addr16	MOVEA R3,R3+	MOVEA @R3+ .addr16	MOVEA R4,R3+	MOVEA @R3+ .addr16	MOVEA R5,R3+	MOVEA @R3+ .addr16	MOVEA R6,R3+	MOVEA @R3+ .addr16	MOVEA R7,R3+	MOVEA @R3+ .addr16

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 @RW RW0,RW0	MOVW @RW 0+8:RW0	MOVW @RW RW0,RW1	MOVW @RW 0+8:RW1	MOVW @RW RW0,RW2	MOVW @RW 0+8:RW2	MOVW @RW RW0,RW3	MOVW @RW 0+8:RW3	MOVW @RW RW0,RW4	MOVW @RW 0+8:RW4	MOVW @RW RW0,RW5	MOVW @RW 0+8:RW5	MOVW @RW RW0,RW6	MOVW @RW 0+8:RW6	MOVW @RW RW0,RW7	MOVW @RW 0+8:RW7
+ 1	MOVW @RW RW1,RW0	MOVW @RW 1+8:RW0	MOVW @RW RW1,RW1	MOVW @RW 1+8:RW1	MOVW @RW RW1,RW2	MOVW @RW 1+8:RW2	MOVW @RW RW1,RW3	MOVW @RW 1+8:RW3	MOVW @RW RW1,RW4	MOVW @RW 1+8:RW4	MOVW @RW RW1,RW5	MOVW @RW 1+8:RW5	MOVW @RW RW1,RW6	MOVW @RW 1+8:RW6	MOVW @RW RW1,RW7	MOVW @RW 1+8:RW7
+ 2	MOVW @RW RW2,RW0	MOVW @RW 2+8:RW0	MOVW @RW RW2,RW1	MOVW @RW 2+8:RW1	MOVW @RW RW2,RW2	MOVW @RW 2+8:RW2	MOVW @RW RW2,RW3	MOVW @RW 2+8:RW3	MOVW @RW RW2,RW4	MOVW @RW 2+8:RW4	MOVW @RW RW2,RW5	MOVW @RW 2+8:RW5	MOVW @RW RW2,RW6	MOVW @RW 2+8:RW6	MOVW @RW RW2,RW7	MOVW @RW 2+8:RW7
+ 3	MOVW @RW RW3,RW0	MOVW @RW 3+8:RW0	MOVW @RW RW3,RW1	MOVW @RW 3+8:RW1	MOVW @RW RW3,RW2	MOVW @RW 3+8:RW2	MOVW @RW RW3,RW3	MOVW @RW 3+8:RW3	MOVW @RW RW3,RW4	MOVW @RW 3+8:RW4	MOVW @RW RW3,RW5	MOVW @RW 3+8:RW5	MOVW @RW RW3,RW6	MOVW @RW 3+8:RW6	MOVW @RW RW3,RW7	MOVW @RW 3+8:RW7
+ 4	MOVW @RW RW4,RW0	MOVW @RW 4+8:RW0	MOVW @RW RW4,RW1	MOVW @RW 4+8:RW1	MOVW @RW RW4,RW2	MOVW @RW 4+8:RW2	MOVW @RW RW4,RW3	MOVW @RW 4+8:RW3	MOVW @RW RW4,RW4	MOVW @RW 4+8:RW4	MOVW @RW RW4,RW5	MOVW @RW 4+8:RW5	MOVW @RW RW4,RW6	MOVW @RW 4+8:RW6	MOVW @RW RW4,RW7	MOVW @RW 4+8:RW7
+ 5	MOVW @RW RW5,RW0	MOVW @RW 5+8:RW0	MOVW @RW RW5,RW1	MOVW @RW 5+8:RW1	MOVW @RW RW5,RW2	MOVW @RW 5+8:RW2	MOVW @RW RW5,RW3	MOVW @RW 5+8:RW3	MOVW @RW RW5,RW4	MOVW @RW 5+8:RW4	MOVW @RW RW5,RW5	MOVW @RW 5+8:RW5	MOVW @RW RW5,RW6	MOVW @RW 5+8:RW6	MOVW @RW RW5,RW7	MOVW @RW 5+8:RW7
+ 6	MOVW @RW RW6,RW0	MOVW @RW 6+8:RW0	MOVW @RW RW6,RW1	MOVW @RW 6+8:RW1	MOVW @RW RW6,RW2	MOVW @RW 6+8:RW2	MOVW @RW RW6,RW3	MOVW @RW 6+8:RW3	MOVW @RW RW6,RW4	MOVW @RW 6+8:RW4	MOVW @RW RW6,RW5	MOVW @RW 6+8:RW5	MOVW @RW RW6,RW6	MOVW @RW 6+8:RW6	MOVW @RW RW6,RW7	MOVW @RW 6+8:RW7
+ 7	MOVW @RW RW7,RW0	MOVW @RW 7+8:RW0	MOVW @RW RW7,RW1	MOVW @RW 7+8:RW1	MOVW @RW RW7,RW2	MOVW @RW 7+8:RW2	MOVW @RW RW7,RW3	MOVW @RW 7+8:RW3	MOVW @RW RW7,RW4	MOVW @RW 7+8:RW4	MOVW @RW RW7,RW5	MOVW @RW 7+8:RW5	MOVW @RW RW7,RW6	MOVW @RW 7+8:RW6	MOVW @RW RW7,RW7	MOVW @RW 7+8:RW7
+ 8	MOVW @RW0,RW0	MOVW @RW0 +d16:RW0	MOVW @RW0 RW0,RW1	MOVW @RW0 +d16:RW1	MOVW @RW0 RW0,RW2	MOVW @RW0 +d16:RW2	MOVW @RW0 RW0,RW3	MOVW @RW0 +d16:RW3	MOVW @RW0 RW0,RW4	MOVW @RW0 +d16:RW4	MOVW @RW0 RW0,RW5	MOVW @RW0 +d16:RW5	MOVW @RW0 RW0,RW6	MOVW @RW0 +d16:RW6	MOVW @RW0 RW0,RW7	MOVW @RW0 +d16:RW7
+ 9	MOVW @RW1,RW0	MOVW @RW1 +d16:RW0	MOVW @RW1 RW1,RW1	MOVW @RW1 +d16:RW1	MOVW @RW1 RW1,RW2	MOVW @RW1 +d16:RW2	MOVW @RW1 RW1,RW3	MOVW @RW1 +d16:RW3	MOVW @RW1 RW1,RW4	MOVW @RW1 +d16:RW4	MOVW @RW1 RW1,RW5	MOVW @RW1 +d16:RW5	MOVW @RW1 RW1,RW6	MOVW @RW1 +d16:RW6	MOVW @RW1 RW1,RW7	MOVW @RW1 +d16:RW7
+ A	MOVW @RW2,RW0	MOVW @RW2 +d16:RW0	MOVW @RW2 RW2,RW1	MOVW @RW2 +d16:RW1	MOVW @RW2 RW2,RW2	MOVW @RW2 +d16:RW2	MOVW @RW2 RW2,RW3	MOVW @RW2 +d16:RW3	MOVW @RW2 RW2,RW4	MOVW @RW2 +d16:RW4	MOVW @RW2 RW2,RW5	MOVW @RW2 +d16:RW5	MOVW @RW2 RW2,RW6	MOVW @RW2 +d16:RW6	MOVW @RW2 RW2,RW7	MOVW @RW2 +d16:RW7
+ B	MOVW @RW3,RW0	MOVW @RW3 +d16:RW0	MOVW @RW3 RW3,RW1	MOVW @RW3 +d16:RW1	MOVW @RW3 RW3,RW2	MOVW @RW3 +d16:RW2	MOVW @RW3 RW3,RW3	MOVW @RW3 +d16:RW3	MOVW @RW3 RW3,RW4	MOVW @RW3 +d16:RW4	MOVW @RW3 RW3,RW5	MOVW @RW3 +d16:RW5	MOVW @RW3 RW3,RW6	MOVW @RW3 +d16:RW6	MOVW @RW3 RW3,RW7	MOVW @RW3 +d16:RW7
+ C	MOVW @RW0+,RW0	MOVW @RW0+ +RW7:RW0	MOVW @RW0+ RW0+,RW1	MOVW @RW0+ +RW7:RW1	MOVW @RW0+ RW0+,RW2	MOVW @RW0+ +RW7:RW2	MOVW @RW0+ RW0+,RW3	MOVW @RW0+ +RW7:RW3	MOVW @RW0+ RW0+,RW4	MOVW @RW0+ +RW7:RW4	MOVW @RW0+ RW0+,RW5	MOVW @RW0+ +RW7:RW5	MOVW @RW0+ RW0+,RW6	MOVW @RW0+ +RW7:RW6	MOVW @RW0+ RW0+,RW7	MOVW @RW0+ +RW7:RW7
+ D	MOVW @RW1+,RW0	MOVW @RW1+ +RW7:RW0	MOVW @RW1+ RW1+,RW1	MOVW @RW1+ +RW7:RW1	MOVW @RW1+ RW1+,RW2	MOVW @RW1+ +RW7:RW2	MOVW @RW1+ RW1+,RW3	MOVW @RW1+ +RW7:RW3	MOVW @RW1+ RW1+,RW4	MOVW @RW1+ +RW7:RW4	MOVW @RW1+ RW1+,RW5	MOVW @RW1+ +RW7:RW5	MOVW @RW1+ RW1+,RW6	MOVW @RW1+ +RW7:RW6	MOVW @RW1+ RW1+,RW7	MOVW @RW1+ +RW7:RW7
+ E	MOVW @RW2+,RW0	MOVW @RW2+ d16:RW0	MOVW @RW2+ RW2+,RW1	MOVW @RW2+ d16:RW1	MOVW @RW2+ RW2+,RW2	MOVW @RW2+ d16:RW2	MOVW @RW2+ RW2+,RW3	MOVW @RW2+ d16:RW3	MOVW @RW2+ RW2+,RW4	MOVW @RW2+ d16:RW4	MOVW @RW2+ RW2+,RW5	MOVW @RW2+ d16:RW5	MOVW @RW2+ RW2+,RW6	MOVW @RW2+ d16:RW6	MOVW @RW2+ RW2+,RW7	MOVW @RW2+ d16:RW7
+ F	MOVW @RW3+,RW0	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr	MOVW @RW3+ addr

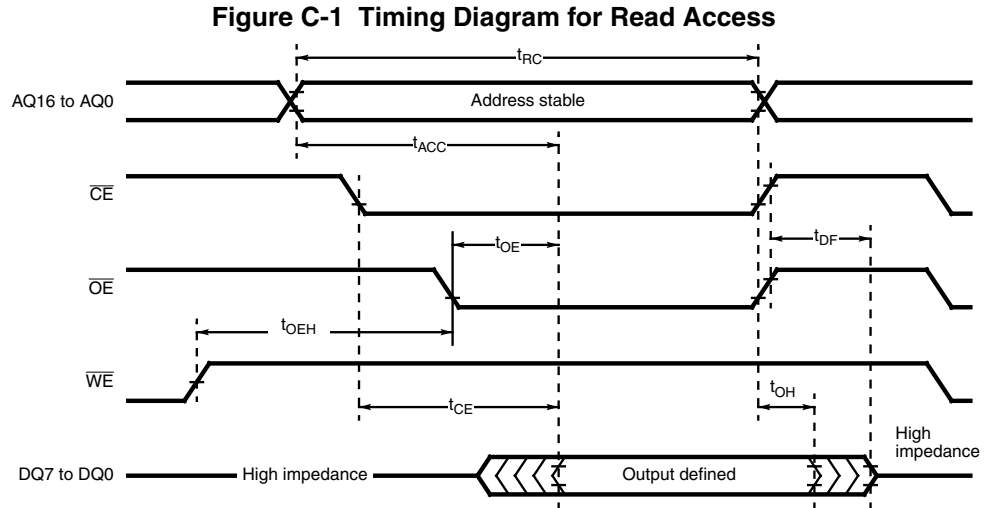
Table B.9-20 XCH Ri, ea instruction (first byte = 7E_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCH R0,R0 : @RW0+d8	XCH R0, @RW0+d8	XCH R1, R1,R0 : @RW0+d8	XCH R1, @RW0+d8	XCH R2, R2,R0 : @RW0+d8	XCH R2, @RW0+d8	XCH R3, R3,R0 : @RW0+d8	XCH R3, @RW0+d8	XCH R4, R4,R0 : @RW0+d8	XCH R4, @RW0+d8	XCH R5, R5,R0 : @RW0+d8	XCH R5, @RW0+d8	XCH R6, R6,R0 : @RW0+d8	XCH R6, @RW0+d8	XCH R7, R7,R0 : @RW0+d8	XCH R7, @RW0+d8
+ 1	XCH R0,R1 : @RW1+d8	XCH R0, @RW1+d8	XCH R1, R1,R1 : @RW1+d8	XCH R1, @RW1+d8	XCH R2, R2,R1 : @RW1+d8	XCH R2, @RW1+d8	XCH R3, R3,R1 : @RW1+d8	XCH R3, @RW1+d8	XCH R4, R4,R1 : @RW1+d8	XCH R4, @RW1+d8	XCH R5, R5,R1 : @RW1+d8	XCH R5, @RW1+d8	XCH R6, R6,R1 : @RW1+d8	XCH R6, @RW1+d8	XCH R7, R7,R1 : @RW1+d8	XCH R7, @RW1+d8
+ 2	XCH R0,R2 : @RW2+d8	XCH R0, @RW2+d8	XCH R1, R1,R2 : @RW2+d8	XCH R1, @RW2+d8	XCH R2, R2,R2 : @RW2+d8	XCH R2, @RW2+d8	XCH R3, R3,R2 : @RW2+d8	XCH R3, @RW2+d8	XCH R4, R4,R2 : @RW2+d8	XCH R4, @RW2+d8	XCH R5, R5,R2 : @RW2+d8	XCH R5, @RW2+d8	XCH R6, R6,R2 : @RW2+d8	XCH R6, @RW2+d8	XCH R7, R7,R2 : @RW2+d8	XCH R7, @RW2+d8
+ 3	XCH R0,R3 : @RW3+d8	XCH R0, @RW3+d8	XCH R1, R1,R3 : @RW3+d8	XCH R1, @RW3+d8	XCH R2, R2,R3 : @RW3+d8	XCH R2, @RW3+d8	XCH R3, R3,R3 : @RW3+d8	XCH R3, @RW3+d8	XCH R4, R4,R3 : @RW3+d8	XCH R4, @RW3+d8	XCH R5, R5,R3 : @RW3+d8	XCH R5, @RW3+d8	XCH R6, R6,R3 : @RW3+d8	XCH R6, @RW3+d8	XCH R7, R7,R3 : @RW3+d8	XCH R7, @RW3+d8
+ 4	XCH R0,R4 : @RW4+d8	XCH R0, @RW4+d8	XCH R1, R1,R4 : @RW4+d8	XCH R1, @RW4+d8	XCH R2, R2,R4 : @RW4+d8	XCH R2, @RW4+d8	XCH R3, R3,R4 : @RW4+d8	XCH R3, @RW4+d8	XCH R4, R4,R4 : @RW4+d8	XCH R4, @RW4+d8	XCH R5, R5,R4 : @RW4+d8	XCH R5, @RW4+d8	XCH R6, R6,R4 : @RW4+d8	XCH R6, @RW4+d8	XCH R7, R7,R4 : @RW4+d8	XCH R7, @RW4+d8
+ 5	XCH R0,R5 : @RW5+d8	XCH R0, @RW5+d8	XCH R1, R1,R5 : @RW5+d8	XCH R1, @RW5+d8	XCH R2, R2,R5 : @RW5+d8	XCH R2, @RW5+d8	XCH R3, R3,R5 : @RW5+d8	XCH R3, @RW5+d8	XCH R4, R4,R5 : @RW5+d8	XCH R4, @RW5+d8	XCH R5, R5,R5 : @RW5+d8	XCH R5, @RW5+d8	XCH R6, R6,R5 : @RW5+d8	XCH R6, @RW5+d8	XCH R7, R7,R5 : @RW5+d8	XCH R7, @RW5+d8
+ 6	XCH R0,R6 : @RW6+d8	XCH R0, @RW6+d8	XCH R1, R1,R6 : @RW6+d8	XCH R1, @RW6+d8	XCH R2, R2,R6 : @RW6+d8	XCH R2, @RW6+d8	XCH R3, R3,R6 : @RW6+d8	XCH R3, @RW6+d8	XCH R4, R4,R6 : @RW6+d8	XCH R4, @RW6+d8	XCH R5, R5,R6 : @RW6+d8	XCH R5, @RW6+d8	XCH R6, R6,R6 : @RW6+d8	XCH R6, @RW6+d8	XCH R7, R7,R6 : @RW6+d8	XCH R7, @RW6+d8
+ 7	XCH R0,R7 : @RW7+d8	XCH R0, @RW7+d8	XCH R1, R1,R7 : @RW7+d8	XCH R1, @RW7+d8	XCH R2, R2,R7 : @RW7+d8	XCH R2, @RW7+d8	XCH R3, R3,R7 : @RW7+d8	XCH R3, @RW7+d8	XCH R4, R4,R7 : @RW7+d8	XCH R4, @RW7+d8	XCH R5, R5,R7 : @RW7+d8	XCH R5, @RW7+d8	XCH R6, R6,R7 : @RW7+d8	XCH R6, @RW7+d8	XCH R7, R7,R7 : @RW7+d8	XCH R7, @RW7+d8
+ 8	XCH R0,R0 : @RW0+dt16	XCH R0, @RW0+dt16	XCH R1, R1,R0 : @RW0+dt16	XCH R1, @RW0+dt16	XCH R2, R2,R0 : @RW0+dt16	XCH R2, @RW0+dt16	XCH R3, R3,R0 : @RW0+dt16	XCH R3, @RW0+dt16	XCH R4, R4,R0 : @RW0+dt16	XCH R4, @RW0+dt16	XCH R5, R5,R0 : @RW0+dt16	XCH R5, @RW0+dt16	XCH R6, R6,R0 : @RW0+dt16	XCH R6, @RW0+dt16	XCH R7, R7,R0 : @RW0+dt16	XCH R7, @RW0+dt16
+ 9	XCH R0,R0 : @RW1+dt16	XCH R0, @RW1+dt16	XCH R1, R1,R0 : @RW1+dt16	XCH R1, @RW1+dt16	XCH R2, R2,R0 : @RW1+dt16	XCH R2, @RW1+dt16	XCH R3, R3,R0 : @RW1+dt16	XCH R3, @RW1+dt16	XCH R4, R4,R0 : @RW1+dt16	XCH R4, @RW1+dt16	XCH R5, R5,R0 : @RW1+dt16	XCH R5, @RW1+dt16	XCH R6, R6,R0 : @RW1+dt16	XCH R6, @RW1+dt16	XCH R7, R7,R0 : @RW1+dt16	XCH R7, @RW1+dt16
+ A	XCH R0,R0 : W2-dt16.A	XCH R0, W2-dt16.A	XCH R1, R1,R0 : W2-dt16.A	XCH R1, W2-dt16.A	XCH R2, R2,R0 : W2-dt16.A	XCH R2, W2-dt16.A	XCH R3, R3,R0 : W2-dt16.A	XCH R3, W2-dt16.A	XCH R4, R4,R0 : W2-dt16.A	XCH R4, W2-dt16.A	XCH R5, R5,R0 : W2-dt16.A	XCH R5, W2-dt16.A	XCH R6, R6,R0 : W2-dt16.A	XCH R6, W2-dt16.A	XCH R7, R7,R0 : W2-dt16.A	XCH R7, W2-dt16.A
+ B	XCH R0,R0 : @RW3+dt16	XCH R0, @RW3+dt16	XCH R1, R1,R0 : @RW3+dt16	XCH R1, @RW3+dt16	XCH R2, R2,R0 : @RW3+dt16	XCH R2, @RW3+dt16	XCH R3, R3,R0 : @RW3+dt16	XCH R3, @RW3+dt16	XCH R4, R4,R0 : @RW3+dt16	XCH R4, @RW3+dt16	XCH R5, R5,R0 : @RW3+dt16	XCH R5, @RW3+dt16	XCH R6, R6,R0 : @RW3+dt16	XCH R6, @RW3+dt16	XCH R7, R7,R0 : @RW3+dt16	XCH R7, @RW3+dt16
+ C	XCH R0,R0 : @RW0+RW7	XCH R0, @RW0+RW7	XCH R1, R1,R0 : @RW0+RW7	XCH R1, @RW0+RW7	XCH R2, R2,R0 : @RW0+RW7	XCH R2, @RW0+RW7	XCH R3, R3,R0 : @RW0+RW7	XCH R3, @RW0+RW7	XCH R4, R4,R0 : @RW0+RW7	XCH R4, @RW0+RW7	XCH R5, R5,R0 : @RW0+RW7	XCH R5, @RW0+RW7	XCH R6, R6,R0 : @RW0+RW7	XCH R6, @RW0+RW7	XCH R7, R7,R0 : @RW0+RW7	XCH R7, @RW0+RW7
+ D	XCH R0,R0 : @RW1+RW7	XCH R0, @RW1+RW7	XCH R1, R1,R0 : @RW1+RW7	XCH R1, @RW1+RW7	XCH R2, R2,R0 : @RW1+RW7	XCH R2, @RW1+RW7	XCH R3, R3,R0 : @RW1+RW7	XCH R3, @RW1+RW7	XCH R4, R4,R0 : @RW1+RW7	XCH R4, @RW1+RW7	XCH R5, R5,R0 : @RW1+RW7	XCH R5, @RW1+RW7	XCH R6, R6,R0 : @RW1+RW7	XCH R6, @RW1+RW7	XCH R7, R7,R0 : @RW1+RW7	XCH R7, @RW1+RW7
+ E	XCH R0,R0 : @PC-dt16	XCH R0, @PC-dt16	XCH R1, R1,R0 : @PC-dt16	XCH R1, @PC-dt16	XCH R2, R2,R0 : @PC-dt16	XCH R2, @PC-dt16	XCH R3, R3,R0 : @PC-dt16	XCH R3, @PC-dt16	XCH R4, R4,R0 : @PC-dt16	XCH R4, @PC-dt16	XCH R5, R5,R0 : @PC-dt16	XCH R5, @PC-dt16	XCH R6, R6,R0 : @PC-dt16	XCH R6, @PC-dt16	XCH R7, R7,R0 : @PC-dt16	XCH R7, @PC-dt16
+ F	XCH R0,R0 : @RW3+	XCH R0, @RW3+	XCH R1, R1,R0 : @RW3+	XCH R1, @RW3+	XCH R2, R2,R0 : @RW3+	XCH R2, @RW3+	XCH R3, R3,R0 : @RW3+	XCH R3, @RW3+	XCH R4, R4,R0 : @RW3+	XCH R4, @RW3+	XCH R5, R5,R0 : @RW3+	XCH R5, @RW3+	XCH R6, R6,R0 : @RW3+	XCH R6, @RW3+	XCH R7, R7,R0 : @RW3+	XCH R7, @RW3+

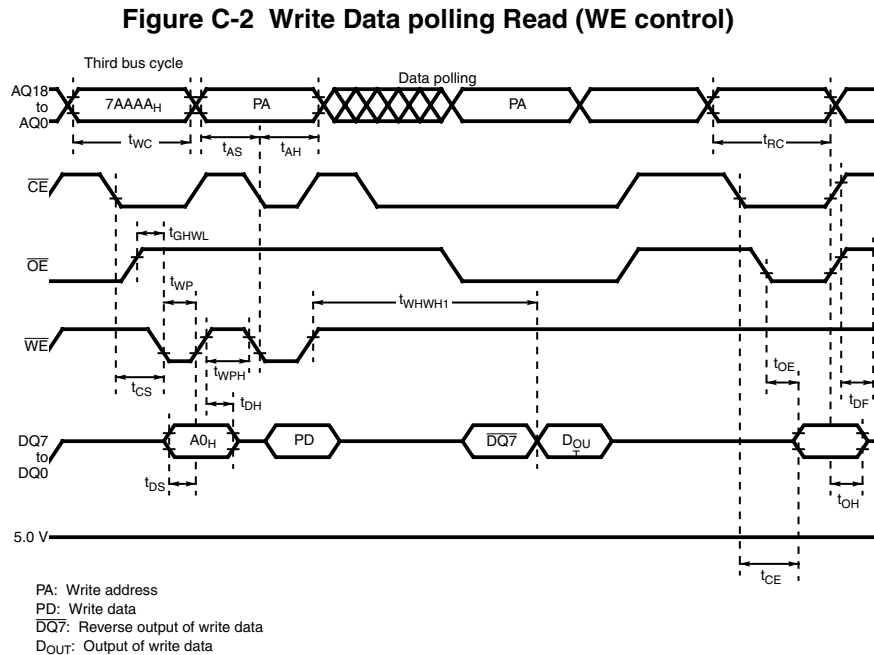
APPENDIX C Timing Diagrams in Flash Memory Mode

Each timing diagram for the external pins of the MB90F594A/MB90F594G/MB90F591A in the Flash Memory mode is shown below.

■ Data read by Read Access



■ Write, Data polling, Read (WE control)

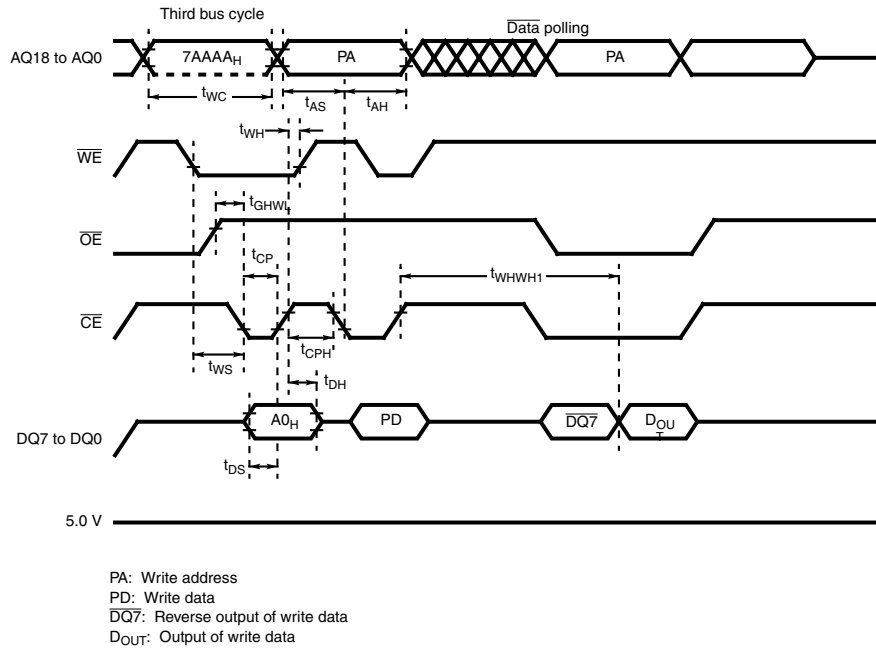


Note:

The last two bus cycle sequences out of the four are described.

■ **Write Data Polling Read (CE control)**

Figure C-3 Timing Diagram for Write Access (CE Control)

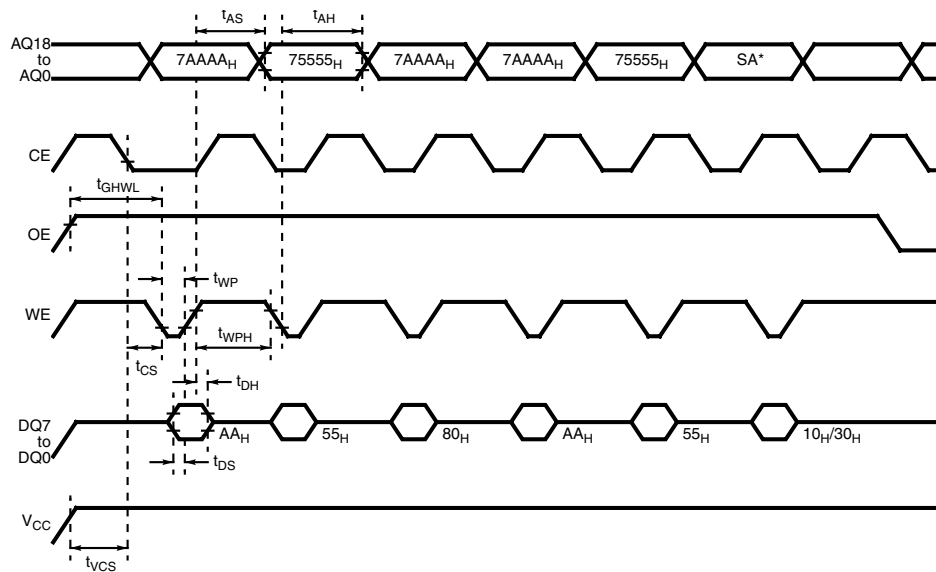


Note:

The last two bus cycle sequences out of the four are described.

■ **Chip Erase/sector Erase Command Sequence**

Figure C-4 Timing Diagram for Write Access (Chip Erasing/Sector Erasing)

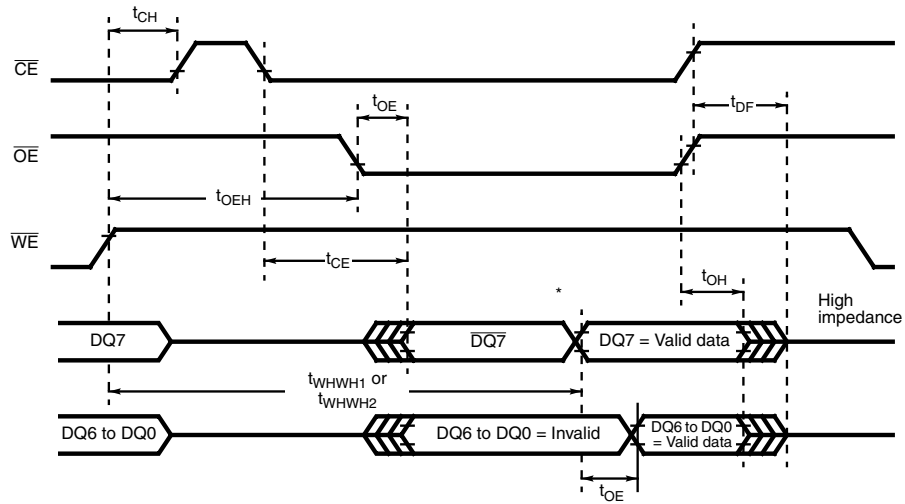


Note:

SA is the sector address at sector erasing. 7AAAA_H (or 6AAAA_H) is the address at chip erasing.

■ **Data Polling**

Figure C-5 Timing Diagram for Data Polling

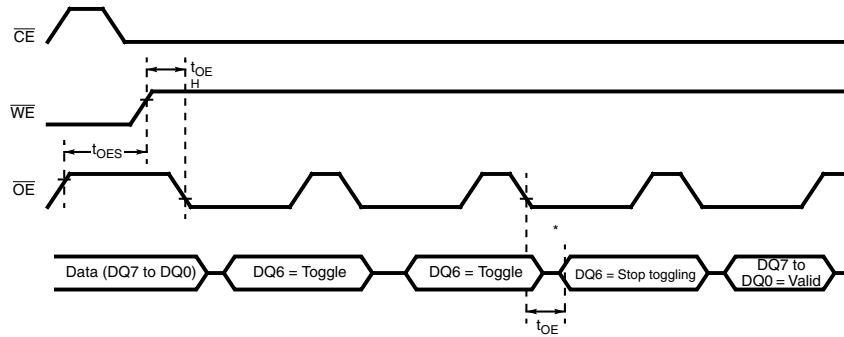


Note:

DQ7 is valid data (The device terminates automatic operation).

■ **Toggle Bit**

Figure C-6 Timing Diagram for Toggle Bit

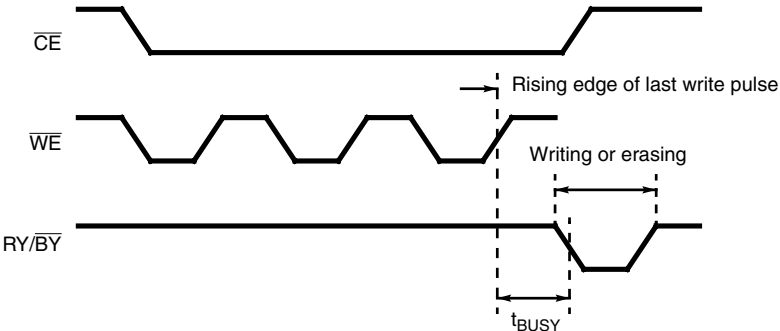


Note:

DQ6 stops toggling (The device terminates automatic operation).

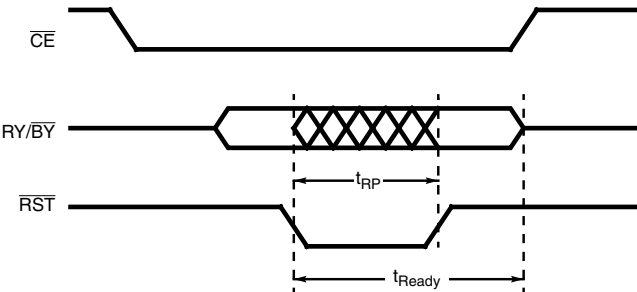
■ RY/BY Timing During Writing/erasing

Figure C-7 Timing Diagram for Output of RY/BY Signal during Writing/Erasing



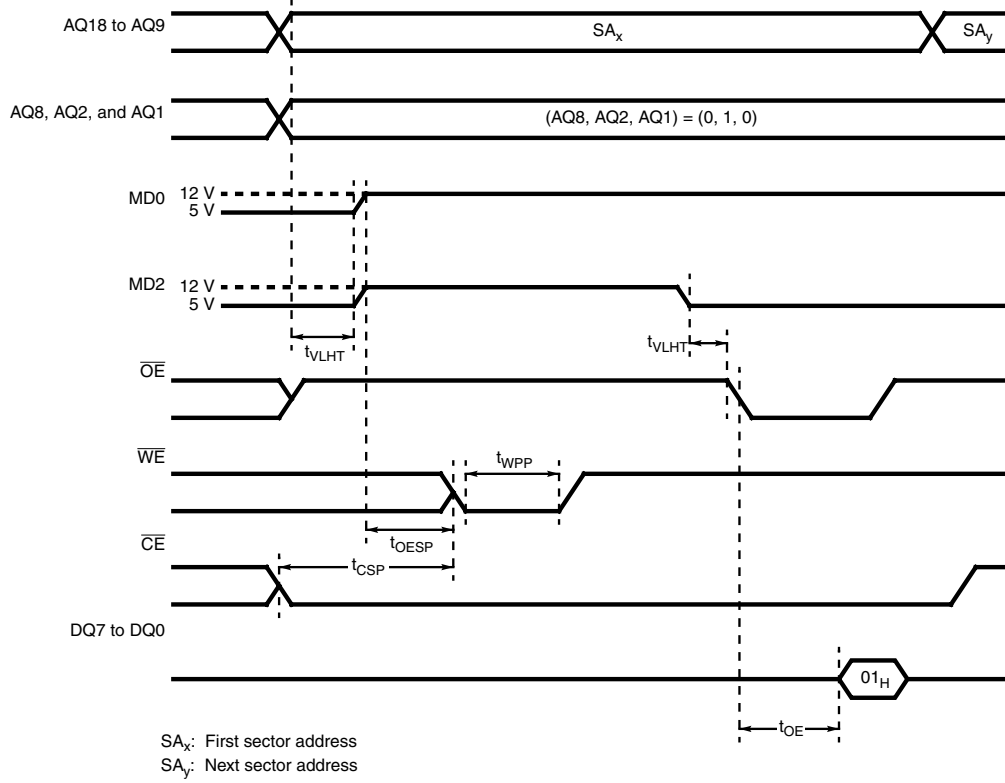
■ RST and RY/BY timing

Figure C-8 Timing Diagram for Output of RY/BY Signal at Hardware Reset



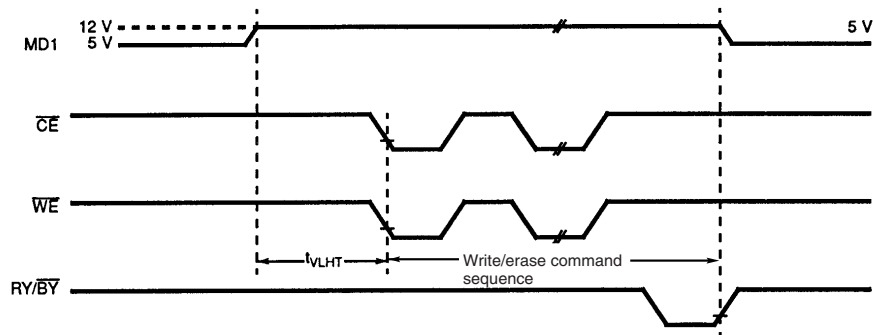
■ Enable Sector Protect/verify Sector Protect

Figure C-9 Enable Sector Protect/Verify Sector Protect



■ Temporary Sector Protect Cancellation

Figure C-10 Temporary Sector Protect Cancellation



APPENDIX D List of MB90590 Interrupt Vectors

The interrupt vector table to be referenced for interrupt processing is allocated to FFFC00_H to FFFFFFF_H in the memory area and also used for software interrupts.

■ List of MB90590 Interrupt Vectors

Table D-1 "MB90590 Interrupt Vectors" lists the interrupt vectors for the MB90590 series.

Table D-1 MB90590 Interrupt Vectors

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 0	FFFEEC _H	FFFED _H	FFFEE _H	Unused	#0	None
·	·	·	·	·	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·
INT 7	FFFEE0 _H	FFFEE1 _H	FFFEE2 _H	Unused	#7	None
INT 8	FFFDC _H	FFFDD _H	FFFDE _H	FFFDF	#8	(RESET vector)
INT 9	FFFDD8 _H	FFFDD9 _H	FFFDDA _H	Unused	#9	ROM correction
INT 10	FFFDD4 _H	FFFDD5 _H	FFFDD6 _H	Unused	#10	<Exception>
INT 11	FFFDD0 _H	FFFDD1 _H	FFFDD2 _H	Unused	#11	Timebase timer
INT 12	FFFDC _H	FFFCD _H	FFFCE _H	Unused	#12	External interrupt (INT0 to INT7)
INT 13	FFFDC8 _H	FFFDC9 _H	FFFCA _H	Unused	#13	CAN 0 RX
INT 14	FFFDC4 _H	FFFDC5 _H	FFFDC6 _H	Unused	#14	CAN 0 TX/NS
INT 15	FFFDC0 _H	FFFDC1 _H	FFFDC2 _H	Unused	#15	CAN 1 RX
INT 16	FFFBC _H	FFFB _H	FFFB _H	Unused	#16	CAN 1 TX/NS
INT 17	FFFBB8 _H	FFFBB9 _H	FFFBA _H	Unused	#17	PPG 0/1
INT 18	FFFBB4 _H	FFFBB5 _H	FFFBB6 _H	Unused	#18	PPG 2/3
INT 19	FFFBB0 _H	FFFBB1 _H	FFFBB2 _H	Unused	#19	PPG 4/5
INT 20	FFFAC _H	FFFAD _H	FFFAE _H	Unused	#20	PPG 6/7
INT 21	FFFA8 _H	FFFA9 _H	FFFAA _H	Unused	#21	PPG 8/9
INT 22	FFFA4 _H	FFFA5 _H	FFFA6 _H	Unused	#22	PPG A/B
INT 23	FFFA0 _H	FFFA1 _H	FFFA2 _H	Unused	#23	16-bit reload timer 0
INT 24	FFF9C _H	FFF9D _H	FFF9E _H	Unused	#24	16-bit reload timer 1
INT 25	FFF98 _H	FFF99 _H	FFF9A _H	Unused	#25	Input capture 0/1

APPENDIX

Table D-1 MB90590 Interrupt Vectors (Continued)

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 26	FFFF94 _H	FFFF95 _H	FFFF96 _H	Unused	#26	Output compare 0/1
INT 27	FFFF90 _H	FFFF91 _H	FFFF92 _H	Unused	#27	Input capture 2/3
INT 28	FFFF8C _H	FFFF8D _H	FFFF8E _H	Unused	#28	Output compare 2/3
INT 29	FFFF88 _H	FFFF89 _H	FFFF8A _H	Unused	#29	Input capture 4/5
INT 30	FFFF84 _H	FFFF85 _H	FFFF86 _H	Unused	#30	Output compare 4/5
INT 31	FFFF80 _H	FFFF81 _H	FFFF82 _H	Unused	#31	A/D converter
INT 32	FFFF7C _H	FFFF7D _H	FFFF7E _H	Unused	#32	I/O timer/Watch timer
INT 33	FFFF78 _H	FFFF79 _H	FFFF7A _H	Unused	#33	Serial I/O
INT 34	FFFF74 _H	FFFF75 _H	FFFF76 _H	Unused	#34	Sound generator
INT 35	FFFF70 _H	FFFF71 _H	FFFF72 _H	Unused	#35	UART 0 RX
INT 36	FFFF6C _H	FFFF6D _H	FFFF6E _H	Unused	#36	UART 0 TX
INT 37	FFFF68 _H	FFFF69 _H	FFFF6A _H	Unused	#37	UART 1 RX
INT 38	FFFF64 _H	FFFF65 _H	FFFF66 _H	Unused	#38	UART 1 TX
INT 39	FFFF60 _H	FFFF61 _H	FFFF62 _H	Unused	#39	UART 2 RX
INT 40	FFFF5C _H	FFFF5D _H	FFFF5E _H	Unused	#40	UART 2 TX
INT 41	FFFF58 _H	FFFF59 _H	FFFF5A _H	Unused	#41	Flash Memory
INT 42	FFFF54 _H	FFFF55 _H	FFFF56 _H	Unused	#42	Delayed interrupt
INT 43	FFFF50 _H	FFFF51 _H	FFFF52 _H	Unused	#43	None
.
.
.
INT 254	FFFC04 _H	FFFC05 _H	FFFC06 _H	Unused	#254	None
INT 255	FFFC00 _H	FFFC01 _H	FFFC02 _H	Unused	#255	None

■ Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Table D-2 "Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers" summarizes the relationships among the interrupt causes, interrupt vectors, and interrupt control registers of the MB90590 series.

Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Interrupt cause	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Reset	N	#08	FFFFDC _H	—	—
INT9 instruction	N	#09	FFFFD8 _H	—	—
Exception	N	#10	FFFFD4 _H	—	—
Timebase timer	N	#11	FFFFD0 _H	ICR00	0000B0 _H
External interrupt (INT0 to INT7)	Y1	#12	FFFFCC _H		
CAN 0 RX	N	#13	FFFFC8 _H	ICR01	0000B1 _H
CAN 0 TX/NS	N	#14	FFFFC4 _H		
CAN 1 RX	N	#15	FFFFC0 _H	ICR02	0000B2 _H
CAN 1 TX/NS	N	#16	FFFFBC _H		
PPG 0/1	N	#17	FFFFB8 _H	ICR03	0000B3 _H
PPG 2/3	N	#18	FFFFB4 _H		
PPG 4/5	N	#19	FFFFB0 _H	ICR04	0000B4 _H
PPG 6/7	N	#20	FFFFAC _H		
PPG 8/9	N	#21	FFFFA8 _H	ICR05	0000B5 _H
PPG A/B	N	#22	FFFFA4 _H		
16-bit reload timer 0	Y1	#23	FFFFA0 _H	ICR06	0000B6 _H
16-bit reload timer 1	Y1	#24	FFFF9C _H		
Input capture 0/1	Y1	#25	FFFF98 _H	ICR07	0000B7 _H
Output compare 0/1	Y1	#26	FFFF94 _H		
Input capture 2/3	Y1	#27	FFFF90 _H	ICR08	0000B8 _H
Output compare 2/3	Y1	#28	FFFF8C _H		
Input capture 4/5	Y1	#29	FFFF88 _H	ICR09	0000B9 _H
Output compare 4/5	Y1	#30	FFFF84 _H		
A/D converter	Y1	#31	FFFF80 _H	ICR10	0000BA _H
I/O timer/Watch timer	N	#32	FFFF7C _H		
Serial I/O	Y1	#33	FFFF78 _H	ICR11	0000BB _H
Sound generator	N	34	FFFF74 _H		

Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Interrupt cause	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
UART 0 RX	Y2	35	FFFF70 _H	ICR12	0000BC _H
UART 0 TX	Y1	36	FFFF6C _H		
UART 1 RX	Y2	37	FFFF68 _H	ICR13	0000BD _H
UART 1 TX	Y1	38	FFFF64 _H		
UART 2 RX	Y2	39	FFFF60 _H	ICR14	0000BE _H
UART 2 TX	Y1	40	FFFF5C _H		
Flash memory	N	41	FFFF58 _H	ICR15	0000BF _H
Delayed interrupt	N	42	FFFF54 _H		

Y1: An EI²OS interrupt clear signal or EI²OS register read access clears the interrupt request flag.

Y2: An EI²OS interrupt clear signal or EI²OS register read access clears the interrupt request flag. A stop request is issued.

N: An EI²OS interrupt clear signal does not clear the interrupt request flag.

Note:

For a peripheral module having two interrupt causes for one interrupt number, an EI²OS interrupt clear signal clears both interrupt request flags.

When EI²OS ends, an EI²OS clear signal is sent to every interrupt flag assigned to each interrupt number.

EI²OS is activated when one of two interrupts assigned to an interrupt control register (ICR) is caused while EI²OS is enabled. This means that an EI²OS descriptor that should essentially be specific to each interrupt cause is shared by two interrupts. Therefore, while one interrupt is enabled, the other interrupt must be disabled.

INDEX

**The index follows on the next page.
This is listed in alphabetic order.**

Index

Numerics

16-bit free-running timer	126
16-bit free-running timer block diagram.....	129
16-bit free-running timer operation	134
16-bit free-running timer timing	135
16-bit I/O timer, block diagram of	127
16-bit reload timer (in internal clock mode), input pin function of	160
16-bit reload timer (with event count function), outline of	152
16-bit reload timer register	154
16-bit reload timer, block diagram of.....	153
16-bit reload timer, internal clock operation of	159
16-bit reload timer, output pin function of.....	162
16-bit reload timer, underflow operation of.....	161
16-bit timer register (TMR)/16-bit reload register (TMRLR), register layout of.....	158
24-bit operand specification	23
2M-bit flash memory feature.....	364
8/16-bit PPG hardware, initial value of.....	190
8/16-bit PPG, function of	174
8/16-bit PPG, selecting count clock for	187

A

A/D converter, block diagram of.....	206
acceptance filter, setting	323
acceptance filtering	319
acceptance mask registers 0 and 1 (AMR0 and AMR1)	308
acceptance mask select register (AMSR)	306
accumulator (A)	30
activation	123
address field, effective	429, 444
address generation type.....	21
address match detection function, block diagram of	352
address match detection function, operation of ...	355
address match detection function, system configuration example of.....	356
addressing.....	428
addressing, direct.....	430
addressing, indirect.....	435
amplitude data register.....	347
analog input enable register.....	205

B

bank addressing type.....	24
bank select prefix.....	38
bit timing register (BTR).....	291
bit timing, setting	323
block diagram.....	5, 83
buffer address pointer (BAP)	63
bus mode setting bit.....	104
bus operation stop (HALT = 0), condition for canceling	287
bus operation stop (HALT = 1), state during.....	287
bus operation stop (HALT=1), condition for setting	287
BVAL bit, caution for disabling message buffers by	331

C

calculating execution cycle count.....	442
CAN controller, block diagram of	273
CAN controller, canceling transmission request from	316
CAN controller, completing transmission of	317
CAN controller, feature of	272
CAN controller, reception flowchart of	322
CAN controller, starting transmission of.....	316
CAN controller, transmission flowchart of.....	317
CE control	485
chip erase/sector erase command sequence	485
CLK asynchronous baud rate	239
CLK synchronous baud rate	239
clock generator, note on	74
clock selection register (CKSCR).....	87
clock selection, status transition of	98
command sequence table.....	372
common register bank prefix (CMR)	39
compare registers 0 and 1 being used, output waveform sample when.....	141
compare registers, output waveform sample with two	142
condition code register (CCR).....	32
continuous mode.....	216
continuous mode, starting EI ² OS in.....	221
control status register.....	131, 146
control status register (ADCS0)	208
control status register (ADCS1)	211

- control status register (CSR)..... 284
control status register (FMCS), flash memory 370
conversion data protection..... 225
counter operation state 163
CPU memory space, outline of 21
cycle count, execution..... 441
- D**
- data counter (DCT) 62
data frame and remote frame, processing for
reception of 320
data polling..... 486
data polling flag (DQ7) 376
data register 130
data register x (x = 0 to 15) (DTRx) 314
data registers (ADCR1 and ADCR0) 214
decrement grade register..... 348
delayed interrupt cause issuance/cancellation
register (DIRR)..... 71
delayed interrupt occurrence 72
delayed interrupt, block diagram of..... 70
direct addressing..... 430
DIV A, Ri and DIVW A, RWi instruction, precautions
for use of..... 41
DIV A, Ri instruction 41
DIV A, Ri instructions without precaution 42
DIVW A, RWi instruction 41
DIVW A, RWi instructions without precaution 42
DLC register x (x = 0 to 15) (DLCRx)..... 313
DTP operation..... 199
DTP request, switching between external interrupt
and..... 200
DTP/external interrupt, note on using 201
- E**
- effective address field 429, 444
EI²OS operation flow 65
EI²OS status register (ISCS) 64
EI²OS, conversion using..... 218
enable sector protect/verify sector protect 488
erasing chip..... 388
erasing flash memory..... 364
erasing sector 389
erasing sector in flash memory 389
example of program patch processing 357
execution cycle count..... 441
execution cycle count, calculating..... 442
extended intelligent I/O service (EI²OS) 45, 60
extended intelligent I/O service descriptor (ISD).... 62
- extended serial I/O interface, interrupt function of
..... 269
external clock..... 242
external event counter 160
external interrupt operation..... 198
external interrupt request..... 200
external shift clock mode 263
- F**
- F²MC-16LX instruction list..... 448
flag change disable prefix (NCC)..... 39
flash memory control signal..... 368
flash memory mode 368
flash memory register 365
flash memory write/erase, detailed explanation of
..... 384
flash memory, block diagram of entire..... 366
flash memory, writing to..... 386
flash microcomputer programmer (power supplied
from programmer), example of minimum
connection to 412
flash microcomputer programmer (user power supply
used), example of minimum connection to
..... 410
frame format, setting..... 323
frequency data register..... 346
- G**
- general-purpose register 29
- H**
- hardware interrupt 44, 53
hardware interrupt operation 54
hardware interrupt, occurrence and release of 55
hardware interrupt, structure of 53
hardware sequence flag 374
hardware standby mode, releasing 95
hardware standby mode, transition to 95
- I**
- I/O map..... 416
I/O port..... 108
I/O port register..... 109
I/O register address pointer (IOA) 63
ID register x (x = 0 to 15) (IDRx)..... 311
IDE register (IDER)..... 295
indirect addressing 435
input capture 144
input capture (2 channels per one module) 127

INDEX

- input capture block diagram 145
- input capture data register 146
- input capture fetch timing, sample of 148
- input capture input timing 149
- input data register (UIDR) and output data register (UODR) 235
- input impedance 205
- input-output circuit 12
- instruction map, structure of 462
- instruction presentation item and symbol, description of 445
- instruction type 427
- intelligent I/O service (EI²OS) function and interrupt 152
- intermittent CPU operation 96
- internal and external clock 242
- internal shift clock mode 263
- Interrupt cause, interrupt vector, and interrupt control register 491
- interrupt control register 491
- interrupt control register (ICR) 48
- interrupt disable instruction 40
- interrupt flow 51
- interrupt level mask register (ILM) 34
- interrupt vector 47, 491
- interrupt, 8/16-bit PPG 189
- interrupt, intelligent I/O service (EI²OS) function and 152
- interrupt/DTP enable register 196
- interrupt/DTP source register 196
- interval interrupt function 117

- L**
- last event indicator register (LEIR) 288
- layout of rate and data register (URD) 236
- lower-power control circuit, outline of 82
- low-power consumption mode, setting 324
- low-power mode control register 84
- low-power mode control register (LPMCR) 85
- low-power mode control register access, note of ... 90
- low-power mode operation 89

- M**
- machine clock, initializing 97
- main clock and PLL clock, switching between 97
- MB90590 interrupt vector, list of 489
- MB90F594A/MB90F594G/MB90F591A/MB90F591G serial programming connection, basic configuration of 402
- memory access mode 102
- memory space map 22
- memory space, multi-byte data allocation in 26
- message buffer 310
- message buffer (data register), list of 281
- message buffer (DLC register and data register), list of 279
- message buffer (ID register), list of 276
- message buffer (x), procedure for reception by ... 327
- message buffer (x), procedure for transmission by 325
- message buffer valid register (BVALR) 294
- mode data 104
- mode pin 103
- multi-byte data, accessing 26
- multi-level message buffer, setting configuration of 329
- multiple interrupt 57

- N**
- negative clock operation 270

- O**
- operation, note on 70
- oscillating clock frequency 405
- oscillation stabilization wait time, setting 94, 95
- output compare 136
- output compare (2 channels per one module) 126
- output compare block diagram 136
- output compare register 137
- output compare register 0, clearing counter upon match with 135
- output compare timing 142
- output compare, control status register of 138
- output data register (UODR) 235
- overflow, clearing counter by 134

- P**
- package dimension 7
- PACSR 353
- PADR0 and PADR1 353
- parity bit 244
- pin assignment 6
- pin function 8
- PLL clock and main clock, switching between 97
- port data register 110
- port direction register 111
- PPG0 operation mode control register (PPGC0) 178

- PPG0, 1 clock select register (PPG01) 182
- PPG1 operation mode control register (PPGC1)
..... 180
- prefix code 40
- prefix code, consecutive 40
- prefix instruction 40
- prefix instruction, restriction on interrupt disable
instruction and 40
- processor status (PS) 32
- program address detection control status register
(PACSR) 353
- program address detection register (PADR0 and
PADR1) 353
- program counter (PC) 35
- program patch processing, example of 357
- PWM control 0 register 336
- PWM1&2 compare register 337
- PWM1&2 select register 338
- R**
- rate and data register (URD) content 236
- read access, data read by 484
- read state, setting flash memory to 385
- receive and transmit error counter (RTEC) 290
- receive overrun 320
- receive overrun register (ROVRR) 304
- received message, storing 319
- reception complete register (RCR) 302
- reception interrupt enable register (RIER) 305
- recommended setting 105
- register bank 36
- register bank pointer (RP) 33
- reload value and pulse width, relationship between 8/
16-bit PPG 186
- remote frame receiving wait register (RFWTR) ... 298
- remote frame, processing for reception of 320
- remote request receiving register (RRTRR) 303
- request level setting register 197
- reset cause 78
- reset cause occurrence 75
- reset input, register not initialized by 76
- reset release, operation after 75
- reset state, setting flash memory to 385
- ROM mirroring module, block diagram of 360
- ROM mirroring register (ROMM) 361
- RST and RY/BY timing 487
- RY/BY timing during writing/erasing 487
- S**
- sector configuration 366
- sector erase timer flag (DQ3) 380
- sector, restarting erasing of flash memory 392
- sector, suspending erasing of flash memory 391
- serial clock input frequency 405
- serial I/O operation 262, 264
- serial I/O prescaler (CDCR) 261
- serial mode control register (UMC) content 231
- serial mode control register (UMC), layout of 231
- serial mode control status register (SMCS) 256
- serial programming connection (power supplied from
programmer), example of 408
- serial programming connection (user power supply
used), example of 406
- serial shift data register (SDR) 260
- set timing of six flags 245
- setting ID 323
- single mode 216
- single mode, starting EI²OS in 219
- sleep mode, releasing 91
- sleep mode, transition to 91
- software interrupt 45, 58
- software interrupt operation 58
- sound control register 344
- sound generator register 343
- sound generator, block diagram of 342
- special register 27
- status flag during transmit and receive operation
..... 249
- status register (USR) content 233
- status register (USR) layout 233
- stepping motor controller register 335
- stepping motor controller, block diagram of 334
- stop mode 217
- stop mode, releasing 93
- stop mode, starting EI²OS in 223
- stop mode, transition to 93
- structure 61
- structure of instruction map 462
- sub-second register 170
- system stack pointer (SSP), user stack pointer (USP)
and 31
- T**
- timebase counter 117
- timebase timer control register (TBTC) 115
- timebase timer, block diagram of 114
- timebase timer, outline of 114

INDEX

timer control register	168
timer control register (TMSCR), register content of	155
timer control register (TMSCR), register layout of	155
timing limit exceeded flag (DQ5)	379
toggle bit.....	486
toggle bit flag (DQ6)	378
toggle bit-2 flag (DQ2).....	382
tone count register	349
transfer data format.....	243
transition to low-power mode, note on	90
transmission cancel register (TCANR).....	299
transmission complete register (TCR).....	300
transmission interrupt enable register (TIER).....	301
transmission request register (TREQR)	296
transmission RTR register (TRTRR)	297
type of instruction	427

U

undefined instruction, exception due to execution of	68
undefined instruction, execution of	68
user power supply	406, 410
user stack pointer (USP) and system stack pointer (SSP)	31

W

watch mode, releasing	92
watch mode, transition to	92
watch timer register.....	167
watch timer, block diagram of	166
watch-dog counter	123
watch-dog stop.....	123
watch-dog timer block diagram	120
watch-dog timer control register (WDTC)	121
WE control	484
write data polling read (CE control).....	485
write, data polling, read (WE control).....	484
writing to flash memory	364

CM44-10105-4E

FUJITSU SEMICONDUCTOR • CONTROLLER MANUAL

F²MC-16LX

16-BIT MICROCONTROLLER

MB90590 Series

HARDWARE MANUAL

July 2002 the fourth edition

Published **FUJITSU LIMITED** Electronic Devices

Edited Technical Information Dept.
