

F²MC-16LX
16-BIT MICROCONTROLLER
MB90340 Series
HARDWARE MANUAL

F²MC-16LX
16-BIT MICROCONTROLLER
MB90340 Series
HARDWARE MANUAL

FUJITSU LIMITED

PREFACE

■ Objectives and intended reader

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

The MB90340 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 MB90340 series for designers who actually use the MB90340 series to design products. Please 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 preliminary manual

CHAPTER 1 “OVERVIEW”

The MB90340 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 “CLOCKS”

This chapter explains the functions and operations of clocks.

CHAPTER 6 “CLOCK MODULATOR”

This chapter explains the functions and operation of the clock modulator.

CHAPTER 7 “RESETS”

This chapter describes resets for the MB90340 series.

CHAPTER 8 “LOW-POWER CONTROL CIRCUIT”

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

CHAPTER 9 “MEMORY ACCESS MODES”

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

CHAPTER 10 “I/O PORTS”

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

CHAPTER 11 “TIMEBASE TIMER”

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

CHAPTER 12 “WATCH-DOG TIMER”

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

CHAPTER 13 “16-BIT I/O TIMER”

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

CHAPTER 14 “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 15 “WATCH TIMER”

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

CHAPTER 16 “8/16-BIT PPG”

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

CHAPTER 17 “DTP/EXTERNAL INTERRUPTS”

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

CHAPTER 18 “8/10-BIT A/D CONVERTER”

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

CHAPTER 19 “CLOCK MONITOR”

This chapter explains the functions and operations of the clock monitor.

CHAPTER 20 “UART”

This chapter explains the functions and operation of the LIN-UART.

CHAPTER 21 “400 kHz I²C INTERFACE”

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

CHAPTER 22 “CAN CONTROLLER”

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

CHAPTER 23 “ADDRESS MATCH DETECTION FUNCTION”

This chapter explains the address match detection function and operation.

CHAPTER 24 “ROM MIRRORING MODULE”

This chapter explains the ROM mirroring module.

CHAPTER 25 “1M/2M/3M-BIT FLASH MEMORY”

This chapter explains the functions and operation of the 1M/2M/3M-bit flash memory.

CHAPTER 26 “EXAMPLES OF MB90F347 SERIAL PROGRAMMING CONNECTION”

This chapter provides examples of F²MC-16LX MB90F347 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, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of Fujitsu semiconductor device; Fujitsu does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. Fujitsu assumes no liability for any damages whatsoever arising out of the use of the information.
- Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of Fujitsu or any third party or does Fujitsu warrant non-infringement of any third-party's intellectual property right or other right by using such information. Fujitsu assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.
- The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).
Please note that Fujitsu 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 of MB90V340(S)	5
1.4	Block Diagram of Flash/Mask ROM version	6
1.5	Pin Assignment	7
1.6	Package Dimensions	11
1.7	Pin Functions	13
1.8	Input-Output Circuits	20
1.9	Handling Device	25
CHAPTER 2	CPU	31
2.1	Outline of the CPU	32
2.2	Memory Space	33
2.3	Memory Space Map	36
2.4	Linear Addressing	38
2.5	Bank Addressing Types	39
2.6	Multi-byte Data in Memory Space	41
2.7	Registers	42
2.7.1	Accumulator (A)	45
2.7.2	User Stack Pointer (USP) and System Stack Pointer (SSP)	46
2.7.3	Processor Status (PS)	47
2.7.4	Program Counter (PC)	50
2.8	Register Bank	51
2.9	Prefix Codes	53
2.10	Interrupt Disable Instructions	56
2.11	Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions	58
CHAPTER 3	INTERRUPTS	61
3.1	Outline of Interrupts	62
3.2	Interrupt Vector	66
3.3	Interrupt Control Registers (ICR)	68
3.4	Interrupt Flow	71
3.5	Hardware Interrupts	73
3.5.1	Hardware Interrupt Operation	74
3.5.2	Occurrence and Release of Hardware Interrupt	75
3.5.3	Multiple interrupts	77
3.6	Software Interrupts	78
3.7	Extended Intelligent I/O Service (EI ² OS)	80
3.7.1	Extended Intelligent I/O Service Descriptor (ISD)	82
3.7.2	EI ² OS Status Register (ISCS)	84
3.8	Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI ² OS)	85
3.9	μDMAC Interrupt	88

3.9.1	μDMAC Functions	89
3.9.2	μDMAC Registers	90
3.9.2.1	DMA descriptor channel specification register (DCSR)	91
3.9.2.2	DMA status register (DSR)	93
3.9.2.3	DMA stop status register (DSSR)	94
3.9.2.4	DMA enable register (DER)	95
3.9.3	DMA Descriptor window register (DDWR)	96
3.9.3.1	Data counter (DCT)	98
3.9.3.2	Value in Data counter	99
3.9.3.3	DMA control register (DMACS)	100
3.9.3.4	Buffer address pointer (BAP)	102
3.9.4	μDMAC Operation	103
3.10	Exceptions	107
CHAPTER 4 DELAYED INTERRUPT		109
4.1	Outline of Delayed Interrupt Module	110
4.2	Delayed Interrupt Register	111
4.3	Delayed Interrupt Operation	112
CHAPTER 5 CLOCKS		113
5.1	Clocks	114
5.2	Block Diagram of the Clock Generation Block	117
5.3	Clock Selection Registers	119
5.3.1	Clock Selection Register (CKSCR)	120
5.3.2	PLL/Subclock Control Register (PSCCR)	124
5.4	Clock Mode	126
5.5	Oscillation Stabilization Wait Interval	130
5.6	Connection of an Oscillator or an External Clock to the Microcontroller	131
CHAPTER 6 CLOCK MODULATOR		133
6.1	Overview	134
6.2	Clock Modulator Control Register (CMCR)	135
6.3	Application Note	137
CHAPTER 7 RESETS		139
7.1	Resets	140
7.2	Reset Cause and Oscillation Stabilization Wait Times	142
7.3	External Reset Pin	144
7.4	Reset Operation	145
7.5	Reset Cause Bits	147
7.6	Status of Pins in a Reset	150
CHAPTER 8 LOW-POWER CONSUMPTION MODE		151
8.1	Overview of Low-Power Consumption Mode	152
8.2	Block Diagram of the Low-Power Consumption Control Circuit	155
8.3	Low-Power Consumption Mode Control Register (LPMCR)	157
8.4	CPU Intermittent Operation Mode	160

8.5	Standby Mode	161
8.5.1	Sleep Mode	163
8.5.2	Timebase Timer Mode	166
8.5.3	Watch Mode	168
8.5.4	Stop Mode	170
8.6	Status Change Diagram	173
8.7	Status of Pins in Standby Mode and during Hold and Reset	175
8.8	Usage Notes on Low-Power Consumption Mode	181
CHAPTER 9	MEMORY ACCESS MODES	185
9.1	Outline of Memory Access Modes	186
9.1.1	Mode Pins	187
9.1.2	Mode Data	188
9.1.3	Memory Space in Each Bus Mode	189
9.2	External Memory Access (Bus Pin Control Circuit)	192
9.2.1	External Memory Access (External Bus Pin Control Circuit) Registers	193
9.2.2	Automatic Ready Function Selection Register (ARSR)	194
9.2.3	External Address Output Control Register (HACR)	196
9.2.4	Bus Control Signal Selection Register (ECSR)	197
9.3	External Memory Access Control Signal Operation	200
9.3.1	Ready Function	202
9.3.2	Hold Function	204
CHAPTER 10	I/O PORTS	205
10.1	I/O Ports	206
10.2	I/O Port Registers	207
10.2.1	Port Data Register	208
10.2.2	Data Direction Register	210
10.2.3	Pull-up Control Register (PUCR)	212
10.2.4	Analog Input Enable Register	213
10.2.5	Input Level Select Register	214
CHAPTER 11	TIMEBASE TIMER	217
11.1	Outline of Timebase Timer	218
11.2	Timebase Timer Control Register	219
11.3	Operations of Timebase Timer	221
CHAPTER 12	WATCH-DOG TIMER	223
12.1	Outline of Watch-Dog Timer	224
12.2	Watch-dog Timer Operation	227
CHAPTER 13	16-BIT I/O TIMER	229
13.1	Outline of 16-Bit I/O Timer	230
13.2	16-Bit I/O Timer Registers	232
13.3	16-bit Free-running Timer	234
13.3.1	Data Register	235
13.3.2	Control Status Register	236

13.3.3	16-bit Free-running Timer Operation	239
13.4	Output Compare	241
13.4.1	Output Compare Register	242
13.4.2	Control Status Register of Output Compare	243
13.4.3	16-bit Output Compare Operation	248
13.5	Input Capture	253
13.5.1	Input Capture Register Details	254
13.5.2	16-bit Input Capture Operation	259
CHAPTER 14	16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)	261
14.1	Outline of 16-Bit Reload Timer (with Event Count Function)	262
14.2	16-Bit Reload Timer (with Event Count Function)	264
14.2.1	Timer Control Status Register (TMCSR)	265
14.2.2	Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)	268
14.3	Internal Clock and External Clock Operations of 16-bit Reload Timer	269
14.4	Underflow Operation of 16-bit Reload Timer	271
14.5	Output Pin Functions of 16-bit Reload Timer	272
14.6	Counter Operation State	273
CHAPTER 15	WATCH TIMER	275
15.1	Outline of Watch Timer	276
15.2	Watch Timer Control Register (WTC)	278
15.3	Watch Timer Operation	280
CHAPTER 16	8/16-BIT PPG	281
16.1	Outline of 8/16-bit PPG	282
16.2	Block Diagram of 8/16-bit PPG	284
16.3	8/16-bit PPG Registers	286
16.3.1	PPG0 Operation Mode Control Register (PPGC0)	287
16.3.2	PPG1 Operation Mode Control Register (PPGC1)	289
16.3.3	PPG0/1 Clock Select Register (PPG01)	291
16.3.4	Reload Register (PRL/PRLH)	293
16.4	Operations of 8/16-bit PPG	294
16.5	Selecting a Count Clock for 8/16-bit PPG	296
16.6	Controlling Pin Output of 8/16-bit PPG Pulses	297
16.7	8/16-bit PPG Interrupts	298
16.8	Initial Values of 8/16-bit PPG Hardware	299
CHAPTER 17	DTP/EXTERNAL INTERRUPTS	301
17.1	Outline of DTP/External Interrupts	302
17.2	DTP/External Interrupt Registers	304
17.3	Operations of DTP/External Interrupts	307
17.4	Switching between External Interrupt and DTP Requests	309
17.5	Notes on using DTP/External Interrupt functions	310
CHAPTER 18	A/D CONVERTER	311
18.1	Features of A/D Converter	312

18.2	Block Diagram of A/D Converter	315
18.3	Registers for A/D Converter	316
18.3.1	Control status register (ADCS0)	317
18.3.2	Control status register (ADCS1)	319
18.3.3	Data Register (ADCR0, ADCR1)	322
18.3.4	Setting Register (ADSR0, ADSR1)	323
18.4	Operation of A/D Converter	327
18.5	Conversion Using EI ² OS	329
18.5.1	Example of activating of EI ² OS in single mode	330
18.5.2	Example of activating of EI ² OS in continuous mode	332
18.5.3	Example of activating of EI ² OS in stop mode	334
18.6	Convert Data Protection Function	336
CHAPTER 19	CLOCK MONITOR FUNCTION	339
19.1	Clock Monitor Function	340
19.2	Clock Output Enable Register (CLKR)	341
CHAPTER 20	UART	343
20.1	Overview of UART	344
20.2	Configuration of UART	348
20.3	UART Pins	353
20.4	UART Registers	354
20.4.1	Serial Control Register (SCR)	355
20.4.2	Serial Mode Register (SMR)	357
20.4.3	Serial Status Register (SSR)	359
20.4.4	Reception and Transmission Data Register (RDR/TDR)	361
20.4.5	Extended Status/Control Register (ESCR)	363
20.4.6	Extended Communication Control Register (ECCR)	365
20.4.7	Baud Rate/Reload Counter Register 0 and 1 (BGR0/1)	367
20.5	UART Interrupts	368
20.5.1	Reception Interrupt Generation and Flag Set Timing	371
20.5.2	Transmission Interrupt Generation and Flag Set Timing	373
20.6	UART Baud Rates	375
20.6.1	Setting the Baud Rate	377
20.6.2	Restarting the Reload Counter	380
20.7	Operation of UART	382
20.7.1	Operation in Asynchronous Mode (Op. Modes 0 and 1)	384
20.7.2	Operation in Synchronous Mode (Operation Mode 2)	386
20.7.3	Operation with LIN Function (Operation Mode 3)	389
20.7.4	Direct Access to Serial Pins	392
20.7.5	Bidirectional Communication Function (Normal Mode)	393
20.7.6	Master-Slave Communication Function (Multiprocessor Mode)	395
20.7.7	LIN Communication Function	399
20.7.8	Sample Flowcharts for UART in LIN communication (Operation Mode 3)	400
20.8	Notes on Using UART	402

CHAPTER 21	400 kHz I²C INTERFACE	405
21.1	I ² C Interface Overview	406
21.2	I ² C Interface Registers	408
21.2.1	Bus Status Register (IBSR)	410
21.2.2	Bus Control Register (IBCR)	413
21.2.3	Ten Bit Slave Address Register (ITBA)	422
21.2.4	Ten Bit Address Mask Register (ITMK)	423
21.2.5	Seven Bit Slave Address Register (ISBA)	425
21.2.6	Data Register (IDAR)	427
21.2.7	Clock Control Register (ICCR)	428
21.3	I ² C Interface Operation	431
21.4	Programming Flow Charts	434
CHAPTER 22	CAN CONTROLLER	437
22.1	Features of CAN Controller	438
22.2	Block Diagram of CAN Controller	439
22.3	List of Overall Control Registers	440
22.4	List of Message Buffers (ID Registers)	442
22.5	List of Message Buffers (DLC Registers and Data Registers)	445
22.6	Classifying the CAN Controller Registers	448
22.6.1	Control Status Register (CSR)	449
22.6.2	Bus Operation Stop Bit (HALT = 1)	454
22.6.3	Last Event Indicator Register (LEIR)	455
22.6.4	Receive and Transmit Error Counters (RTEC)	457
22.6.5	Bit Timing Register (BTR)	458
22.6.6	Message Buffer Valid Register (BVALR)	461
22.6.7	IDE register (IDER)	462
22.6.8	Transmission Request Register (TREQR)	463
22.6.9	Transmission RTR Register (TRTRR)	464
22.6.10	Remote Frame Receiving Wait Register (RFWTR)	465
22.6.11	Transmission Cancel Register (TCANR)	466
22.6.12	Transmission Complete Register (TCR)	467
22.6.13	Transmission Interrupt Enable Register (TIER)	468
22.6.14	Reception Complete Register (RCR)	469
22.6.15	Remote Request Receiving Register (RRTRR)	470
22.6.16	Receive Overrun Register (ROVRR)	471
22.6.17	Reception Interrupt Enable Register (RIER)	472
22.6.18	Acceptance Mask Select Register (AMSR)	473
22.6.19	Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)	475
22.6.20	Message Buffers	477
22.6.21	ID Register x (x = 0 to 15) (IDRx)	478
22.6.22	DLC Register x (x = 0 to 15) (DLCRx)	480
22.6.23	Data Register x (x = 0 to 15) (DTRx)	481
22.7	Transmission of CAN Controller	483
22.8	Reception of CAN Controller	485
22.9	Reception Flowchart of CAN Controller	488
22.10	How to Use the CAN Controller	489

22.11	Procedure for Transmission by Message Buffer (x)	491
22.12	Procedure for Reception by Message Buffer (x)	493
22.13	Setting Configuration of Multi-level Message Buffer	495
22.14	Setting the redirection of CAN1 RX/TX pin	497
22.15	Setting the CAN Direct Mode Register	499
22.16	Precautions when Using CAN Controller	500
CHAPTER 23 ADDRESS MATCH DETECTION FUNCTION		503
23.1	Outline of the Address Match Detection Function	504
23.2	Registers of the Address Match Detection Function	505
23.3	Operation of the Address Match Detection Function	507
23.4	Example of the Address Match Detection Function	508
CHAPTER 24 ROM MIRRORING MODULE		511
24.1	Outline of ROM Mirroring Module	512
24.2	ROM Mirroring Register (ROMM)	513
CHAPTER 25 0.5M/1M/2M/3M/4M-BIT FLASH MEMORY		515
25.1	Overview of 0.5M/1M/2M/3M/4M-bit Flash Memory	516
25.2	Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory	517
25.3	Write/Erase Modes	520
25.4	Flash Memory Control Status Register (FMCS)	522
25.5	Starting the Flash Memory Automatic Algorithm	524
25.6	Confirming the Automatic Algorithm Execution State	526
25.6.1	Data Polling Flag (DQ7)	528
25.6.2	Toggle Bit Flag (DQ6)	530
25.6.3	Timing Limit Exceeded Flag (DQ5)	531
25.6.4	Sector Erase Timer Flag (DQ3)	532
25.6.5	Toggle Bit-2 Flag (DQ2)	534
25.7	Detailed Explanation of Writing to and Erasing Flash Memory	536
25.7.1	Setting The Read/Reset State	537
25.7.2	Writing Data	538
25.7.3	Erasing All Data (Erasing Chips)	540
25.7.4	Erasing Optional Data (Erasing Sectors)	541
25.7.5	Suspending Sector Erase	543
25.7.6	Restarting Sector Erase	544
25.8	Notes on using 0.5M/1M/2M/3M/4M-bit Flash Memory	545
25.9	Flash Security Feature	547
25.10	Example of Programming 1M-bit Flash Memory	548
CHAPTER 26 EXAMPLES OF MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), 346CA(S), F347A(S), F347CA(S), F349/C(S) SERIAL PROGRAMMING CONNECTION		553
26.1	Basic Configuration of MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) Serial Programming Connection	554
26.2	Example of Serial Programming Connection (User Power Supply Used)	558

26.3	Example of Serial Programming Connection (Power Supplied from the Programmer)	560
26.4	Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)	562
26.5	Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)	564
APPENDIX		567
APPENDIX A I/O Maps		568
APPENDIX B Instructions		583
B.1	Instruction Types	584
B.2	Addressing	585
B.3	Direct Addressing	587
B.4	Indirect Addressing	593
B.5	Execution Cycle Count	600
B.6	Effective address field	603
B.7	How to Read the Instruction List	604
B.8	F ² MC-16LX Instruction List	607
B.9	Instruction Map	621
APPENDIX C Timing Diagrams in Flash Memory Mode		643
APPENDIX D List of Interrupt Vectors		651
INDEX		655

CHAPTER 1

OVERVIEW

The MB90340 Series is a family member of the F²MC-16LX micro controllers.

- 1.1 "Product Overview"
- 1.2 "Features"
- 1.3 "Block Diagram MB90V340"
- 1.4 "Block Diagram Flash/Mask ROM version"
- 1.5 "Pin Assignment"
- 1.6 "Package Dimensions"
- 1.7 "Pin Functions"
- 1.8 "Input-Output Circuits"
- 1.9 "Handling Device"

1.1 Product Overview

Table 1.1-1 “Product overview” provides a quick outlook of the MB90340 Series.

■ Product overview

Table 1.1-1 Product overview

Features	MB90V340(S)	MB90F342/C(S),MB90F343/C(S) MB90F345/C(S),MB90F346A(S) MB90F346CA(S),MB90F347A(S) MB90F347CA(S),MB90F349/C(S)	MB90341/C(S),MB90342/C(S), MB90346A(S),MB90346CA(S), MB90347A(S),MB90347CA(S), MB90348/C(S),MB90349/C(S)
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, x6, 1/2 when PLL stops) Minimum instruction execution time: 42 ns (4 MHz osc. PLL x 6)		
ROM memory	External	Boot-block, Flash memory 64K bytes : MB90F346A(S),MB90F346CA(S) 128K bytes : MB90F347A(S),MB90F347CA(S) 256K bytes : MB90F342/C(S),MB90F349/C(S) 384K bytes : MB90F343/C(S) 512K bytes : MB90F345/C(S)	Boot-block, Mask ROM 64K bytes : MB90346A(S),MB90346CA(S) 128K bytes : MB90341/C(S),MB90347A(S), MB90347CA(S),MB90348/C(S) 256K bytes : MB90342/C(S),MB90349/C(S)
RAM	30 Kbytes	2K bytes : MB90F346A(S),MB90F346CA(S) 6K bytes : MB90F347A(S),MB90F347CA(S) 16 Kbytes : MB90F342/C(S),MB90F349/C(S) 20K bytes : MB90F343/C(S),MB90F345/C(S)	2K bytes : MB90346A(S),MB90346CA(S) 6K bytes : MB90347A(S),MB90347CA(S) 16 Kbytes : MB90341/C(S),MB90342/C(S), MB90348/C(S),MB90349/C(S)
Technology	0.35μm CMOS with on-chip voltage regulator for internal power supply	0.35μm CMOS with on-chip voltage regulator for internal power supply + Flash memory with on-chip charge pump for programming voltage	0.35μm CMOS with on-chip voltage regulator for internal power supply
Package	PGA-299	QFP-100, LQFP-100	

1.2 Features

Table 1.2-1 “MB90340 features” lists the features of the MB90340 series.

■ Features

Table 1.2-1 MB90340 features (1/2)

Parameter	Part Number	
	MB90F342/C(S), MB90F343/C(S), MB90F345/C(S), MB90F346A(S), MB90F346CA(S), MB90F347A(S), MB90F347CA(S), MB90F349/C(S), MB90341/C(S), MB90342/C(S), MB90346A(S), MB90346CA(S), MB90347A(S), MB90347CA(S), MB90348/C(S), MB90349/C(S)	MB90V340(S)
UART	4 channels	5 channels
	Wide range of baud rate settings using a dedicated reload timer Special synchronous options for adapting to different synchronous serial protocols LIN functionality working either as master or slave LIN device	
I ² C (400Kbit/s)	devices with 'C'-suffix: 2ch devices without 'C'-suffix: --	2 channels
A/D Converter	devices with 'C'-suffix: 24ch devices without 'C'-suffix: 16ch	24 input channels
	10-bit or 8-bit resolution Conversion time: Minimum 3 μ s include sample time (per one channel)	
16-bit Reload Timer (4 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	
16-bit I/O Timer (2 channels)	Signals an interrupt when overflowing Supports Timer Clear when a match with Output Compare (Channel 4) Operation clock freq.: f_{sys} , $f_{sys}/2^1$, $f_{sys}/2^2$, $f_{sys}/2^3$, $f_{sys}/2^4$, $f_{sys}/2^5$, $f_{sys}/2^6$, $f_{sys}/2^7$ (f_{sys} = System clock freq.) I/O Timer 0 (clock input FRCK0) corresponds to ICU 0/1/2/3, OCU 0/1/2/3 I/O Timer 1 (clock input FRCK1) corresponds to ICU 4/5/6/7, OCU 4/5/6/7	
16-bit Output Compare (8 channels)	Signals an interrupt when a match with 16-bit I/O Timer 16-bit compare registers. A pair of compare registers can be used to generate an output signal.	
16-bit Input Capture (8 channels)	Rising edge, falling edge or rising & falling edge sensitive Signals an interrupt upon external event	
8/16-bit Programmable Pulse Generator (8 channels)	Supports 8-bit and 16-bit operation modes Sixteen 8-bit reload counters Sixteen 8-bit reload registers for L pulse width Sixteen 8-bit reload registers for H pulse width 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 Operation clock freq.: f_{sys} , $f_{sys}/2^1$, $f_{sys}/2^2$, $f_{sys}/2^3$, $f_{sys}/2^4$ or $102.4\mu s$ $f_{osc}=@5MHz$ (f_{sys} = System clock frequency, f_{osc} = Oscillation clock frequency)	

Table 1.2-1 MB90340 features (2/2)

Parameter	Part Number	
	MB90F342/C(S), MB90F343/C(S), MB90F345/C(S), MB90F346A(S), MB90F346CA(S), MB90F347A(S), MB90F347CA(S), MB90F349/C(S), MB90341/C(S), MB90342/C(S), MB90346A(S), MB90346CA(S), MB90347A(S), MB90347CA(S), MB90348/C(S), MB90349/C(S)	MB90V340(S)
CAN Interface	2 ch: MB90F342/C(S), MB90F343/C(S), MB90F345/C(S), MB90341/C(S), MB90342/C(S) 1 ch: MB90F346A(S), MB90F346CA(S), MB90F347A(S), MB90F347CA(S), MB90F349/C(S), MB90346A(S), MB90346CA(S), MB90347A(S), MB90347CA(S), MB90348/C(S), MB90349/C(S)	3 channels
	Conforms to CAN Specification Version 2.0 Part A and B Automatic re-transmission in case of error Automatic transmission responding to Remote Frame Prioritized 16 message buffers for data and ID's Supports multiple messages Flexible configuration of acceptance filtering: Full bit compare / Full bit mask / Two partial bit masks Supports up to 1Mbps	
External Interrupt (16 channels)	Can be programmed edge sensitive or level sensitive	
D/A converter	-	2 channels
Up to 100kHz Subclock for low power operation	devices without 'S'-suffix: yes devices with 'S'-suffix: --	
I/O Ports	Virtually all external pins can be used as general purpose I/O All push-pull outputs Bit-wise programmable as input/output or peripheral signal Programmable in groups of 8 as CMOS / Automotive inputs (default) TTL input level programmable for external bus (default for external reset vector fetch)	
Flash Memory	Supports automatic programming, Embedded Algorithm ^{TM*} Write/Erase/Erase-Suspend/Resume commands A flag indicating completion of the algorithm Number of erase cycles: 10,000 times Data retention time: 10 years Boot block configuration Erase can be performed on each block Block protection with external programming voltage Flash Security Feature for protecting the content of the Flash (MB90F346A)	
ROM security	Protects ROM contents. (MASK ROM product only)	

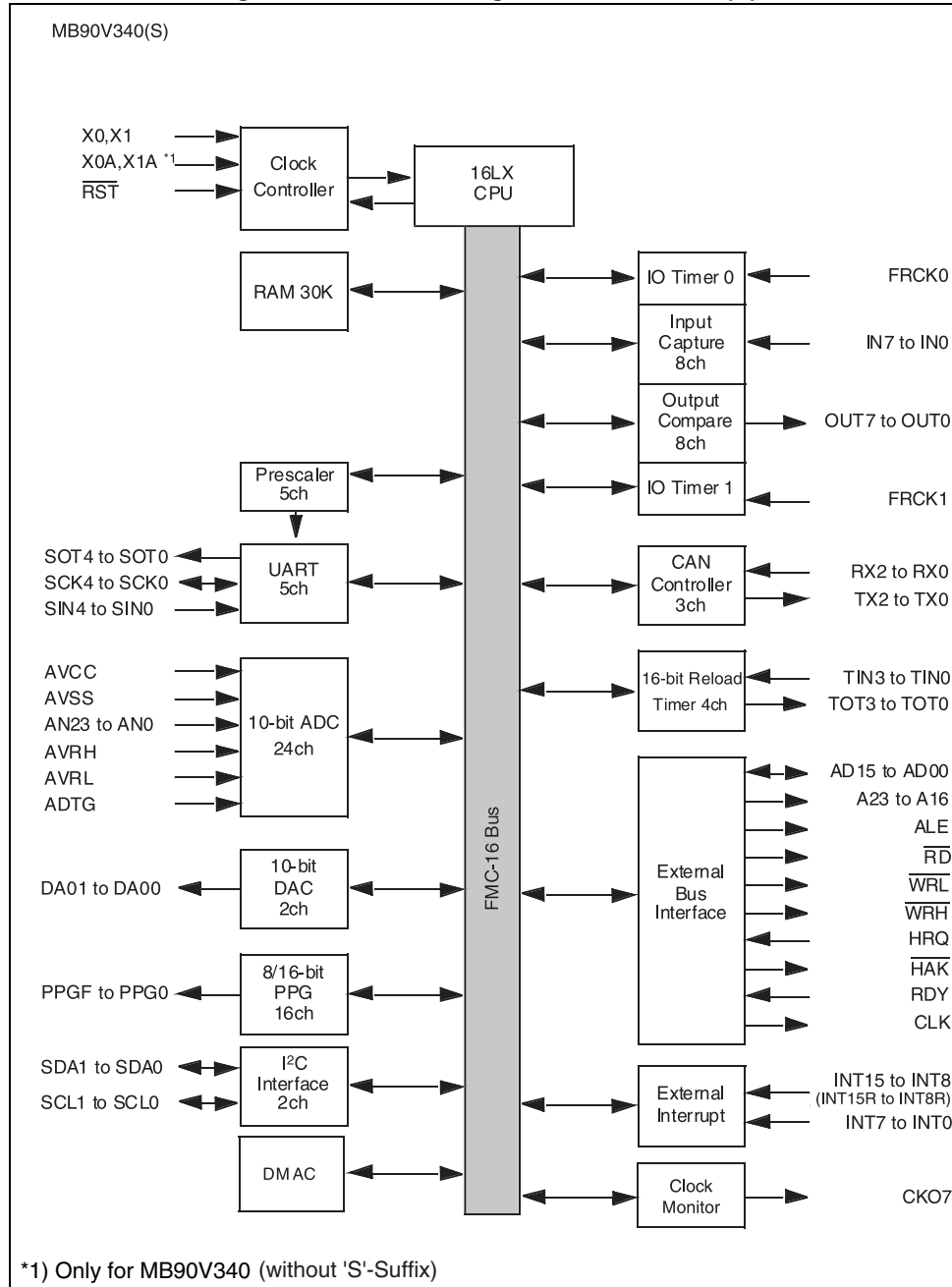
*: Embedded Algorithm is a trade mark of Advanced Micro Devices Inc.

1.3 Block Diagram of MB90V340(S)

Figure 1.3-1 “Block diagram of MB90V340(S)” shows a block diagram of the MB90V340(S).

■ Block diagram of MB90V340(S)

Figure 1.3-1 Block diagram of MB90V340(S)

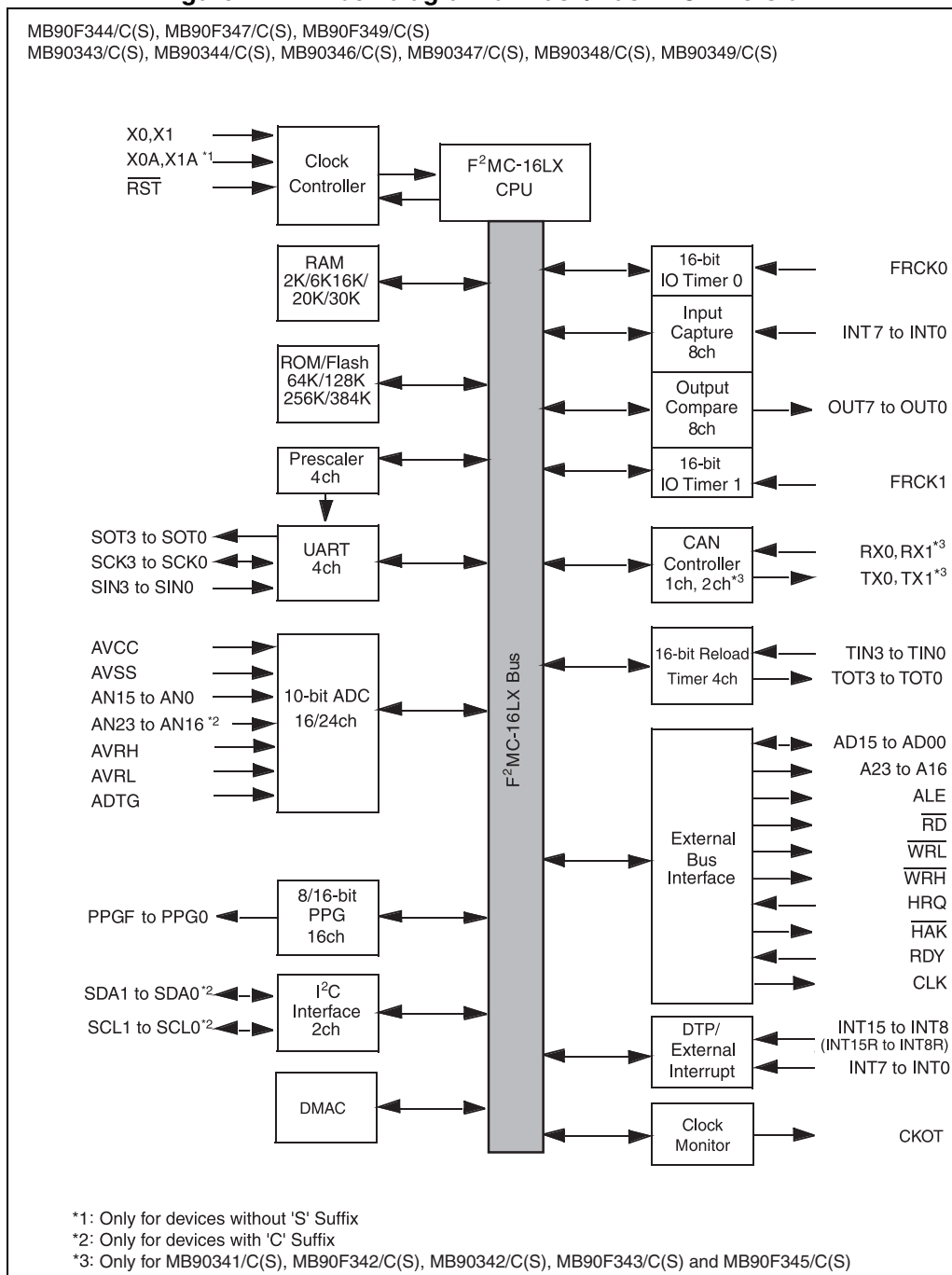


1.4 Block Diagram of Flash/Mask ROM version

Figure 1.4-1 “Block diagram of Flash/Mask ROM version” shows a block diagram of Flash/Mask ROM version.

■ Block diagram of Flash/Mask ROM version

Figure 1.4-1 Block diagram of Flash/Mask ROM version



1.5 Pin Assignment

This chapter shows the pin assignments for the MB90340 series.

■ Pin assignment

- Products without 'C' - suffix

Figure 1.5-1 Pin assignment of Flash/Mask ROM version for QFP-100

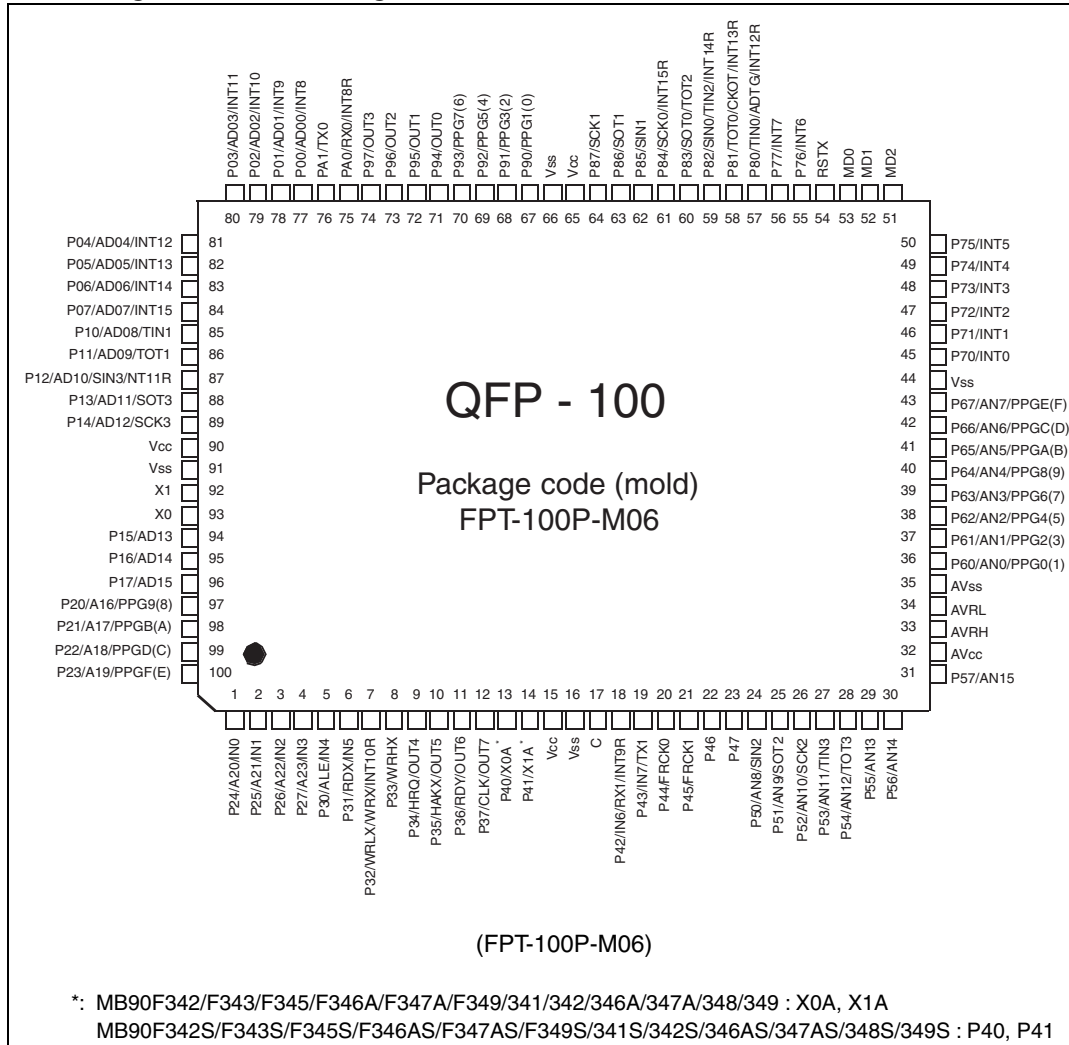
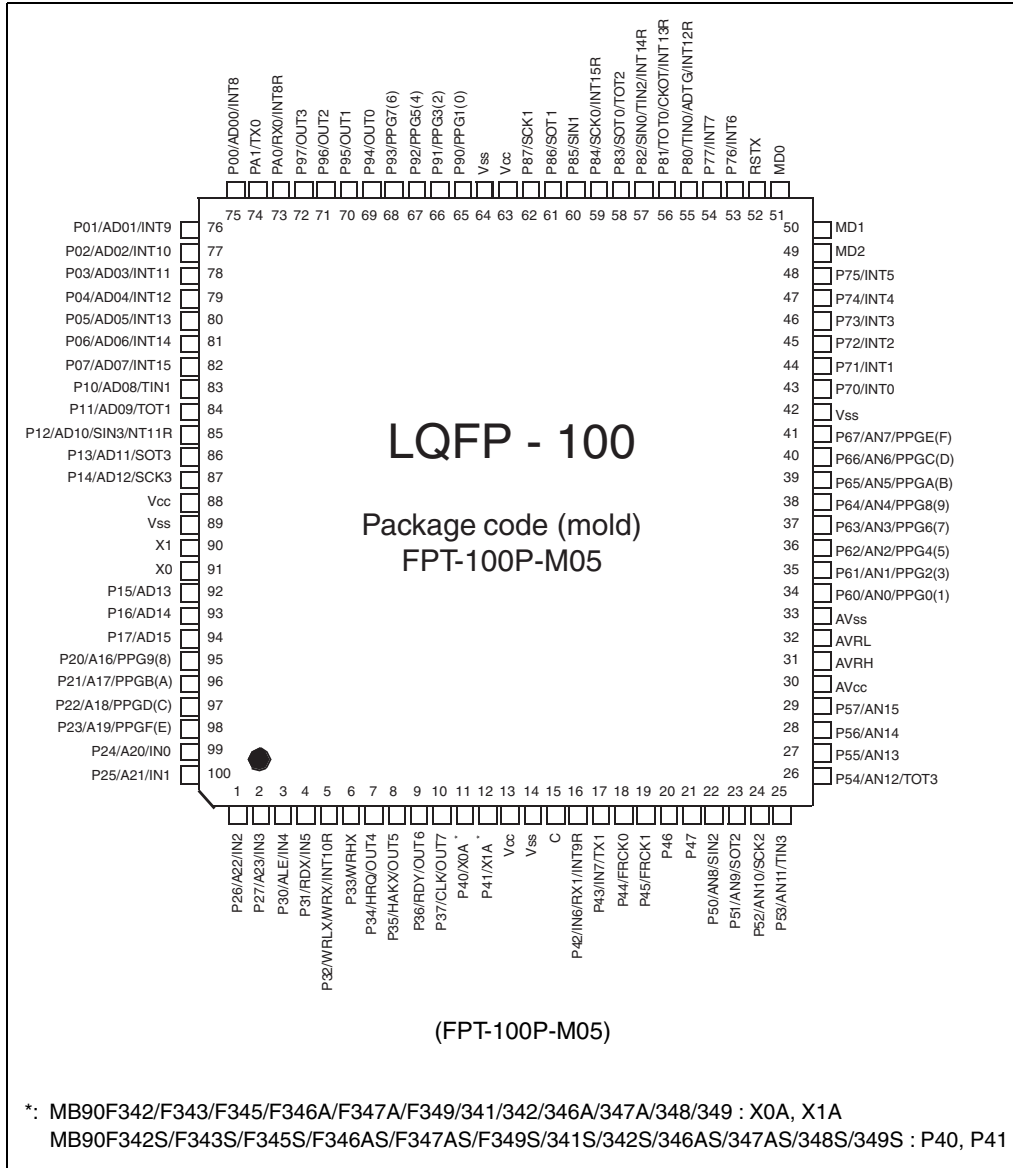


Figure 1.5-2 Pin assignment of Flash/Mask ROM version for LQFP-100



*: MB90F342/F343/F345/F346A/F347A/F349/341/342/346A/347A/348/349 : X0A, X1A
MB90F342S/F343S/F345S/F346AS/F347AS/F349S/341S/342S/346AS/347AS/348S/349S : P40, P41

- Products without 'C' - suffix

Figure 1.5-3 Pin assignment of Flash/Mask ROM version for QFP-100

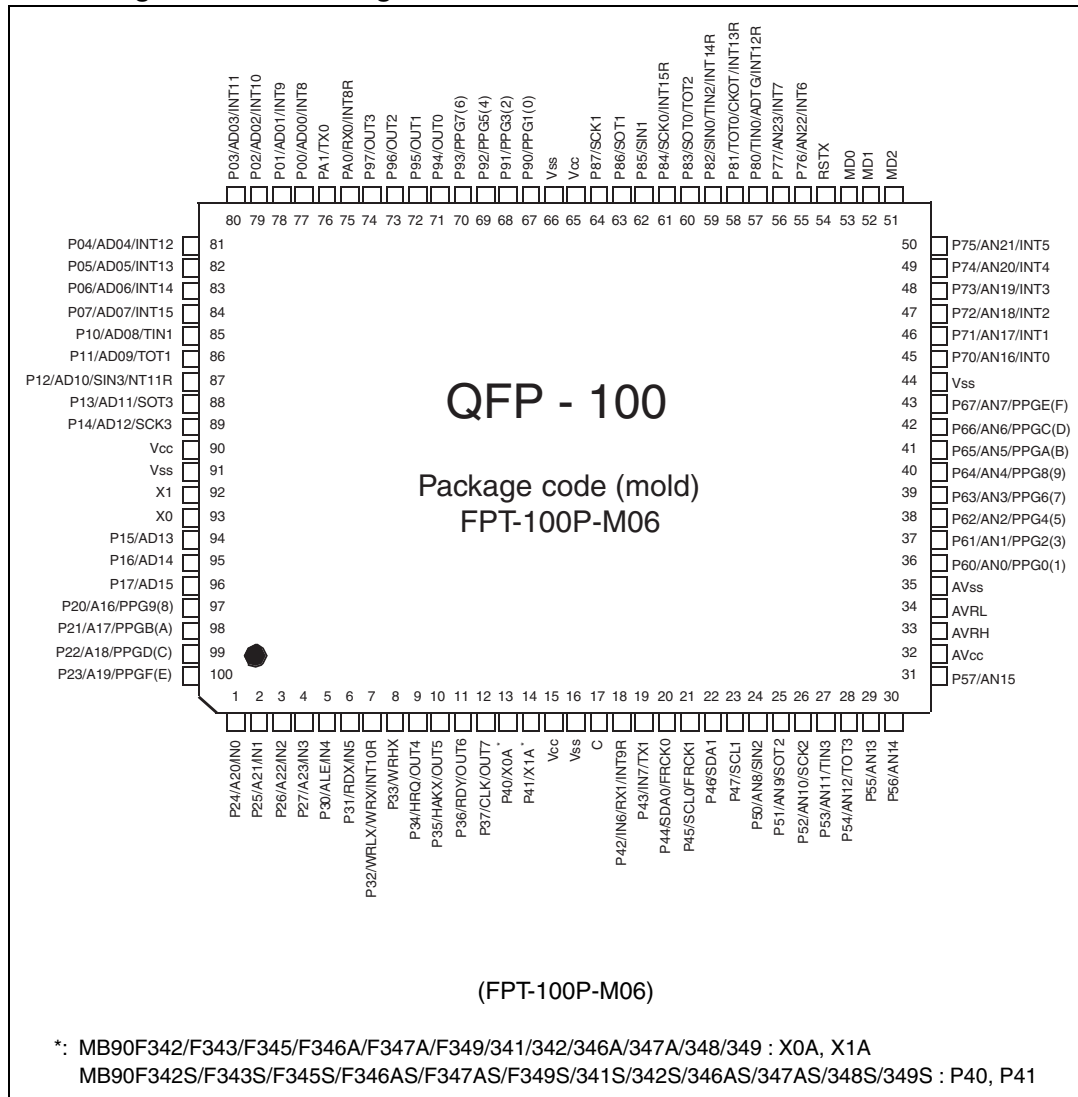
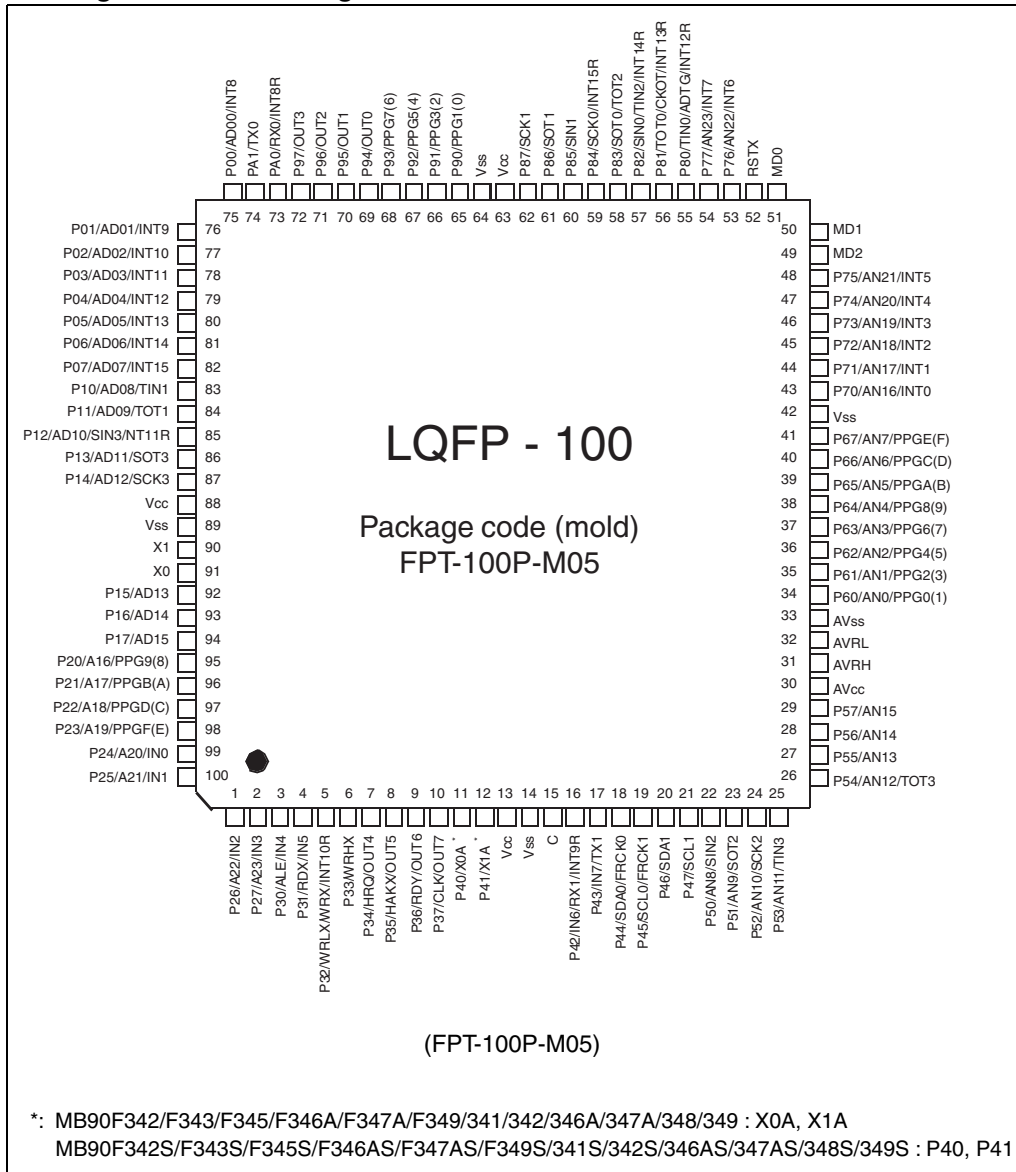


Figure 1.5-4 Pin assignment of Flash/Mask ROM version for LQFP-100



1.6 Package Dimensions

Figure 1.6-1 “Package dimensions” shows the package dimensions of the MB90340 series.

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

■ Package dimensions

Figure 1.6-1 FPT-100P-M05+ Package dimensions

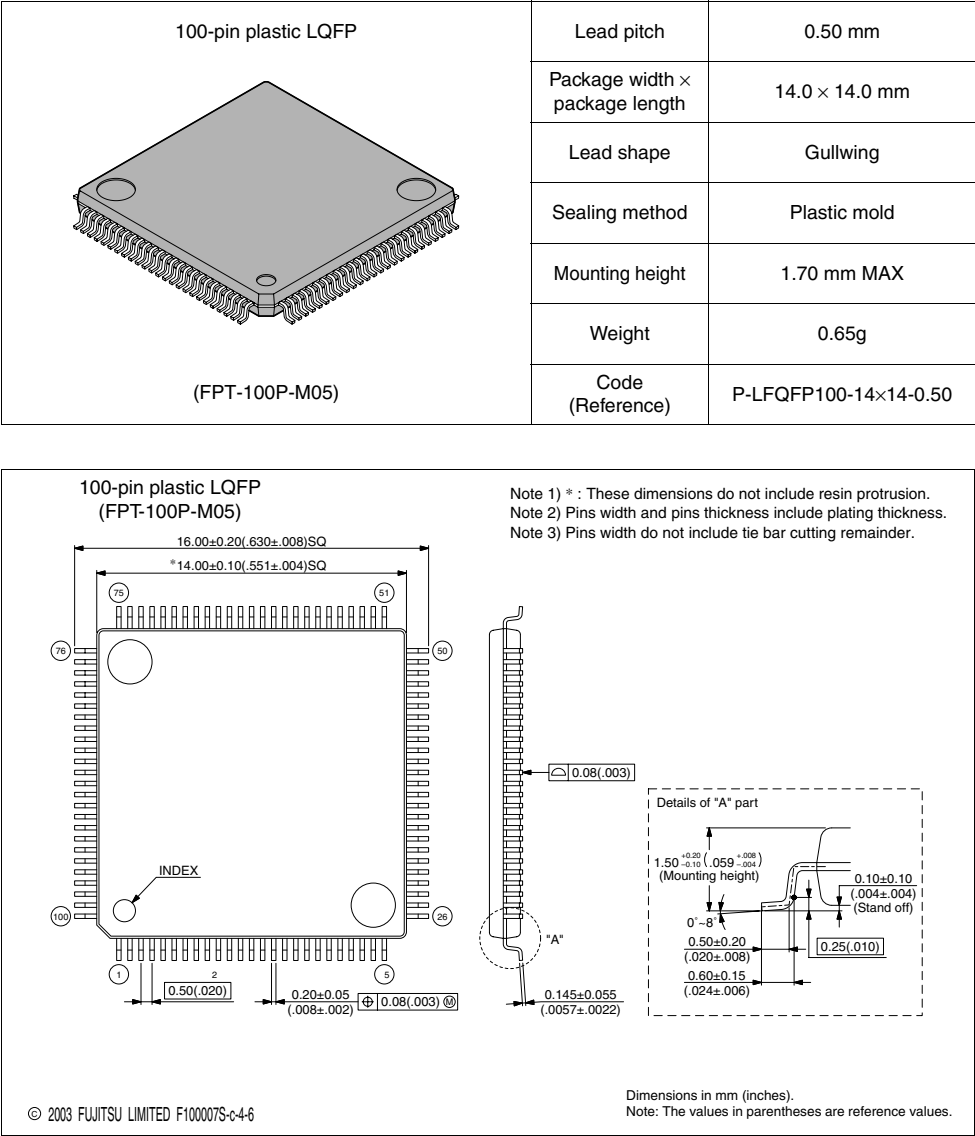
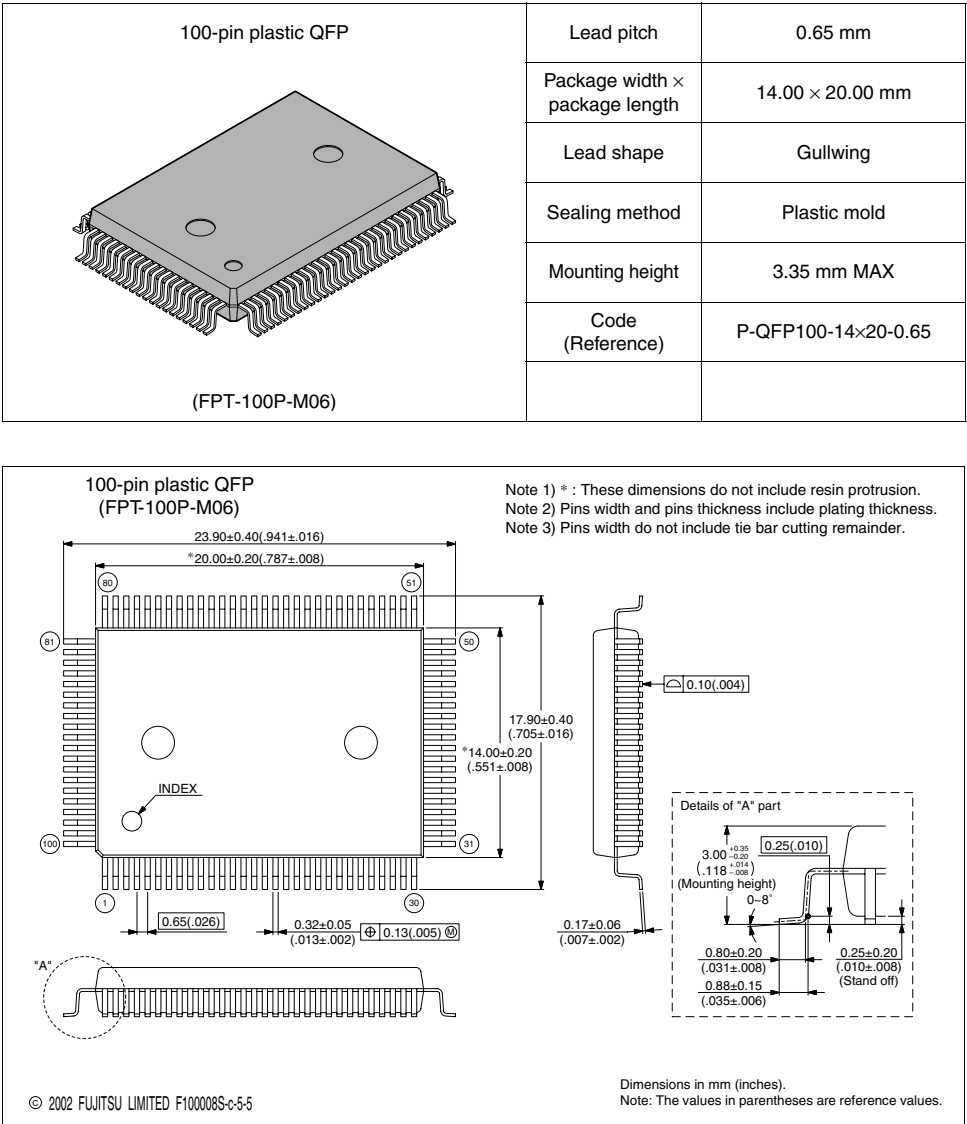


Figure 1.6-2 FPT-100P-M06 Package dimensions



1.7 Pin Functions

Table 1.7-1 “Pin description” describes the pin functions of the MB90340 series.

■ Pin functions

Table 1.7-1 Pin description (1/7)

Pin no.		Pin name	Circuit type	Function
LQFP100 ^{*2}	QFP100 ^{*1}			
90	92	X1	A	Oscillation output
91	93	X0		Oscillation input
52	54	$\overline{\text{RST}}$	E	Reset input
75 to 82	77 to 84	P00 to P07	G	General purpose I/O
		AD00 to AD07		I/O pins for lower 8 bits of external address/data bus. These functions is enabled when the external bus is enabled.
		INT8 to INT15		External interrupt request input pins for INT8 to INT15
83	85	P10	G	General purpose I/O
		AD08		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		TIN1		Event input pin for the reload timer 1.
84	86	P11	G	General purpose I/O
		AD09		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		TOT1		Output pin for the reload timer 1.
85	87	P12	N	General purpose I/O
		AD10		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		SIN3		Serial data input pin for UART3.
		INT11R		Second external interrupt request input pin for INT11.
86	88	P13	G	General purpose I/O
		AD11		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		SOT3		Serial data output pin for UART3.

Table 1.7-1 Pin description (2/7)

Pin no.		Pin name	Circuit type	Function
LQFP100*2	QFP100*1			
87	89	P14	G	General purpose IO
		AD12		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		SCK3		Clock I/O pin for UART3.
92	94	P15	N	General purpose IO
		AD13		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		SIN4		Serial data input pin for UART4 (MB90V340 only).
93	95	P16	G	General purpose IO
		AD14		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		SOT4		Serial data output pin for UART4 (MB90V340 only).
94	96	P17	G	General purpose IO
		AD15		I/O pin for the external address/data bus. This function is enabled when the external bus is enabled.
		SCK4		Clock I/O pin for UART4 (MB90V340 only).
95 to 98	97 to 100	P20 to P23	G	General purpose IO
		A16 to A19		Output pins for A16 to A17 of the external address bus. This function is enabled when the external bus is enabled.
		PPG9,PPGB,P PGD,PPGF		Output pins for PPGs.
99 to 2	1 to 4	P24 to P27	G	General purpose IO
		A20 to A23		Output pins for A20 to A23 of the external address bus. This function is enabled when the external bus is enabled.
		IN0 to IN3		Data sample input pins for input captures ICU0 to ICU3.
3	5	P30	G	General purpose IO
		ALE		Address latch enable output pin. This function is enabled when the external bus is enabled.
		IN4		Data sample input pin for input capture ICU4.
4	6	P31	G	General purpose IO
		RDX		Read strobe output pin for the data bus. This function is enabled when the external bus is enabled.
		IN5		Data sample input pin for input capture ICU5.

Table 1.7-1 Pin description (3/7)

Pin no.		Pin name	Circuit type	Function
LQFP100* ²	QFP100* ¹			
5	7	P32	G	General purpose IO
		WRLX /WRX		Write strobe output pin for the data bus. This function is enabled when both the external bus and the WR/WRL pin output are enabled. WRL is used to write-strobe 8 lower bits of the data bus in 16-bit access while WR is used to write-strobe 8 bits of the data bus in 8-bit access.
		RX2		RX input pin for CAN2 Interface (MB90V340 only).
		INT10R		Second external interrupt request input pin for INT10.
6	8	P33	G	General purpose IO
		WRHX		Write strobe output pin for the 8 higher bits of the data bus. This function is enabled when the external bus is enabled, when the external bus 16-bit mode is selected, and when the WRH output pin is enabled.
		TX2		TX Output pin for CAN2 (MB90V340 only).
7	9	P34	G	General purpose IO
		HRQ		Hold request input pin. This function is enabled when both the external bus and the hold function are enabled.
		OUT4		Waveform output pin for output compares OCU4.
8	10	P35	G	General purpose IO
		HAKX		Hold acknowledge output pin. This function is enabled when both the external bus and the hold function are enabled.
		OUT5		Waveform output pin for output compares OCU5.
9	11	P36	G	General purpose IO
		RDY		Ready input pin. This function is enabled when both the external bus and the external ready function are enabled.
		OUT6		Waveform output pin for output compares OCU6.
10	12	P37	G	General purpose IO
		CLK		CLK output pin. This function is enabled when both the external bus and CLK output are enabled.
		OUT7		Waveform output pin for output compares OCU7.

Table 1.7-1 Pin description (4/7)

Pin no.		Pin name	Circuit type	Function
LQFP100*2	QFP100*1			
11 to 12	13 to 14	P40 to P41	F	General purpose IO (only for devices with S-suffix)
		X0A, X1A	B	Oscillator input pins for sub-clock (only for devices without S-suffix)
16	18	P42	F	General purpose IO
		IN6		Data sample input pin for input capture ICU6.
		RX1		RX input pin for CAN1 Interface (MB90(F)343/(F)344 only).
		INT9R		Second external interrupt request input pin for INT10.
17	19	P43	F	General purpose IO
		IN7		Data sample input pin for input capture ICU7.
		TX1		TX Output pin for CAN1 (MB90(F)343/(F)344 only).
18	20	P44	H	General purpose IO
		SDA0		Serial data I/O pin for I ² C 0 (only devices with C-suffix)
		FRCK0		Input for the 16-bit IO Timer 0
19	21	P45	H	General purpose IO
		SCL0		Serial clock I/O pin for I ² C 0 (only devices with C-suffix)
		FRCK1		Input for the 16-bit IO Timer 1
20	22	P46	H	General purpose IO
		SDA1		Serial data I/O pin for I ² C 1 (only devices with C-suffix)
21	23	P47	H	General purpose IO
		SCL1		Serial clock I/O pin for I ² C 1 (only devices with C-suffix)
22	24	P50	O	General purpose IO
		AN8		Analog input pin for the A/D converter
		SIN2		Serial data input pin for UART2.
23	25	P51	I	General purpose IO
		AN9		Analog input pin for the A/D converter
		SOT2		Serial data output pin for UART2.
24	26	P52	I	General purpose IO
		AN10		Analog input pin for the A/D converter
		SCK2		Clock I/O pin for UART2.

Table 1.7-1 Pin description (5/7)

Pin no.		Pin name	Circuit type	Function
LQFP100* ²	QFP100* ¹			
25	27	P53	I	General purpose IO
		AN11		Analog input pin for the A/D converter
		TIN3		Event input pin for the reload timer 3.
26	28	P54	I	General purpose IO
		AN12		Analog input pin for the A/D converter
		TOT3		Output pin for the reload timer 3.
27	29	P55	I	General purpose IO
		AN13		Analog input pin for the A/D converter
28 to 29	30 to 31	P56 to P57	J	General purpose IO
		AN14 to AN15		Analog input pins for the A/D converter
		DA00 to DA01		D/A converter analog output pins.
34 to 41	36 to 43	P60 to P67	I	General purpose IO
		AN0 to AN7		Analog input pins for the A/D converter.
		PPG0, 2, 4, 6, 8, A, C, E		Output pins for PPGs.
43 to 48, 53, 54	45 to 50, 55, 56	P70 to P77	I	General purpose IO
		AN16 to AN23		Analog input pins for the A/D converter (only devices with C-suffix)
		INT0 to INT7		External interrupt request input pins for INT0 to INT7.
55	57	P80	F	General purpose IO
		TIN0		Event input pin for the reload timer 0.
		ADTG		Trigger input pin for the A/D converter.
		INT12R		Second external interrupt request input pin for INT12.
56	58	P81	F	General purpose IO
		TOT0		Output pin for the reload timer 0.
		CKOT		Output pin for the clock monitor.
		INT13R		Second external interrupt request input pin for INT13.

Table 1.7-1 Pin description (6/7)

Pin no.		Pin name	Circuit type	Function
LQFP100*2	QFP100*1			
57	59	P82	M	General purpose IO
		SIN0		Serial data input pin for UART0.
		TIN2		Event input pin for the reload timer 2.
		INT14R		Second external interrupt request input pin for INT14.
58	60	P83	F	General purpose IO
		SOT0		Serial data output pin for UART0.
		TOT2		Output pin for the reload timer 2.
59	61	P84	F	General purpose IO
		SCK0		Clock I/O pin for UART0.
		INT15R		Second external interrupt request input pin for INT15.
60	62	P85	M	General purpose IO
		SIN1		Serial data input pin for UART1.
61	63	P86	F	General purpose IO
		SOT1		Serial data output pin for UART1.
62	64	P87	F	General purpose IO
		SCK1		Clock I/O pin for UART1.
65 to 68	67 to 70	P90 to P93	F	General purpose IO
		PPG1, 3, 5, 7		Output pins for PPGs.
69 to 72	71 to 74	P94 to P97	F	General purpose IO
		OUT0 to OUT3		Waveform output pins for output compares OCU0 to OCU3. This function is enabled when the OCU enables waveform output.
73	75	PA0	F	General purpose IO
		RX0		RX input pin for CAN0 Interface. When the CAN function is used, output from the other functions must be stopped.
		INT8R		Second external interrupt request input pin for INT8.
74	76	PA1	F	General purpose IO
		TX0		TX Output pin for CAN0.
30	32	AVCC	K	Vcc power input pin for analog circuits

Table 1.7-1 Pin description (7/7)

Pin no.		Pin name	Circuit type	Function
LQFP100*2	QFP100*1			
31	33	AVRH	L	Reference voltage input for the A/D Converter. Supply voltage to AVCC pin must be kept higher than or equal to AVRH pin voltage specially when the supply voltage to AVRH is turned on or off.
32	34	AVRL	K	Lower reference voltage input for the A/D Converter
33	35	AVSS	K	Vss power input pin for analog circuits
50 to 51	52 to 53	MD1 to MD0	C	Input pins for specifying the operating mode. The pins must be directly connected to Vcc or Vss
49	51	MD2	D	Input pin for specifying the operating mode. The pins must be directly connected to Vcc or Vss
13 63 88	15 65 90	VCC		Power (3.5V to 5.5V) input pins
13 42 64 89	16 44 66 91	VSS		Power (0V) input pins
15	17	C	K	This is the power supply stabilization capacitor pin. It should be connected to a higher than or equal to 0.1μF ceramic capacitor.

*1: FPT-100P-M06

*2: FPT-100P-M05

1.8 Input-Output Circuits

Table 1.8-1 “I/O circuit types” lists the input-output circuits.

■ Input-output circuits

Table 1.8-1 I/O circuit types (1/5)

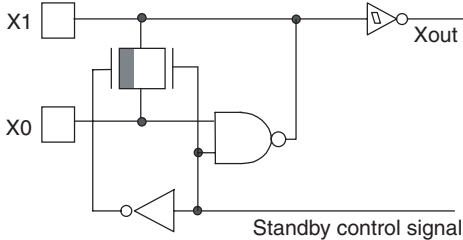
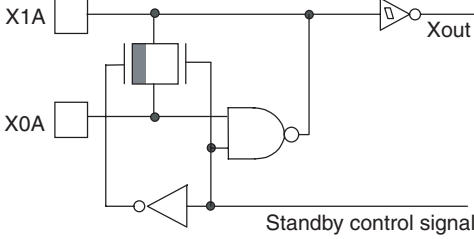
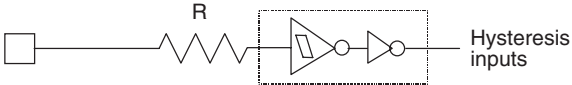
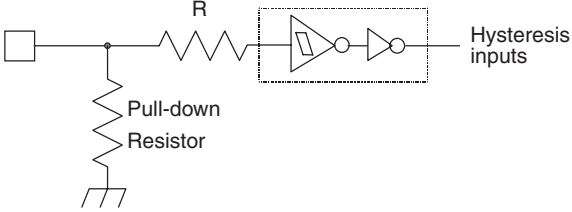
Type	Circuit	Remarks
A		Oscillation circuit • High-speed oscillation feedback resistor = approx. 1 MΩ
B		Oscillation circuit • Low-speed oscillation feedback resistor = approx. 10 MΩ
C		Mask ROM and EVA device: • CMOS Hysteresis input pin Flash device: • CMOS input pin
D		Mask ROM and EVA device: • CMOS Hysteresis input pin • Pull-down resistor value: approx. 50 kΩ Flash device: • CMOS input pin • No Pull-down

Table 1.8-1 I/O circuit types (2/5)

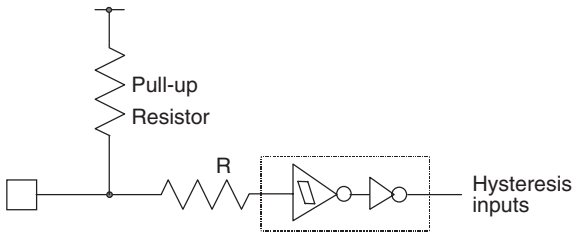
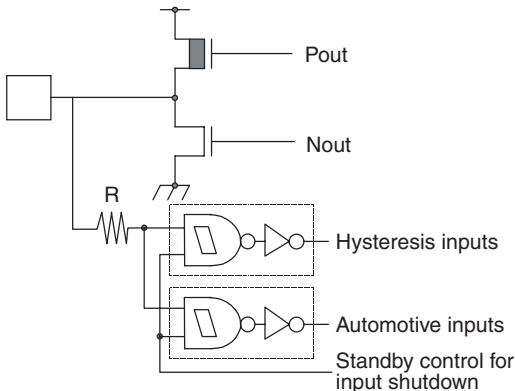
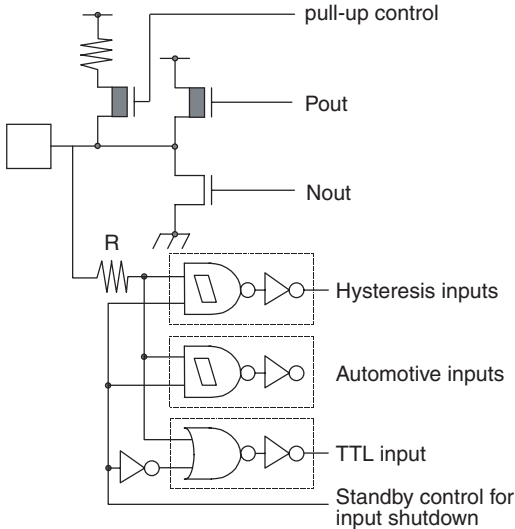
Type	Circuit	Remarks
E		CMOS Hysteresis input pin Pull-up resistor value: approx. 50 kΩ
F		- CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$) - CMOS hysteresis inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function)
G		- CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$) - CMOS hysteresis inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function) - TTL input (With the standby-time input shutdown function) - Programmable pull-up resistor: 50kΩ approx.

Table 1.8-1 I/O circuit types (3/5)

Type	Circuit	Remarks
H		- CMOS level output ($I_{OL} = 3\text{ mA}$, $I_{OH} = -3\text{ mA}$) - CMOS hysteresis inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function)
I		- CMOS level output ($I_{OL} = 4\text{ mA}$, $I_{OH} = -4\text{ mA}$) - CMOS hysteresis inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function) - A/D analog input
J		- CMOS level output ($I_{OL} = 4\text{ mA}$, $I_{OH} = -4\text{ mA}$) - D/A analog output - CMOS hysteresis inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function) - A/D analog input

Table 1.8-1 I/O circuit types (4/5)

Type	Circuit	Remarks
K		Power supply input protection circuit
L		- A/D converter ref+ (AVRH) power supply input pin, With the protection circuit - Flash devices do not have a protection circuit against VCC for pin AVRH
M		- CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$) - CMOS inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function)
N		- CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$) - CMOS inputs (With the standby-time input shutdown function) - Automotive input (With the standby-time input shutdown function) - TTL input (With the standby-time input shutdown function) - Programmable pull-up resistor: 50kΩ approx

Table 1.8-1 I/O circuit types (5/5)

Type	Circuit	Remarks
O		• CMOS level output ($I_{OL} = 4\text{ mA}$, $I_{OH} = -4\text{ mA}$) • CMOS inputs (With the standby-time input shutdown function) • Automotive input (With the standby-time input shutdown function) • A/D analog input

1.9 Handling Device

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

- Preventing latch-up
 - Treatment of unused pins
 - Using external clock
 - Precautions for when not using a sub clock signal
 - Notes on during operation of PLL clock mode
 - 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
 - Note on Energization
 - Initialization
 - Stabilization of power supply voltage
 - Output of Port0 to port3 during power-on (External-bus mode)
 - Note on using CAN Function
 - Flash security Function (except for MB90F346A)
-

■ 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.

For the same reason, also be careful not to let the analog power-supply voltage (AV_{CC} , $AVRH$) exceed the digital power-supply voltage.

● Treatment of unused pins

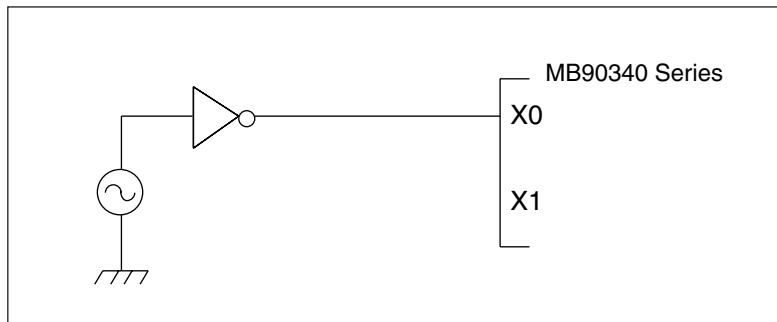
Leaving unused input pins open may result in misbehavior or latch up and possible permanent damage of the device. Therefore they must be pulled up or pulled down through resistors. In this case those resistors should be more than 2 k Ω .

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 1.9-1 Using external clock



● Precautions for when not using a sub clock signal

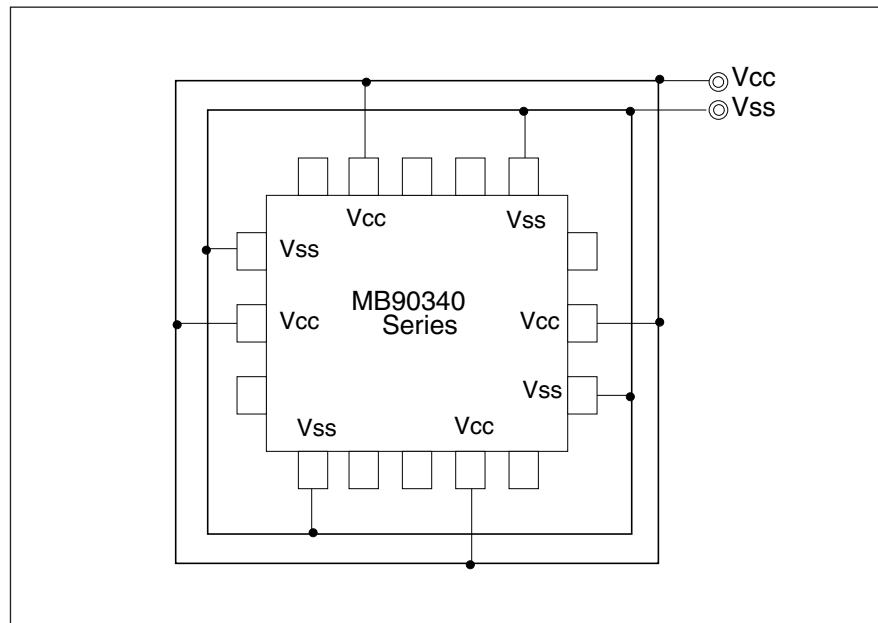
If you do not connect pins X0A and X1A to an oscillator, use pull-down handling on the X0A pin, and leave the X1A pin open.

● 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.

● Power supply pins (V_{CC}/V_{SS})

- If there are multiple V_{CC} and V_{SS} pins, from the point of view of device design, pins to be of the same potential are connected the inside of the device to prevent such malfunctioning as latch up.
To reduce unnecessary radiation, prevent malfunctioning of the strobe signal due to the rise of ground level, and to keep the recommended DC characteristics specified as the total output current, be sure to connect the V_{CC} and V_{SS} pins to the power supply and ground externally.
- Connect V_{CC} and V_{SS} to the device from the power supply with lowest possible impedance.
- As a measure against power supply noise, connect a capacitor of about 0.1 μF as a bypass capacitor between V_{CC} and V_{SS} in the vicinity of V_{CC} and V_{SS} pins of the device

Figure 1.9-2 Power supply pins (V_{CC}/V_{SS})

● Pull-up/down resistors

The MB90340 Series does not support internal pull-up/down resistors (except Port0 to Port3: programmable pull-up 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.

● 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} , AV_{RH} , AV_{RL}) and analog inputs (AN0 to AN23) 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 AV_{RH} 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 as $AV_{CC} = V_{CC}$, $AV_{SS} = AV_{RH} = AV_{RL} = V_{SS}$.

● Notes on Energization

To prevent malfunction of the internal voltage regulator, supply voltage profile while turning on the power supply should be slower than 50 μ s from 0.2 V to 2.7 V.

● Initialization

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

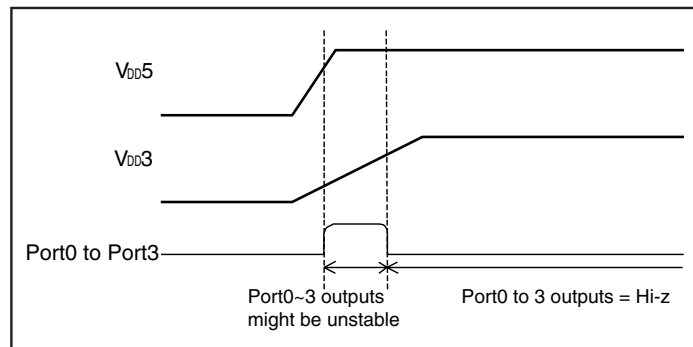
● Stabilization of power supply voltage

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

● Output of Port0 to Port3 during Power-on (External-bus mode)

As shown below, when power is turned on in External-Bus mode, there is a possibility that output signal of Port0 to Port3 might be unstable.

Figure 1.9-3 Output of Port0 to Port3 when power is turning



● Note on using CAN Function

The clock modulation function can't be used, because the function is tested now. So, at using CAN, the DIRECT bit of the CAN direct mode register (CDMR) must be set "1". (See Table 1.9-1). If the DIRECT bit is not set correctly, the device does not operate normally.

Table 1.9-1 Setting of the Clock Modulation and CAN Direct Mode

	Setting of the clock modulation (CMCR: PMOD bit)	Setting of CAN direct mode (CDMR: DIRECT bit)
Enabled setting	0: Disabled the clock modulation (Initial state)	1: Enabled the CAN direct mode
Setting prohibited	1: Enabled the clock modulation	0: Disabled the CAN direct mode (Initial state)

Note: For details on the clock modulation, see "CHAPTER 6 CLOCK MODULATOR".

For details on the CAN direct mode, see "22.13 Setting Configuration of Multi-level Message Buffer".

● Flash security Function (except for MB90F346A)

The security bit is located in the area of the flash memory.

If protection code 01_H is written in the security bit, the flash memory is in the protected state by security.

Therefore please do not write 01_H in this address if you do not use the security function.

Please refer to following table for the address of the security bit.

	Flash memory size	Address for security bit
MB90F347A	Embedded 1 Mbit Flash Memory	FE0001_H
MB90F342 MB90F349	Embedded 2 Mbit Flash Memory	FC0001_H
MB90F343	Embedded 3 Mbit Flash Memory	F90001_H
MB90F345	Embedded 4 Mbit Flash Memory	F80001_H

CHAPTER 2

CPU

This chapter explains the CPU.

- 2.1 “Outline of the 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 the 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 the 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: 42 ns (at 4-MHz oscillation, 6 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
 - Up to 16 channels of DMAC
- 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

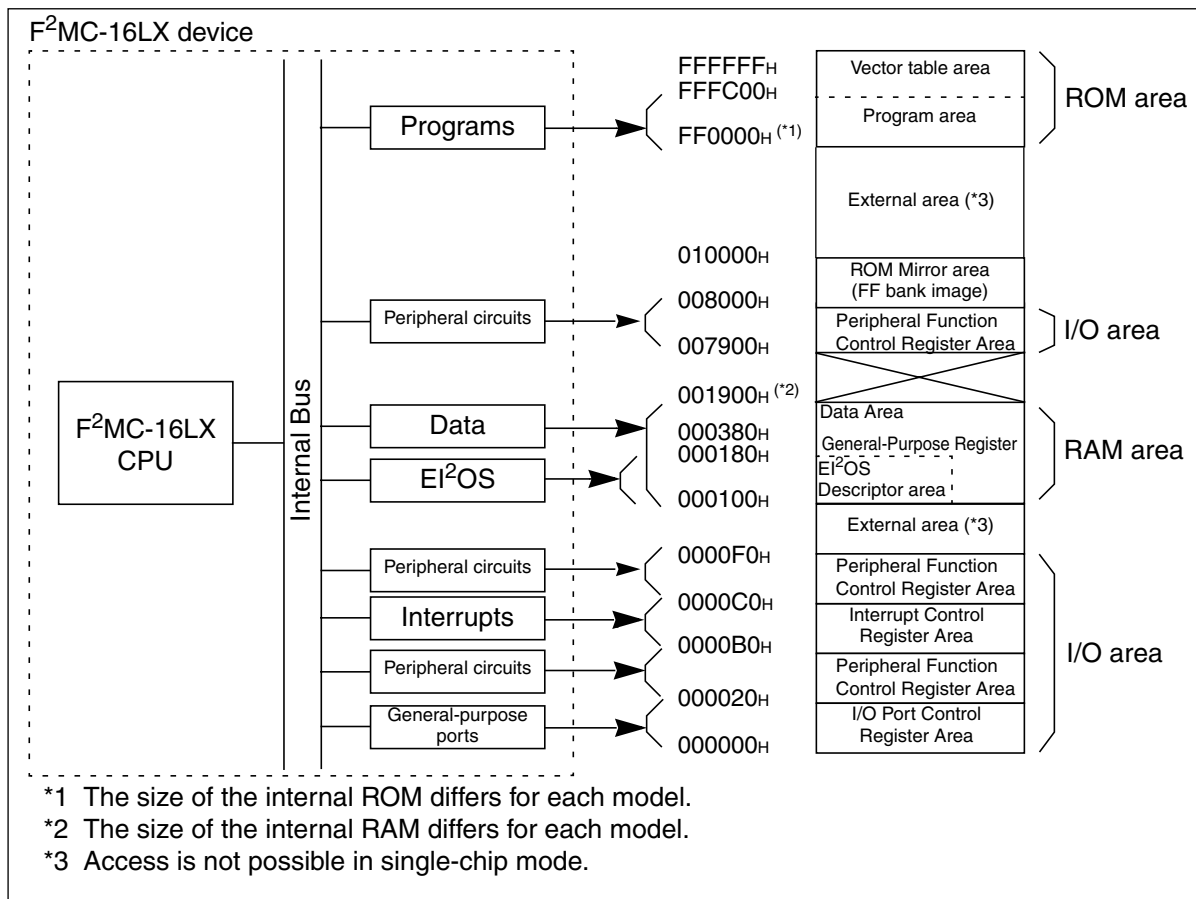
An F²MC-16LX CPU has a 16-Mbyte memory space. All data program input and output managed by the F²MC-16LX CPU are located in this 16-Mbyte memory space. The CPU accesses the resources by indicating their addresses using a 24-bit address bus.

■ Outline of CPU memory space

All I/O, programs and data are located in the 16-megabyte memory space of the F²MC-16LX CPU. The CPU is able to access each resource through an address indicated by the 24-bit address bus.

Figure 2.2-1 “Sample relationship between F²MC-16LX system and memory” shows a sample relationship between the F²MC-16LX system and memory map.

Figure 2.2-1 Sample relationship between F²MC-16LX system and memory



■ ROM area

- Vector table area (address: FFFC00_H to FFFFFFF_H)

This area is used as a vector table for vector call instructions, interrupt vectors, and reset vectors.

This area is allocated at the highest addresses of the ROM area. The start address of the corresponding processing routine is set as data in each vector table address.

- Program area (address: Up to FFFBFF_H)

ROM is built in as an internal program area.

The size of internal ROM differs for each model.

■ RAM area

- Data area (address: From 000100_H to 0018FF_H (for 6KByte))

The static RAM is built in as an internal data area.

The size of internal RAM differs for each model.

- General-purpose register area (address: 000180_H to 00037F_H)

Auxiliary registers used for 8-bit, 16-bit, and 32-bit arithmetic operations and transfer are allocated in this area.

Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.

When this area is used as a general-purpose register, general-purpose register addressing enables high-speed access with short instructions.

- Extended intelligent I/O service (EI²OS) descriptor area (address: 000100_H to 00017F_H)

This area retains the transfer modes, I/O addresses, transfer count, and buffer addresses.

Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.

■ I/O area

- Interrupt control register area (address: 0000B0_H to 0000BF_H)

The interrupt control registers (ICR00 to ICR15) correspond to all peripheral functions that have an interrupt function. These registers set interrupt levels and control the extended intelligent I/O service (EI²OS).

- Peripheral function control register area

(address: 000020_H to 0000AF_H, 0000C0_H to 0000EF_H, 007900_H to 007FFF_H)

This register controls the built-in peripheral functions and inputs and outputs data.

- I/O port control register area (address: 000000_H to 00001F_H)

This register controls I/O ports, and inputs and outputs data.

■ Address generation types

The F²MC-16LX has the following two addressing modes:

- Linear addressing

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

- This register 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 MB90340 Series is shown in Figure 2.3-1 “Memory space map”.

■ Memory space map

The ROM data in the high-order portion of FF-bank can be seen as an image in the higher 00-bank in order to support the small model C compiler. Since the low-order 16 bits are identical, this part of the ROM data can be referenced without using the far specification in the pointer declaration.

For example, when 00C000_H is accessed, the contents of ROM at FFC000_H are read. However, since the ROM area in the FF bank exceeds 32 Kbytes, its entire image cannot be mirrored in the 00 bank.

The image between FF8000_H and FFFFFFF_H is visible in bank 00, whereas the data between FF0000_H and FF7FFF_H is only visible in bank FF.

Figure 2.3-1 Memory Space Map (1/2)

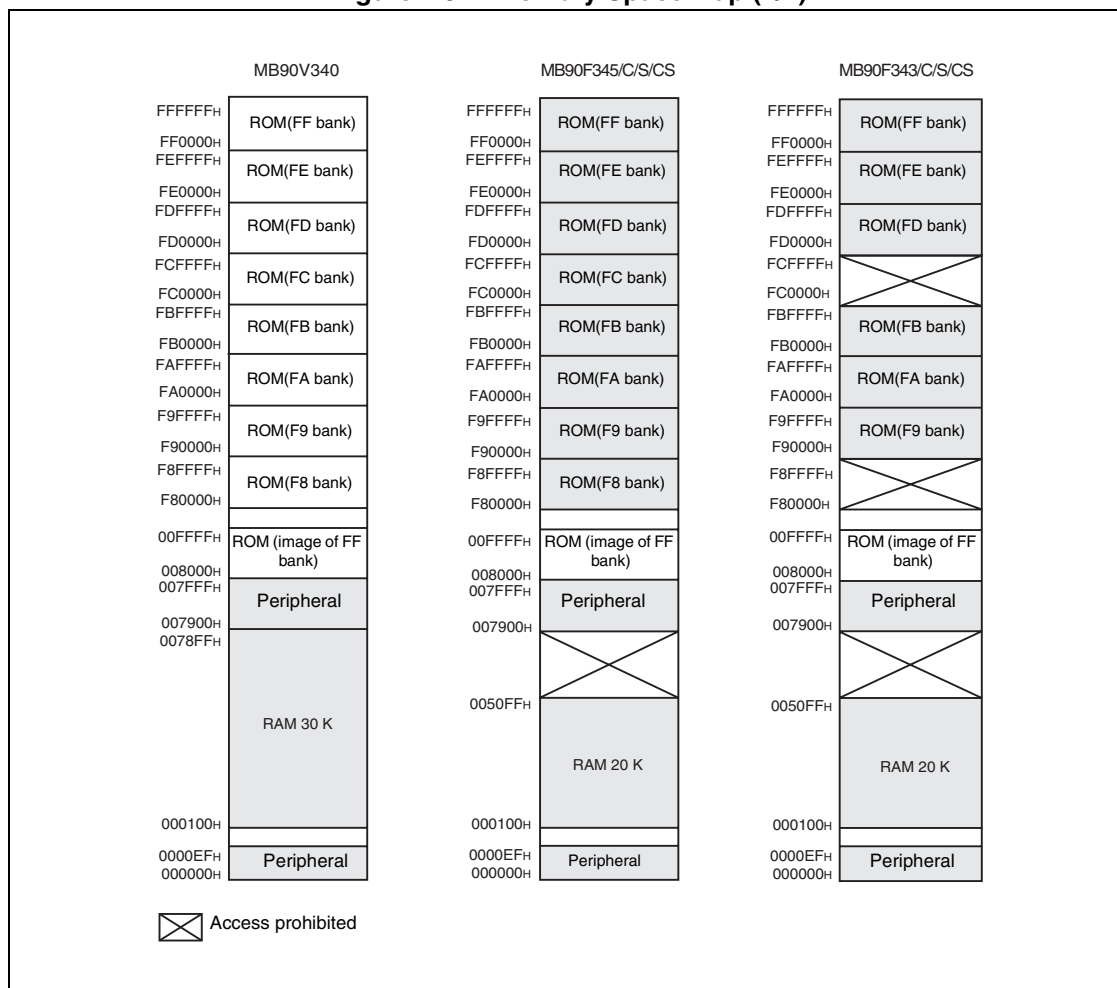
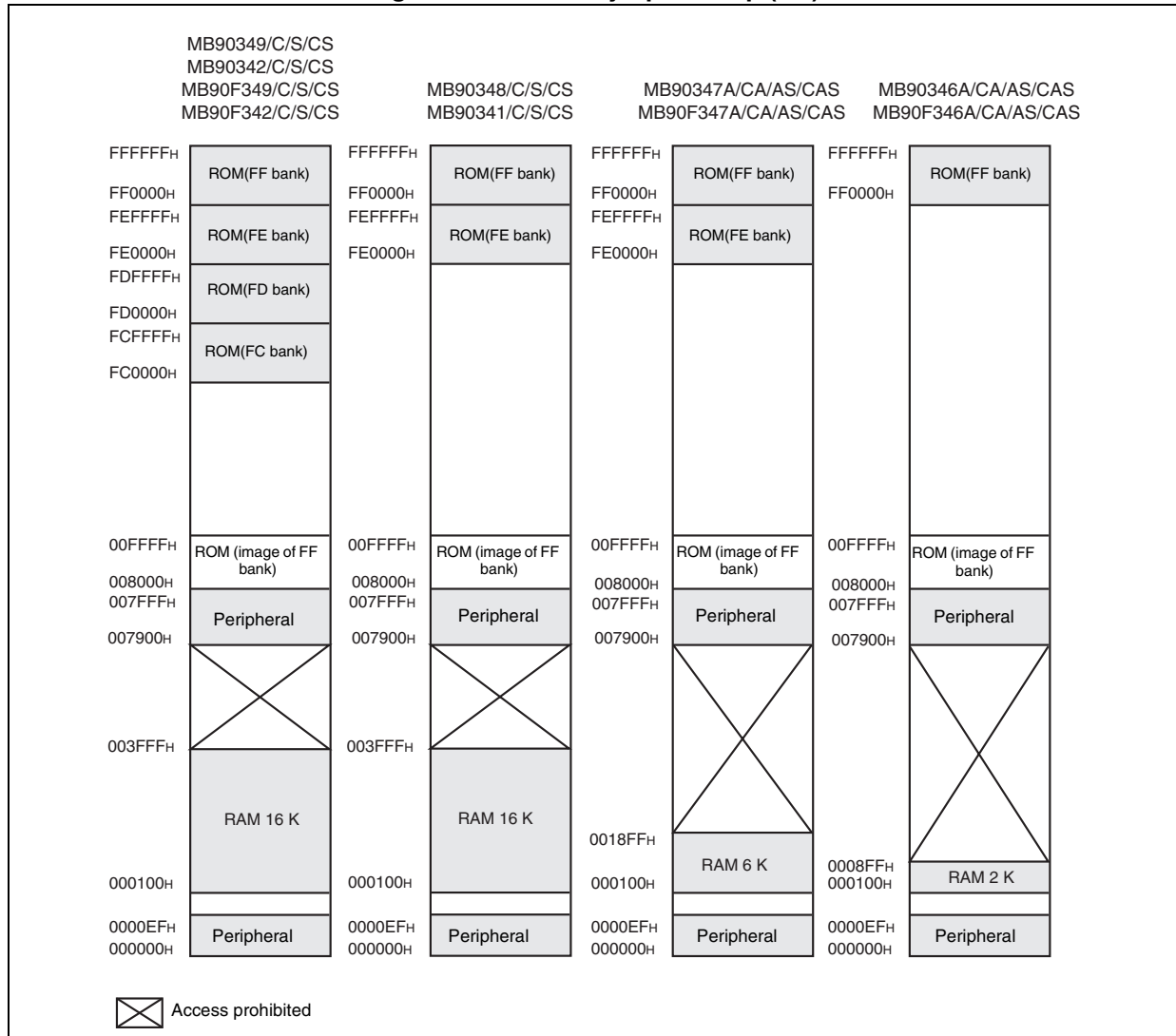


Figure 2.3-1 Memory Space Map (2/2)

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)

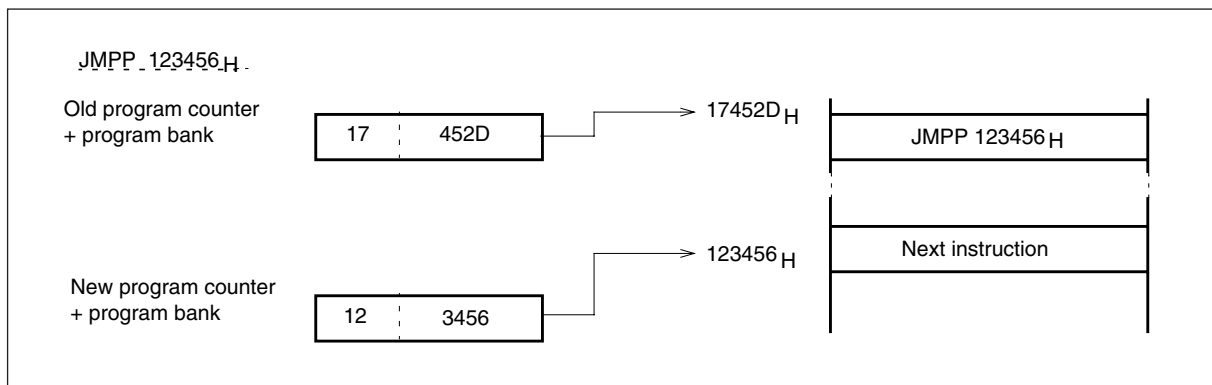
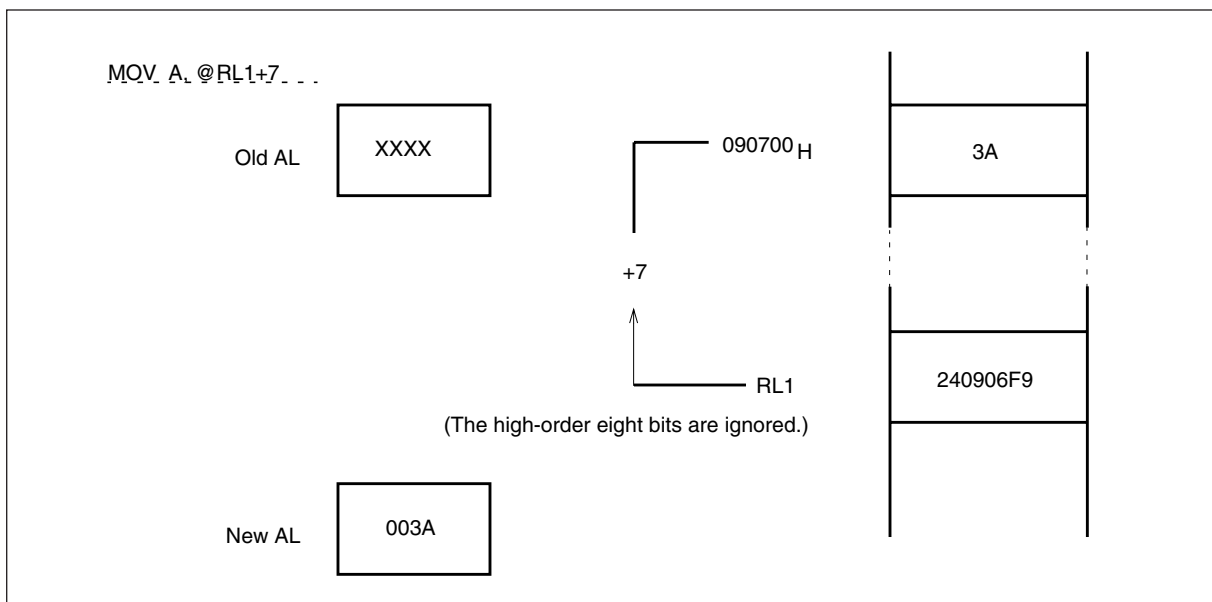


Figure 2.4-2 Example of linear method (32-bit register indirect specification)



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 USB 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.

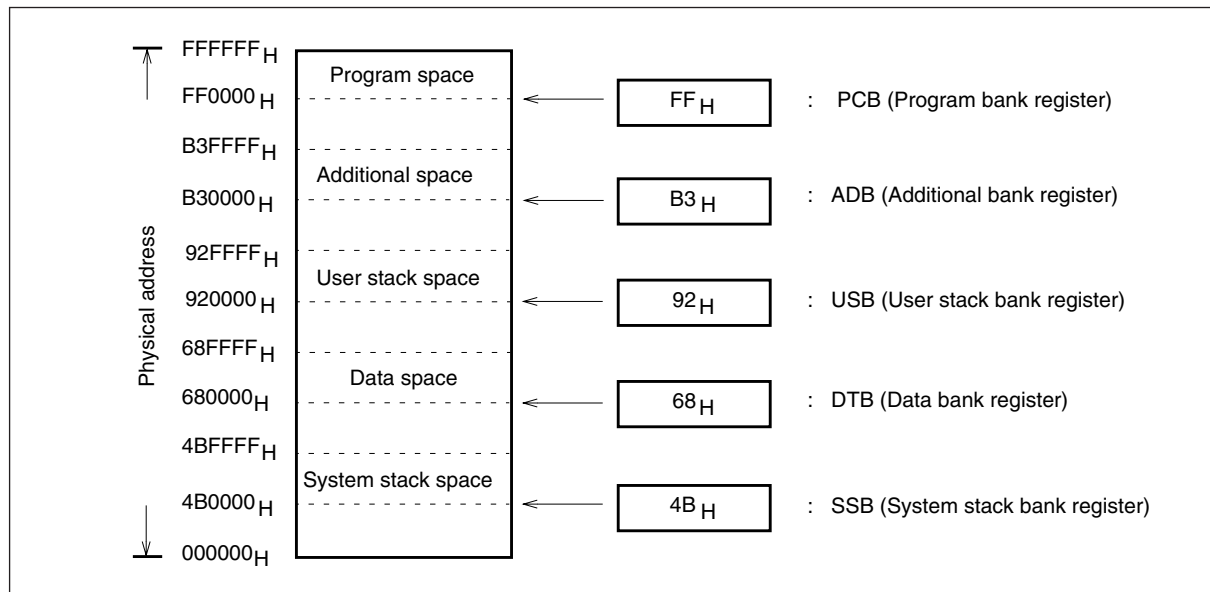
Table 2.5-1 “Default space” lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency. To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank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

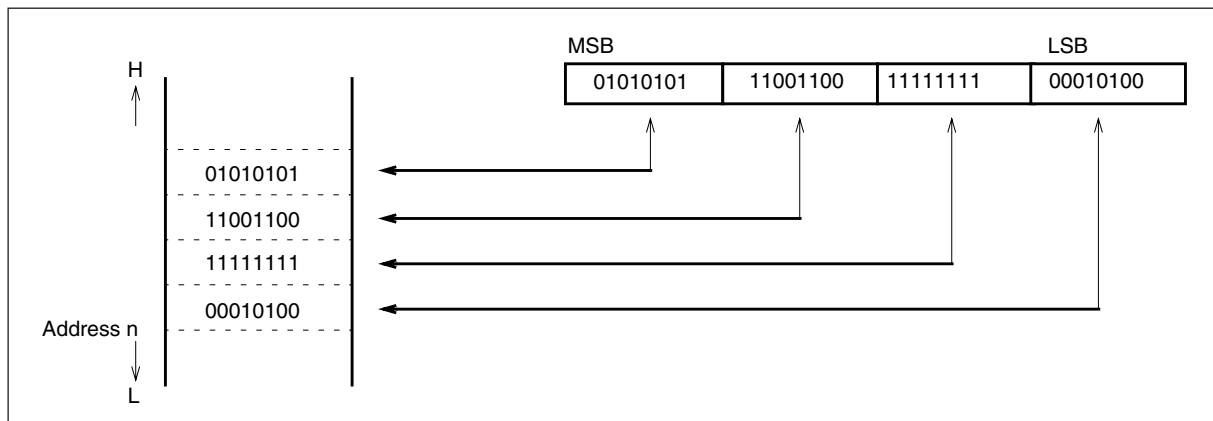
Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the low-order 16 bits are transferred before the high-order 16 bits.

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.

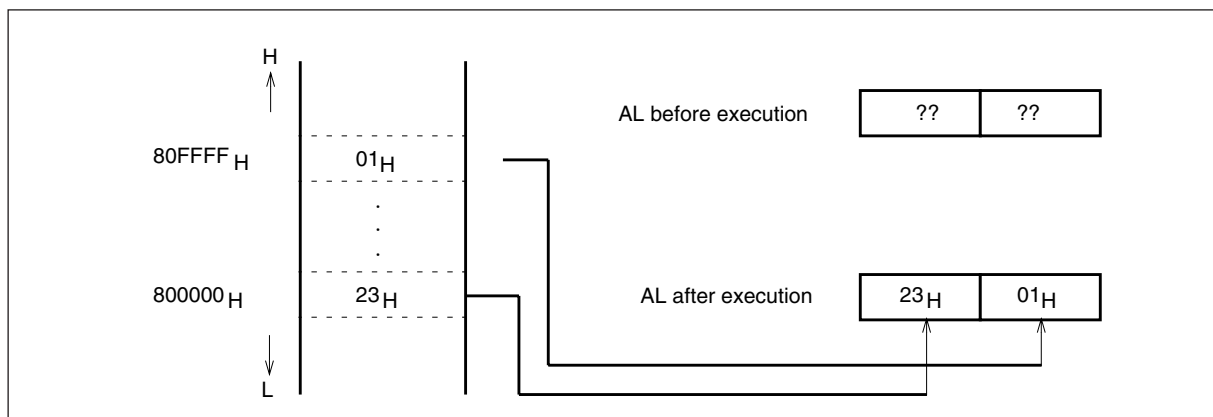
Figure 2.6-1 Sample allocation of multi-byte data in memory



■ 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, 080FFFFH ” is an example of an instruction accessing multi-byte data.

Figure 2.6-2 Execution of MOVW A, 080FFFFH



2.7 Registers

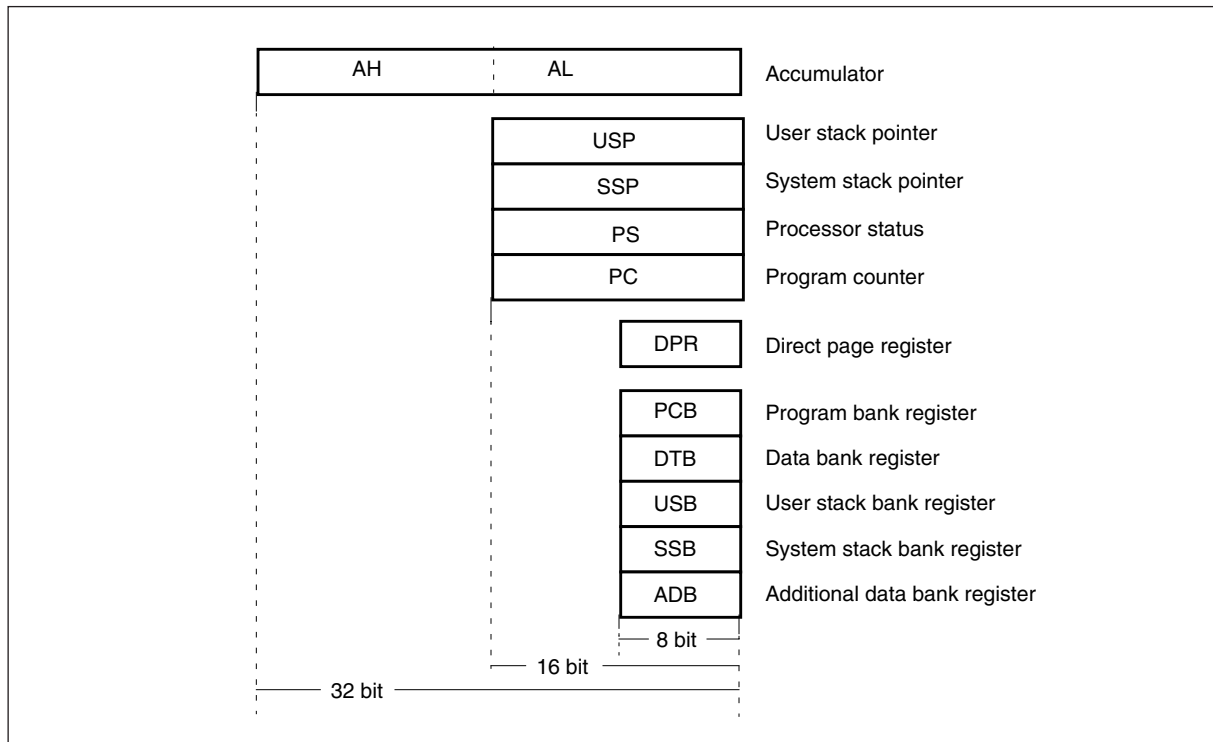
The F²MC-16LX registers are largely classified into two types: special registers in the CPU and general-purpose registers in memory. The special registers are dedicated internal hardware of the CPU, and they have specific use defined by the CPU architecture. The general-purpose registers share the CPU 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 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.

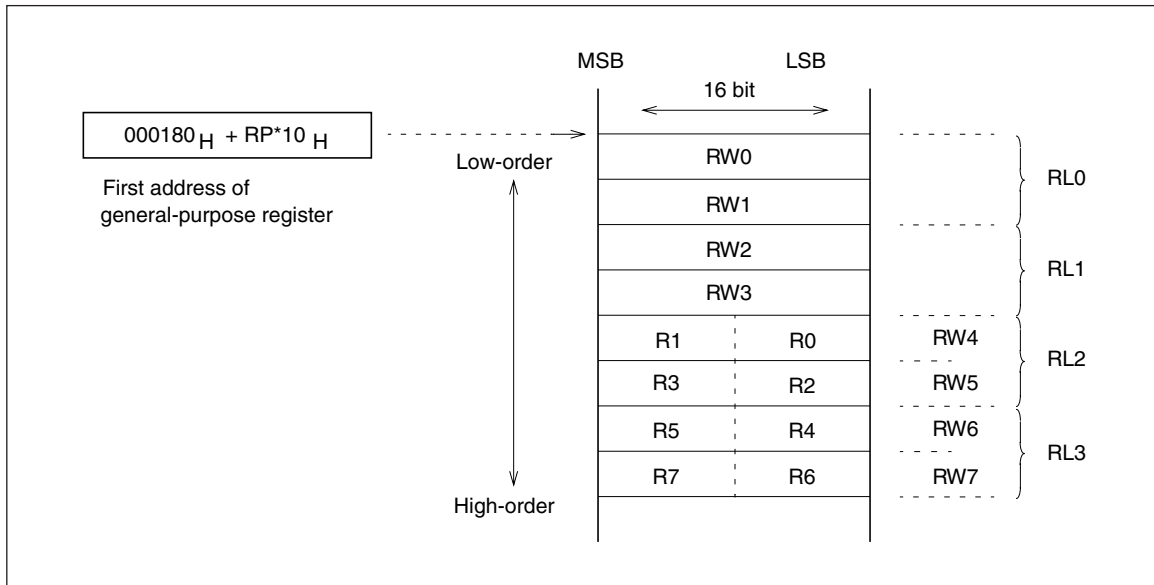
Figure 2.7-1 Special registers

■ 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

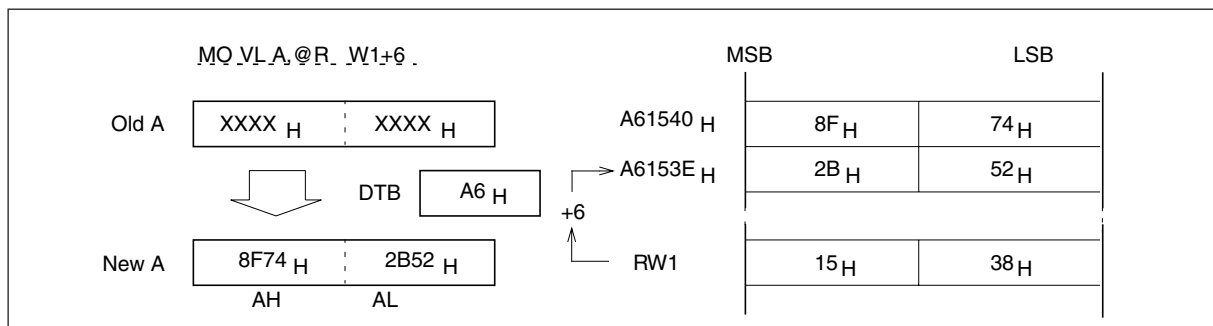
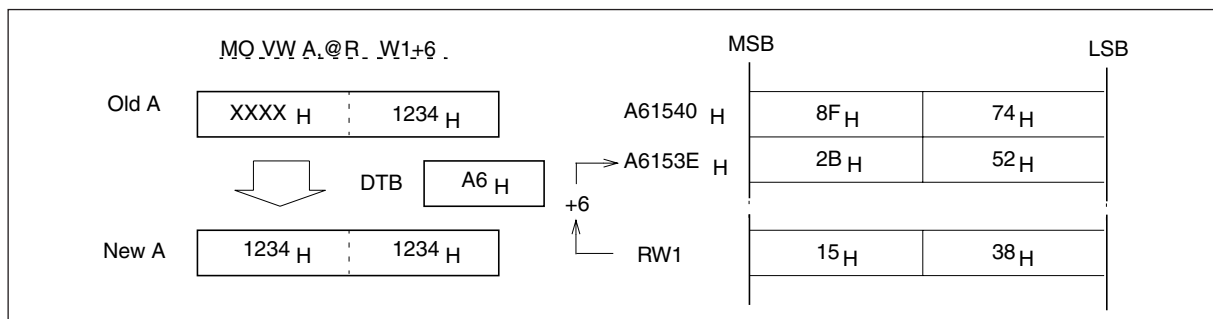


Figure 2.7-4 AL-AH transfer



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

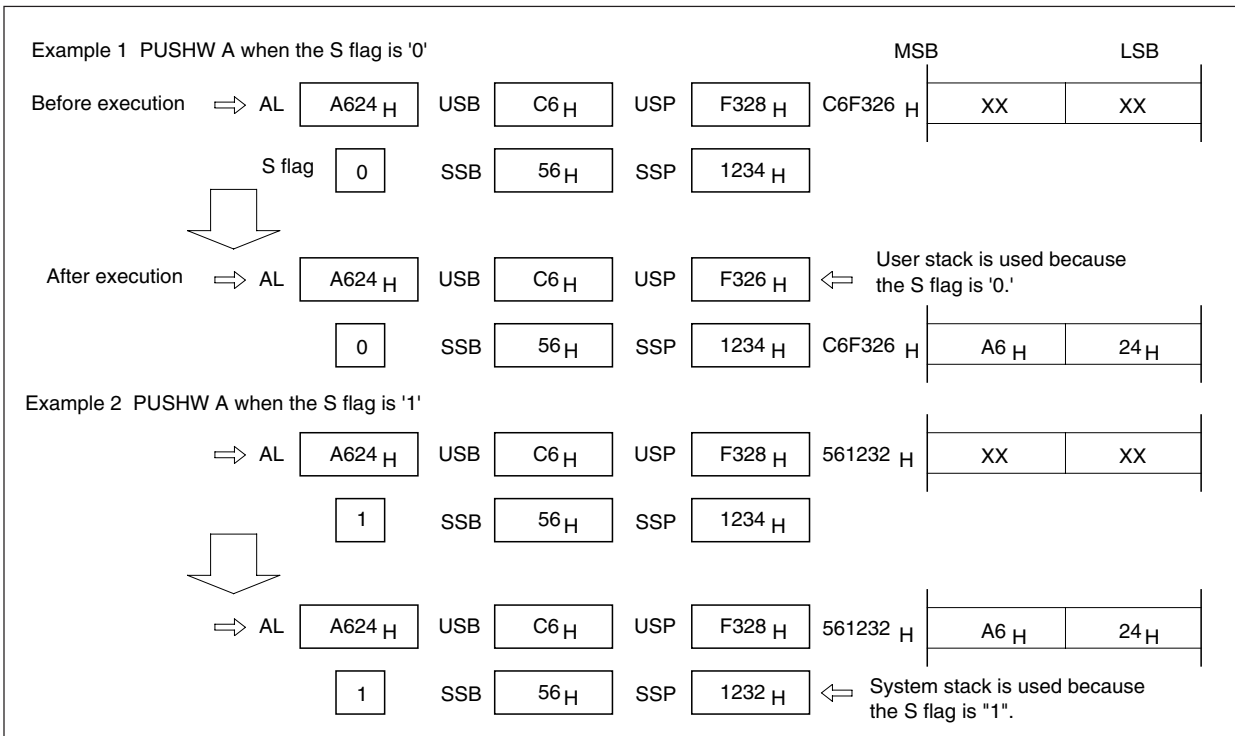
USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data when a push/pop instruction or subroutine is executed.

■ 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.

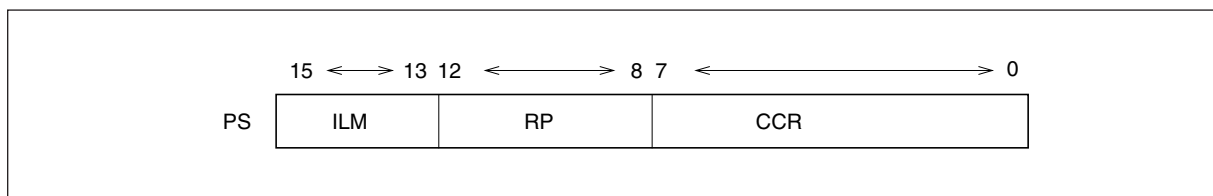
2.7.3 Processor Status (PS)

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.

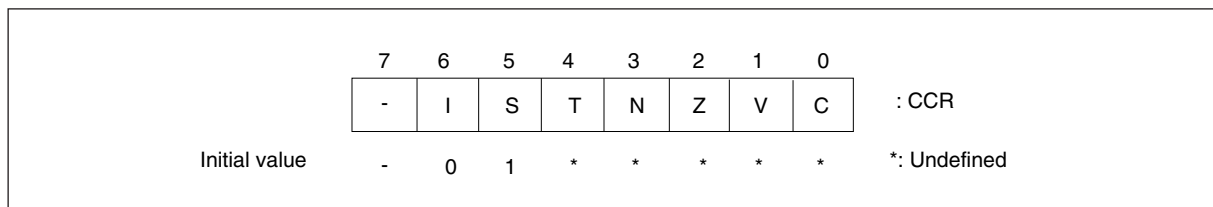
Figure 2.7-6 Processor status (PS) structure



■ 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:

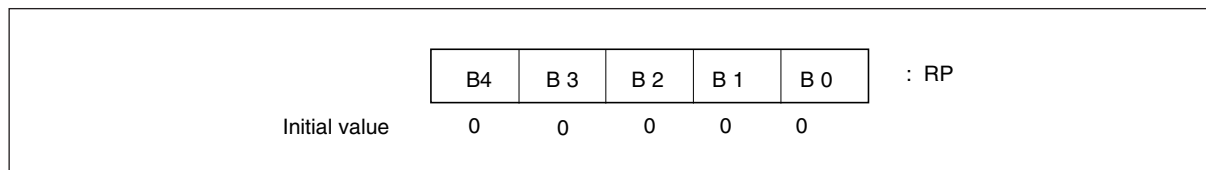
The C flag is set when a carry-up or carry-down from the MSB occurs as a result of operation execution, and is otherwise cleared.

■ 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: $[00180_H + (RP) \cdot 10_H]$ (see Figure 2.7-8 "Register bank pointer (RP)"). The RP register consists of five bits, and can take a value between 00_H and 1F_H. 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)

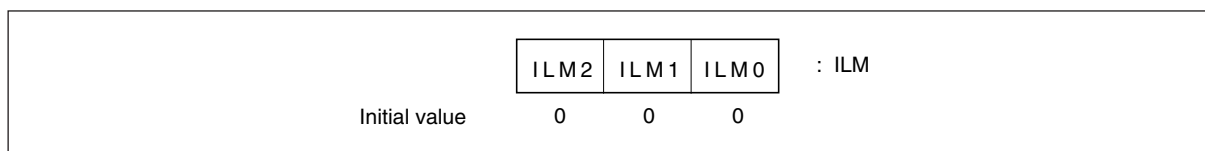


Table 2.7-1 Levels indicated by the interrupt level mask (ILM) register

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

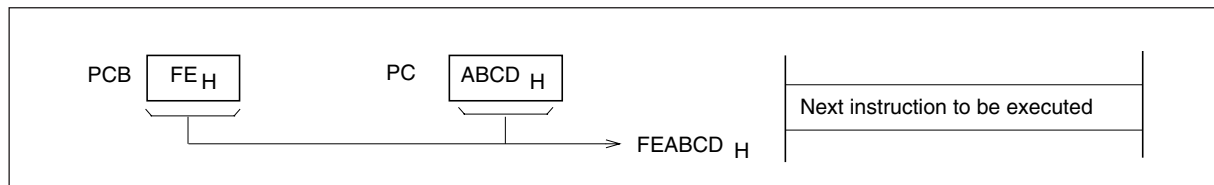
2.7.4 Program Counter (PC)

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.

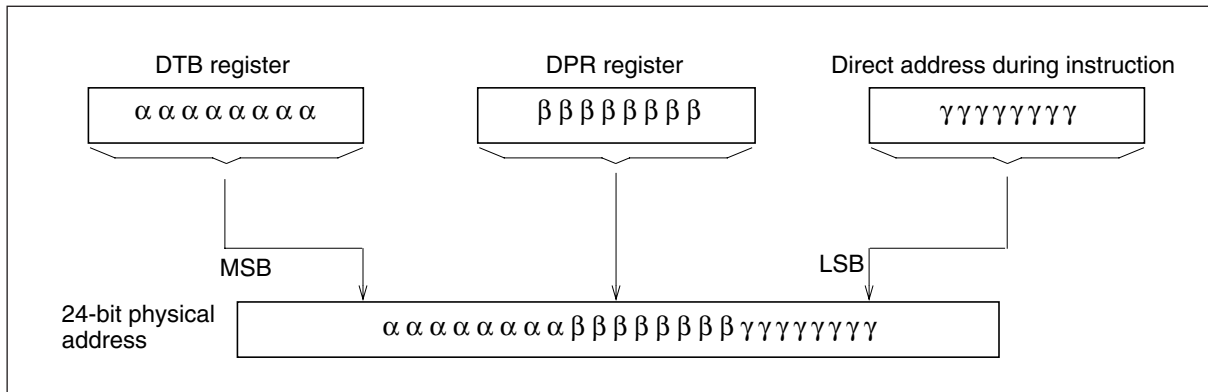
Table 2.8-2 Relationship between registers

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

- 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

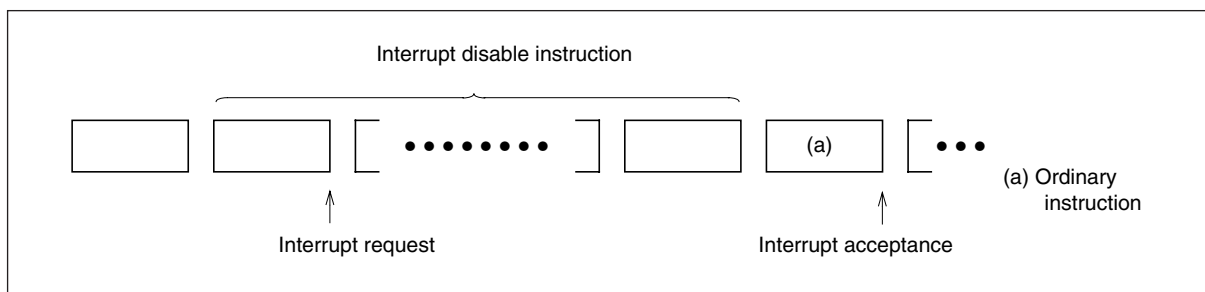
Interrupt requests are not sampled for the following ten instructions:

- 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".

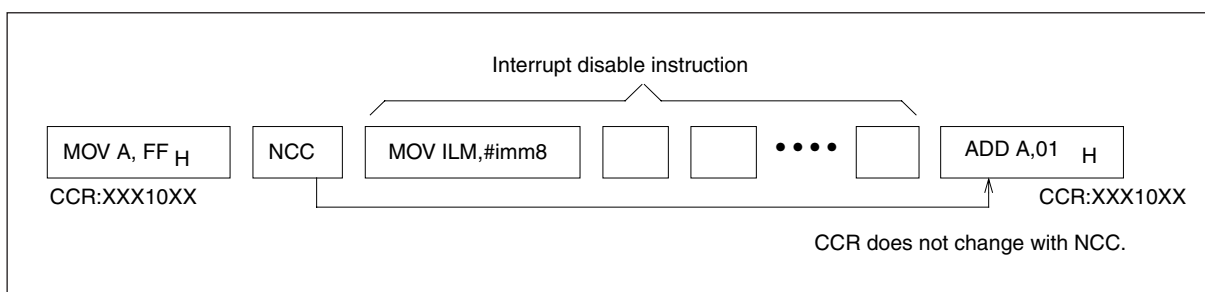
Figure 2.10-1 Interrupt disable instruction



■ 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

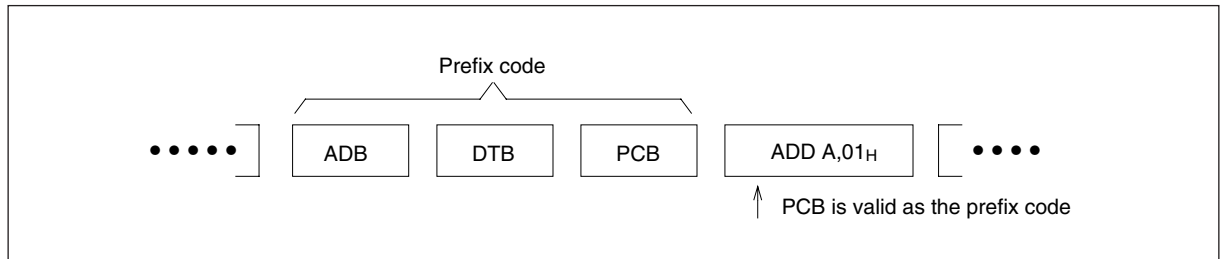


■ Consecutive prefix codes

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 MB90340 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 “μDMAC Interrupt”
- 3.10 “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 five 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
 - **DMAC:** 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.

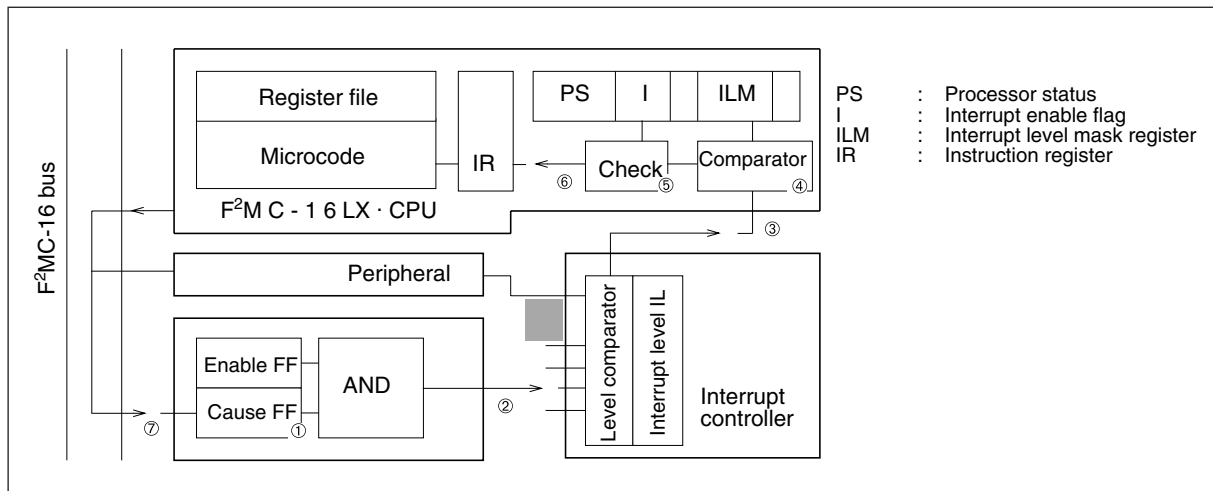
● Specifying an interrupt level

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

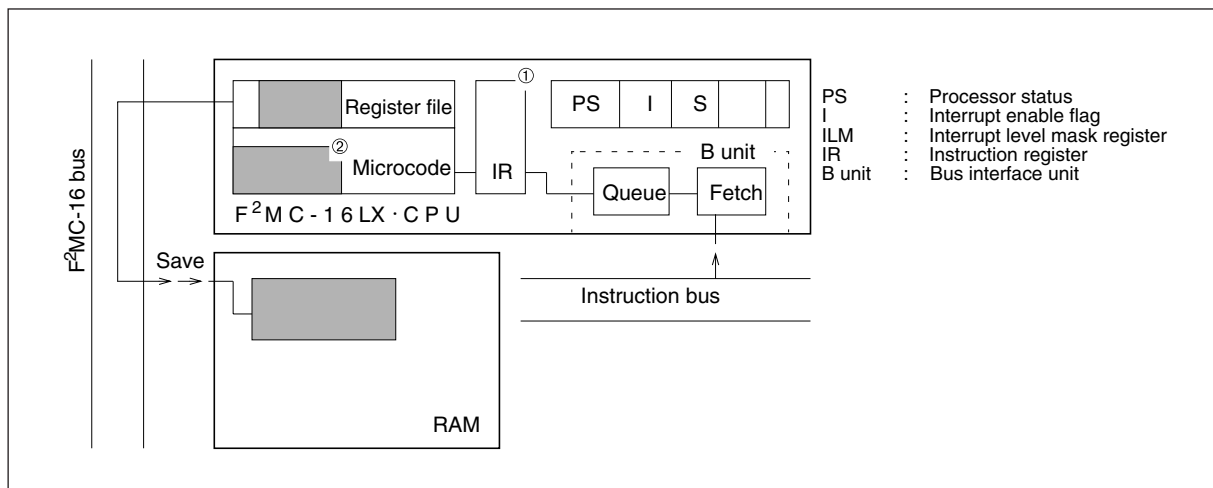


■ Software interrupts

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

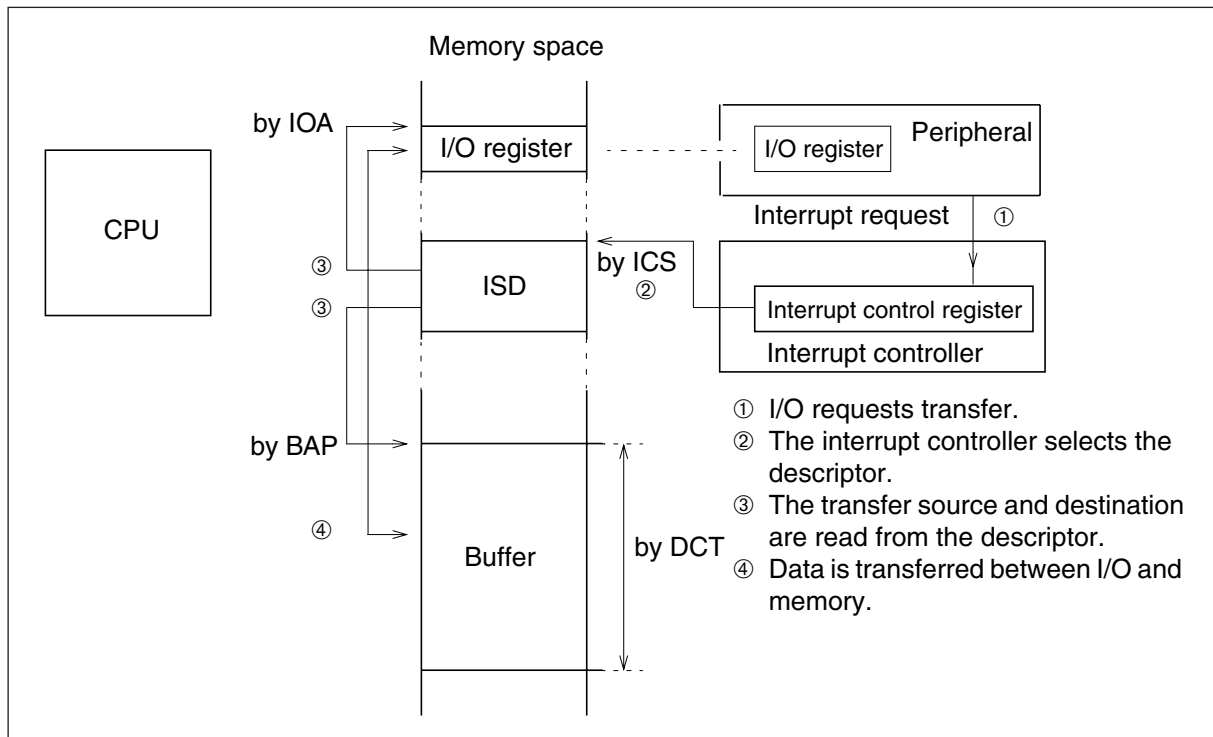


■ Extended intelligent I/O service (EI²OS)

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.

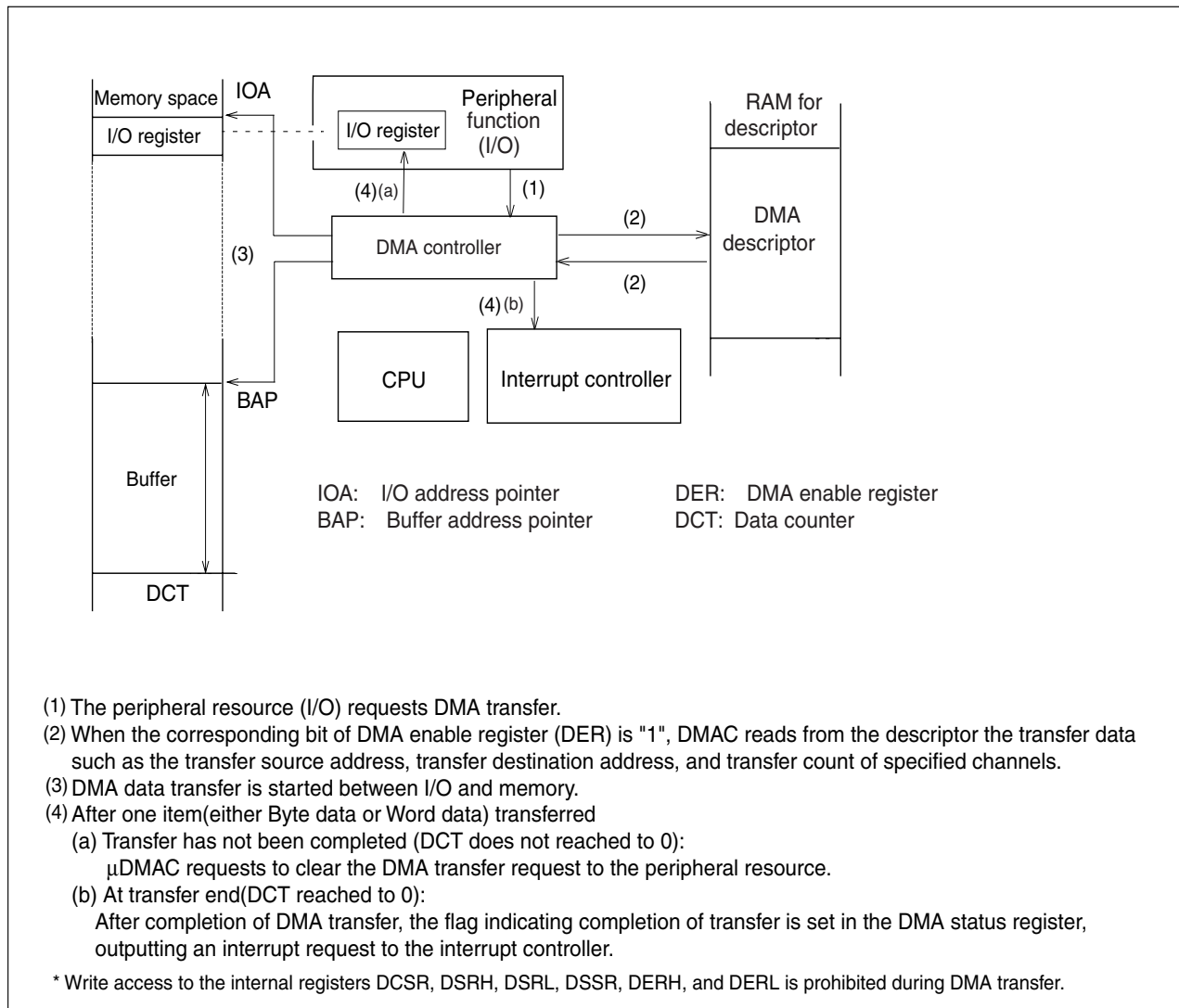
Figure 3.1-3 Overview of the extended intelligent I/O service (EI²OS)

■ Direct memory access (DMA)

DMA is a function used to automatically transfer data between peripheral functions and memory. Previous data transfers by an interrupt processing program is provided in the same way as the direct memory access (DMA). When a transfer of data of a specified count is completed, an interrupt processing program is automatically executed.

Interrupt by DMA is a type of hardware interrupts.

Figure 3.1-4 Overview of the Direct Memory access (DMA)



■ 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 (1/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address L	Vector address H	Vector address bank	Mode register
		Number	Address				
INT 0 *	--	--	--	FFFFFC _H	FFFFFD _H	FFFFFE _H	Unused
INT 1 *	--	--	--	FFFFF8 _H	FFFFF9 _H	FFFFFA _H	Unused
.	--	--	--	.	.	.	.
INT 7 *	--	--	--	FFFFE0 _H	FFFFE1 _H	FFFFE2 _H	Unused
INT 8	Reset	--	--	FFFFDC _H	FFFFDD _H	FFFFDE _H	FFFFDF _H
INT 9	INT9 instruction	--	--	FFFFD8 _H	FFFFD9 _H	FFFFDA _H	Unused
INT 10	Exception	--	--	FFFFD4 _H	FFFFD5 _H	FFFFD6 _H	Unused
INT 11	CAN0 RX	ICR00	0000B0 _H	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Unused
INT 12	CAN0 TX/NS			FFFFCC _H	FFFFCD _H	FFFFCE _H	Unused
INT 13	CAN1 RX / Input Capture6	ICR01	0000B1 _H	FFFFC8 _H	FFFFC9 _H	FFFFCA _H	Unused
INT 14	CAN1 TX/NS /Input Caputer7			FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Unused
INT 15	CAN2 RX / I ² C0	ICR02	0000B2 _H	FFFFC0 _H	FFFFC1 _H	FFFFC2 _H	Unused
INT 16	CAN2 TX/NS			FFFFBC _H	FFFFBD _H	FFFFBE _H	Unused
INT 17	16-bit ReloadTimer0	ICR03	0000B3 _H	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Unused
INT 18	16-bit ReloadTimer1			FFFFB4 _H	FFFFB5 _H	FFFFB6 _H	Unused
INT 19	16-bit ReloadTimer2	ICR04	0000B4 _H	FFFFB0 _H	FFFFB1 _H	FFFFB2 _H	Unused
INT 20	16-bit ReloadTimer3			FFFFAC _H	FFFFAD _H	FFFFAE _H	Unused
INT 21	PPG 0/1/4/5	ICR05	0000B5 _H	FFFFA8 _H	FFFFA9 _H	FFFFAA _H	Unused
INT 22	PPG 2/3/6/7			FFFFA4 _H	FFFFA5 _H	FFFFA6 _H	Unused

Table 3.2-1 Interrupt vectors (2/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address L	Vector address H	Vector address bank	Mode register
		Number	Address				
INT 23	PPG 8/9/C/D	ICR06	0000B6 _H	FFFFA0 _H	FFFFA1 _H	FFFFA2 _H	Unused
INT 24	PPG A/B/E/F			FFFF9C _H	FFFF9D _H	FFFF9E _H	Unused
INT 25	Time Base Timer	ICR07	0000B7 _H	FFFF98 _H	FFFF99 _H	FFFF9A _H	Unused
INT 26	External Interrupt 0-3/8-11			FFFF94 _H	FFFF95 _H	FFFF96 _H	Unused
INT 27	Watch Timer	ICR08	0000B8 _H	FFFF90 _H	FFFF91 _H	FFFF92 _H	Unused
INT 28	External Interrupt 4-7/12-15			FFFF8C _H	FFFF8D _H	FFFF8E _H	Unused
INT 29	A/D Converter	ICR09	0000B9 _H	FFFF88 _H	FFFF89 _H	FFFF8A _H	Unused
INT 30	I/O Timer 0/1			FFFF84 _H	FFFF85 _H	FFFF86 _H	Unused
INT 31	Input Capture 4/5 / I ² C1	ICR10	0000BA _H	FFFF80 _H	FFFF81 _H	FFFF82 _H	Unused
INT 32	Output Compare 0/1/4/5			FFFF7C _H	FFFF7D _H	FFFF7E _H	Unused
INT 33	Input Capture 0-3	ICR11	0000BB _H	FFFF78 _H	FFFF79 _H	FFFF7A _H	Unused
INT 34	Output Compare 2/3/6/7			FFFF74 _H	FFFF75 _H	FFFF76 _H	Unused
INT 35	UART 0 RX	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	Unused
INT 36	UART 0 TX			FFFF6C _H	FFFF6D _H	FFFF6E _H	Unused
INT 37	UART 1 RX / UART3 RX	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	Unused
INT 38	UART 1 TX / UART 3 TX			FFFF64 _H	FFFF65 _H	FFFF66 _H	Unused
INT 39	UART 2 RX / UART 4 RX	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	Unused
INT 40	UART 2 TX / UART 4 TX			FFFF5C _H	FFFF5D _H	FFFF5E _H	Unused
INT 41	Flash Memory	ICR15	0000BF _H	FFFF58 _H	FFFF59 _H	FFFF5A _H	Unused
INT 42	Delayed Interrupt			FFFF54 _H	FFFF55 _H	FFFF56 _H	Unused
INT 43	--	--	--	FFFF50 _H	FFFF51 _H	FFFF52 _H	Unused
.	--	--	--	.	.	.	.
.	--	--	--	.	.	.	.
.	--	--	--	.	.	.	.
INT 254	--	--	--	FFFC04 _H	FFFC05 _H	FFFC06 _H	Unused
INT 255	--	--	--	FFFC00 _H	FFFC01 _H	FFFC02 _H	Unused

*: 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.

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:

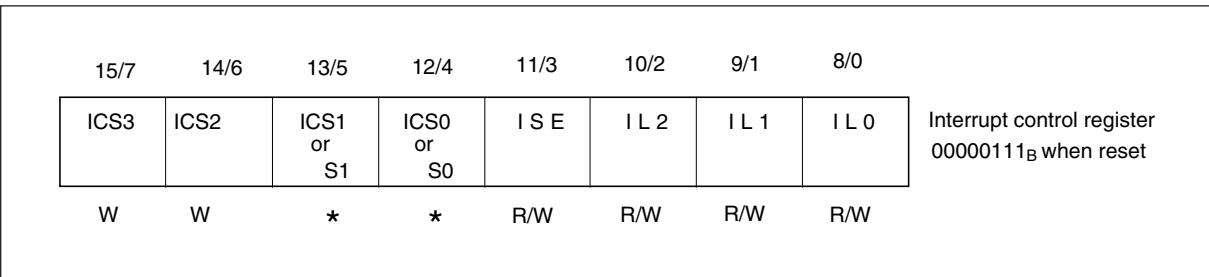
- Setting an interrupt level for corresponding peripherals
- Selecting whether to use an ordinary interrupt or extended intelligent I/O service for the corresponding peripherals
- Selecting the extended intelligent I/O service channel

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

■ 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

IL2	IL1	ILO	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

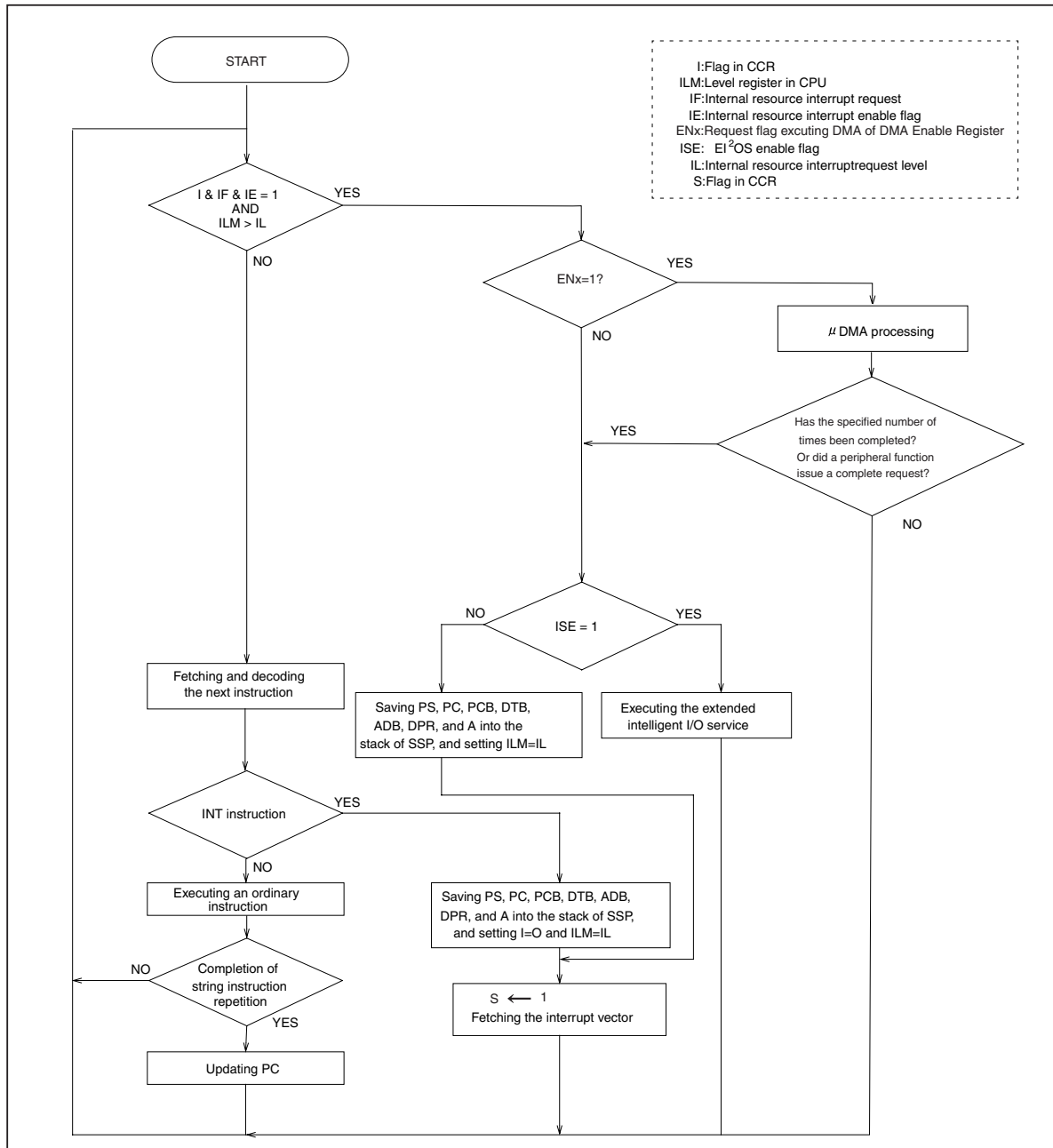
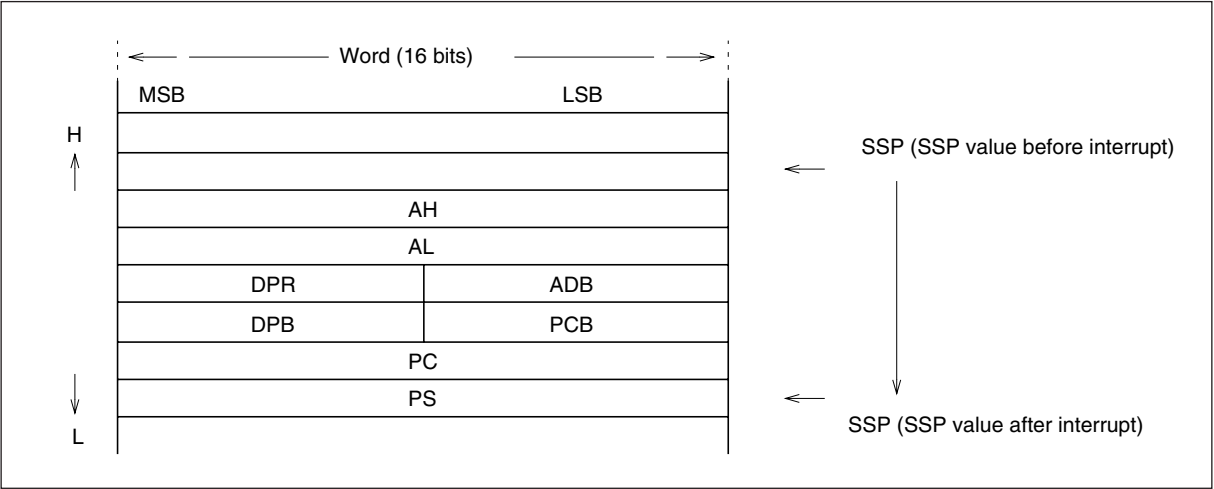


Figure 3.4-2 Register saving during interrupt processing



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.

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.

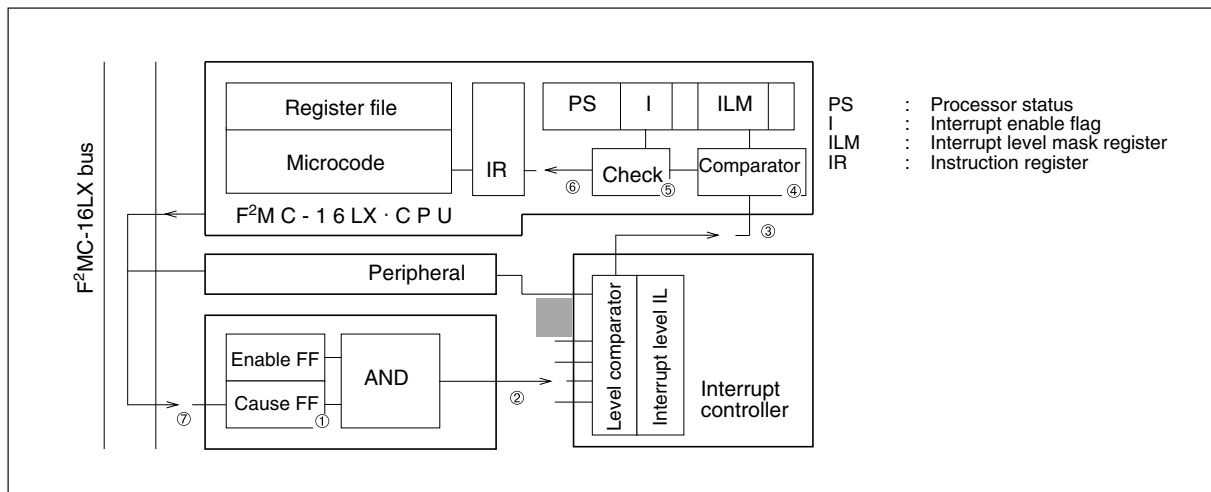
Figure 3.5-1 “Occurrence and release of hardware interrupt” illustrates the flow from the occurrence of a hardware interrupt until there is no interrupt request in the interrupt processing program.

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.

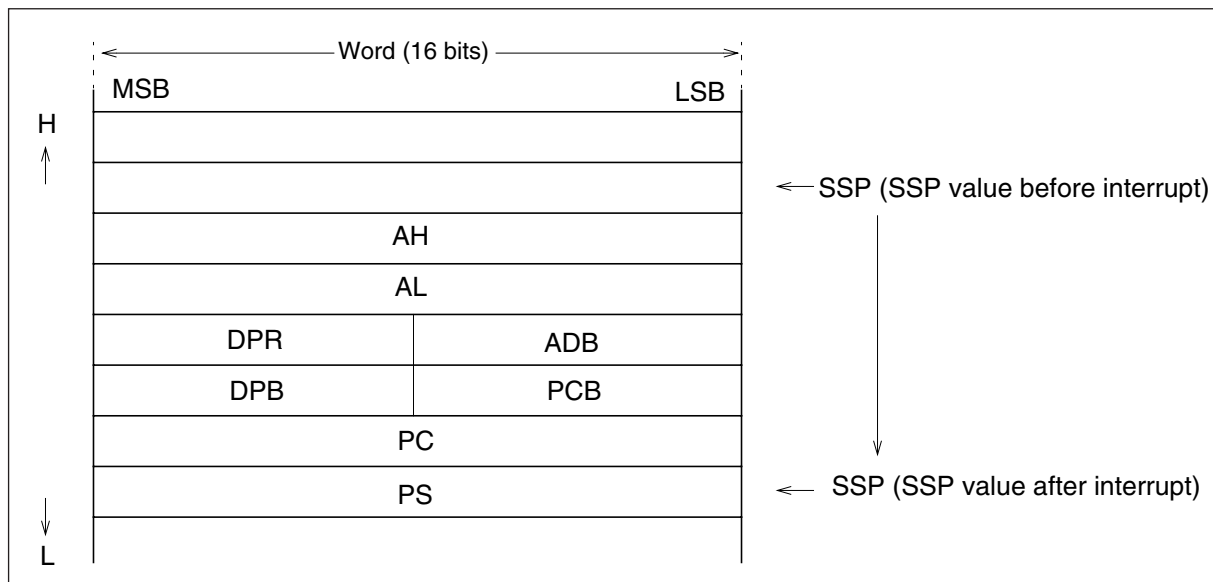
If an interrupt occurs during interrupt processing, a higher-level interrupt is processed first.

■ 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.

Figure 3.5-2 Registers saved in 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 interrupt vectors

Table D-1 “Interrupt vectors” lists the interrupt vectors of the MB90340 series.

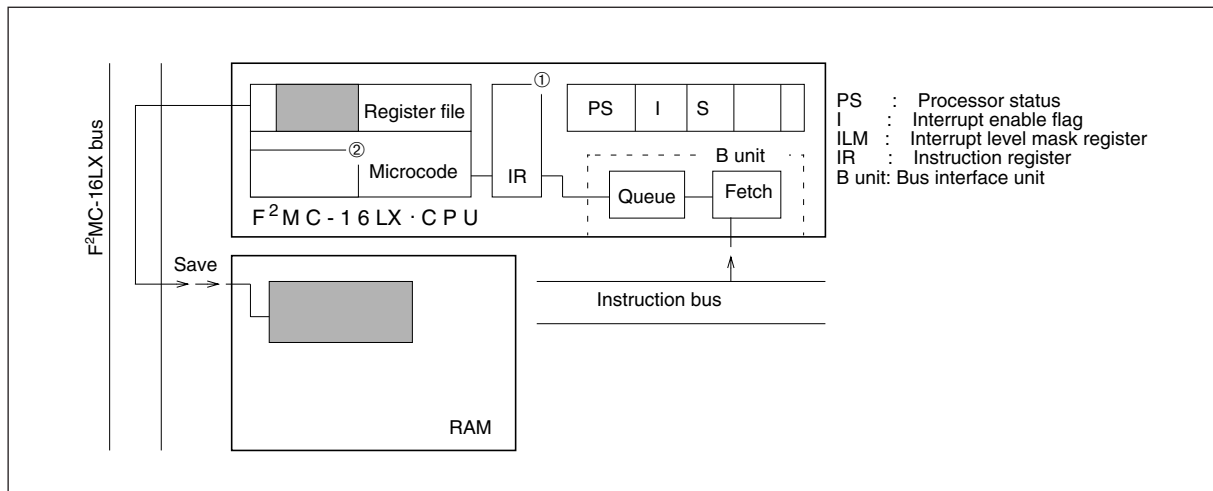
As shown in Table D-1 “Interrupt vectors”, software interrupts share the same interrupt vector area with hardware interrupts.

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 MB90340 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:

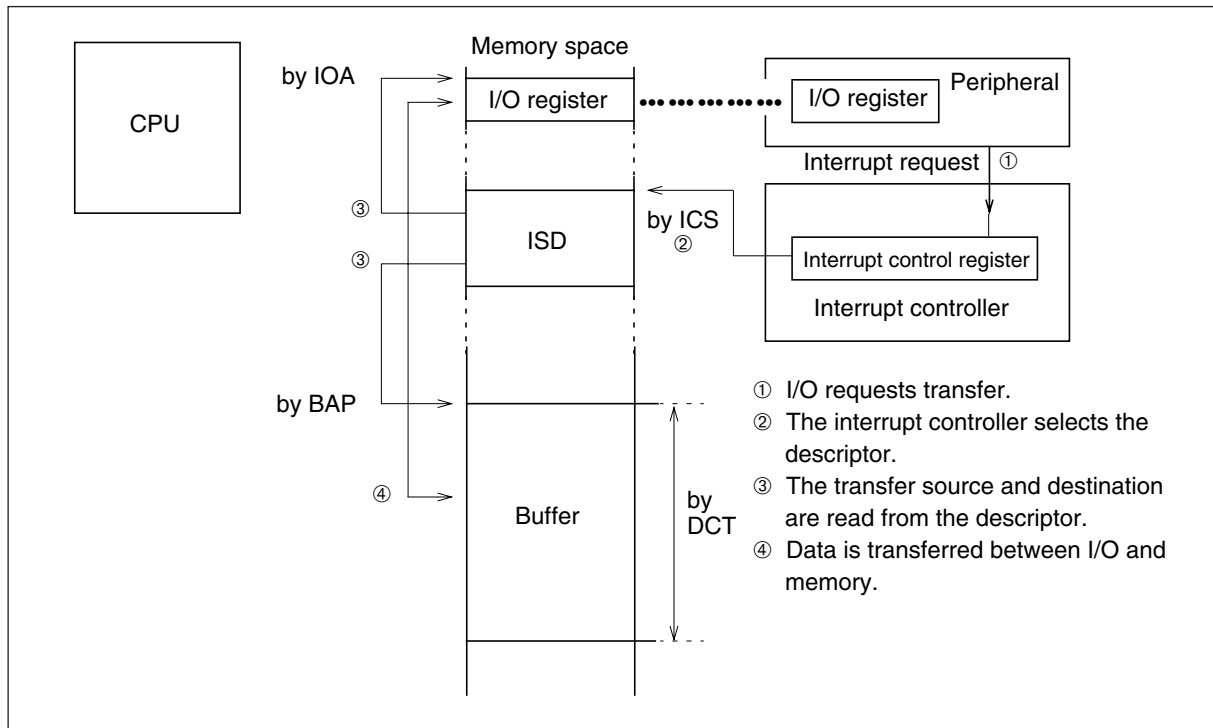
- The program size can be small because it is not necessary to write a transfer program.
- No internal register is used for transfer, eliminating the need for register saving and increasing the transfer speed.
- Transfer can be terminated from I/O, preventing unnecessary data from being transferred.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

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)

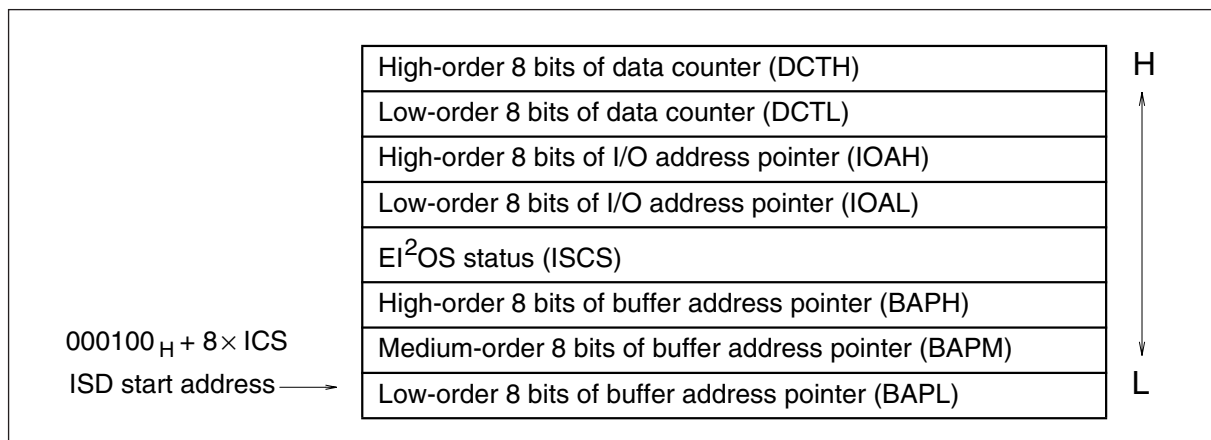
The extended intelligent I/O service descriptor exists between 000100_H and 00017F_H in internal RAM, and consists of the following items:

- 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.

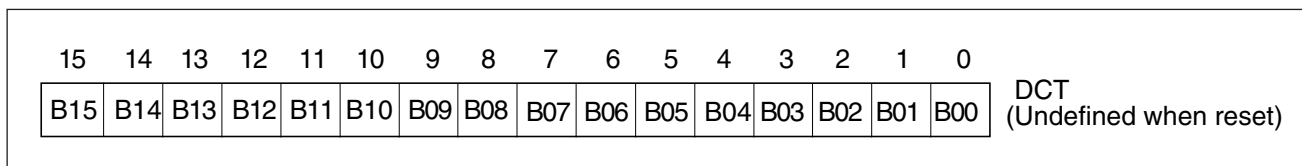
Figure 3.7-2 Extended intelligent I/O service descriptor configuration



■ 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.

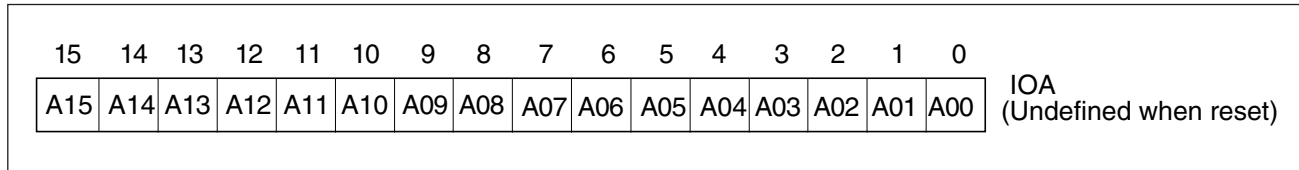
Figure 3.7-3 Data counter configuration



■ 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.

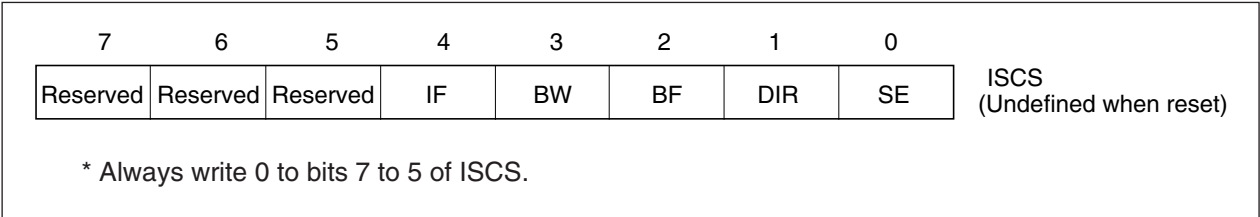
3.7.2 EI²OS Status Register (ISCS)

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



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

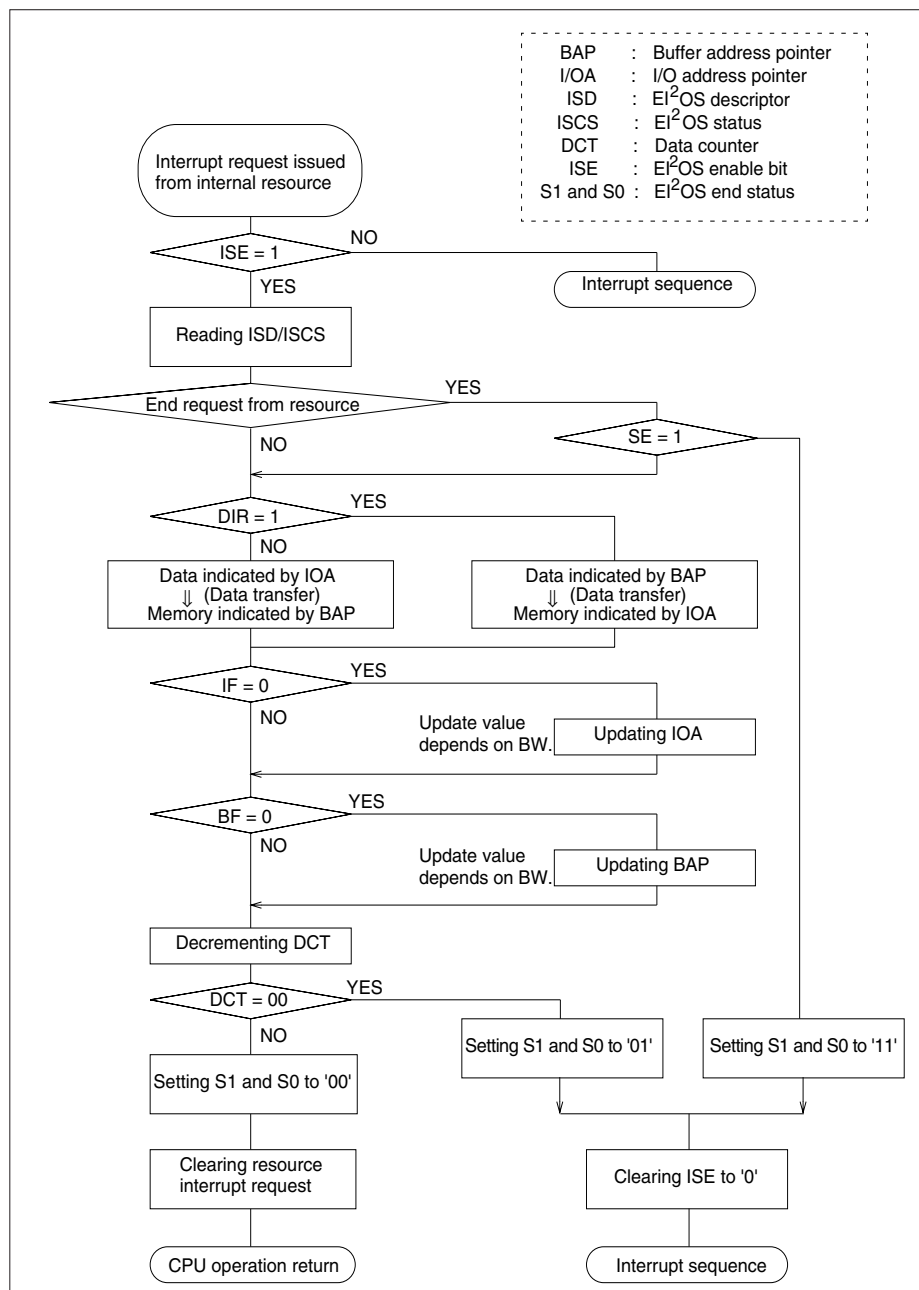
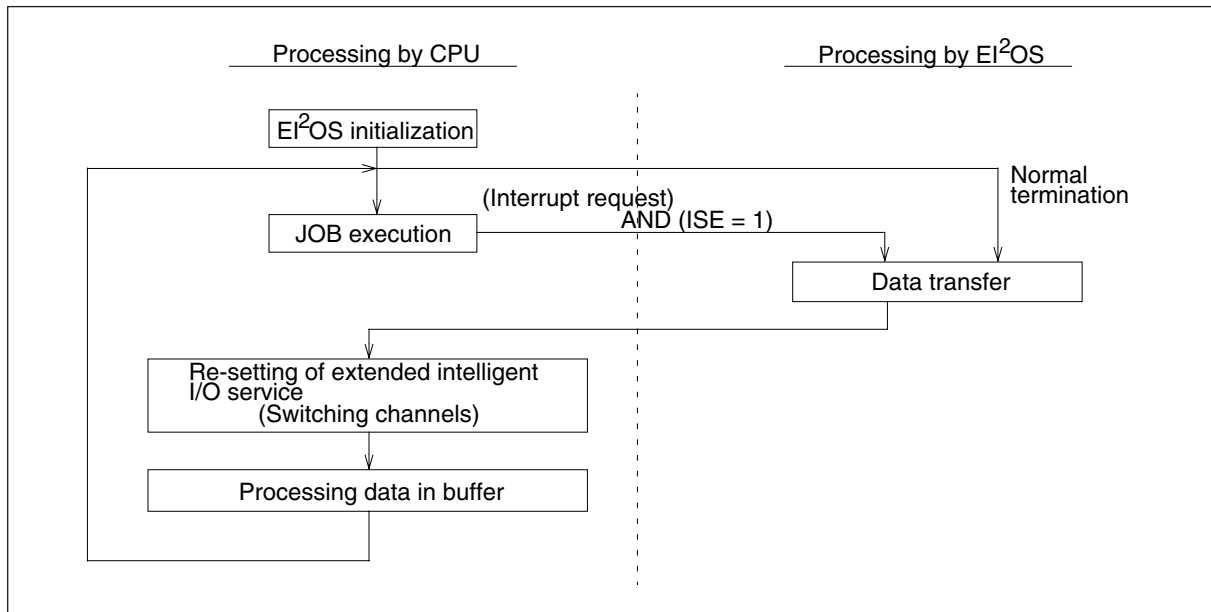


Figure 3.8-2 EI²OS use flow

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 3.8-3 "Interrupt handling times") 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 3.8-3 "Interrupt handling times") 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

Table 3.8-3 Interrupt handling times

Address pointed to by the stack pointer	Interpolation value [cycles]
External 8-bit	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

3.9 μ DMAC Interrupt

μ DMAC is a simplified DMA with the same function as EI²OS

3.9.1 μ DMAC Functions

3.9.2 μ DMAC Registers

3.9.3 DMA Descriptor window register (DDWR)

3.9.4 DMAC Operation

3.9.1 μ DMAC Functions

This section describes the μ DMAC functions.

■ μ DMAC functions

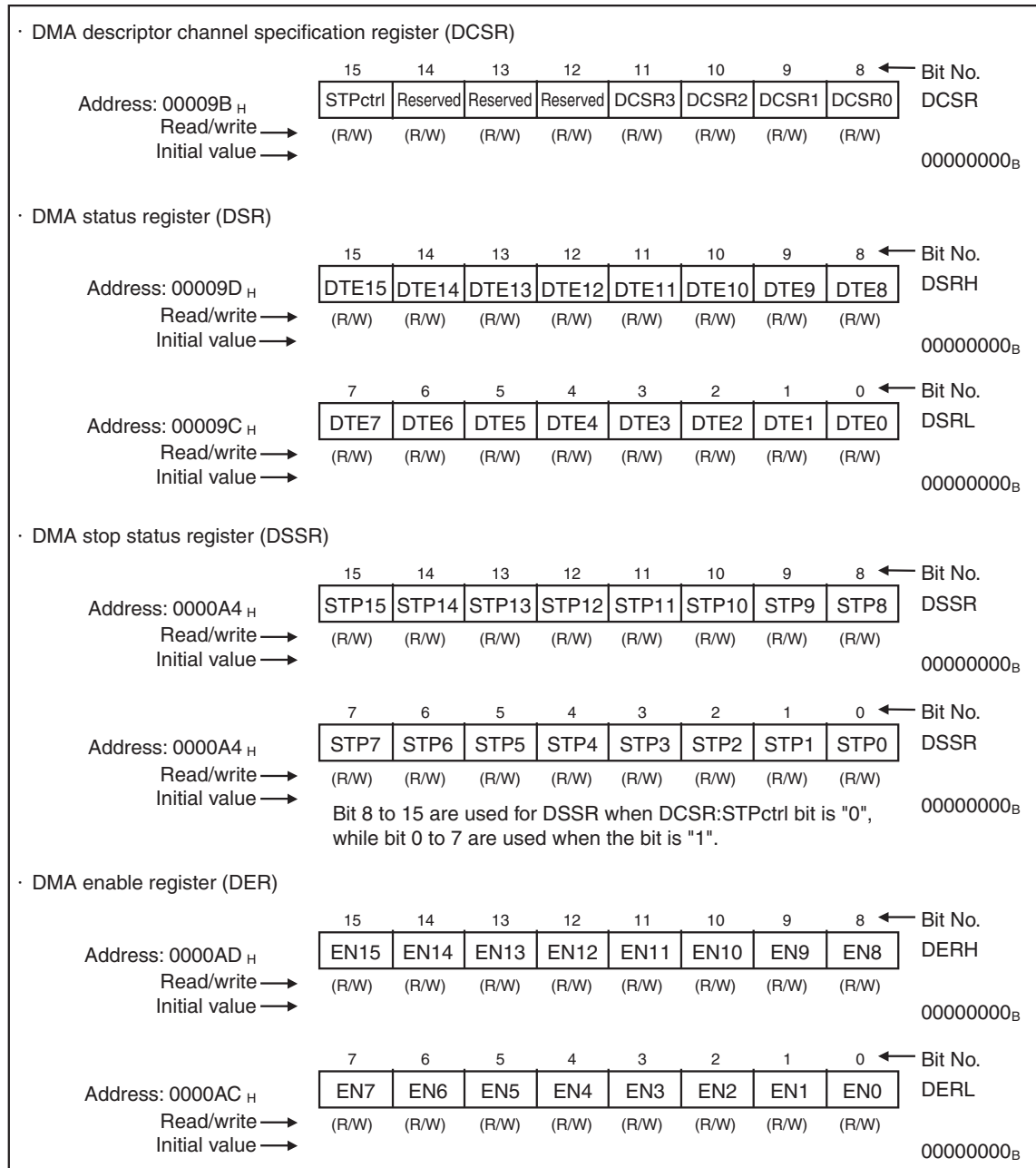
μ DMAC function has the following functions:

- Automatic data transfer between peripheral resources (I/O) and memory
- CPU program execution stopped during starting of DMA
- 16 DMA transfer channels (The smaller the channel number, the higher the DMA transfer priority.)
- Allow selection of whether or not to increment the transfer source, transfer destination addresses.
- DMA transfer started by peripheral resource (I/O) interrupts.
- DMA transfer controlled by: (a) DMA enable register (DER), (b) DMA stop status register (DSSR), (c) DMA status register (DSR), (d) DMA descriptor channel specification register (DCSR), and (e) Descriptor (DMACS)
- STOP requests are issued as a means to stop DMA transfers from a peripheral resource.
- After the end of a DMA transfer, a flag is set to the bit corresponding to the transfer end channel of the DMA status register, and an end interrupt is then output to the interrupt controller.

3.9.2 μ DMAC Registers

μ DMAC has four registers: DCSR, DSR, DSSR, and DER. The DMA descriptor used to set DMA transfer is explained in Section 3.9.3 DMA Descriptor window register (DDWR).

■ Registers List

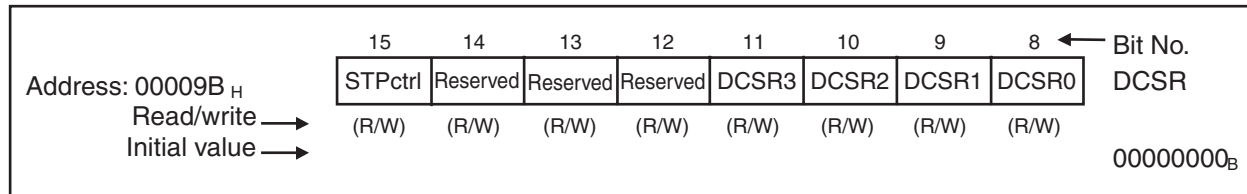


3.9.2.1 DMA descriptor channel specification register (DCSR)

This section describes the DMA descriptor channel specification register (DCSR).

■ DMA descriptor channel specification register (DCSR)

Figure 3.9-1 DMA descriptor channel specification register (DCSR) configuration



[bit 11 to 8] DCSR_x: DMA Descriptor Channel Specification

Table 3.9-1 Relationship between DCSR and Selected Channels

DCSR3 to 0	Selected Channels	Resource Interrupt Request
0000	0	16-bit Reload Timer 0
0001	1	16-bit Reload Timer 1
0010	2	16-bit Reload Timer 2
0011	3	External Interrupt 0-3, 8-11
0100	4	External Interrupt 4-7, 12-15
0101	5	A/D Converter
0110	6	Input Capture 4/5, I ² C1
0111	7	Output Compare 0/1/4/5
1000	8	Input Capture 0-3
1001	9	Output Compare 2/3/6/7
1010	10	UART0 RX
1011	11	UART0 TX
1100	12	UART1 RX, UART3 RX
1101	13	UART1 TX, UART3 TX
1110	14	UART2 RX, UART4 RX
1111	15	UART2 TX, UART4 TX

One (descriptor channel) of 16 channels is selected by the DCSR setting. For details, see Section 3.9.3 “DMA descriptor window register (DDWR)”.

[bit 15] STPctrl: STP Control Bit

STPctrl bit	Function
0 [Initial value]	STP8 to 15 are selected as DSSR.
1	STP0 to 7 are selected as DSSR.

[bit 14 to 12] Reserved: Reserved Bits

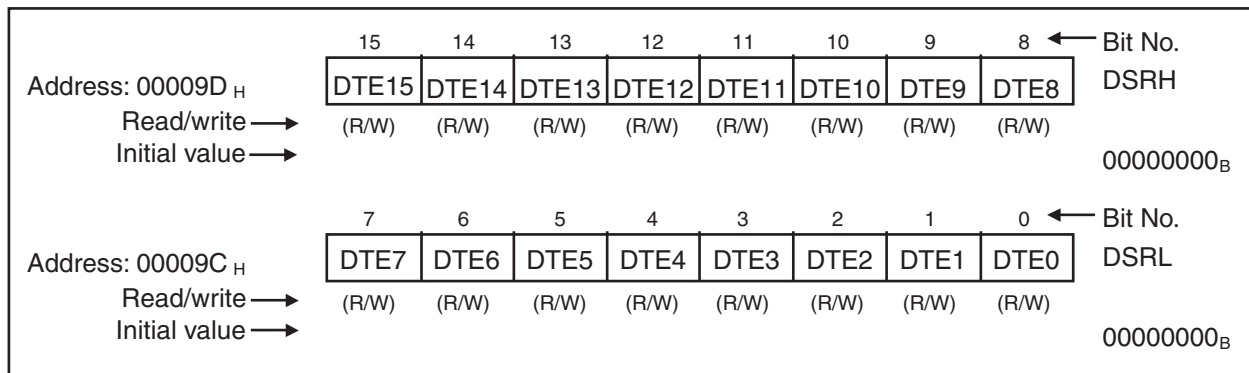
Bit 12 to 14 (three bits) are reserved bits, so bit manipulation instructions cannot be executed for them. When setting DSCR, always write “0” to these bits. Reading these bits always returns “0”.

3.9.2.2 DMA status register (DSR)

This section describes the DMA status register (DSR).

■ DMA status register (DSR)

Figure 3.9-2 DMA status register (DSR) configuration



[bit 15 to 0] DTE_x: DMA Status

DTE _x bit	Function
0 [Initial value]	Indicates no interrupt request present. When DTE _x is “0”, write “0” to this bit. Writing “1” to it is prohibited.
1	Indicates that DMA transfer completed and interrupt requested. When DTE _x is “1”, writing “0” to this bit clears it to “0”; writing “1” to it retains the previous data.

Note:

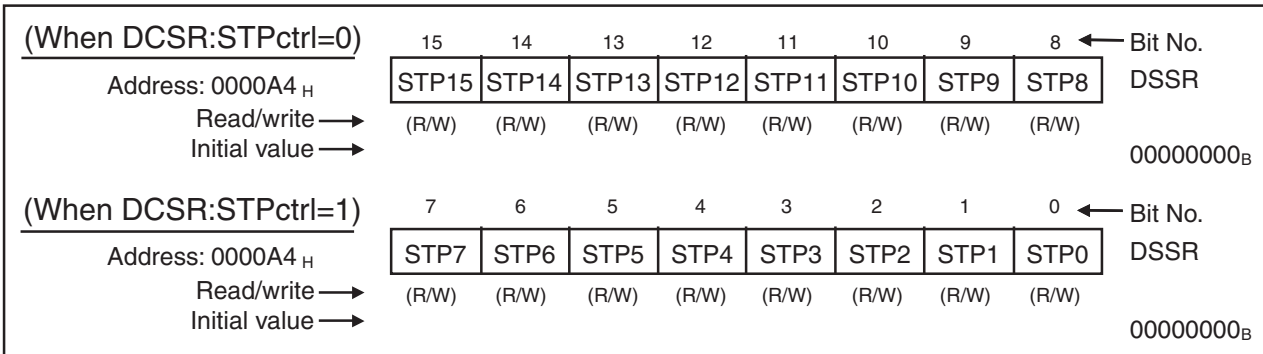
For write access to this register, always use a read - modify - write instruction.

3.9.2.3 DMA stop status register (DSSR)

This section describes the DMA stop status register (DSSR)

■ DMA stop status register (DSSR)

Figure 3.9-3 DMA stop status register (DSSR) configuration



[bit 15 to 0] STPx: DMA Stop Status

STPx bit	Function
0 [Initial value]	No STOP request is accepted in a DMA transfer. When STPx is “0”, write “0” to this bit. Writing “1” to it is prohibited.
1	Indicates that DMA transfer stopped due to STOP request issued by resource during DMA transfer. When the DMA control register SE bit is “1” and the corresponding channel receives the STOP request, the corresponding bit of the DMA enable register is cleared to “0”. When STPx is “1”, writing “0” to this bit clears it to “0”; writing “1” retains the previous data.

Note:

DSSR is controlled by the MSB of DCSR (STPctrl). When STPctrl is “0”, bit 8 to 15 are selected as DSSR; when STPctrl is “1”, bit 0 to 7 are. Since the initial value of STPctrl is “0”, bit 8 to 15 are selected as DSSR at first.

When writing to DSSR, use an RMW (read-modify-write) instruction.

The following three channels supports the STOP request:

Channels	Corresponding STPx bits	Resource
ch10	STP10	UART0RX
ch12	STP12	UART1RX, UART3RX
ch14	STP14	UART2RX, UART4RX

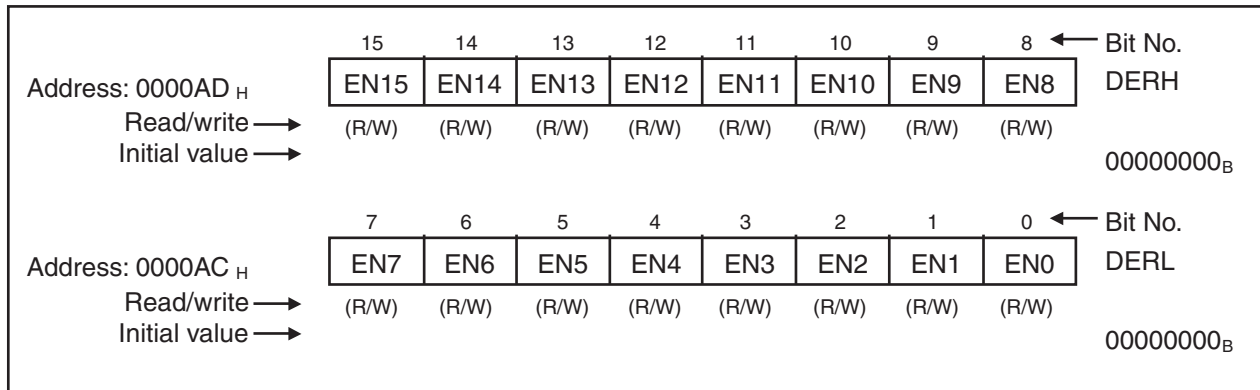
* Bits other than STP10, STP12, and STP14 have no meaning.

3.9.2.4 DMA enable register (DER)

This section describes DMA enable register (DER).

■ DMA enable register (DER)

Figure 3.9-4 DMA enable register (DER) configuration



[bit 15 to 0] ENx: DMA Enable

ENx bit	Function
0 [Initial value]	The interrupt request from the resource is output to the interrupt controller. (The interrupt request from the resource is not processed as a DMA start request.)
1	The interrupt request from the resource is processed as a DMA start request. At completion of DMA transfer, the interrupt request is output to the interrupt controller. This bit is cleared to "0" when the DMA transfer byte count becomes "0" or when DMA transfer is stopped by the STOP request from the resource.

When writing to DER, use an RMW (read-modify-write) instruction.

Note:

Before entering the sleep mode, these bits must be cleared to "0".

3.9.3 DMA Descriptor window register (DDWR)

The DMA descriptor consists of 8 bytes x 16 channels and is used to set DMA transfer. One channel selected from the 16 channels is mapped to the DMA descriptor window register (DDWR), making it possible to access this channel. DDWR addresses are “0000D0_H” to “0000D7_H”.

■ Configuration of DMA Descriptor window register (DDWR)

The DMA descriptor consists of 8 bytes x16 channels and the structure of each channel is as shown in Figure 3.9-5 “Configuration of DMA Descriptor Window Register (DDWR)” The descriptor for the channel selected by the DMA descriptor channel specification register (DCSR) or by the interrupt request channel number is mapped onto the DMA descriptor window register (DDWR). For the relationship between the DMA descriptor channel specification register (DCSR) and the channels to be selected, see Table 3.9-1 “Relationship between DCSR and Selected Channels”.

Figure 3.9-5 Configuration of DMA Descriptor Window Register (DDWR)

Address	
0000D7 _H	Upper 8 bits of data counter (DCTH)
0000D6 _H	Lower 8 bits of data counter (DCTL)
0000D5 _H	Upper 8 bits of I/O Register address pointer (IOAH)
0000D4 _H	Lower 8 bits of I/O Register address pointer (IOAL)
0000D3 _H	DMA Control register (DMACS)
0000D2 _H	Upper 8 bits of buffer address pointer (BAPH)
0000D1 _H	Middle 8 bits of buffer address pointer (BAPM)
0000D0 _H	Lower 8 bits of buffer address pointer (BAPL)

■ Each Register of DMA Descriptor

Each register making up the DMA descriptor is explained in the following sections. The registers must be initialized before setting ENx to “1” because their initial values become undefined when they are reset.

Note:

Access to the DDWR register is not allowed over 2 machine clock cycles after the setting of the DCSR register is changed.

3.9.3.1 Data counter (DCT)

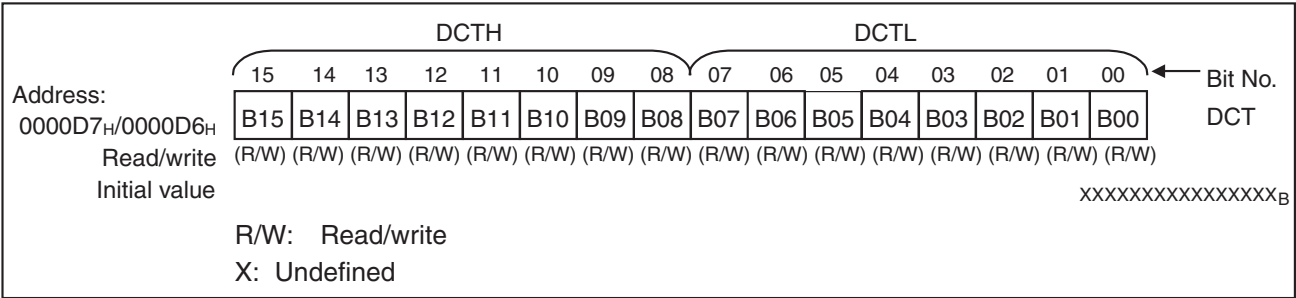
This section describes the Data counter (DCT).

■ Data counter (DCT)

The data counter (DCT) is a 16-bit register to store the transfer count. After each data transfer, the data counter is decremented by 1 irrespective of whether the transfer is word or byte transfer. When the data counter becomes “0”, DMA transfer ends. Figure 3.9-6 “Data Counter (DCT) Configuration” shows the configuration of the data counter (DCT).

When “0” is set to DCT, the maximum data transfer count (65536) is set.

Figure 3.9-6 Data Counter (DCT) Configuration



■ Value in Data counter

The table below explains the relationship between the number of bytes and the value in the DCT register.

DMACS register		DCT register
BW bit	BYTEL bit	
0	-	N
1	0	N/2
1	1	(N+1)/2

N: Number of bytes

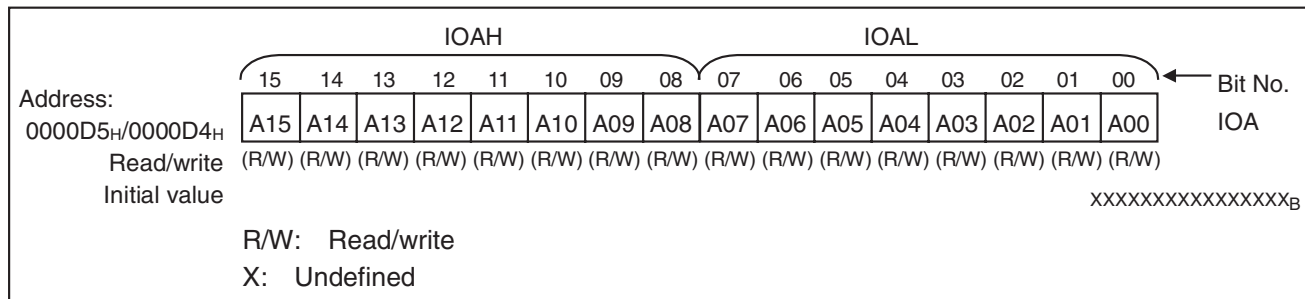
3.9.3.2 I/O register address pointer (IOA)

This section describes the I/O register address pointer (IOA).

■ I/O register address pointer (IOA)

The I/O register address pointer (IOA) is a 16-bit register and indicates the lower addresses (A15 to A00) of the I/O register. Its upper addresses (A23 to A16) are all “0”, allowing any I/O address space in the range from the “000000_H” address to the “00FFFF_H” address to be specified. When “Update Performed” is specified with the IF bit (this bit selects whether to update IOA or fix IOA) of the DMA control register (DMACS), IOA is incremented by 1 at byte transfer; and incremented by 2 at word transfer. When “Update Not Performed” is specified with the IF bit, IOA is fixed (IOA does not change). Figure 3.9-7 “I/O Register Address Pointer (IOA) Configuration” shows the configuration of the data counter (IOA).

Figure 3.9-7 I/O Register Address Pointer (IOA) Configuration



3.9.3.3 DMA control register (DMACS)

This section describes the DMA control register (DMACS).

■ DMA control register (DMACS)

The DMA control register (DMACS) is 8-bit long and is used to specify whether to update or fix the buffer address pointer and the I/O register address pointer, to specify the transfer data format (byte or word), to specify the transfer direction, to specify byte transfer, and to issue wait instructions. Figure 3.9-8 "Configuration of DMACS" shows the configuration of DMACS.

Figure 3.9-8 Configuration of DMACS

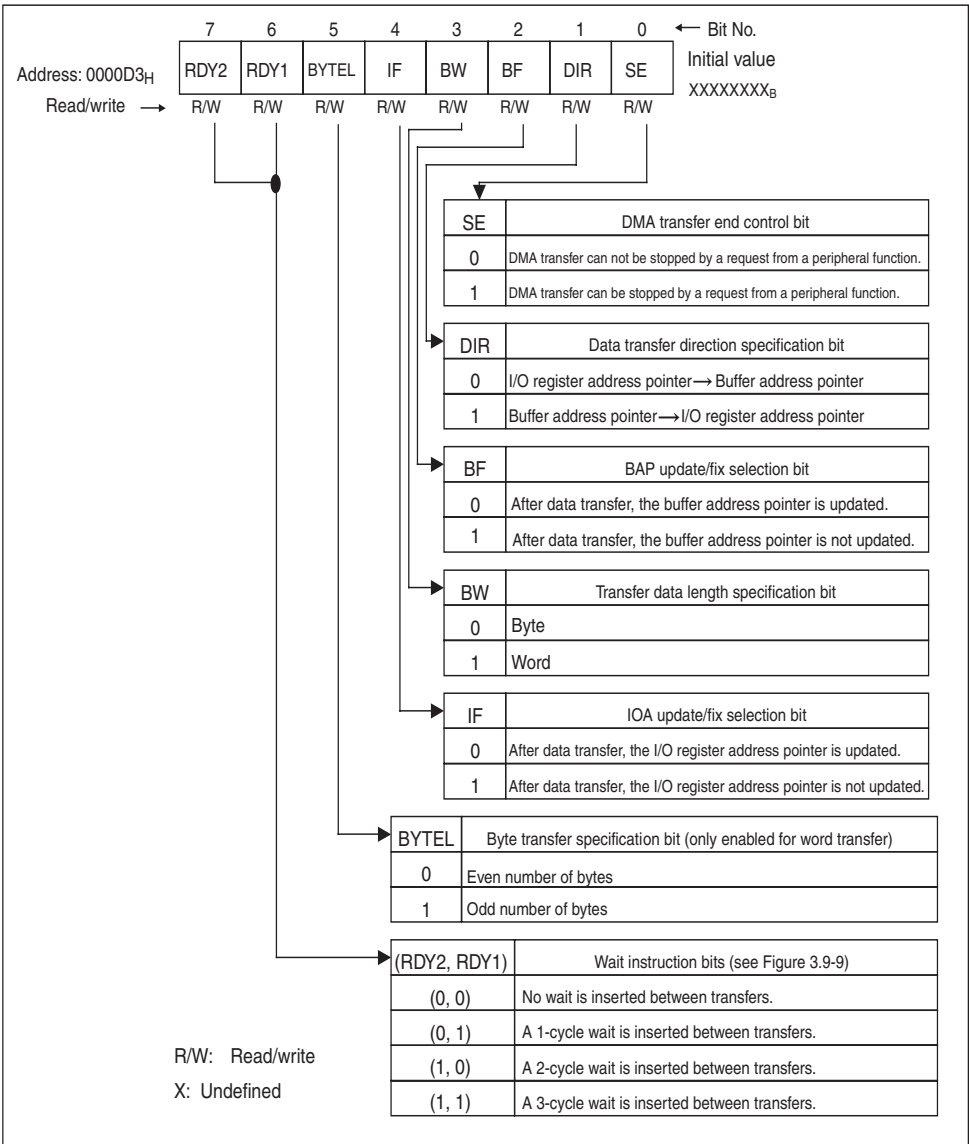
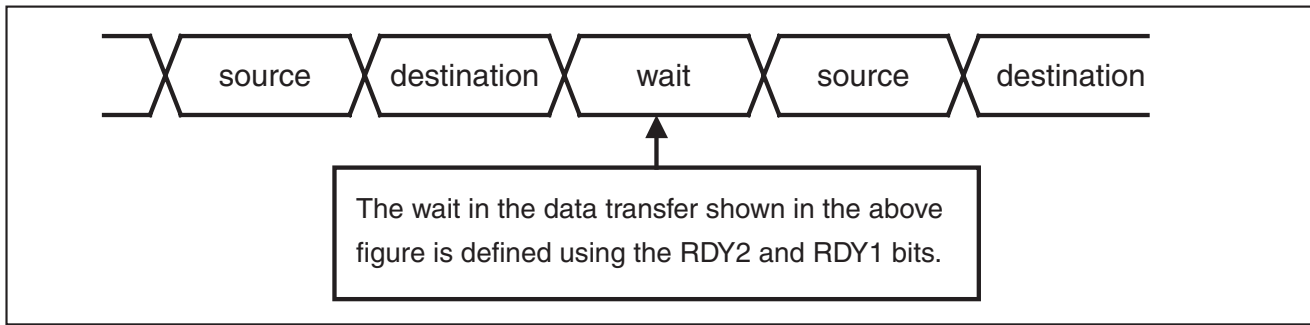


Figure 3.9-9 Explanation of Wait Instruction Bits

Note:

If writing transmission data to LIN-UART by using μ DMAC, not setting RDY2 and RDY1 bit of DMACS register in (0, 0).

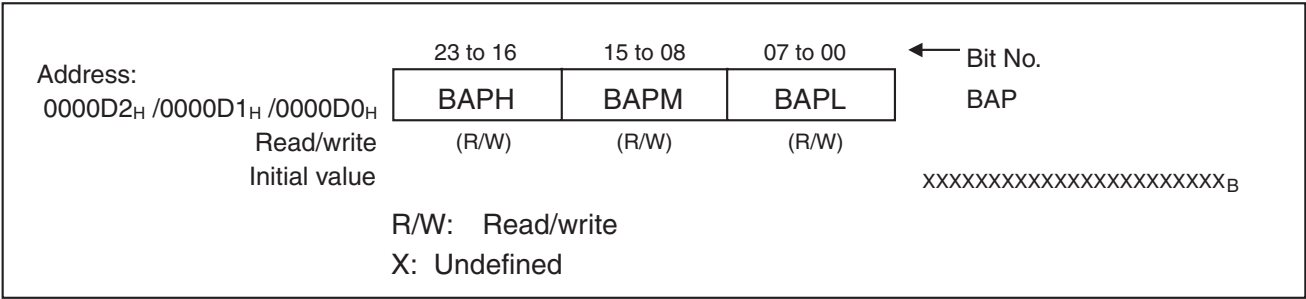
3.9.3.4 Buffer address pointer (BAP)

This section describes the Buffer address pointer (BAP).

■ Buffer address pointer (BAP)

The buffer address pointer (BAP) is a 24-bit register and is used to store addresses that will be used for DMA transfer. There is an independent BPA for each DMA channel, so each DMA channel can transfer data between any address in the 16-MByte address space and I/O. When "Update Performed" is specified using the BF bit (selects whether to update BAP or fix BAP) of the DMA control register (DMACS), the BAP lower 16 bits (BAPM, BAPL) are incremented by "1" at byte transfer and incremented by "2" at word transfer. Its upper 8 bits do not change. Figure 3.9-10 "Buffer Address Pointer (BAP)" shows the configuration of the buffer address pointer (BAP).

Figure 3.9-10 Buffer Address Pointer (BAP)



Notes:

- The area that can be specified using the I/O register address pointer (IOA) is "000000_H" to "00FFFF_H".
- The area that can be specified using the buffer address pointer (BAP) is "000000_H" to "FFFFFF_H".
- The following may not be specified for IOA and BAP: addresses of μ DMA internal registers (DCSR, DSRH, DSRL, DSSR, DERH, DERL); addresses of DMA descriptor window register (DDWR).

3.9.4 μ DMAC Operation

This section describes the μ DMAC operation.

■ DMAC Operation

Figure 3.9-11 "DMAC Operation" shows the DMAC operation.

Data transfer using DMAC is performed in the following order:

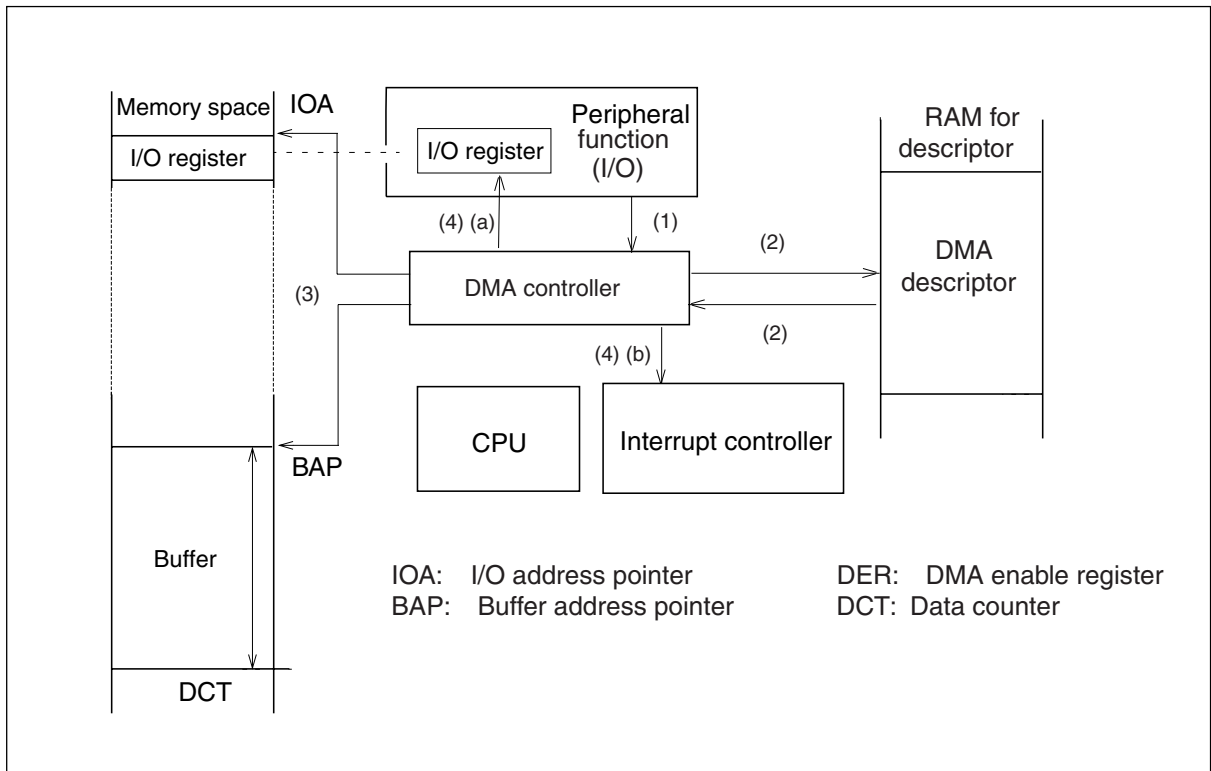
- (1) The peripheral resource (I/O) requests DMA transfer.
- (2) When the corresponding bit of DMA enable register (DER) is "1", DMAC reads from the descriptor the transfer data such as the transfer source address, transfer destination address, and transfer count of specified channels.
- (3) DMA data transfer is started between I/O and memory.
- (4) After one item(either Byte data or Word data) transferred
 - (a) Transfer has not been completed (DCT does not reached to 0):

μ DMAC requests to clear the DMA transfer request to the peripheral resource.
 - (b) At transfer end(DCT reached to 0):

After completion of DMA transfer, the flag indicating completion of transfer is set in the DMA status register, outputting an interrupt request to the interrupt controller.

Note,

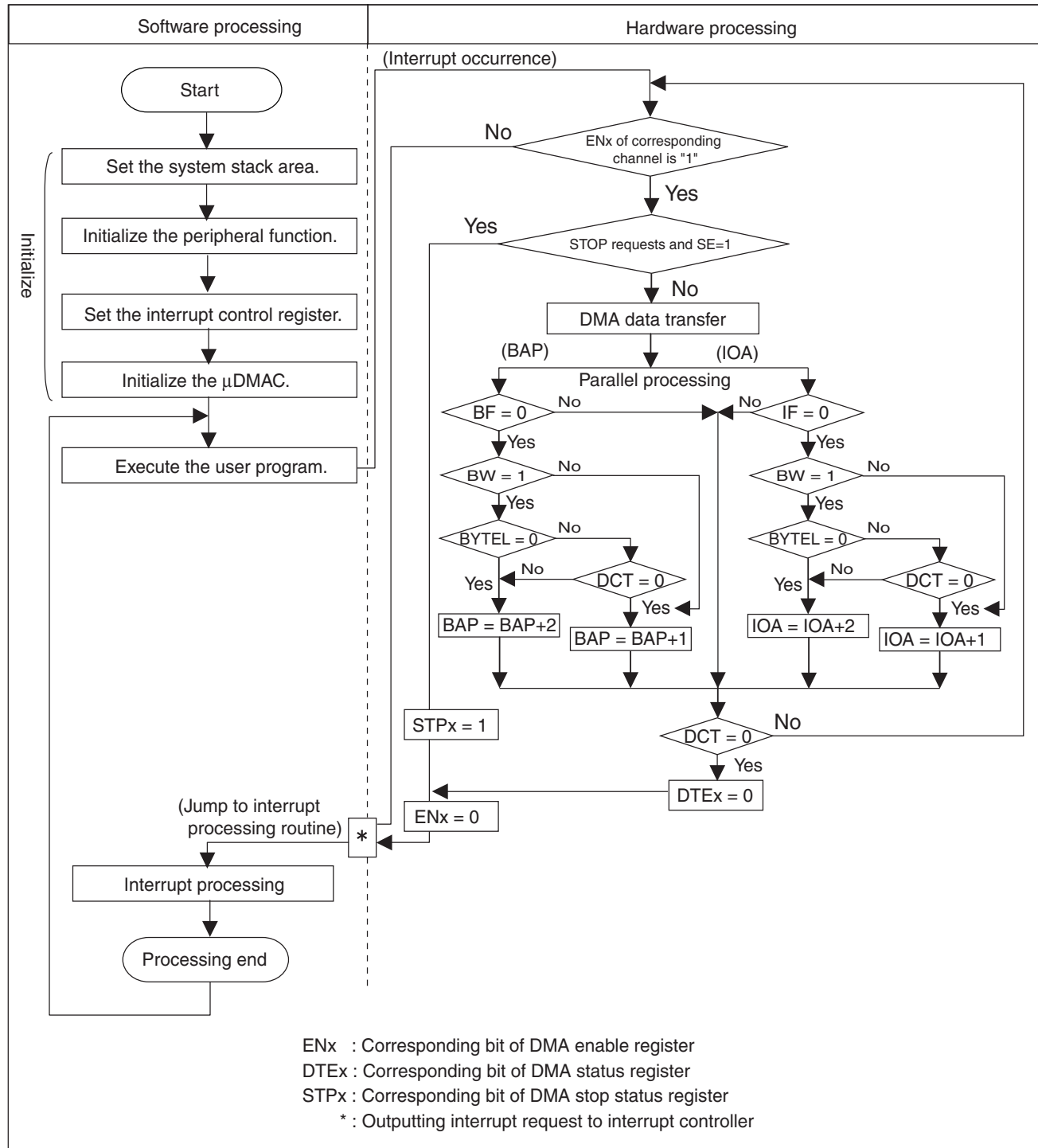
Write access to the internal registers DCSR, DSRH, DSRL, DSSR, DERH, and DERL is prohibited during DMA transfer.

Figure 3.9-11 DMAC Operation

■ Procedure for using DMAC

Figure 3.9-12 "Procedure for Using DMAC" illustrates the procedure for using DMAC.

Figure 3.9-12 Procedure for Using DMAC



■ Data transferring cycle number (Internal transfer)

If μ DMAC should keep bus permission and data should transfer inside LSI, the transfer cycle number is equal to following;

Table 3.9-2 Keoo bus permission → Start transferring

Matching between DCSR3 to DCSR0 of DCSR and interrupt request channel	Matching failed between DCSR3 to DCSR0 of DCSR and interrupt request channel
1 machine cycle	2 machine cycles

Table 3.9-3 Transferring cycle

Address pointer		DMACS		Cycle number
BAP	IOA	BW	BYTEL	
-	-	0	-	$4 + (\text{RDY2}, \text{RDY1})^{*1}$ machine cycle
Odd	Even	1	-	$6 + (\text{RDY2}, \text{RDY1})^{*2}$ machine cycle
Even	Odd			
Odd	Odd	1	-	$8 + (\text{RDY2}, \text{RDY1})^{*2}$ machine cycle
Even	Even	1	-	$4 + (\text{RDY2}, \text{RDY1})^{*1}$ machine cycle

*1: $(\text{RDY2}, \text{RDY1}) = 0$ at the last transferring

*2: At $\text{BYTEL} = 1$, the values are equal to 4 cycle, and $(\text{RDY2}, \text{RDY1}) = 0$.

At $\text{BYTEL} = 0$, $(\text{RDY2}, \text{RDY1}) = 0$.

Odd: Odd address

Even: Even address

3.10 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”

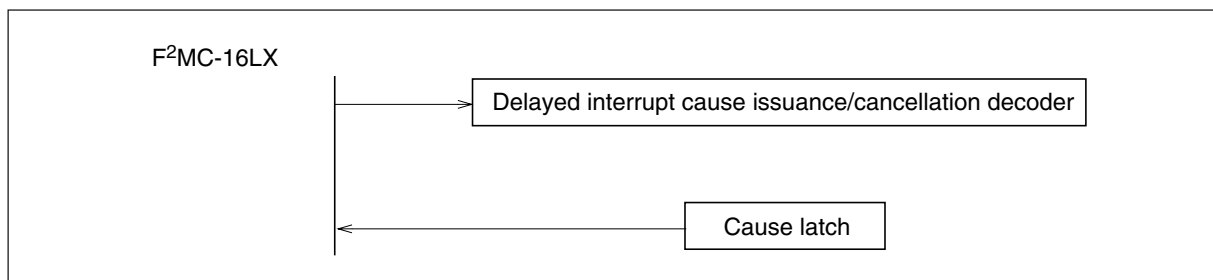
4.1 Outline of Delayed Interrupt Module

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

The delayed interrupt signal is activated by writing "1" to the corresponding bit of DIRR and inactivated by writing "0" to the same bit. Therefore, the interrupt bit in DIRR should be cleared to "0" within the interrupt service routine. Otherwise the same interrupt is serviced again right after the first interrupt service is completed.

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)

Figure 4.2-1 Delayed Interrupt Cause/Cancel Register (DIRR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
00009F _H	-	-	-	-	-	-	-	R0	XXXXXXXX0 _B
	-	-	-	-	-	-	-	R/W	
R/W : Readable and writable X : Undefined - : Undefined bit									

Table 4.2-1 Functional Explanation of Each Bit of the Delayed Interrupt Cause/Cancel Register (DIRR)

Bit name		Function
bit15 to bit9	-: Undefined bit	- When these bits are read, the values are undefined. - Writing to these bits does not affect operation.
bit8	R0: Delayed interrupt request output bit	- This bit sets the generation/cancel of a delayed interrupt request. - When this bit is "1", a delayed interrupt request is output. - When this bit is "0", the delayed interrupt request is cleared. - When a reset is specified, interrupt causes are canceled (cleared to "0").

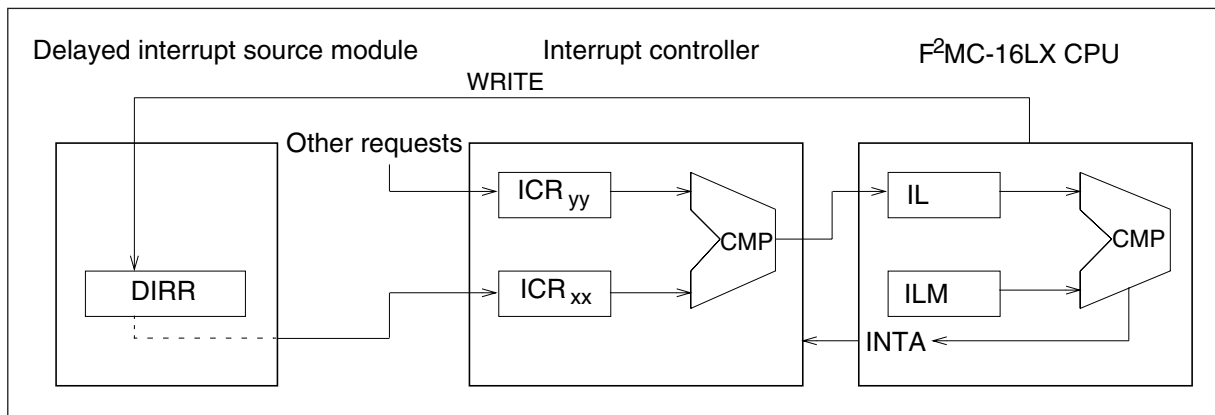
4.3 Delayed Interrupt Operation

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

CLOCKS

This chapter explains the clocks used by MB90340 series microcontrollers.

- 5.1 “Clocks”
- 5.2 “Block Diagram of the Clock Generation Block”
- 5.3 “Clock Selection Registers”
- 5.4 “Clock Mode”
- 5.5 “Oscillation Stabilization Wait Interval”
- 5.6 “Connection of an Oscillator or an External Clock to the Microcontroller”

5.1 Clocks

The clock generation block controls the operation of the internal clock that controls operation of the CPU and peripheral functions. This internal clock is called the machine clock. One internal clock cycle is called one machine cycle. Other clocks include a clock generated by source oscillation, called an oscillation clock, and a clock generated by the internal PLL oscillation, called a PLL clock.

■ Clocks

The clock generation block contains the oscillation circuit that generates the oscillation clock. An external oscillator is attached to this circuit. The oscillation clock can also be supplied by inputting an external clock to the clock generation block. The clock generation block also contains the PLL clock multiplier circuit, which generates five clocks whose frequencies are multiples of the oscillation clock frequency. The clock generation block controls the oscillation stabilization wait interval and PLL clock multiplication as well as internal clock operation by changing the clock with a clock selector.

● Oscillation clock (HCLK)

The oscillation clock is generated either from an external oscillator attached to the oscillation circuit or by the input of an external clock.

● Sub-clock (SCLK)

The sub-clock runs the watch timer and can also be used as a low-speed machine clock.

The sub-clock is generated either from an external oscillator attached to the oscillation circuit or by the input of an external clock.

Frequency of the sub-clock is oscillation clock frequency divided by 2 or 4. The division ratio is determined by SCDS bit of PLL/Subclock Control Register (PSCCR).

● Main clock (MCLK)

The main clock, whose frequency is the oscillation clock frequency divided by 2, supplies the clock input to the timebase timer and the clock selector.

● PLL clock (PCLK)

The PLL clock is obtained by multiplying the oscillation clock frequency with the internal PLL clock multiplier circuit (PLL oscillation circuit). Selection can be made from among five different PLL clocks.

● Machine clock (ϕ)

The machine clock controls the operation of the CPU and peripheral functions. One clock cycle is regarded as one machine cycle ($1/\phi$). An operating machine clock can be selected from among the main clock (whose frequency is the source clock frequency divided by 2) and the other five clocks (whose frequencies are multiples of the source clock frequency).

Note:

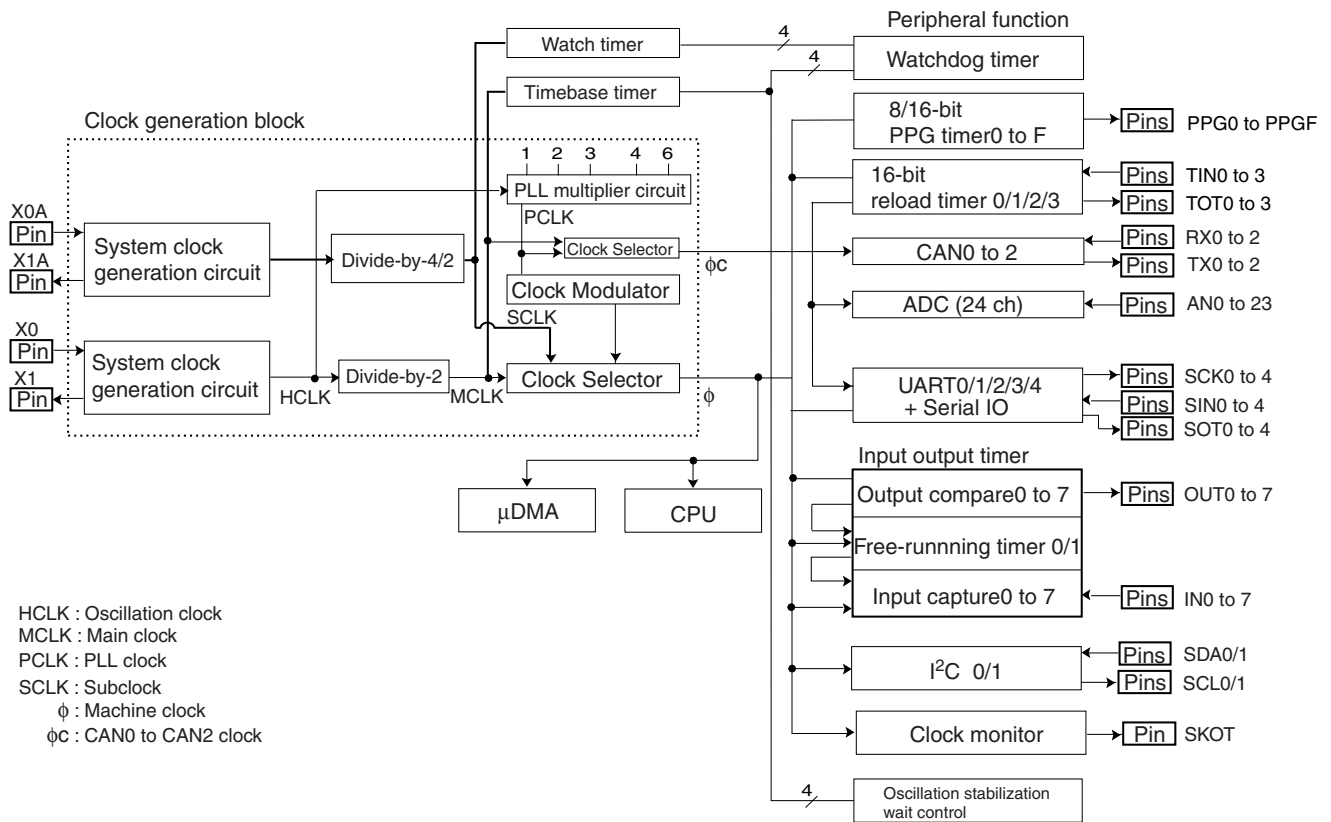
When the operating voltage is 5V, the OSC source oscillation can be between 3MHz and 16MHz. When an external clock source is used, its frequency can be between 3MHz and 24MHz. The highest operating frequency for the CPU and peripheral resource circuits is 24MHz, however. Normal operation is not guaranteed if a multiplication factor resulting in a higher frequency than 24MHz is specified. For example, if the external clock frequency is 16MHz, only 1 can be specified as the multiplication factor.

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

■ Clock Supply Map

Since the machine clock generated in the clock generation block is supplied as the clock that controls the operation of the CPU and peripheral functions, the operation of the CPU and the peripheral functions is affected by switching between the main clock, the PLL clock and the subclock (clock mode) and by a change in the PLL clock multiplier. Since some peripheral functions receive frequency-divided output from the timebase timer, a peripheral unit can select the clock best suited for this operation. Figure 5.1-1 “Clock supply map” shows the clock supply map.

Figure 5.1-1 Clock supply map



5.2 Block Diagram of the Clock Generation Block

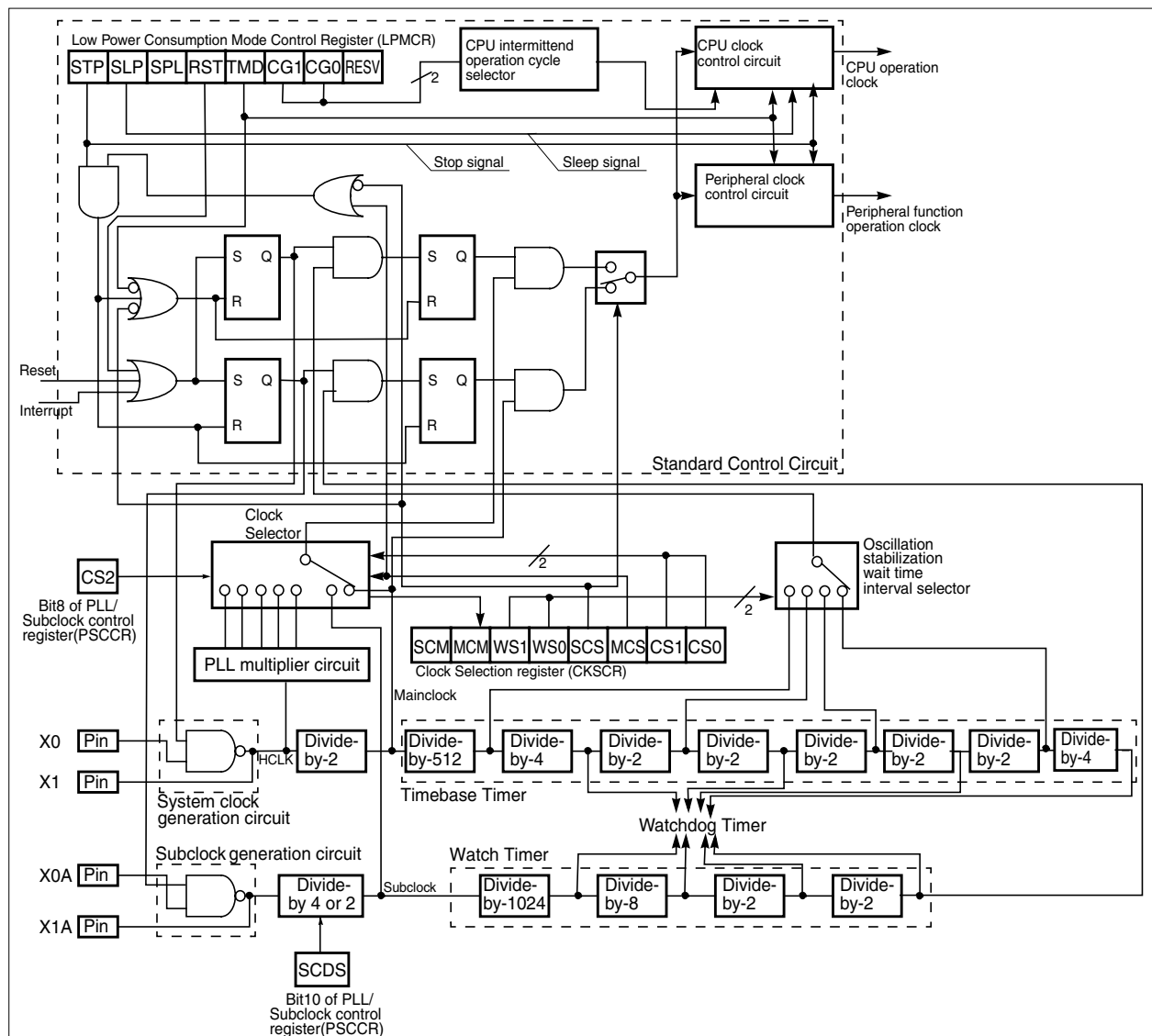
The clock generation block consists of five blocks:

- System clock generation circuit
- PLL multiplier circuit
- Clock selector
- Clock selection register (CKSCR)
- Oscillation stabilization wait interval selector

■ Block Diagram of the Clock Generation Block

Figure 5.2-1 “Block Diagram of the Clock Generation Block” shows a block diagram of the clock generation block.

Figure 5.2-1 Block Diagram of the Clock Generation Block



- Oscillation clock generation circuit

The oscillation clock generation circuit generates an oscillation clock (HCLK) from an external oscillator attached to it. Alternatively, an external clock can be input to this circuit.

- Sub-clock generation circuit

The sub-clock generation circuit generates a sub-clock (SCLK) from an external oscillator attached to it. An external clock can be also input to this circuit.

- PLL multiplier circuit

The PLL multiplier circuit multiplies the oscillation clock frequency through PLL oscillation and supplies a clock whose frequency is a multiple of the oscillation clock frequency to the CPU clock selector.

- Clock selector

From among the main clock, five different PLL clocks and subclock, the clock selector selects the clock that is supplied to the CPU and peripheral clock control circuits.

- Clock selection register (CKSCR)

The clock selection register is used to switch between the main clock, PLL clocks and subclock, also used to select an oscillation stabilization wait interval and a PLL clock multiplier.

- PLL/Subclock Control Register (PSCCR)

The PLL/subclock control register is used to select multiplication factor of the PLL (CS2 bit in this register in addition to CS1 and CS0 bit in the CKSCR register) and to specify division factor (1/4 or 1/2) of the subclock.

- Oscillation stabilization wait interval selector

This oscillation stabilization wait interval selector selects an oscillation stabilization wait interval for the oscillation clock when the stop mode is released or when a watchdog timer reset occurs. Selection is made from among three different timebase timer outputs. In all other cases, an oscillation stabilization wait interval is not selected.

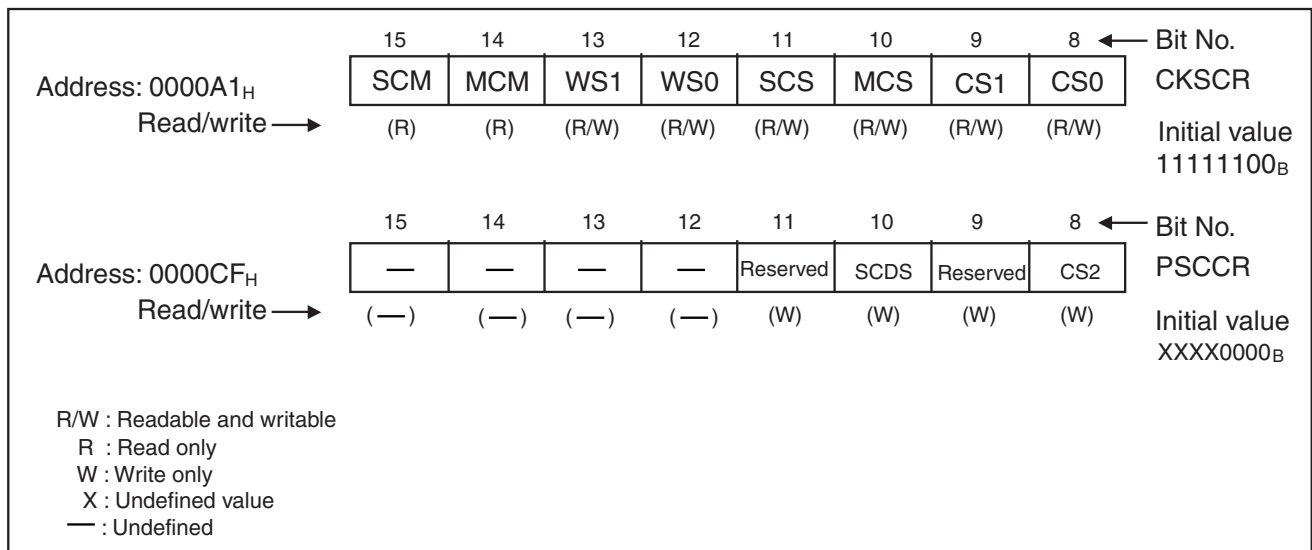
5.3 Clock Selection Registers

This section lists the clock selection registers and describes the function of each register in details.

■ Clock Selection Registers

Figure 5.3-1 “Clock selection registers” shows the clock selection register (CKSCR) and PLL/Subclock Control Register (PSCCR)

Figure 5.3-1 Clock selection registers



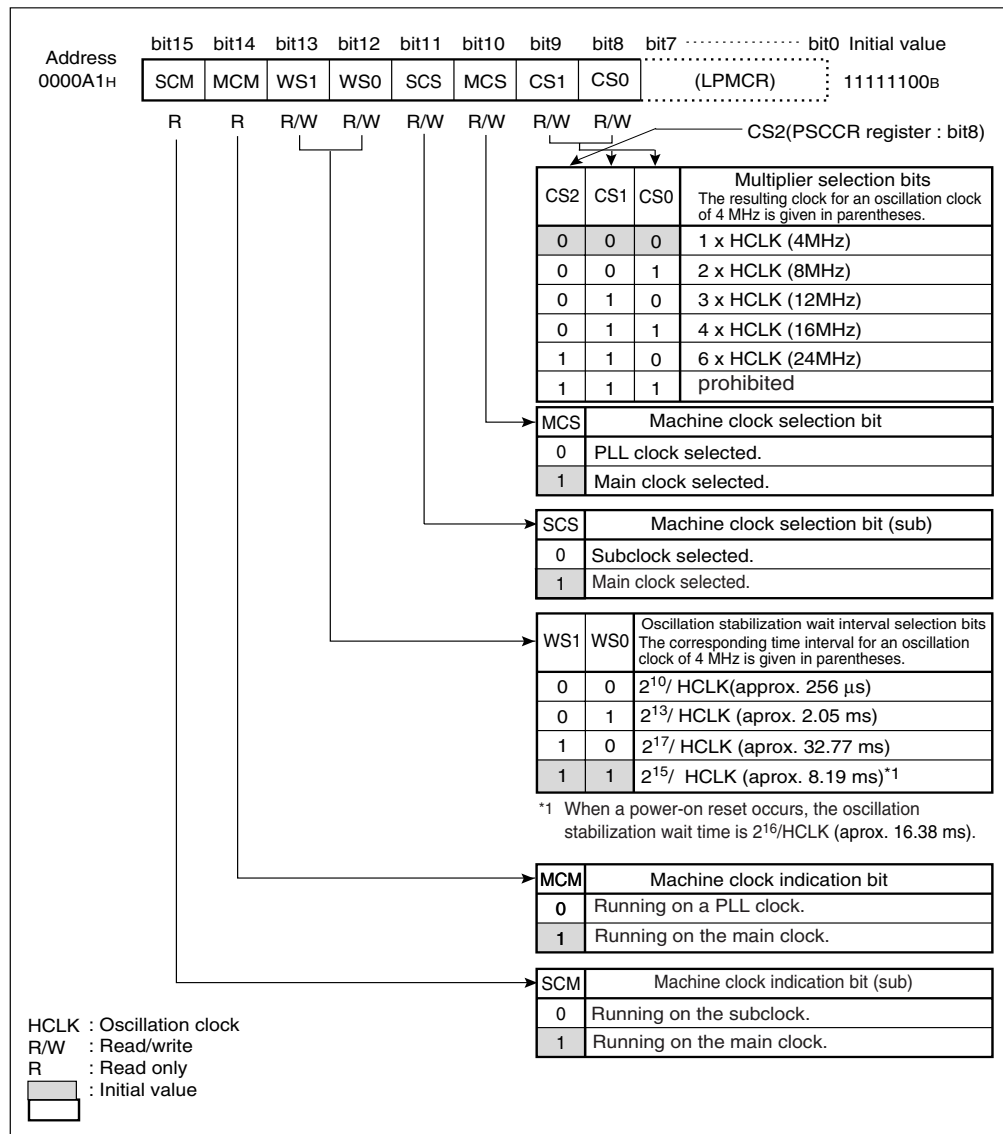
5.3.1 Clock Selection Register (CKSCR)

The clock selection register (CKSCR) is used to switch between the main clock, PLL clocks and subclock, also used to select an oscillation stabilization wait interval and a PLL clock multiplier.

■ Configuration of the Clock Selection Register (CKSCR)

Figure 5.3-2 “Configuration of the Clock Selection Register (CKSCR)” shows the configuration of the clock selection register (CKSCR). Table 5.3-1 “Function Description of Each Bit of the Clock Selection Register (CKSCR)” describes the function of each bit of the clock selection register (CKSCR).

Figure 5.3-2 Configuration of the Clock Selection Register (CKSCR)



Note:

The machine clock selection bit is initialized to main clock selection at a reset.

Table 5.3-1 Function Description of Each Bit of the Clock Selection Register (CKSCR) (1/3)

Bit name		Function
bit 15	SCM: Machine clock indication bit (sub)	- This bit indicates whether the main clock or sub-clock has been selected as the machine clock. - When this bit is “0”, the sub-clock has been selected. When it is “1”, the main clock has been selected. - If SCS = 1 and SCM = 0, the main clock oscillation stabilization wait interval is in effect.
bit 14	MCM: Machine clock indication bit	- This bit indicates whether the main clock or a PLL clock has been selected as the machine clock. - When this bit is 0, a PLL clock has been selected. When it is 1, the main clock has been selected. - If MCS = 0 and MCM = 1, the PLL clock oscillation stabilization wait interval is in effect.
bit 13 bit 12	WS1 and WS0: Oscillation stabilization wait interval selection bits	- These bits select an oscillation stabilization wait interval of the oscillation clock when stop mode was released, when transition occurred from sub-clock mode to main clock mode, or when transition occurred from sub-clock mode to PLL clock mode. - These bits are initialized to “11_B” by all reset causes. Note: The oscillation stabilization wait interval must be set to a value appropriate for the oscillator used. See Section 7.2, “Reset Causes and Oscillation Stabilization Wait Intervals.” These bits can be reset to “00_B” only for main clock mode. Reference: When main clock mode is switched to PLL clock mode, the oscillation stabilization wait interval is fixed at $2^{14}/\text{HCLK}$ (that is, the oscillation stabilization wait interval is approximately 4.1 ms when the oscillation clock frequency is 4 MHz). When sub-clock mode is switched to PLL clock mode or when PLL stop mode is released, the oscillation stabilization wait interval uses the specified values in the WS1 and WS0 bits. For PLL oscillation stabilization, at least $2^{14}/\text{HCLK}$ is required. Accordingly, when sub-clock mode is switched to PLL clock mode, or when PLL clock mode is switched to PLL stop mode, set WS1 and WS0 bits to “10_B” or “11_B”.

Table 5.3-1 Function Description of Each Bit of the Clock Selection Register (CKSCR) (2/3)

Bit name		Function
bit 11	SCS: Machine clock selection bit (sub)	- This bit specifies whether the main clock or sub-clock is selected as the machine clock. - If 0 is written when this bit is 1, the mode is switched to the subclock mode by synchronizing with the subclock (about 130 μs when the sub-clock oscillation frequency is 32kHz and is divided by 4). - If this bit has been set to “0” and “1” is written to it, the oscillation stabilization wait interval for the main clock starts. As a result, the timebase timer is automatically cleared. - When the sub-clock has been selected, the operating clock frequency is the oscillation clock frequency divided by 2 or 4 (that is, the machine clock frequency is 16kHz or 8 kHz when the oscillation clock frequency is 32 kHz). - The subclock division ratio is determined by SCDS bit of PLL/Subclock Control Register (PSCCR). - If both the SCS and MCS bits are “0”, the SCS bit takes precedence, that is, the sub-clock is selected. - This bit is initialized to “1” by all reset causes. Note: The oscillation stabilization wait time of the subclock (about 2s, when the sub-clock oscillation frequency is 32 kHz and is divided by 4) is generated when the power is turned on or the stop mode is canceled. Thus, if the mode is switched from the main clock mode to the subclock mode during this period, an oscillation stabilization wait time is generated.
bit 10	MCS: Machine clock selection bit	- This bit specifies whether the main clock or a PLL clock is selected as the machine clock. - When this bit is “0”, a PLL clock is selected. When it is 1, the main clock is selected. - If this bit has been set to “1” and “0” is written to it, the oscillation stabilization wait interval for the PLL clock starts. As a result, the timebase timer is automatically cleared, and the TBOF bit of the timebase timer control register (TBTC) is also cleared. - When main clock mode is switched to PLL clock mode, the oscillation stabilization wait interval is fixed at $2^{14}/\text{HCLK}$ (that is, the oscillation stabilization wait interval is approximately 4.1 ms when the oscillation clock frequency is 4 MHz). When sub-clock mode is switched to PLL clock mode or when PLL stop mode is released, the oscillation stabilization wait interval uses the specified values in the WS1 and WS0 bits. For PLL oscillation stabilization, at least $2^{14}/\text{HCLK}$ is required. Accordingly, when sub-clock mode is switched to PLL clock mode, or when PLL clock mode is switched to PLL stop mode, set the WS1 and WS0 bits to “10_B” or “11_B.” - When the main clock has been selected, the operating clock frequency is the oscillation clock frequency divided by 2 (that is, the operating clock is 2 MHz when the oscillation clock frequency is 4 MHz). - This bit is initialized to “1” by all reset causes. Note: When the MCS bit is “1”, write “0” to it only when the timebase timer interrupt is masked by the TBIE bit of the timebase timer control register (TBTC) or the interrupt level register (ILM).

Table 5.3-1 Function Description of Each Bit of the Clock Selection Register (CKSCR) (3/3)

Bit name		Function																												
bit 9 bit 8	CS1 and CS0: Multiplier selection bits	• These bits and CS2 bits in PSCCR register select a PLL clock multiplier. • Selection can be made from among five different multipliers. • These bits are initialized to “00_B” by all reset causes. • Recommended setting of CS2,1,0: <table><tr><th>CS2</th><th>CS1</th><th>CS0</th><th>PLL clock Multiplex time</th></tr><tr><td>0</td><td>0</td><td>0</td><td>x 1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>x 2</td></tr><tr><td>0</td><td>1</td><td>0</td><td>x 3</td></tr><tr><td>0</td><td>1</td><td>1</td><td>x 4</td></tr><tr><td>1</td><td>1</td><td>0</td><td>x 6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>prohibited</td></tr></table> Note: When the MCS or MCM bit is “0”, writing to these bits is not possible. Write to the CS0, CS1 and CS2 bits only after setting the MCS bit to “1” (main clock mode).	CS2	CS1	CS0	PLL clock Multiplex time	0	0	0	x 1	0	0	1	x 2	0	1	0	x 3	0	1	1	x 4	1	1	0	x 6	1	1	1	prohibited
CS2	CS1	CS0	PLL clock Multiplex time																											
0	0	0	x 1																											
0	0	1	x 2																											
0	1	0	x 3																											
0	1	1	x 4																											
1	1	0	x 6																											
1	1	1	prohibited																											

HCLK: Oscillation clock

5.3.2 PLL/Subclock Control Register (PSCCR)

PLL/Subclock control register adds the selection of a PLL clock multiplier.

■ Configuration of the PLL/Subclock Control Register (PSCCR)

Figure 5.3-3 “Configuration of the PLL/Subclock Control Register (PSCCR)” shows the configuration of the PLL/Subclock control register (PSCCR).

Figure 5.3-3 Configuration of the PLL/Subclock Control Register (PSCCR)

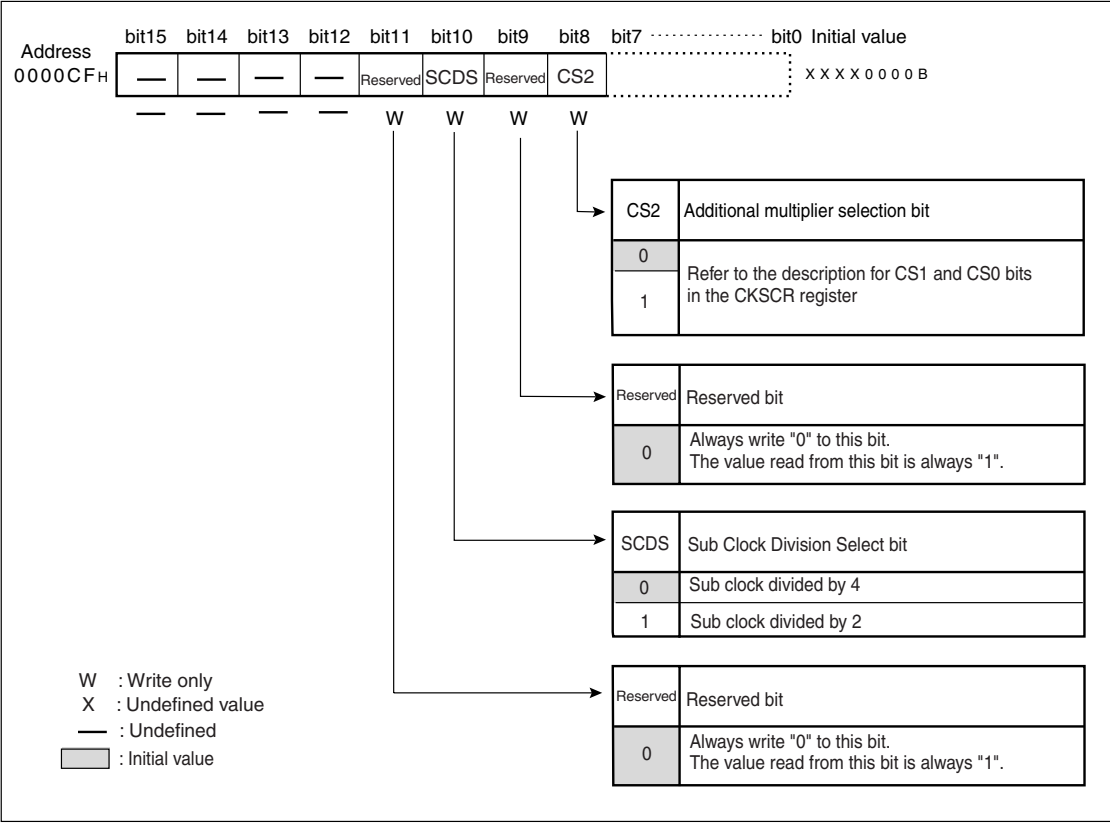


Table 5.3-2 Function Description of Each Bit of the PLL/Subclock Control Register (PSCCR)

Bit name		Function
bit 15 to bit 12	Unused bits	- These bits are unused. - Writing to these bits has no effect. - Reading these bits always returns “1”.
bit 11 to bit 9	Reserved: Reserved bit	- These bits are reserved bits. - Always write “0” to these bits - Reading these bits always returns “1”.
bit 10	SCDS: Sub clock division ratio select bit	- This bit selects the sub clock division ratio. - Writing “0” to this bit sets operating clock in sub clock mode to sub clock oscillation frequency divided by 4. - Writing “1” to this bit sets the clock to the oscillation frequency divided by 2. - This bit is initialized to “0” by all reset causes.
bit 8	CS2: Multiplier selection bit2	- This bit and CS1 and CS0 bits of the Clock selection register (CKSCR) select a PLL clock multiplier. - About the relationship between setting CS2,CS1 and CS0 bits and the PLL clock multiplier selection, please refer to the description for the CS1 and CS0 bits of the CKSCR register. - This bit is initialized to “0” by all reset causes. Note: When the MCS or MCM bit is “0”, changing the setting of this bit is not allowed. Change this bit only after setting the MCS bit to “1” (main clock mode).

Note:

The PSCCR is a write-only register, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

5.4 Clock Mode

Three clock modes are provided: main clock mode, PLL clock mode and sub-clock mode.

■ Main Clock Mode, PLL Clock Mode and Sub-clock Mode

● Main clock mode

In main clock mode, a clock whose frequency is the oscillation clock frequency divided by 2 is used as the operating clock for the CPU and peripheral resources, and the PLL clocks are disabled.

● PLL clock mode

In PLL clock mode, a PLL clock is used as the operating clock for the CPU and peripheral resources. A PLL clock multiplier is selected with the clock selection register (CKSCR: CS1 and CS0) and PLL/Subclock control register (PSCCR: CS2).

● Sub-clock mode

In sub-clock mode, a clock whose frequency is the oscillation clock frequency divided by 2 or 4 is used as the operating clock for the CPU and peripheral resources, and the main clock and PLL clocks are disabled.

The subclock division ratio is determined by SCDS bit of PLL/Subclock Control Register (PSCCR).

■ Clock Mode Transition

Transition among main clock mode, PLL clock mode, and sub-clock mode is performed by writing to the MCS and SCS bits of the clock selection register (CKSCR).

● Transition from main clock mode to PLL clock mode

When the MCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in main clock mode, switching from the main clock to a PLL clock occurs after the PLL clock oscillation stabilization wait interval ($2^{14}/HCLK$).

● Transition from PLL clock mode to main clock mode

When the MCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in PLL clock mode, switching from the PLL clock to the main clock occurs when the edges of the PLL clock and the main clock coincide (after 1 to 12 PLL clocks).

● Transition from main clock mode to sub-clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in main clock mode, switching from the main clock to a sub-clock occurs by synchronizing with the subclock (about 130μs when the sub-clock oscillation frequency is 32kHz and is divided by 4).

● Transition from sub-clock mode to main clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in sub-clock mode, switching from the sub-clock to the main clock occurs after the main clock oscillation stabilization wait interval. The oscillation stabilization wait interval is selected based on the WS1 and WS0 bits of the clock selection register (CKSCR).

● Transition from PLL clock mode to sub-clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in PLL clock mode, switching from the PLL clock to the sub-clock occurs.

● Transition from sub-clock mode to PLL clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in sub-clock mode, switching from the sub-clock to a PLL clock occurs after the main clock oscillation stabilization wait interval. The oscillation stabilization wait interval is selected based on the WS1 and WS0 bits of the clock selection register (CKSCR).

Notes:

Even though the MCS and SCS bits of the clock selection register (CKSCR) are rewritten, machine clock switching does not occur immediately. When operating a resource that depends on the machine clock, confirm that machine clock switching has been performed by referring to the MCM and SCM bits of the clock selection register (CKSCR) before operating the resource.

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

If both the SCS and MSC bits are “0”, the SCS bit takes precedence, that is, sub-clock mode is selected.

■ Selection of a PLL Clock Multiplier

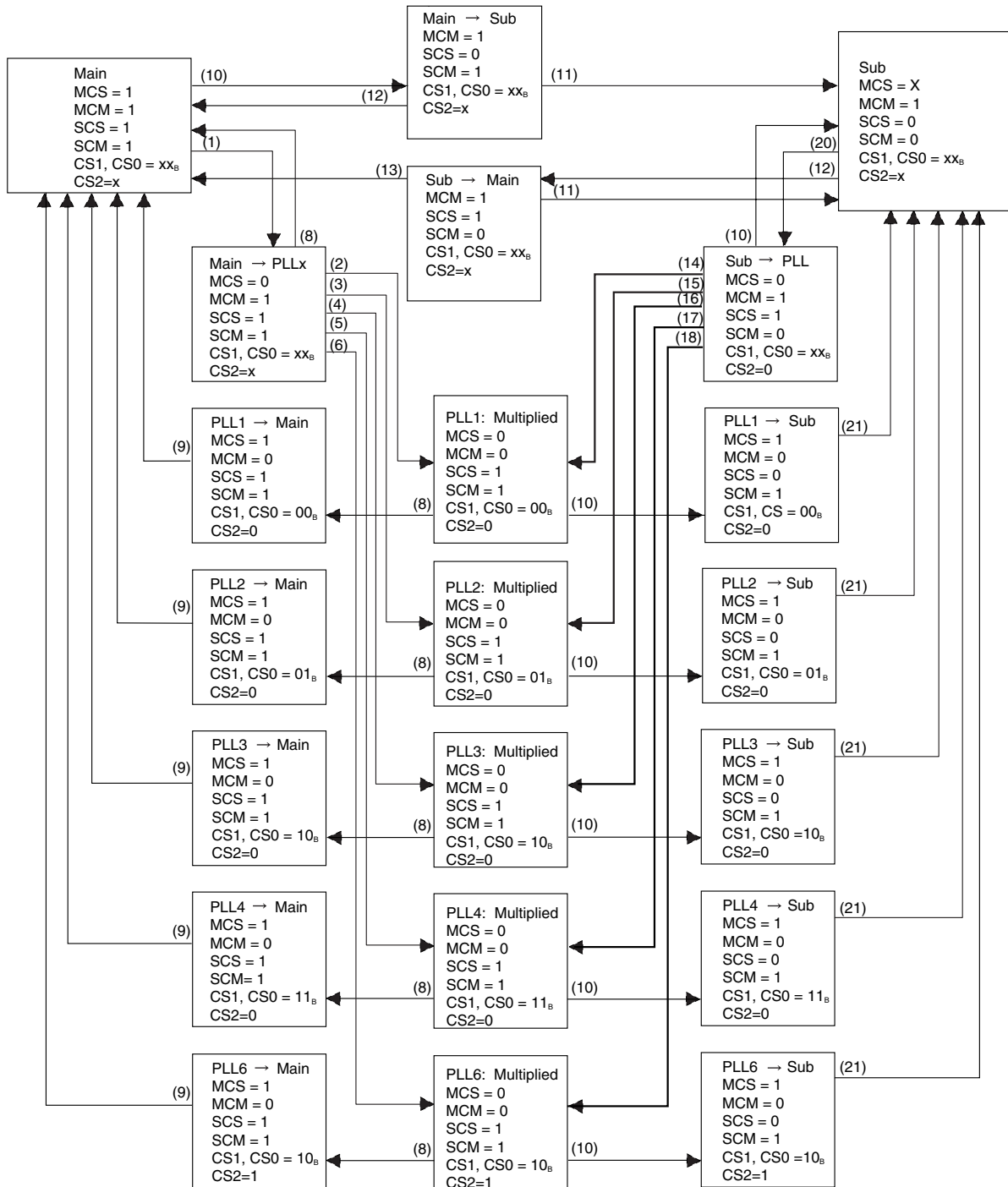
Writing a value from “00_B” to “11_B” to the CS1 and CS0 bits of the clock selection register (CKSCR) and “0” or “1” to the CS2 bit of the PLL/Subclock control register (PSCCR) selects a PLL clock multiplier of 1 to 4 (when CS2=0), 2 to 6 (when CS2=1, CS1=1, CS0=0).

■ Machine Clock

The machine clock may be a PLL clock output from the PLL multiplier circuit, a clock whose frequency is the source oscillation frequency divided by 2, or a clock whose frequency is the sub-clock oscillation frequency divided by 4 or 2. This machine clock is supplied to the CPU and peripheral functions. The main clock, PLL clock, or sub-clock can be selected by writing to the MCS or SCS bit of the clock selection register (CKSCR). And the sub-clock division ratio, 4 or 2, can be selected by writing to SCDS bit of the PLL/Subclock control register (PSCCR).

Figure 5.4-1 “Status Change Diagram for Machine Clock Selection” shows the status change caused by machine clock switching.

Figure 5.4-1 Status Change Diagram for Machine Clock Selection



- (1) Writing “0” to the MCS bit
- (2) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 00_B & CS2 = 0
- (3) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 01_B & CS2 = 0
- (4) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 10_B & CS2 = 0
- (5) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 11_B & CS2 = 0
- (6) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 10_B & CS2 = 1
- (8) Writing “1” to the MCS bit (including reset)
- (9) Timing of synchronization between the PLL clock and the main clock
- (10) Writing “0” to the SCS bit
- (11) Timing of synchronization between the main clock and the sub-clock.
- (12) Writing “1” to the SCS bit (MCS = 1)
- (13) End of main clock oscillation stabilization wait
- (14) End of main clock oscillation stabilization wait & CS1 and CS0 = 00_B & CS2 = 0
- (15) End of main clock oscillation stabilization wait & CS1 and CS0 = 01_B & CS2 = 0
- (16) End of main clock oscillation stabilization wait & CS1 and CS0 = 10_B & CS2 = 0
- (17) End of main clock oscillation stabilization wait & CS1 and CS0 = 11_B & CS2 = 0
- (18) End of main clock oscillation stabilization wait & CS1 and CS0 = 10_B & CS2 = 1
- (20) Writing “1” to the SCS bit (MCS = 0)
- (21) Timing of synchronization between the PLL clock and the sub-clock

MCS	: Machine clock selection bit of the clock selection register (CKSCR)
MCM	: Machine clock indication bit of the clock selection register (CKSCR)
SCS	: Machine clock selection bit of the clock selection register (CKSCR) (sub)
SCM	: Machine clock indication bit of the clock indication register (CKSCR) (sub)
CS1, CS0	: Multiplier selection bits of the clock selection register (CKSCR)
CS2	: Multiplier selection bits of the PLL/subclock control register (PSCCR)

Notes:

- The initial value for the machine clock setting is main clock (CKSCR: MCS = 1, SCS = 1).
- If both the SCS and MCS bits are “0”, the SCS bit takes precedence, that is, the sub-clock is selected.
- When sub-clock mode is switched to PLL clock mode, set the WS1 and WS0 bits of CKSCR to “10_B” or “11_B.”

5.5 Oscillation Stabilization Wait Interval

When the power is turned on, when stop mode is released, or switching from the sub-clock to the main clock or from sub-clock to the PLL clock occurs, an oscillation stabilization wait interval is required after oscillation begins because there is no oscillation. When switching from the main clock to the PLL clock or from the main clock to the sub-clock occurs, an oscillation stabilization wait interval is required.

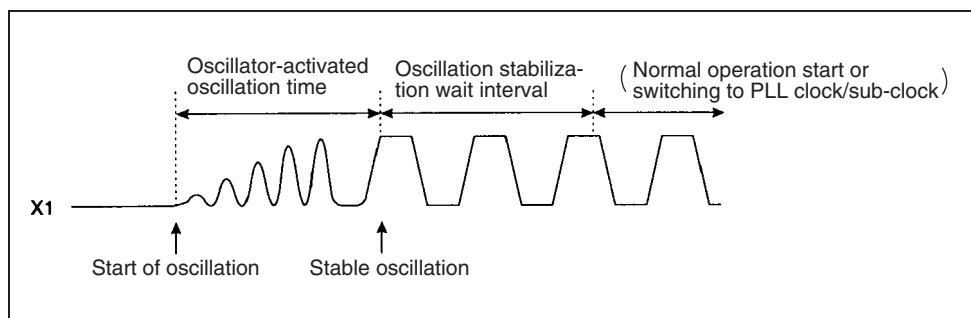
■ Oscillation Stabilization Wait Interval

Ceramic and crystal oscillators generally require several to dozens of ms to stabilize at their natural frequency (oscillation frequency) when oscillation starts. For this reason, CPU operation is not allowed immediately after oscillation starts but is allowed only after full oscillation stabilization. After the oscillation stabilization wait interval has elapsed, the clock is supplied to the CPU. Because the oscillation stabilization time depends on the type of oscillator (crystal, ceramic, etc.), the proper oscillation stabilization wait interval for the oscillator used must be selected. An oscillation stabilization wait interval is selected by setting the clock selection register (CKSCR).

When clock mode is switched from main clock to PLL clock, main clock to subclock, subclock to main clock, or subclock to PLL clock, the CPU runs in the clock mode set before switching for the oscillation stabilization wait time. After the oscillation stabilization wait time has elapsed, the CPU changes to the specified clock mode.

Figure 5.5-1 “Operation Immediately after Oscillation Starts” shows the operation immediately after oscillation starts.

Figure 5.5-1 Operation Immediately after Oscillation Starts



5.6 Connection of an Oscillator or an External Clock to the Microcontroller

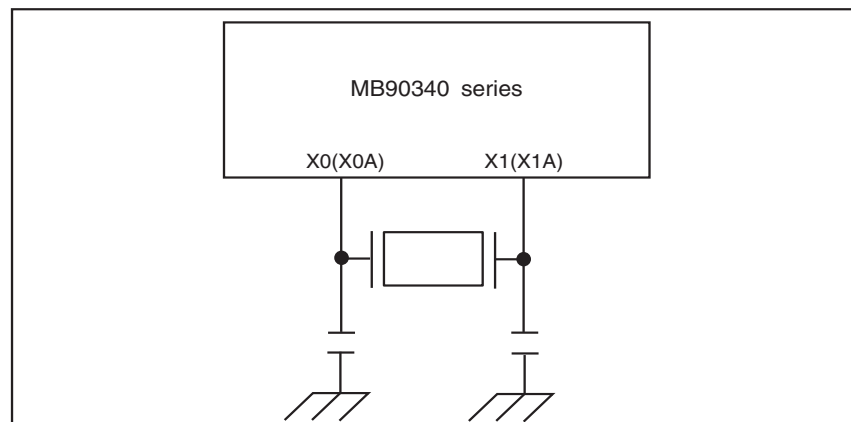
The MB90340 series microcontroller contains a system clock generation circuit. Connecting an external oscillator to this circuit generates the system clock. Alternatively, an externally generated clock can be input to the microcontroller.

■ Connection of an Oscillator or an External Clock to the Microcontroller

● Example of connecting a crystal or ceramic oscillator to the microcontroller

Connect a crystal or ceramic oscillator as shown in the example in Figure 5.6-1 “Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller”.

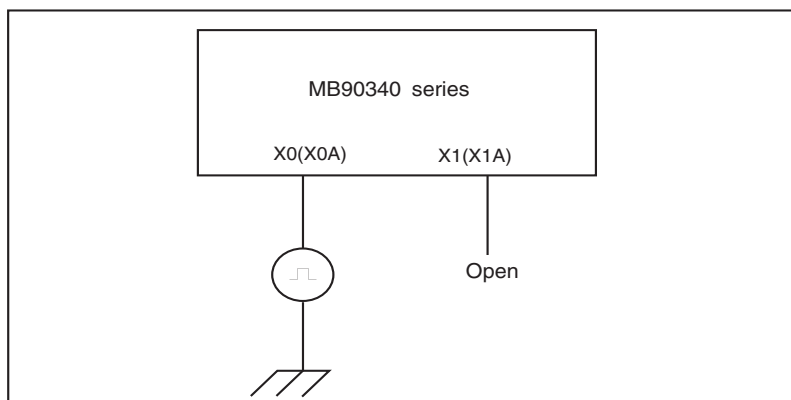
Figure 5.6-1 Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller



● Example of connecting an external clock to the microcontroller

As shown in the example in Figure 5.6-2 “Example of Connecting an External Clock to the Microcontroller”, connect an external clock to pin X0 (X0A). Pin X1 (X1A) must be open.

Figure 5.6-2 Example of Connecting an External Clock to the Microcontroller



CHAPTER 6

CLOCK MODULATOR

This chapter provides an overview of the Clock Modulator and its features. It describes the register structure and operation of the Clock Modulator.

6.1 “Overview”

6.2 “Clock Modulator Control Register (CMCR)”

6.3 “Application Note”

Note: This function is being evaluated, so cannot be available.

6.1 Overview

This section describes an overview of the Clock Modulator.

■ Overview

The clock modulator is intended for the reduction of electromagnetic interference - EMI, by spreading the spectrum of the clock signal over a wide range of frequencies.

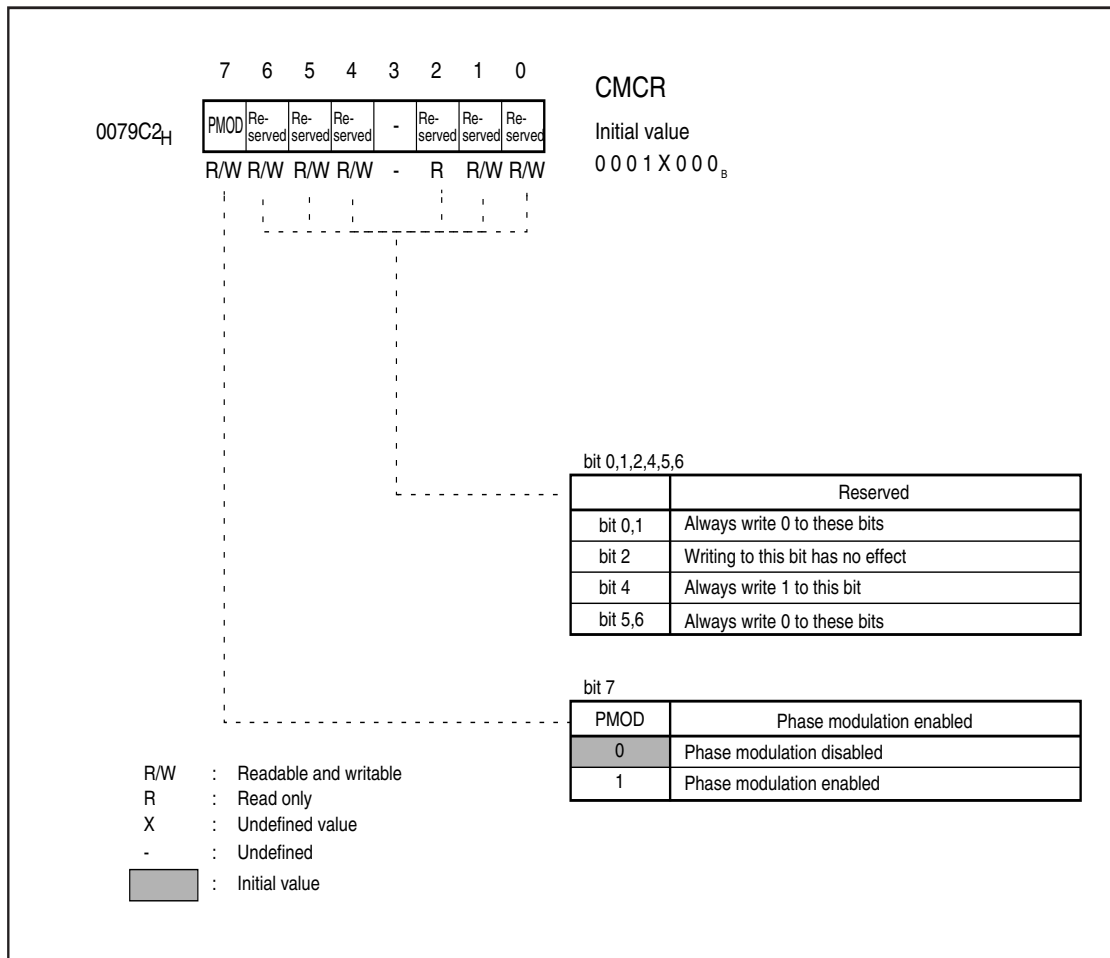
The module is fed with an unmodulated reference clock provided by the PLL circuit. This reference clock is phase modulated by a triangular waveform.

6.2 Clock Modulator Control Register (CMCR)

The Control Register (CMCR) set the phase clock modulator to enable or disable.

■ Clock modulator control register (CMCR)

Figure 6.2-1 Configuration of the clock modulator control register (CMCR)



■ Clock modulator control register contents

Table 6.2-1 Function of each bit of the clock modulator control register

Bit name		Function
bit7	PMOD: Phase modulation enable bit	Writing “0”: Phase modulation disabled (default). Writing “1”: Modulator enabled, MCU is running with phase modulated clock • To enable the modulator, PMOD must be set to 1. • Before the modulator can be enabled, the PLL must deliver a stable reference clock (PLL lock time must be elapsed - refer to the CLOCK chapter in the hardware manual). • Whenever the PLL output frequency is changed or the PLL is switched off e.g. in power down modes, the modulator must be disabled before -> PMOD=0 and 4 NOP cycles must follow the PMOD-bit access. • After enabling the clock modulator, the clock switches immediately to modulated clock without glitches in the clock signal. Please refer to the application note for a description of the recommended startup sequence. • The status of the clock signal is indicated by PMOD. PMOD=1 clock is phase modulated. • The pulse width of the phase modulated clock signal can vary +/- 0.8 ns. E.g. at F0 = 24 MHz unmodulated input clock, T0 = 41.7ns. Tmodmin = 41.7ns - 0.8ns = 40.9ns --> Fmodmax = 1/40.9 ns = 25.45 MHz.
bit 6 to 5	Reserved	Always write 0 to these bits.
bit 4	Reserved	Always write 1 to this bit.
bit 3	Undefined	
bit 2	Reserved	This bit read only and writing to this bit has no effect. Reading this bit always returns 0.
bit 1 to 0	Reserved	Always write 0 to these bits.

6.3 Application Note

Startup/stop sequence for the clock modulator.

■ Recommended startup sequence for the clock modulator

start	1.	Switch on PLL
	2.	Wait PLL lock time (refer to the MCM flag description in the CLOCK chapter of the hardware manual).
	3.	Enable the clock modulator (PMOD=1). The clock switches immediately to modulated clock ... running...
stop	4.	Disable modulator PMOD=0
	5.	4 NOP cycles
	6.	Disable PLL, switch to power down mode, etc.

Note:

Do not enable the modulator before the PLL lock time has elapsed. Do not disable the PLL while the modulator is running.

■ CAN Direct Mode Setting

When CAN module is used, the DIRECT bit of CAN direct mode register (CDMR) should be set to following. If the DIRECT bit should not be set correct, the CAN module will be causing a malfunction.

	Clock modulation setting (CMCR: PMOD bit)	CAN direct mode setting (CDMR: DIRECT bit)
Enable	0 : Disabled clock modulation (initial value)	1 : Enabled CAN direct mode
Prohibited	1 : Enabled clock modulation	0 : Disabled CAN direct mode (initial value)

Note:

The clock modulator can not be used, because of now testing. So, only following setting can be used; PMOD=0 (initial value), DIRECT=1

For details of CAN direct mode register (CDMR), see "22.15 Setting the CAN Direct Mode Register".

CHAPTER 7

RESETS

This chapter describes resets for the MB90340-series microcontrollers.

7.1 “Resets”

7.2 “Reset Causes and Oscillation Stabilization Wait Times”

7.3 “External Reset Pin”

7.4 “Reset Operation”

7.5 “Reset Cause Bits”

7.6 “Status of Pins in a Reset”

7.1 Resets

If a reset is generated, the CPU immediately stops the current execution process and waits for the reset to be cleared. The CPU then begins processing at the address indicated by the reset vector.

The four causes of a reset are as follows

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

■ Causes of a reset

Table 7.1-1 “Causes of a reset” lists the causes of a reset.

Table 7.1-1 Causes of a reset

Type of reset	Cause	Machine clock	Watchdog timer	Oscillation stabilization wait
Power-on	When the power is turned on	Main clock (MCLK)	Stop	Yes
External pin	L level input to $\overline{\text{RST}}$ pin	Main clock (MCLK)	Stop	No
Software	A "0" is written to the RST bit of the low power consumption mode control register (LPMCR).	Main clock (MCLK)	Stop	No
Watchdog timer	Watchdog timer overflow	Main clock (MCLK)	Stop	No

Main clock: Oscillation clock frequency divided by 2

● External reset

An external reset is generated by the L level input to an external reset pin ($\overline{\text{RST}}$ pin). The minimum required period of the L level is 500ns. The oscillation stabilization wait time is not required for external resets.

In the MB90340 series the external reset has to be min. 100 μs for wake-up from Main-Time base timer mode.

If the reset cause is generated during a write operation (during the execution of a transfer instruction such as MOV), the CPU waits for the reset to be cleared after completion of the instruction only for reset requests via the $\overline{\text{RST}}$ pin. Therefore, the normal write operation is completed even though a reset is input concurrently.

Note that a reset may prevent the data transfer requested by a string-processing instruction (such as MOVS) from being completed because the reset is accepted before a specified number of bytes are transferred.

Also note that when the RDY signal of the external bus interface is kept active, the external reset takes its effect after 16 machine clock cycles.

When the external reset is applied while in the stop mode, subclock mode, sub-sleep mode and watch mode, the external reset signal must be kept active for more than the duration of the required stabilization time of the external oscillator plus 100 μ s.

When the external reset is applied while in the timebase timer mode, the external reset signal must be kept active for more than 100 μ s.

● Software reset

A software reset is an internal reset generated by writing "0" to the RST bit of the low power consumption mode control register (LPMCR). The oscillation stabilization wait time is not required for a software reset.

● Watchdog timer reset

A watchdog timer reset is generated by a watchdog timer overflow that occurs when "0" is written to the WTE bit of the watchdog timer control register (WDTC) within a given time after the watchdog timer is activated. The oscillation stabilization wait time can be set by the clock selection register (CKSCR).

● Power-on reset

A power-on reset is generated when the power is turned on. The oscillation stabilization wait times at $2^{16}/\text{HCLK}$ (approx. 16.38 ms, HCLK = 4 MHz). When the oscillation stabilization wait time has elapsed, the reset is executed.

Reference Definition of clocks

HCLK: Oscillation clock

MCLK: Main clock

ϕ : Machine clock (CPU operating clock)

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

See Chapter 5 "Clocks," for details on machine clocks.

7.2 Reset Cause and Oscillation Stabilization Wait Times

The MB90340 series has four reset causes. The oscillation stabilization wait time for a reset depends on the reset cause.

■ Reset causes and oscillation stabilization wait times

Table 7.2-1 “Reset causes and oscillation stabilization wait times” summarizes reset causes and oscillation stabilization wait times.

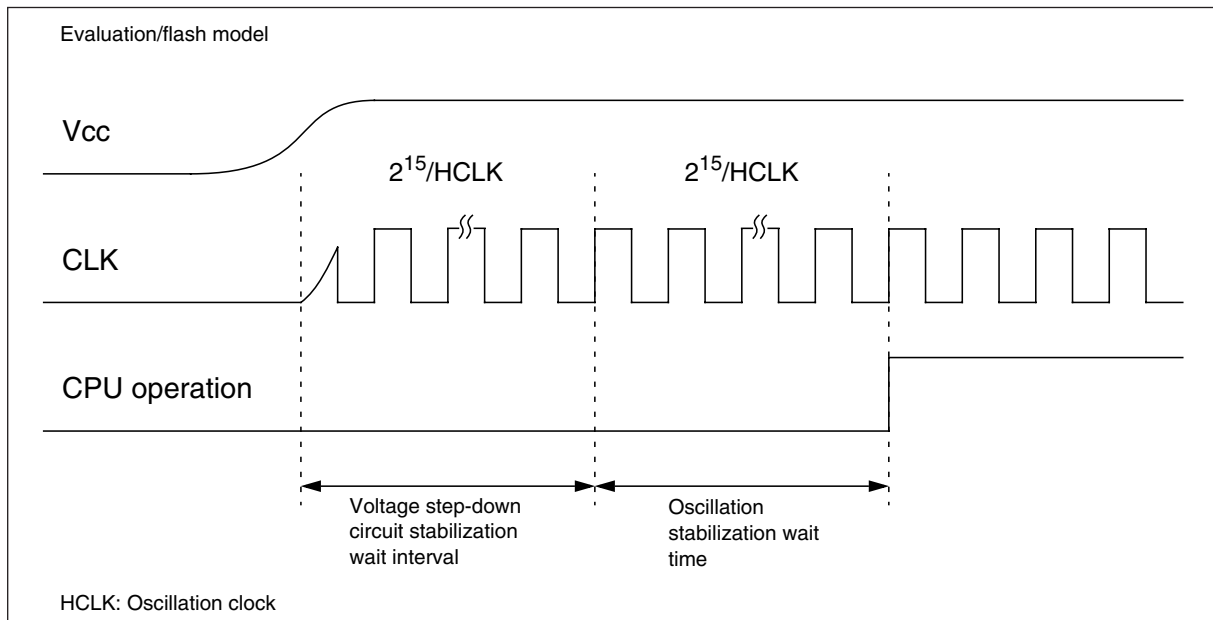
Table 7.2-1 Reset causes and oscillation stabilization wait times

Reset cause	Oscillation stabilization wait time The corresponding time interval for an oscillation clock frequency of 4 MHz is given in parentheses.
Power-on reset	always $2^{16}/\text{HCLK}$ (approx. 16.38 ms at 4 MHz oscillator)
Watchdog timer	None; though bits WS1 and WS0 are initialized to "11".
External reset via the $\overline{\text{RST}}$ pin	None; though bits WS1 and WS0 are initialized to "11".
Software reset	None; though bits WS1 and WS0 are initialized to "11".

HCLK: Oscillation clock

WS1 and WS0: Oscillation stabilization wait time selection bits of the clock selection register (CKSCR)

Figure 7.2-1 “Oscillation stabilization wait times at a power-on reset” shows the oscillation stabilization wait times at a power-on reset.

Figure 7.2-1 Oscillation stabilization wait times at a power-on reset**Note:**

Ceramic and crystal oscillators generally require an oscillation stabilization wait time of several milliseconds to 10 to 20 milliseconds, until stabilization at a natural frequency is attained. A proper oscillation stabilization wait time must be set for the particular oscillator used.

See Section 5.5 “Oscillation Stabilization Wait Time”, for details about oscillation stabilization wait times.

■ Oscillation stabilization wait and reset state

A reset operation in response to a power-on reset and other resets during stop mode or sub-clock mode is performed after the oscillation stabilization wait time has elapsed. This time interval is generated by the timebase timer. If the external reset has not been cleared after the interval, the reset operation is performed after the external reset is cleared.

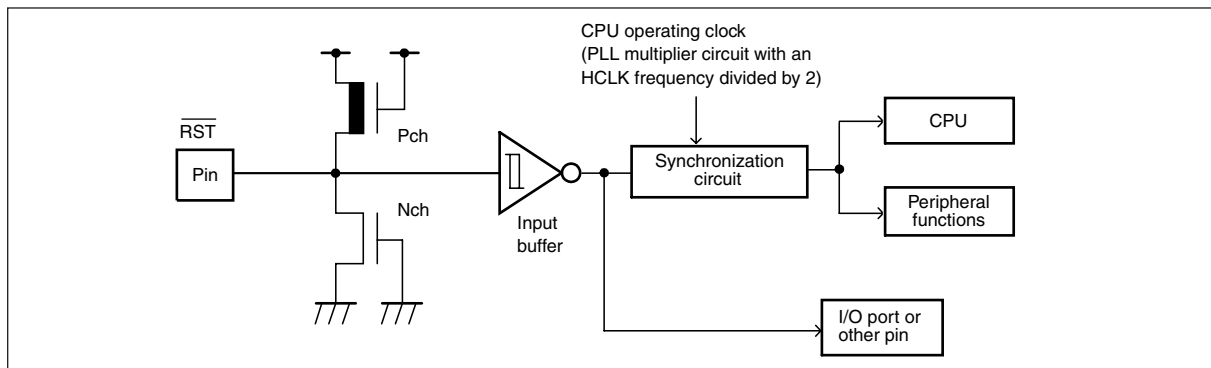
7.3 External Reset Pin

The external reset pin ($\overline{\text{RST}}$ pin) is an input pin used exclusively for a reset. Inputting an L level signal generates an internal reset. For the MB90340-series, resets are generated in synchronization with the CPU operating clock. However, initialization of external pin is asynchronous with the CPU operating clock.

■ Block diagrams of the external reset pin

- Block diagram of the external reset pin

Figure 7.3-1 Block diagram of the external reset pin



Note:

Inputs to the $\overline{\text{RST}}$ are accepted during cycles in which memory is not affected to prevent memory from being destroyed by a reset during a write operation.

A clock is required to initialize the internal circuit. In particular, an operation with an external clock requires clock input together with reset input.

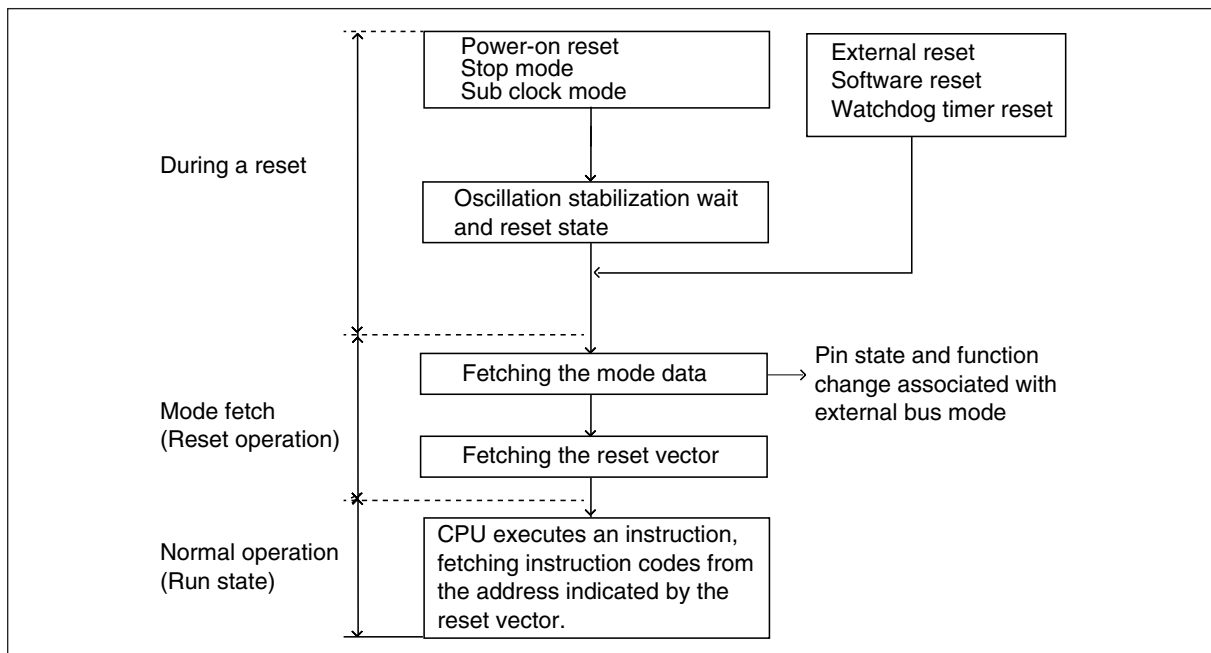
7.4 Reset Operation

When the reset signal is inactivated, the reset vector and mode data is fetched from the predetermined locations depending on the setting of the mode pins. This operation, the mode fetch, then defines the operation mode of the CPU and the start address of the first instruction. For the power on reset, reset from the stop mode or sub-clock mode, the mode fetch is preformed after the oscillation stabilization wait time is elapsed.

■ Overview of reset operation

Figure 7.4-1 “Reset operation flow” shows the reset operation flow.

Figure 7.4-1 Reset operation flow



■ Mode pins

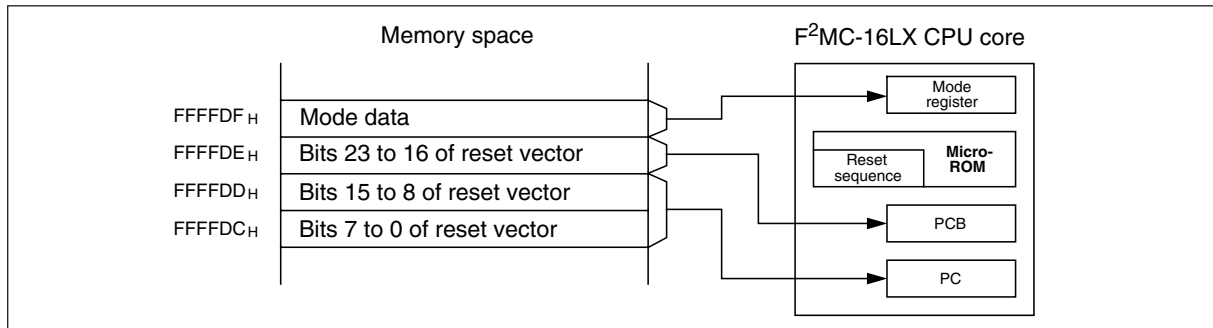
Setting the mode pins (MD0 to MD2) specifies how to fetch the reset vector and the mode data. Fetching the reset vector and the mode data is performed in the reset sequence. See Section 9.1.1 “Mode Pins”, for details on mode pins.

■ Mode fetch

When the reset is cleared, the CPU transfers the reset vector and the mode data to the appropriate registers in the CPU core by hardware. The reset vector and mode data are allocated to the four bytes from "FFFFDC_H" to "FFFFDF_H". The CPU outputs these addresses to the bus immediately after the reset is cleared and then fetches the reset vector and mode data. Using mode fetching, the CPU can begin processing at the address indicated by the reset vector.

Figure 7.4-2 “Transfer of reset vector and mode data” shows the transfer of the reset vector and mode data.

Figure 7.4-2 Transfer of reset vector and mode data



● Mode data (address: FFFFDF_H)

Only a reset operation changes the contents of the mode register. The mode register setting is valid after a reset operation. See Section 9.1.2 “Mode Data”, for details on mode data.

● Reset vector (address: FFFFD_C_H to FFFFDE_H)

The reset vector points to the start address after the reset sequence. The CPU starts to execute the first instruction stored in the start address.

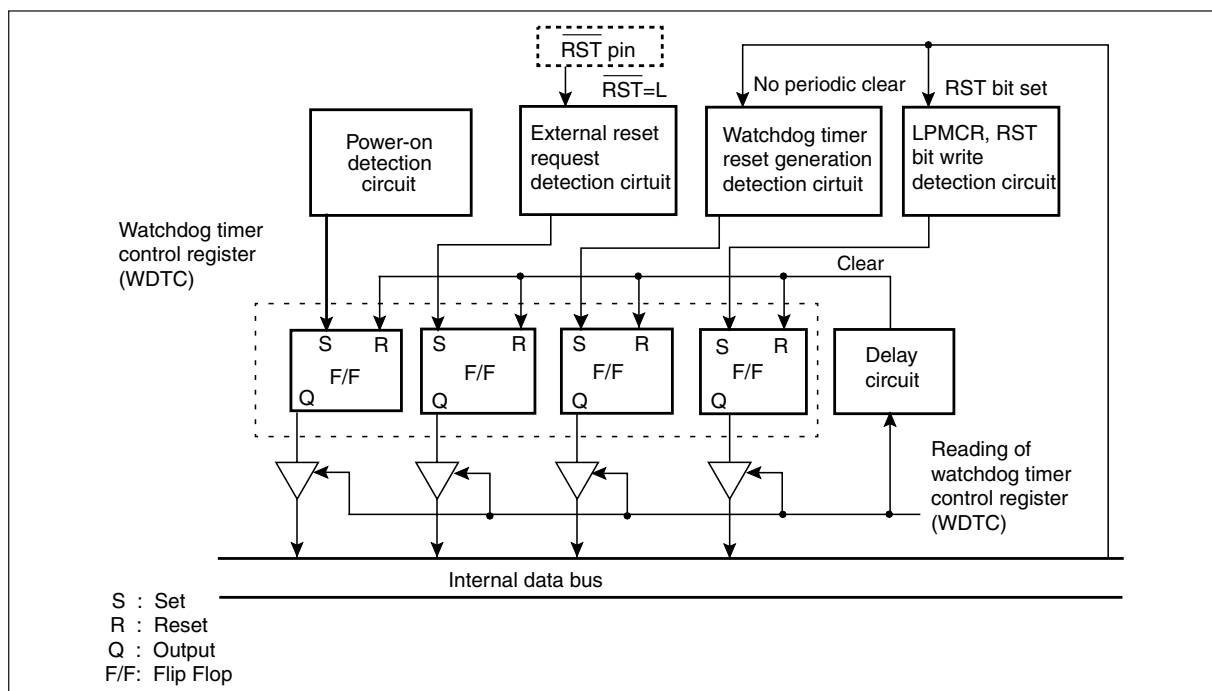
7.5 Reset Cause Bits

A reset cause can be identified by reading the watchdog timer control register (WDTC).

Reset cause bits

As shown in Figure 7.5-1 “Block diagram of reset cause bits”, a flip-flop is associated with each reset cause. The contents of the flip-flops are obtained by reading the watchdog timer control register (WDTC). If the cause of a reset must be identified after the reset has been cleared, the value read from the WDTC should be processed by the software and a branch made to the appropriate program.

Figure 7.5-1 Block diagram of reset cause bits



■ Correspondence between reset cause bits and reset causes

Figure 7.5-2 “Configuration of reset cause bits (watchdog timer control register)” shows the configuration of the reset cause bits of the watchdog timer control register (WDTC). Table 7.5-1 “Correspondence between reset cause bits and reset causes” maps the correspondence between the reset cause bits and reset causes. See “Watchdog timer control register (WDTC)” in Section 12.1 “Outline of Watch-Dog Timer”, for details.

Figure 7.5-2 Configuration of reset cause bits (watchdog timer control register)

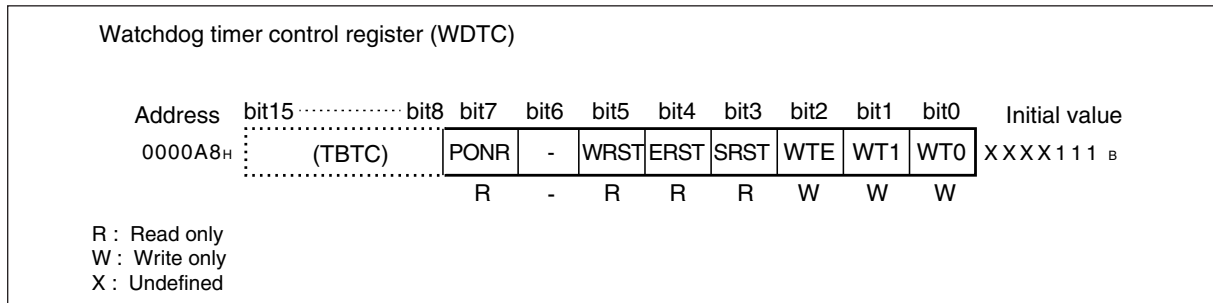


Table 7.5-1 Correspondence between reset cause bits and reset causes

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

*: Previous state defined

X: Undefined

■ Notes about reset cause bits

● Multiple reset causes generated at the same time

When multiple reset causes are generated at the same time, the corresponding reset cause bits of the watchdog timer control register (WDTC) are also set to "1". If, for example, an external reset request via the $\overline{\text{RST}}$ pin and the watchdog timer overflow occur at the same time, the ERST and the WRST bits are both set to "1".

● Power-on reset

For a power-on reset, because the PONR bit is set to "1" but all other reset cause bits are undefined, the software should be programmed so that it will ignore all reset cause bits except the PONR bit if it is "1".

- Clearing the reset cause bits

The reset cause bits are cleared only when the watchdog timer control register (WDTC) is read. Any bit corresponding to a reset cause that has already been generated is not cleared even though another reset is generated (a setting of "1" is retained).

Note:

If the power is turned on under conditions where no power-on reset occurs, the value in WDTC register may not be guaranteed.

7.6 Status of Pins in a Reset

This section describes the status of pins when a reset occurs.

■ Status of pins during a reset

The status of pins during a reset depends on the settings of mode pins (MD2 to MD0).

- When internal vector mode has been set: (MD2 to MD0 = "011_B")

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

About status of each pins during reset, please see “8.7 Status of Pins in Standby Mode and during Hold and Reset”.

■ Status of pins after mode data is read

The status of pins after mode data has been read depends on the mode data (M1 and M0 = "00").

- When single-chip mode has been selected (M1, and M0 = "00_B")

All I/O pins (resource pins) are high impedance.

Note:

For those pins that change to high impedance when a reset cause is generated, confirm that devices connected to the pins do not malfunction.

CHAPTER 8

LOW-POWER CONSUMPTION MODE

This chapter explains the low-power consumption mode of MB90340 series microcontrollers.

- 8.1 “Overview of Low-Power Consumption Mode”
- 8.2 “Block Diagram of the Low-Power Consumption Control Circuit”
- 8.3 “Low-Power Consumption Mode Control Register (LPMCR)”
- 8.4 “CPU Intermittent Operation Mode”
- 8.5 “Standby Mode”
- 8.6 “Status Change Diagram”
- 8.7 “Status of Pins in Standby Mode and during Hold and Reset”
- 8.8 “Usage Notes on Low-Power Consumption Mode”

8.1 Overview of Low-Power Consumption Mode

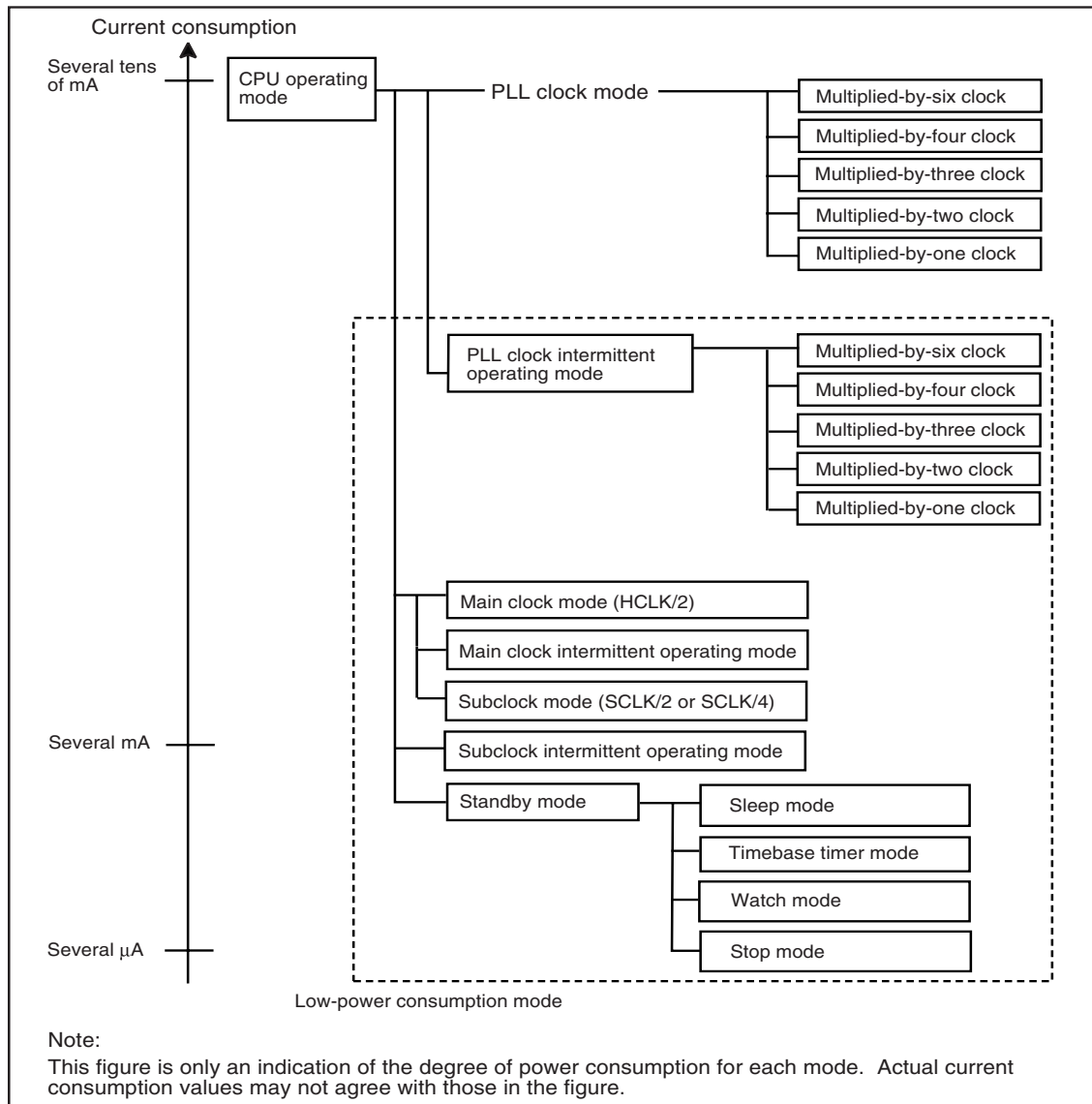
The MB90340 series has the following CPU operating modes, any of which can be used depending on operating clock selection and clock operation control:

- Clock mode (PLL clock mode, main clock mode, or sub-clock mode)
- CPU intermittent operating mode (PLL clock intermittent operating mode, main clock intermittent operating mode, or sub-clock intermittent operating mode)
- Standby mode (sleep mode, timebase timer mode, stop mode, or watch mode)

■ CPU Operating Modes and Current Consumption

Figure 8.1-1 “CPU Operating Mode and Current Consumption” shows the relationship between the CPU operating modes and current consumption.

Figure 8.1-1 CPU Operating Mode and Current Consumption



■ Clock Mode

● PLL clock mode

In this mode, a PLL clock that is a multiple of the oscillation clock (HCLK) is used to operate the CPU and peripheral functions.

● Main clock mode

In this mode, the main clock, with the oscillation clock (HCLK) frequency divided by 2 is used to operate the CPU and peripheral functions. In the main clock mode, the PLL multiplier circuit is inactive.

● Sub-clock mode

In this mode, the sub-clock, with the oscillation clock (SCLK) frequency divided by 2 or 4 is used to operate the CPU and peripheral functions. In the sub-clock mode, the main clock and PLL multiplier circuit are inactive.

The subclock oscillation stabilization wait time of $2 \sim 14 / \text{SCLK}$ (Approx. 2s @32KHz, 1/4 division) takes place when power-on reset and reactivation from stop mode. If a transition from main clock mode from subclock mode is performed during this oscillation stabilization wait time, actual transition may be delayed.

Reference:

For the clock mode, see Section 5.4 “Clock Mode”.

■ CPU Intermittent Operating Mode

In this mode, the CPU is operated intermittently while high-speed clock pluses are supplied to peripheral functions, thereby reducing power consumption. In this mode, intermittent clock pulses are supplied only to the CPU while it is accessing a register, internal memory, peripheral function, or external unit.

■ Standby Mode

In this mode, the low-power consumption control circuit stops supplying the clock to the CPU (sleep mode) or the CPU and peripheral functions (timebase timer mode) or stops the oscillation clock itself (stop mode), thereby reducing power consumption.

● PLL sleep mode

The PLL sleep mode is activated to stop the CPU operating clock in the PLL clock mode. Components excluding the CPU operate on the PLL clock.

● Main sleep mode

The main sleep mode is activated to stop the CPU operating clock in the main clock mode. Components excluding the CPU operate on the main clock.

● Sub-sleep mode

The subsleep mode activated to stop the CPU operating clock in the subclock mode. Components excluding the CPU operate on the subclock oscillation frequency divided four or two.

- Timebase timer mode

The timebase timer mode causes the operation of functions, excluding the oscillation clock, timebase timer, and watch timer, to stop. All functions other than the timebase timer and watch timer are inactivated.

- Watch mode

The watch mode operates the watch timer only. The subclock operates and the main clock and PLL multiplier circuit stop.

- Stop mode

The stop mode cause the oscillation to stop. All functions are inactivated.

Note:

Because the stop mode turns the oscillation clock off, data can be retained by the lowest power consumption.

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

8.2 Block Diagram of the Low-Power Consumption Control Circuit

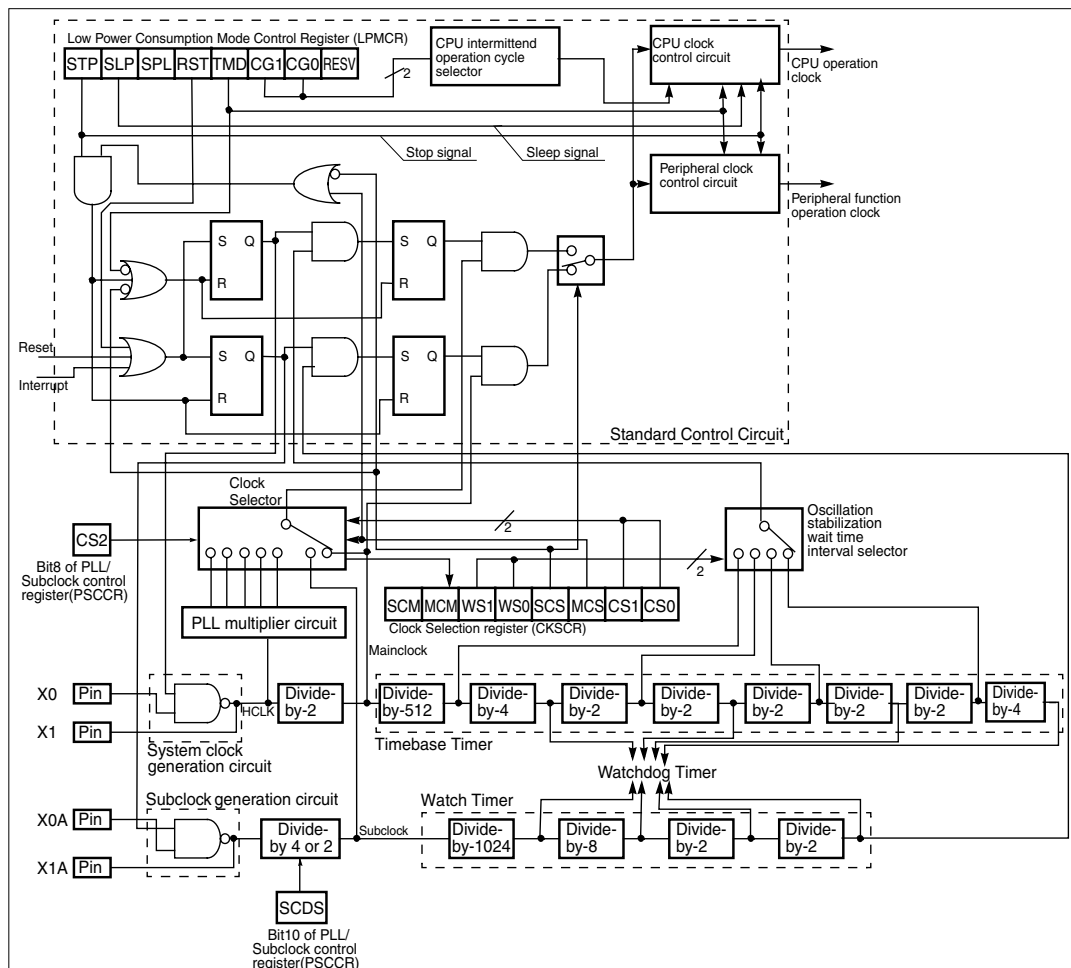
The low-power consumption control circuit consists of the following seven blocks:

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

■ Block Diagram of the Low-Power Consumption Control Circuit

Figure 8.2-1 “Block Diagram of the Low-Power Consumption Control Circuit” shows a block diagram of the low-power consumption control circuit.

Figure 8.2-1 Block Diagram of the Low-Power Consumption Control Circuit



- CPU intermittent operation selector

This selector selects the number of clock pulses to halt the CPU during the CPU intermittent operation mode.

- Standby control circuit

The standby control circuit controls the CPU clock control and the peripheral clock control circuits and turns the low-power consumption mode on and off.

- CPU clock control circuit

This circuit controls clocks supplied to the CPU. This circuit controls clocks supplied to peripheral functions for the peripheral clock control circuit.

- Peripheral clock control circuit

This circuit controls clocks supplied to peripheral functions.

- Pin high-impedance control circuit

This circuit makes external pins high-impedance in the timebase timer mode and stop mode. For pins with the pull-up option, this circuit disconnects the pull-up resistor in the stop mode.

- Internal reset generation circuit

This circuit generates an internal reset signal.

- Low-power consumption mode control register (LPMCR)

This register is used to switch to and release the standby mode and to set the CPU intermittent operation function.

8.3 Low-Power Consumption Mode Control Register (LPMCR)

This register switches to or releases the low-power consumption mode. This register also sets the number of CPU clock pulses to halt during the CPU intermittent operation mode.

■ Low-Power Consumption Mode Control Register (LPMCR)

Figure 8.3-1 “Configuration of the low-power consumption mode control register (LPMCR)” shows the configuration of the low-power consumption mode control register (LPMCR).

Figure 8.3-1 Configuration of the low-power consumption mode control register (LPMCR)

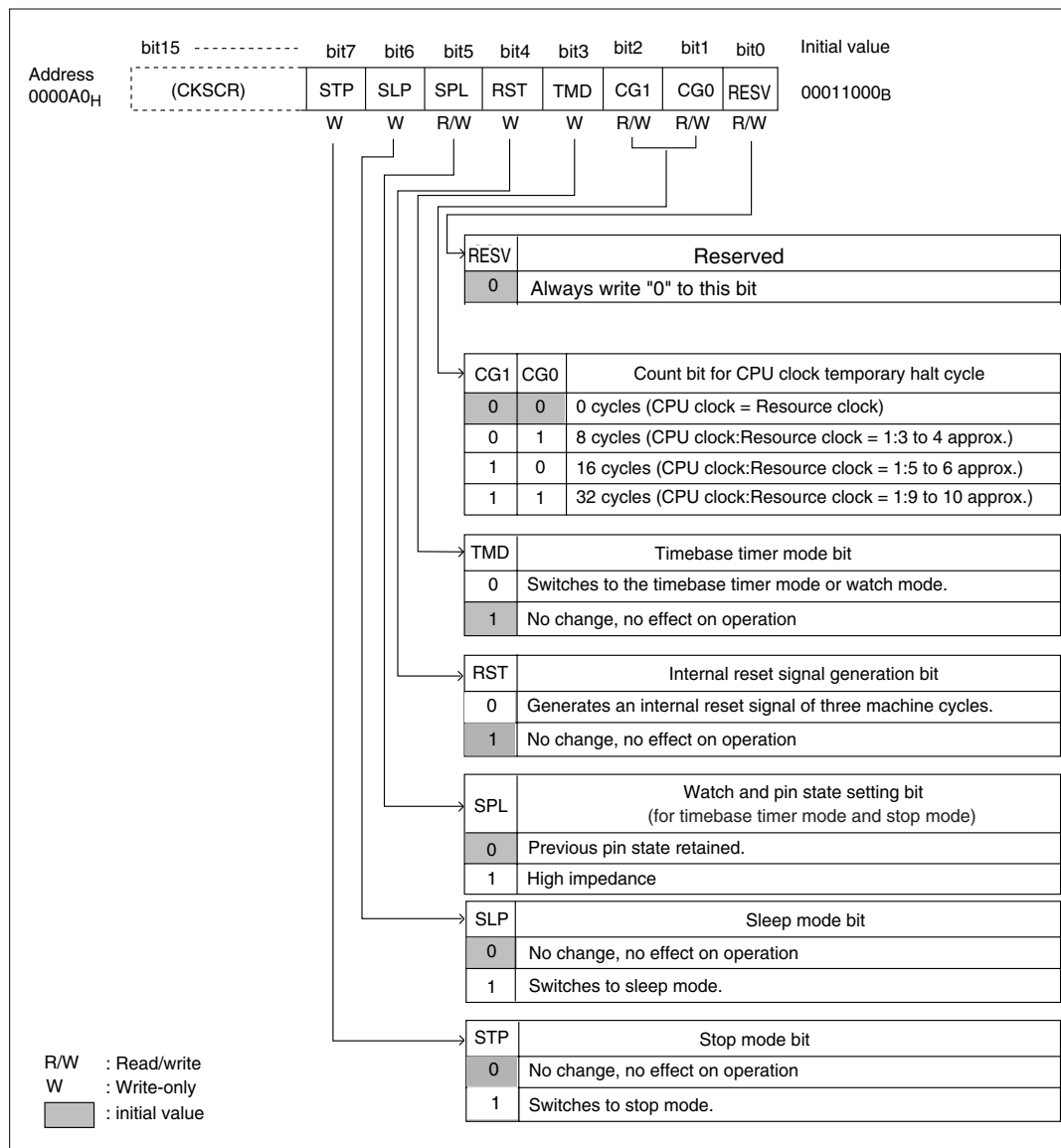


Table 8.3-1 Function Description of Each Bit of the Low-power Consumption Mode Control Register (LPMCR)

Bit name		Function
bit 7	STP: Stop mode bit	• This bit indicates switching to the stop mode. • When 1 is written to this bit, a switch to the stop mode is performed. • Writing 0 in this bit has no effect on operation. • This bit is cleared to 0 by a reset or when an interrupt request occurs. • The read value of this bit is always 0.
bit 6	SLP: Sleep mode bit	• This bit indicates switching to a sleep mode. • When 1 is written to this bit, a switch to a sleep mode is performed. • Writing 0 in this bit has no effect on operation. • This bit is cleared to 0 by a reset or when an interrupt request occurs. • The read value of this bit is always 0.
bit 5	SPL: Pin state setting bit (for watch mode, timebase timer mode, and stop mode)	• This bit is enabled only in the watch mode, timebase timer mode, and stop mode. • When this bit is 0, the level of the external pins is retained. • When this bit is 1, the status of the external pins changes to high-impedance. • This bit is initialized to 0 by a reset.
bit 4	RST: Internal reset signal generation bit	• When 0 is written to this bit, an internal reset signal of three machine cycles is generated. • Writing 1 in this bit has no effect on operation. • The read value of this bit is always 1.
bit 3	TMD: Watch/timebase timer mode bit	• This bit indicates switching to the watch mode or timebase timer mode. • When 0 is written to this bit in the main clock mode or PLL clock mode, a switch to timebase timer mode is performed. • When 0 is written to this bit in the subclock mode, a switch to the watch mode is performed. • This bit is cleared to 1 by a reset or when an interrupt request occurs. • The read value of this bit is always 1.
bit 2 bit 1	CG1, CG0: Bits for selecting clock count for CPU temporary halt cycle	• These bits set the number of CPU clock pulses per cycle to halt the CPU for the CPU intermittent operation function. • The clock supplied to the CPU is stopped for the specified number of pulses after the execution of each instruction. • Four types of clock counts are selectable. • These bits are initialized to 00_B by a reset.
bit 0	RESV: Reserved	Note: Always write "0" to this bit.

Notes:

- Switching to a low-power consumption mode (including the stop mode, sleep mode, timebase timer mode, and watch mode) is performed by writing the low-power consumption mode control register. Only the instructions listed in Table 8.3-2 "Instructions to Be Used for Switching to a Low-Power Consumption Mode" should be used for this purpose. If other instructions are used for switching to a low-power consumption mode, operation cannot be assured.
- The low-power consumption mode transition instruction in Table 3.8-2 "Instructions used for change to low-power consumption mode" must always be followed by an array of instructions highlighted by a line below.

```

MOV LPMCR, #H' xx ; The low-power consumption mode transition instruction in Table 3.8-2
┌ NOP
├ NOP
├ JMP $+3 ; jump to next instruction
└ MOV A, #H' 10 ; any instruction

```

The device does not guarantee its operation after returning from the low-power consumption mode if you place an array of instructions other than the one enclosed in the line.

- To access the low-power consumption mode control register (LPMCR) with C language, refer to "Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode" in the section 3.8.8 "Precautions when Using Low-Power Consumption Mode".
- When word-length is used for writing the low-power consumption mode control register, even addresses must be used. Using odd addresses to switch to a low-power consumption mode may result in a malfunction.
- To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode, watch mode or timebase timer mode, disable the output of peripheral functions, and set the STP bit to 1 or set the TMD bit to 0.

Table 8.3-2 Instructions at Transition to Low-power Consumption Mode

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

8.4 CPU Intermittent Operation Mode

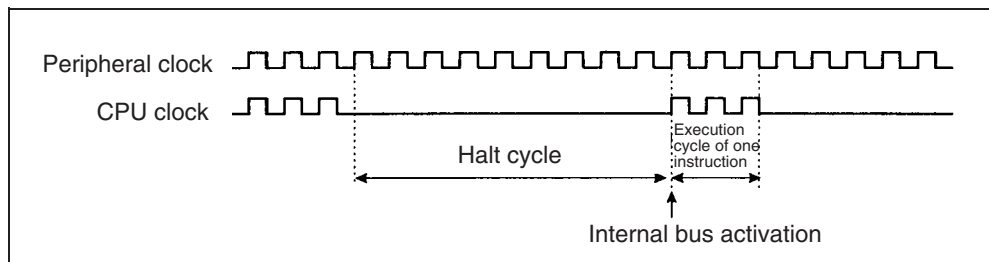
This mode is used for intermittent operation of the CPU while external buses and peripheral functions continue to operate at high speeds. The purpose of this mode is to reduce power consumption.

■ CPU Intermittent Operation Mode

This mode halts the supply of the clock pulse to the CPU for a certain period. The halt occurs after the execution of every instruction that accesses a register, internal memory (ROM and RAM), I/O, peripheral functions, or the external bus. Internal bus cycle activation is therefore delayed. While high-speed peripheral clock pulses are supplied to peripheral functions, the execution speed of the CPU is reduced, thereby enabling low-power consumption processing.

- The low-power consumption mode control register (LPMCR: CG1 and CG0) is used to select the number of clock pulses per halt cycle of the clock supplied to the CPU.
- External bus operation uses the same clock as that used for peripheral functions.
- Instruction execution time in the CPU intermittent mode can be calculated. A correction value should be obtained by multiplying the execution count of instructions that access a register, internal memory, internal peripheral functions, or the external bus by the number of clock pulses per halt cycle. Add this corrective value to the normal execution time. Figure 8.4-1 “Clock Pulses during the CPU Intermittent Operation” shows the operating clock pulses during the CPU intermittent operation mode.

Figure 8.4-1 Clock Pulses during the CPU Intermittent Operation



8.5 Standby Mode

The standby mode includes the sleep (PLL sleep, main sleep, subsleep), watch, and stop modes.

■ Operation Status during Standby Mode

Table 8.5-1 “Operation Statuses during Standby Mode” shows operation statuses during standby mode.

Table 8.5-1 Operation Statuses during Standby Mode

Standby mode		Condition for switch	Main clock	Sub-clock	Machine clock	CPU	Peripheral	Pin	Release event			
Sleep mode	PLL sleep mode	SCS=1 MCS=0 SLP=1	Active		Active		Active	Active	External reset Interrupt			
	Main sleep mode	SCS=1 MCS=1 SLP=1										
	Subsleep mode	SCS=0 SLP=1 MCS=X	Inactive									
Timebase timer mode	Timebase timer mode (SPL=0)	SCS=1 MCS=X TMD=0	Active	Active		Inactive	Inactive ^{*1}	Retained	External reset Interrupt ^{*3}			
	Timebase timer mode (SPL=1)	SCS=1 MCS=X TMD=0					Hi-z ^{*6}					
Watch mode	Watch mode (SPL=0)	TMD=0 MCS=X SCS=0	Inactive				Inactive			Inactive ^{*2}	Retained	External reset Interrupt ^{*4}
	Watch mode (SPL=1)	TMD=0 MCS=X SCS=0								Hi-z ^{*6}		
Stop mode	Stop mode (SPL=0)	STP=1	Inactive	Inactive						Inactive	Retained	External reset Interrupt ^{*5}
	Stop mode (SPL=1)	STP=1									Hi-z ^{*6}	

*1: The timebase timer and watch timer operate.

*2: The watch timer operates.

*3: Watch timer, timebase timer and external interrupt

*4: Watch timer and external interrupt

*5: External interrupt

*6: Input pins for DTP/External interrupt are enabled.

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

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

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

TMD: Watch and timebase timer mode bit of low-power consumption mode control register (LPMCR)

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

SCS: Machine clock selection bit (sub) of clock selection register (CKSCR)

Hi-z: High-impedance

Note:

For those external pins shared between port functions and peripheral functions, disable output for the peripheral functions then set the STP bit to "1" or reset the TMD bit to "0" to set these pins in high impedance state.

8.5.1 Sleep Mode

This mode causes the CPU operating clock to stop while other components continue to operate. When the low-power consumption mode control register (LPMCR) indicates a switch to a sleep mode, a switch to the PLL sleep mode occurs if the PLL clock mode has been set. A switch to the main clock mode occurs if the main clock mode has been set. A switch to the subsleep mode occurs if the subclock mode has been set.

■ Switching to Sleep Mode

Writing 1 in the SLP bit and the TMD bit and 0 in the STP bit of the low-power consumption mode control register (LPMCR) triggers a switch to a sleep mode. At this time, if the MSC bit is 0 and the SCS bit is 1 in the clock selection register (CKSCR), a switch to the PLL sleep mode is triggered. If the MSC bit and the SCS bit are 1, a switch to the main sleep mode is triggered. If the SCS bit is 0, a switch to the subsleep mode is triggered.

Note:

When 1 is written to the SLP and STP bits at the same time, the STP bit setting overrides the SLP bit setting and the mode switches to the stop mode. When 1 is written to the SLP bit and 0 is written to the TMD bit at the same time, the TMD bit setting overrides the SLP bit setting and the mode switches to the timebase timer mode or watch mode.

● Data retention function

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

● Hold function

During a sleep mode, the external bus hold function is active. This function sets the hold status when requested to do so.

● Operation during an interrupt request

Writing 1 in the SLP bit of the low-power consumption mode control register during an interrupt request does not trigger a switch to a sleep mode. If the CPU does not accept the interrupt, the CPU executes the next instruction. If the CPU accepts the interrupt, CPU operation immediately branches to the interrupt processing routine.

● Status of pins

During a sleep mode, all pins (excluding those used for bus I/O or bus control) retain their previous status.

■ Release of Sleep Mode

The low-power consumption control circuit releases sleep modes when a reset is input or an interrupt occurs.

● Return by a reset

A sleep mode is initialized to the main clock mode by a reset.

Note:

When the sub-sleep mode is switched to main clock mode using an external reset pin ($\overline{\text{RST}}$ pin), input level “L” for at least the oscillation time of the oscillator ^{*1} + 100 μs + 16 machine cycles.

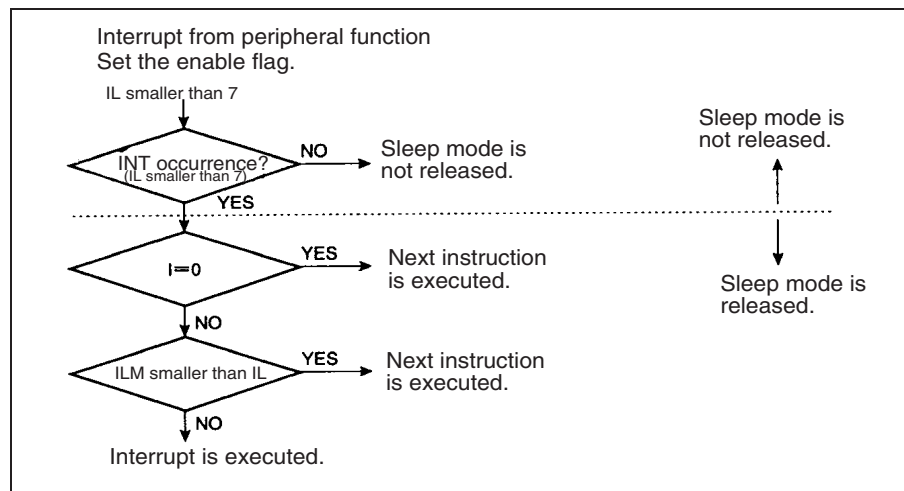
*1: The oscillation time of the oscillator is the time required to reach 90% of amplitude. It takes several to dozens of ms for crystal oscillators, hundreds of μs to several ms for FAR/ceramic oscillators, and 0 ms for external clocks.

● Return by an interrupt

If an interrupt request of higher than level seven is issued from a peripheral circuit during a sleep mode, the sleep mode is released. After the mode is released, the interrupt is handled as an ordinary interrupt. If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the sleep mode.

Figure 8.5-1 “Release of Sleep Mode by Interrupt Occurrence” shows the release of a sleep mode when an interrupt occurs.

Figure 8.5-1 Release of Sleep Mode by Interrupt Occurrence

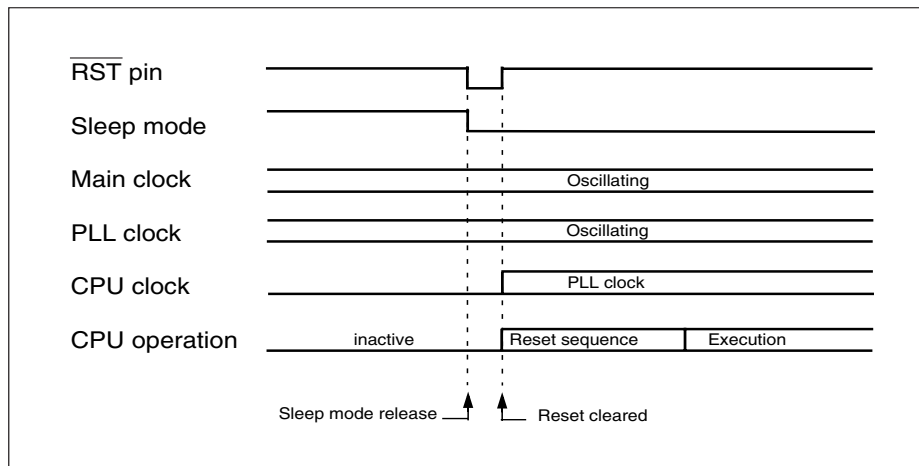


Note:

When interrupt processing is executed, the CPU normally executes the instruction that follows the instruction in which switching to a sleep mode has been specified. The CPU then proceeds to interrupt processing. If the switching to sleep mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

Figure 8.5-2 “Release of the Sleep Mode (External Reset)” shows a return from the Sleep mode.

Figure 8.5-2 Release of the Sleep Mode (External Reset)



8.5.2 Timebase Timer Mode

This mode causes all functions, excluding oscillation, the timebase timer, and the watch timer, to stop. In this mode, only the timebase timer and watch timer operate.

■ Switching to the Timebase Timer Mode

When 0 is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the PLL clock mode or main clock mode (CKSCR: SCS = 1), switching to the timebase timer mode occurs.

● Data retention function

In the timebase timer mode, the contents of dedicated registers, such as accumulators, and the internal RAM are retained.

● Hold function

During the timebase timer mode, because the external bus hold function is inactive, a hold request is not accepted even if it is input. If a hold request is input during switching to the timebase timer mode, the bus may remain at high-impedance and prevent the $\overline{\text{HAK}}$ signal from going low.

● Operation during an interrupt request

Writing 0 in the TMD bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the timebase timer mode.

● Status of pins

Whether the external pins in the timebase timer mode retain the state they had immediately before switching to the timebase timer mode or go to the high-impedance state can be controlled by the low-power consumption mode control register (LPMCR: SPL).

■ Release of Timebase Timer Mode

The low-power consumption control circuit releases the timebase timer mode when a reset is input or an interrupt occurs.

● Return by a reset

The timebase timer mode is initialized to the main clock mode by a reset.

● Return by an interrupt

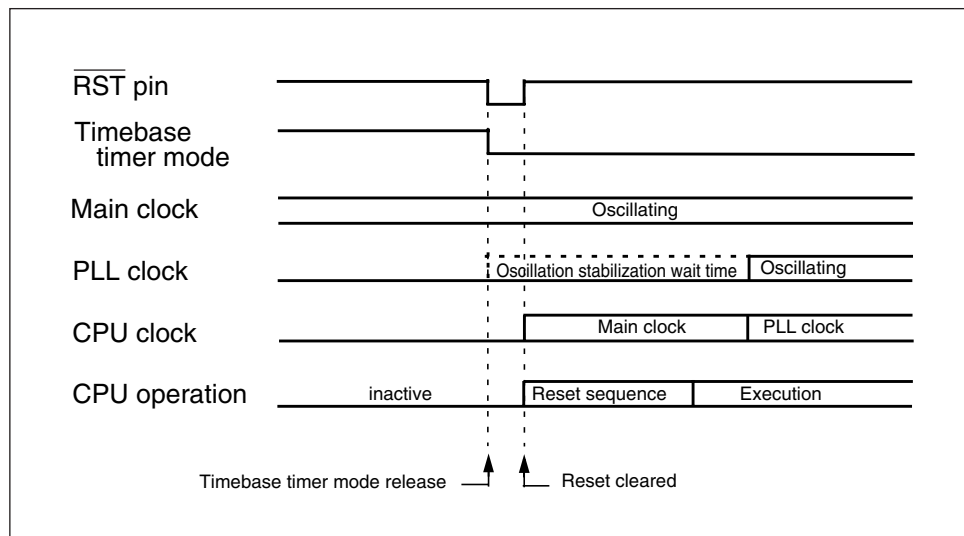
If a watch timer, timebase timer, or external interrupt generates an interrupt request at an interrupt level higher than 7 during the timebase timer mode (IL2, IL1, and IL0 of the interrupt control register (ICR) do not indicate 111_B), the low-power consumption mode control circuit releases the timebase timer mode. After the mode is released, the interrupt is handled as an ordinary interrupt. If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), or interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the timebase timer mode.

Note:

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the timebase timer mode has been specified. The CPU then proceeds to interrupt processing. If the switching to the timebase timer mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

Figure 8.5-3 “Release of the timebase timer mode (External Reset)” shows a return from the timebase timer mode.

Figure 8.5-3 Release of the timebase timer mode (External Reset)



8.5.3 Watch Mode

This mode causes all functions, excluding the subclock and watch timer, to stop. In this mode almost all chip functions stop.

■ Switching to the Watch Mode

When 0 is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the subclock run mode (CKSCR: SCS = 0), switching to the watch mode occurs.

● Data retention function

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

● Hold function

During the watch mode, because the external bus hold function is inactive, a hold request is not accepted even if it is input. If a hold request is input during switching to the watch mode, the bus may remain at high-impedance and prevent the $\overline{\text{HAK}}$ signal from going low.

● Operation during an interrupt request

Writing 1 in the TMD bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the watch mode.

● Status of pins

Whether the external pins in the watch mode retain the state they had immediately before switching to the watch mode or go to the high-impedance state can be controlled by the SPL bit of the low-power consumption mode control register (LPMCR).

■ Release of Watch Mode

The low-power consumption control circuit releases the watch mode when a reset is input or an interrupt occurs.

● Return by a reset

The watch mode is released by a reset to the oscillation stabilization wait reset state. The reset sequence is executed after the oscillation stabilization wait time.

The period of reset assertion depends on quality of main clock crystal oscillator.

● Return by an interrupt

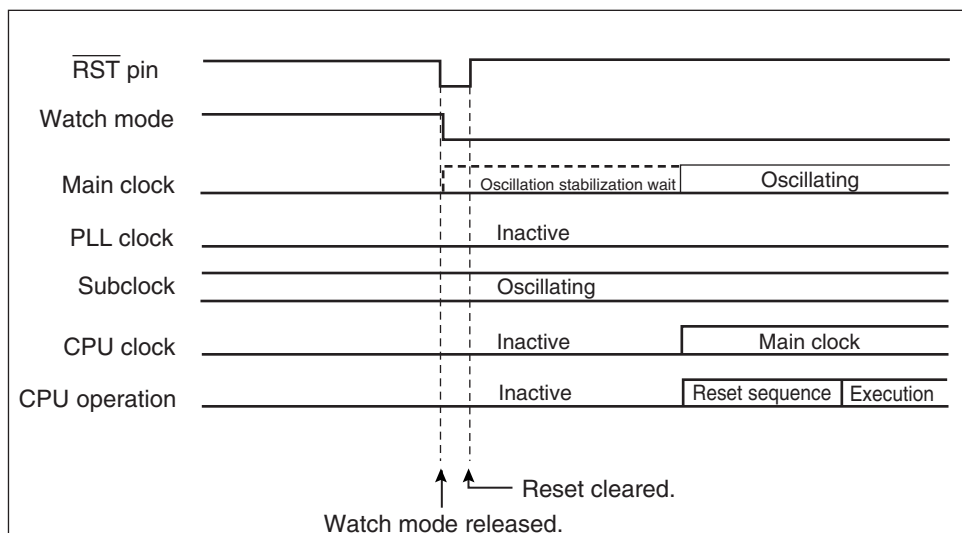
If a watch timer or external interrupt generates an interrupt request at an interrupt level higher than 7 during the watch mode (IL2, IL1, and IL0 of the interrupt control register (ICR) do not indicate 111_B), the low-power consumption mode control circuit releases the watch mode and causes switching to the subclock mode immediately. After the switching, the interrupt is handled as an ordinary interrupt. If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the watch mode.

Note:

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the watch mode has been specified. The CPU then proceeds to interrupt processing. If the switching to the watch mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

Figure 8.5-4 “Release of the Watch Mode (External Reset)” shows a return from the watch mode.

Figure 8.5-4 Release of the Watch Mode (External Reset)



8.5.4 Stop Mode

Because this mode causes oscillation to stop and inactivates all functions, data can be retained by the lowest power consumption.

■ Switching to the Stop Mode

When 1 is written to the STP bit of the low-power consumption mode control register (LPMCR), switching to the stop mode occurs.

● Data retention function

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

● Hold function

During the stop mode, because the external bus hold function is inactive, a hold request is not accepted even if it is input. If a hold request is input during switching to the stop mode, the bus may remain at high-impedance and prevent the $\overline{\text{HAK}}$ signal from going low.

● Operation during an interrupt request

Writing 1 in the STP bit of the low-power consumption mode control register (LPMCR) does not trigger a switch to the stop mode.

● Status of pins

Whether the external pins in the stop mode retain the state they had immediately before switching to the stop mode or go to the high-impedance state can be controlled by the SPL bit of the low-power consumption mode control register (LPMCR).

Note:

If "1" is specified for both the STP bit and the SLP bit of the low-power consumption mode control (LPMCR) simultaneously, the STP bit takes precedence, and a transition to stop mode occurs.

For those external pins shared between port functions and peripheral functions, disable output for the peripheral functions then set the STP bit to "1" or reset the TMD bit to "0" to set these pins in high impedance state.

■ Release of Stop Mode

The low-power consumption control circuit releases the stop mode when a reset is input or an interrupt occurs. Because the oscillation clock (HCLK) and sub-clock (SCLK) are halted, the stop mode is released after the oscillation stabilization wait interval of the main clock or sub-clock.

● Return by a reset

After the stop mode is released by a reset, the oscillation stabilization wait state is set. The reset sequence is executed after the oscillation stabilization wait time.

The period of reset assertion depends on quality of main clock crystal oscillator.

● Return by an interrupt

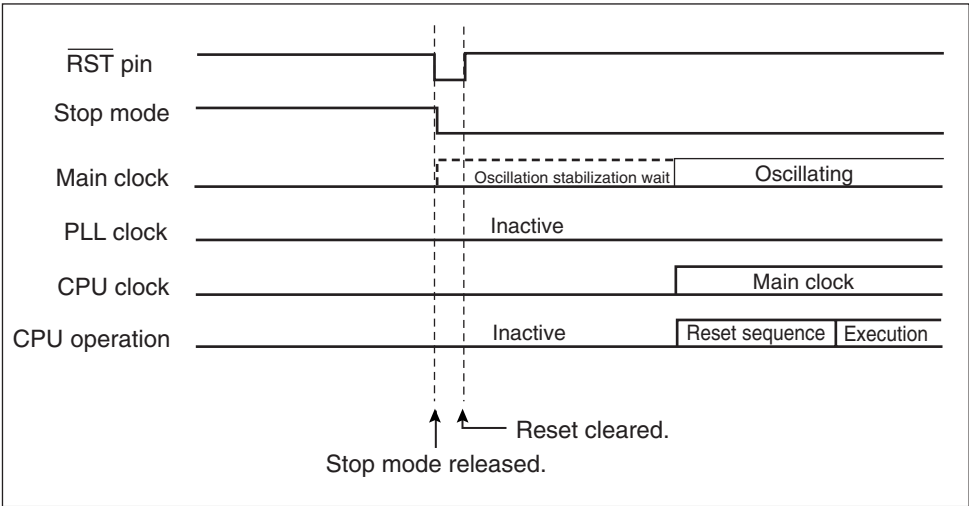
If an external interrupt generates an interrupt request at interrupt level higher than 7 during the stop mode (IL2, IL1, and IL0 of the interrupt control register (ICR) do not indicate 111_B), the low-power consumption mode control circuit releases the stop mode. The interrupt is then handled as an ordinary interrupt after the oscillation stabilization wait time of the main clock specified by the WS1 and WS0 bits of the clock selection register (CKSCR). If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the stop mode.

Notes:

- When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the stop mode has been specified. The CPU then proceeds to interrupt processing. If the switching to the stop mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.
- In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires $2^{14}/\text{HCLK}$ or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10_B" or "11_B".

Figure 8.5-5 "Release of the Stop Mode (External Reset)" shows a return from the stop mode.

Figure 8.5-5 Release of the Stop Mode (External Reset)

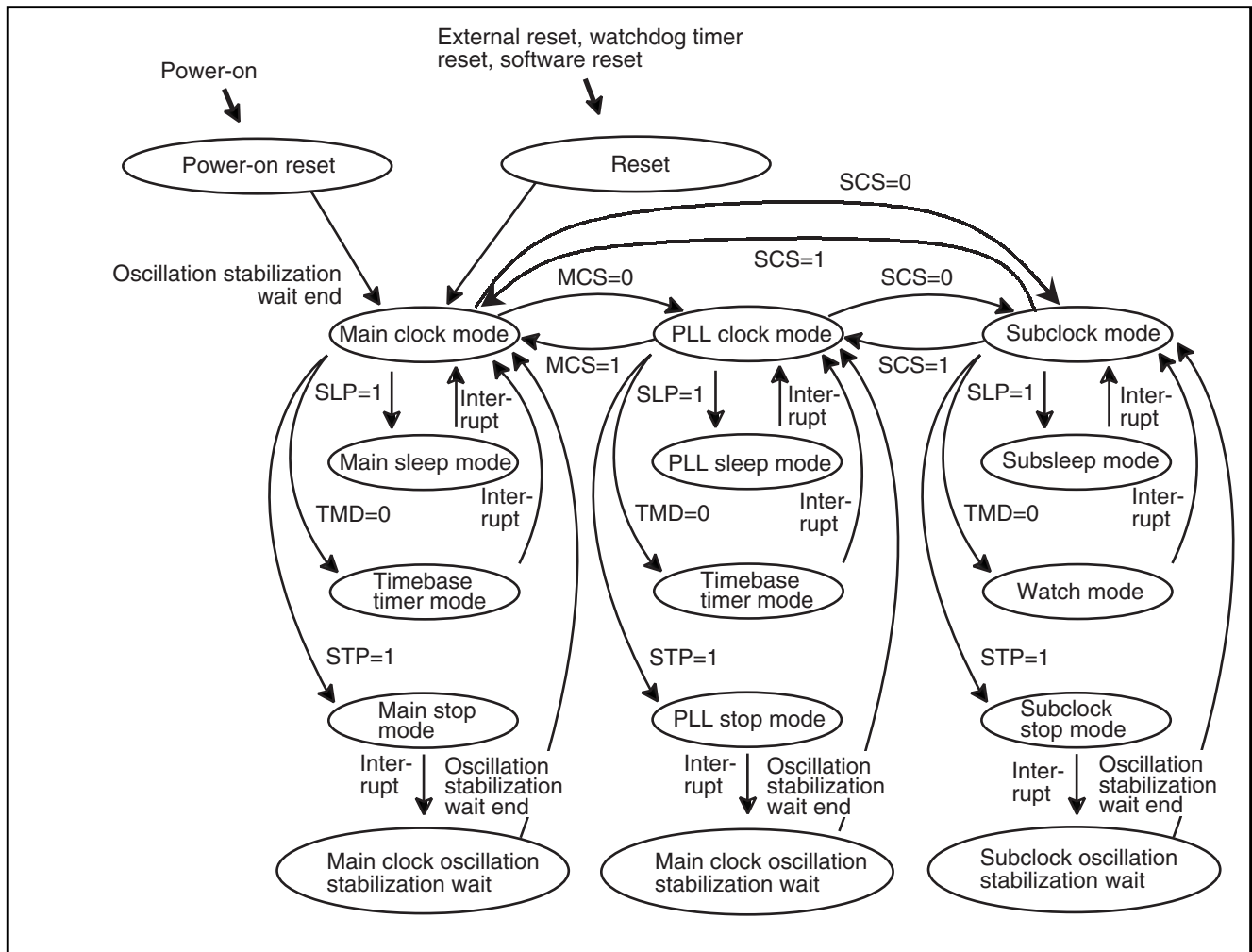


8.6 Status Change Diagram

Figure 5.6-1 shows the status change diagram of the MB90340 series.

■ Status Change Diagram

Figure 8.6-1 Status Change Diagram



Note:

During reading this diagram, refer to the priority of the standby modes (see “Priorities of the STP, SLP, and TMD Bits” in Section 8.3 “Low-Power Consumption Mode Control Register (LPMCR)”).

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

■ Operation Status in Each Operating Mode Table

Table 8.6-1 “Operation Status in Each Operating Mode” lists the operation status in each operating mode.

Table 8.6-1 Operation Status in Each Operating Mode

Operation status	Main clock	Subclock	PLL clock	CPU	Peripheral	Watch	Timebase timer	Clock source
PLL	Active	Active	Active	Active	Active	Active	Active	PLL clock
PLL sleep				Inactive				
Timebase timer *1					Inactive			
PLL stop	Inactive	Inactive	Inactive			Inactive	Inactive	
PLL oscillation stabilization wait	Active	Active	Active	Active	Active	Active		
Main	Active	Active	Inactive	Active	Active	Active	Active	Main clock
Main sleep				Inactive				
Timebase timer *2					Inactive			
Main stop	Inactive	Inactive				Inactive	Inactive	
Main oscillation stabilization wait	Active	Active		Active	Active			
Sub	Inactive	Active	Inactive	Active	Active	Active	Inactive	Sub-clock
Subsleep				Inactive				
Watch		Inactive			Inactive			
Subclock stop						Inactive		
Subclock oscillation stabilization wait				Active	Active			
Power-on reset	Active	Active	Inactive	Inactive	Inactive	Active	Active	Main clock
Reset			Active					

*1: During the PLL clock mode

*2: During the main clock mode

8.7 Status of Pins in Standby Mode and during Hold and Reset

The status of pins in the standby mode and during hold and reset are described for each memory access mode.

■ Status of Each Pin in the Single Chip Mode

Table 8.7-1 "Status of Each Pin in the Single Chip Mode" shows the Status of Each Pin in the Single Chip Mode.

Table 8.7-1 Status of Each Pin in the Single Chip Mode

Pin name	Standby/timebase timer/watch mode		Reset		
	Sleep	Stop ^{*6}			
		SPL = 0		SPL = 1	
P10 to P17 P20 to P27 P30 to P37 P40 to P47 P50 to P57 P60 to P67 P80 to P87 P90 to P97 PA0 to PA1	Status before the mode retained ^(*2)	Input disabled ^{*4} / status before the mode retained ^{*2}	Input disabled ^{*4} / output Hi-Z ^{*5}	Input not possible ^{*3} / output Hi-Z ^{*5}	
P00 to P07 ^{*7}		Input enabled ^{*1} / status before the mode retained ^{*2}	Input enabled ^{*1} / output Hi-Z ^{*5}		
P70 to P77					
PA0, P42, P32, P12, P80 to P82, P84 ^{*8}					

*1: "Input enabled" means that the input function is available. The pull-up or pull-down option or external input is required. When the pin is used as an output port, the status is the same as other ports.

*2: "Status before the mode retained" means that it keep previous output state when output or that the input is "not possible" when input.

*3: "Input not possible" means that the gate connected to the input pin is functioning as input but the internal circuit can not accept this input signal because the internal circuit is not functioning.

*4: "Input disabled" means that the gate connected to the input pin is disabled.

*5: "Output Hi-Z" means that the pin driving transistor is placed in the drive prohibited state to set the pin to high impedance.

*6: The pull-up option is disconnected from each port pin in stop mode.

*7: These pins are "Input enabled" in STOP mode only if INTxR bit of External interrupt source select register (EISSR) is "0". Otherwise they are "Input disabled".

*8: These pins are "Input enabled" in STOP mode only if INTxR bit of External interrupt source select register (EISSR) is "1". Otherwise they are "Input disabled".

Note:

In the stop mode up to sixteen external signals can be input and the frequency of each signal must be 1 KHz or less.

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

Table 8.7-2 “Status of Each Pin in the External Bus 16-bit Data Bus Mode” shows the Status of Each Pin in the External Bus 16-bit Data Bus Mode.

Table 8.7-2 Status of Each Pin in the External Bus 16-bit Data Bus Mode

Pin name	Standby/timbase timer/watch mode			Hold	Reset	At internal ROM access immediately after reset release	At internal ROM access after external ROM access		
	Sleep	Stop							
		SPL = 0	SPL = 1						
P00 to P07 (AD00 to AD07) P10 to P17 (AD08 to AD15)	Input not possible / output Hi-Z	Input disabled / output Hi-Z	Input disabled / output Hi-Z*5	Input not possible / output Hi-Z	Input not possible / output Hi-Z	Output Hi-Z / input enabled	Output Hi-Z / input enabled		
P20 to P27 (A16 to A23)	Output state *1,*3	Output state *1,*3		Input not possible / output Hi-Z*3		Output state*1	Hold the address immediately before		
P37(CLK)	Input not possible / output enabled*2,*3	Input not possible / output state *1,*3		Input not possible / output enabled*2,*3		Output Hi-Z / input enabled	Output Hi-Z / input enabled		
P36(RDY)	Status before the mode retained*4	Input disabled / status before the mode retained*4		Input not possible / output Hi-Z*3					
P35(HAK)				“L” output*3					
P34(HRQ)				“1” input*3					
P33(WRH)	“H” output*3	“H” output*3		Input not possible / output Hi-Z*3	"H" output			“H” output	“H” output
P32(WRL)				Input not possible / output Hi-Z					
P31(RD)									
P30(ALE)	"L" output	"L" output							
P40 to P47 P50 to P57 P60 to P67 P80 to P87 P90 to P97 PA0 to PA1	Status before the mode retained*4	Input disabled / status before the mode retained*4	Input enabled / output Hi-Z*5	Status before the mode retained*4	Input not possible / output Hi-Z	Output Hi-Z / input enabled	Output Hi-Z / input enabled		
P70 to P77		Input enabled / status before the mode retained*4							
PA0, P42, P32, P12, P80 to P82, P84 *6									

- *1: “Output state” means that the output is being driven by the internal circuit but the internal circuit is on hold. Thus the output value is fixed to either “0” or “1”. When the internal circuit is in operation, the output value changes accordingly except during reset.
- *2: “Output enabled” means that the output is being driven and propagate the signal from the internal circuit in operation.
- *3: When used as output, it retains the previous output value.
- *4: “Status before the mode retained” means that it keeps the previous output state when output or that input is “not possible” when input. “Input not possible” means that the gate connected to the input pin is functioning as input but the internal circuit can not accept this input signal because the internal circuit is not functioning.
- *5: “Input disabled” means that the gate connected to the input pin is disabled.
- *6: These pins are “Input enabled” in STOP mode if the INTxR bit of External interrupt source select register (EISSR) is “1”. Otherwise they are disabled.

Note:

For those external pins shared between port functions and peripheral functions, disable output for the peripheral functions then set the STP bit to "1" or reset the TMD bit to "0" to set these pins in high impedance state.

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

Table 8.7-3 “Status of Each Pin in the External Bus 8-bit Data Bus Mode” shows the Status of Each Pin in the External Bus 8-bit Data Bus Mode.

Table 8.7-3 Status of Each Pin in the External Bus 8-bit Data Bus Mode

Pin name	Standby mode			Hold	Reset	At internal ROM access immediately after reset release	At internal ROM access after external ROM access
	Sleep	Stop					
		SPL = 0	SPL = 1				
P00 to P07 (AD00 to AD07) P10 to P17 (AD08 to AD15)	Input not possible / output Hi-Z	Input disabled/ output Hi-Z	Input disabled / output Hi-Z *5	Input not possible / output Hi-Z	Input not possible / output Hi-Z	Output Hi-Z / input enabled	Output Hi-Z / input enabled
P20 to P27 (A16 to A23)	Output state *1,*3	Output state *1,*3		Input not possible / output Hi-Z *3		Output state *1	Hold the address immediately before
P37(CLK)	Input not possible / output enabled*2,*3	Input not possible / output state *1,*3		Input not possible / output enabled *2,*3		Output Hi-Z / input enabled	Output Hi-Z / input enabled
P36(RDY)	Status before the mode retained*4	Input disabled/ status before the mode retained*4		Input not possible / output Hi-Z *3			
P35(HAK)				“L” output *3			
P34(HRQ)				“1” input *3			
P33				Input not possible /status before the mode retained*4 output Hi-Z*3			
P32(WR)	“H” output*3	“H” output*3		Input not possible /status before the mode retained*4 output Hi-Z*3			
P31(RD)	"H" output	"H" output		Input not possible / output Hi-Z	"H" output	“H” output	“H” output
P30(ALE)	"L" output	"L" output		Input not possible / output Hi-Z	"L" output	“L” output	“L” output
P40 to P47 P50 to P57 P60 to P67 P80 to P87 P90 to P97	Status before the mode retained*4	Input disabled/ status before the mode retained*4	Input enabled / output Hi-Z *5	Status before the mode retained*4	Input disabled/ output Hi-Z	Output Hi-Z / input enabled	Output Hi-Z / input enabled
P51 to P54 P90, P91, P95, PA0		Input enabled/ status before the mode retained					
P70 to P77							
PA0, P42, P32, P12, P80 to P82, P84*6							

- *1: “Output state” means that the output is being driven by the internal circuit but the internal circuit is on hold. Thus the output value is fixed to either “0” or “1”. When the internal circuit is in operation, the output value changes accordingly except during reset.
- *2: “Output enabled” means that the output is being driven and propagate the signal from the internal circuit in operation.
- *3: When used as output, it retains the previous output value.
- *4: “Status before the mode retained” means that it keeps the previous output state when output or that input is “not possible” when input. “Input not possible” means that the gate connected to the input pin is functioning as input but the internal circuit can not accept this input signal because the internal circuit is not functioning.
- *5: “Input disabled” means that the gate connected to the input pin is disabled.
- *6: These pins are “Input enabled” in STOP mode if the INTxR bit of External interrupt source select register (EISSR) is “1”. Otherwise they are disabled.

Note:

For those external pins shared between port functions and peripheral functions, disable output for the peripheral functions then set the STP bit to "1" or reset the TMD bit to "0" to set these pins in high impedance state.

8.8 Usage Notes on Low-Power Consumption Mode

Note the following five items when using the low-power consumption mode:

- **Switching to a standby mode and interrupt**
 - **Release of a standby mode by an interrupt**
 - **Release of the stop mode**
 - **Oscillation stabilization wait time Oscillation**
 - **Clock mode switching**
-

■ Switching to a Standby Mode and Interrupt

When there is a pending interrupt request to the CPU, writing "1" to the STP bit or SLP bit or writing "1" to the TMD bit in the LPMCR register does not take its effect. In other words, the desired mode transition is not performed even after the interrupt service is completed. If the interrupt is being serviced already and the interrupt request is cleared and if there are no other pending requests, the desired mode transition can be performed. Note that this behavior is not dependent on whether the setting in the PS register allows the CPU to accept the interrupt request.

● Release of the Standby Mode by an Interrupt

If an interrupt request of interrupt level higher than seven priority is issued from a peripheral function during the sleep, timebase timer, or stop mode, the standby mode is released, which does not depend on whether the CPU accepts the interrupt.

After the release of the standby mode by an interrupt, normal processing is performed. The CPU branches to the interrupt handling routine provided that the priority of the interrupt request indicated by the interrupt level setting bits (IL2, IL1, and IL0 of ICR) is higher than the interrupt level mask register (ILM) and the interrupt enable flag ((I) of the condition code register (CCR) is set to 1 (enabled).

If the interrupt is not accepted, the CPU starts the execution with the instruction following the instruction in which switching to the standby mode has been specified.

When interrupt is accepted, the CPU first executes the instruction following the instruction in which switching to the standby mode has been specified. The CPU then proceeds to interrupt processing.

Depending on the condition when switching to a standby mode was performed, however, the CPU may proceed to interrupt processing before executing the next instruction.

Note:

If the CPU should not branch to the interrupt processing routine immediately after a return, action such as interrupt disabling must be taken before a standby mode is set.

● Release of the Stop Mode

The stop mode can be released by an input that has been set as an external interrupt input cause before the system enters the stop mode. As an input cause, an H-level signal, L-level signal, rising edge, or falling edge can be selected.

■ Oscillation Stabilization Wait Time

● Clock oscillation stabilization wait time

Because the oscillator is halted in the stop mode, an oscillation stabilization wait time is required. A time period selected by the WS1 and WS0 bits of the clock selection register (CKSCR) is used as the oscillation stabilization wait time.

● PLL clock oscillation stabilization wait time

In main clock mode, the PLL multiplication circuit stops. When changing to PLL clock mode, it is necessary to reserve the PLL clock oscillation stabilization wait time. The CPU runs in main clock mode till the PLL clock oscillation stabilization wait time has elapsed. When the main clock mode is switched to PLL clock mode, the PLL clock oscillation stabilization wait time is fixed at $2^{14}/\text{HCLK}$ (HCLK: oscillation clock).

In sub-clock mode, the main clock and PLL multiplication circuit stop. When changing to PLL clock mode, it is necessary to reserve the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of the main clock and PLL clock oscillation stabilization wait times. The PLL clock oscillation stabilization wait time, however, requires $2^{14}/\text{HCLK}$ or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10_B" or "11_B".

In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires $2^{14}/\text{HCLK}$ or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10_B" or "11_B".

● Clock mode switching

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

● Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode

To access the low-power consumption mode control register (LPMCR) with assembler language

To set the low power consumption mode control register (LPMCR) to enter the standby mode, use the instruction listed in Table 3.8-2.

The standby mode transition instruction in Table 3.8-2 must always be followed by an array of instructions highlighted by a line below.

MOV LPMCR, #H' xx; The low-power consumption mode transition instruction in Table 3.8-2

```

NOP
NOP
JMP $+3          ; Jump to the next instruction
MOV A, #H' 10    ; Arbitrary instruction

```

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

To access the low-power consumption mode (LPMCR) with C language

To enter the standby mode using the low power consumption mode control register (LPMCR), use one of the following methods 1. to 3. to access the register:

1. Specify the standby mode transition instruction as a function and insert two `__wait_nop()` built-in functions after that instruction. If any interrupt other than the interrupt to return from the standby mode can occur within the function, optimize the function during compilation to suppress the LINK and UNLINK instructions from occurring.

Example: Watch mode or timebase timer mode transition function

```

void enter_watch(){
    IO_LPMCR.byte = 0x10; /* Set LPMCR TMD bit to 0 */
    __wait_nop();
    __wait_nop();
}

```

2. Define the standby mode transition instruction using `__asm` statements and insert two NOP and JMP instructions after that instruction.

Example: Transition to sleep mode

```

__asm("MOV I:_IO_LPMCR, #H' 58); /* Set LPMCR SLP bit to 1 */
__asm("NOP");
__asm("OP");
__asm("JMP $+3"); /* Jump to the next instruction*/

```

3. Define the standby mode transition instruction between `#pragma asm` and `#pragma endasm` and insert two NOP and JMP instructions after that instruction.

Example: Transition to stop mode

```

#pragma asm
    MOV I:_IO_LPMCR, #H' 98 /* Set LPMCR SLP bit to 1 */
    NOP
    NOP
    JMP $+3 /* Jump to the next instruction */
#pragma endasm

```


CHAPTER 9

MEMORY ACCESS MODES

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

9.1 “Outline of Memory Access Modes”

9.2 “External Memory Access (Bus Pin Control Circuit)”

9.3 “External Memory Access Control Signal Operation”

9.1 Outline of Memory Access Modes

In the F²MC-16LX, various modes are provided for access methods and access areas.

■ Memory Access Modes

Table 9.1-1 Mode Pins and Modes

Operation mode	Bus mode	Access mode
RUN	Single chip	-
	Internal ROM, external bus	8 bits
		16 bits
	External ROM, external bus	8 bits
		16 bits
Flash programming	-	-

● 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 M_x bit in mode data. By selecting an operation mode, normal operation activation or flash serial programming 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.

● Access mode

Access mode means the mode for controlling the external data bus width. The access mode is specified by the MD_x mode setting pin and the S₀ bit in mode data. By selecting an access mode, an 8- or 16-bit external data bus is specified.

9.1.1 Mode Pins

Table 9.1-2 “Mode Pins and Modes” lists the operations that can be specified by combining the three external pins MD2 to MD0.

■ Mode pins

Table 9.1-2 Mode Pins and Modes

Mode pin setting			Mode name	Reset vector access area	External data bus width	Remarks
MD2	MD1	MD0				
0	0	0	External vector mode 0	External	8 bits	
0	0	1	External vector mode 1	External	16 bits	Reset vector, 16-bit bus width access
0	1	0	Reserved			
0	1	1	Internal vector mode	Internal	(Mode data)	Reset sequence and later segments are controlled based on mode data.
1	0	0	Reserved			
1	0	1				
1	1	0	Flash memory serial programming *	--	--	--
1	1	1	Flash memory	--	--	Mode when parallel writer is used

*: Flash serial programming mode involves other pin settings than the mode pins. For details, see CHAPTER 26
EXAMPLES OF MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S)
SERIAL PROGRAMMING CONNECTION.

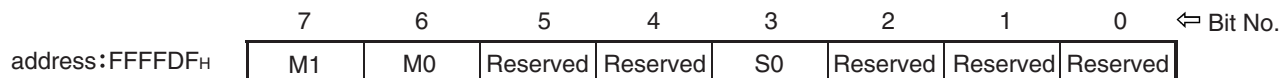
9.1.2 Mode Data

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 9.1-1 Mode Data Configuration

Mode data address



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

The M1 and M0 bits are used to specify the operation mode after the reset sequence is completed. Table 9.1-3 “M1 and M0 (Bus Mode Setting Bit) Functions” shows the relationship between the M1 and M0 bits and the functions.

Table 9.1-3 M1 and M0 (Bus Mode Setting Bit) Functions

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

[Bit 3] S0 (mode setting bit)

The S0 bit is used to specify the bus mode or access mode after the reset sequence is completed. Table 9.1-4 “S0 (Mode Setting Bit) Functions” shows the relationship between the S0 bit and the functions.

Table 9.1-4 S0 (Mode Setting Bit) Functions

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

9.1.3 Memory Space in Each Bus Mode

Figure 9.1-2 “Relationship between Access Areas and Physical Addresses for Each Bus Mode” shows the correspondence between the access areas and physical addresses for each bus mode.

■ Memory Space in Each Bus Mode

Figure 9.1-2 Relationship between Access Areas and Physical Addresses for Each Bus Mode

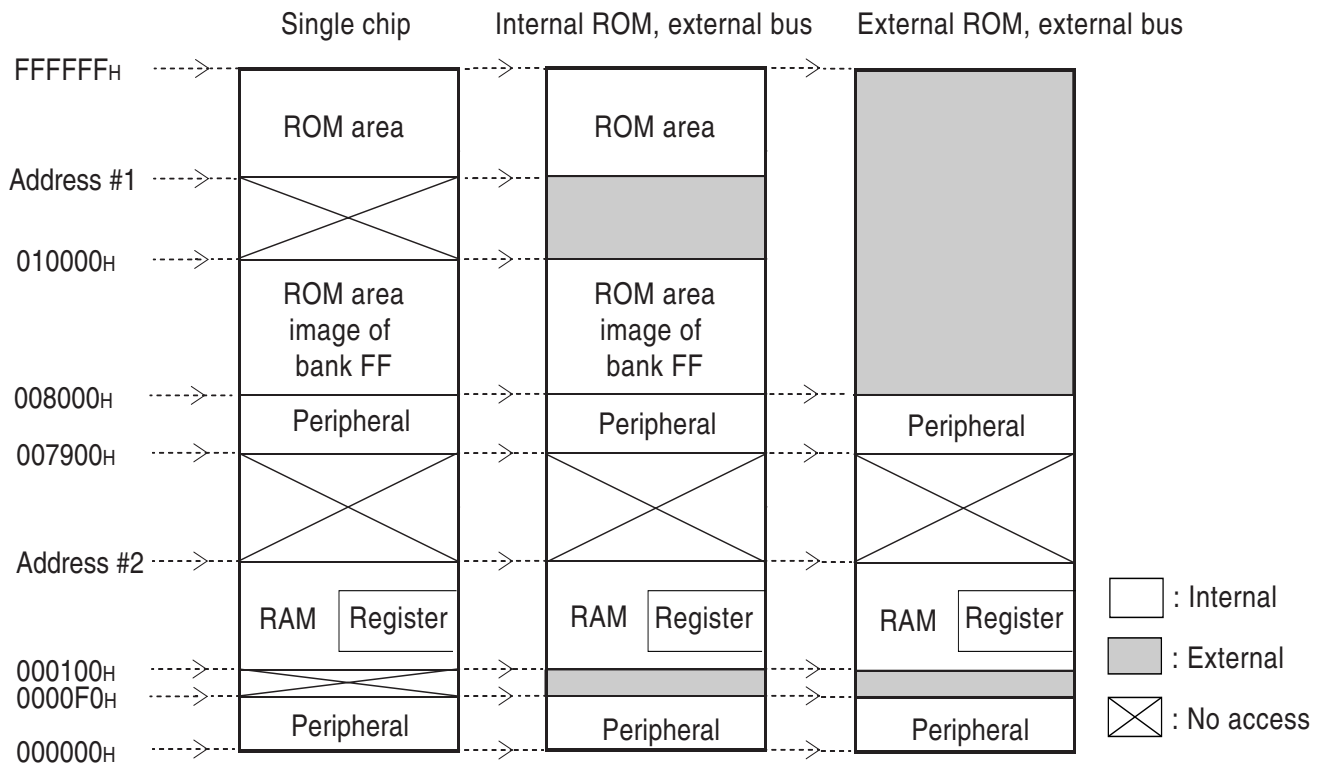


Table 9.1-5 Relationship between addresses #1 and #2 by product type

TYPE	Address #1	Address #2
MB90341/C/S/CS	FE0000 _H *	004000 _H
MB90(F)342/C/S/CS	FC0000 _H *	004000 _H
MB90F343/C/S/CS	F90000 _H	005100 _H
MB90F345/C/S/CS	F80000 _H	005100 _H
MB90(F)346A/CA/AS/CAS	FF0000 _H	000900 _H
MB90(F)347A/CA/AS/CAS	FE0000 _H	001900 _H
MB90348/C/S/CS	FE0000 _H	004000 _H
MB90(F)349/C/S/CS	FC0000 _H	004000 _H
MB90V340/S	F80000 _H	007900 _H

*: On MB90F343/C/S/CS, Address FC0000_H to FCFFFF_H and address F80000_H to F8FFFF_H is reserved area, and not used as ROM space.

■ Recommended Setting

Table 9.1-6 “Example of Recommended Settings for Mode Pins and Mode Data” lists an example of recommended settings for mode pins and mode data.

Table 9.1-6 Example of Recommended Settings for Mode Pins and Mode Data

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

External pins have signal functions that depend on each mode.

Table 9.1-7 External Pin Functions for Each Mode

Pin name	Function			
	Single chip	External bus expansion		Flash programming
		8 bits	16 bits	
P07 to P00	Port	AD07 to AD00		DQ07 to DQ00
P17 to P10		A15 to AD08	AD15 to AD08	AQ15 to AQ08
P27 to P20		A23 to A16*		AQ07 to AQ00
P30		ALE		AQ16
P31		$\overline{RD}$		$\overline{CE}$
P32		$\overline{WR}$	$\overline{WRL}$	$\overline{OE}$
P33		Port	$\overline{WRH}$ *	$\overline{WE}$
P34		HRQ*		AQ17
P35		$\overline{HAK}$ *		AQ18
P36		RDY*		$\overline{BYTE}$
P37		CLK*		$\overline{RY/BY}$

*: The upper address output pins and the $\overline{WRL/WR}$, $\overline{WRH}$, HRQ, $\overline{HAK}$, RDY, and CLK pins can be used as ports through function selection. See Section 9.2 “External Memory Access (Bus Pin Control Circuit)” for details.

9.2 External Memory Access (Bus Pin Control Circuit)

The external bus pin control circuit controls the external bus pins for external expansion of the CPU address and data buses.

External Memory Access (Bus Pin Control Circuit)

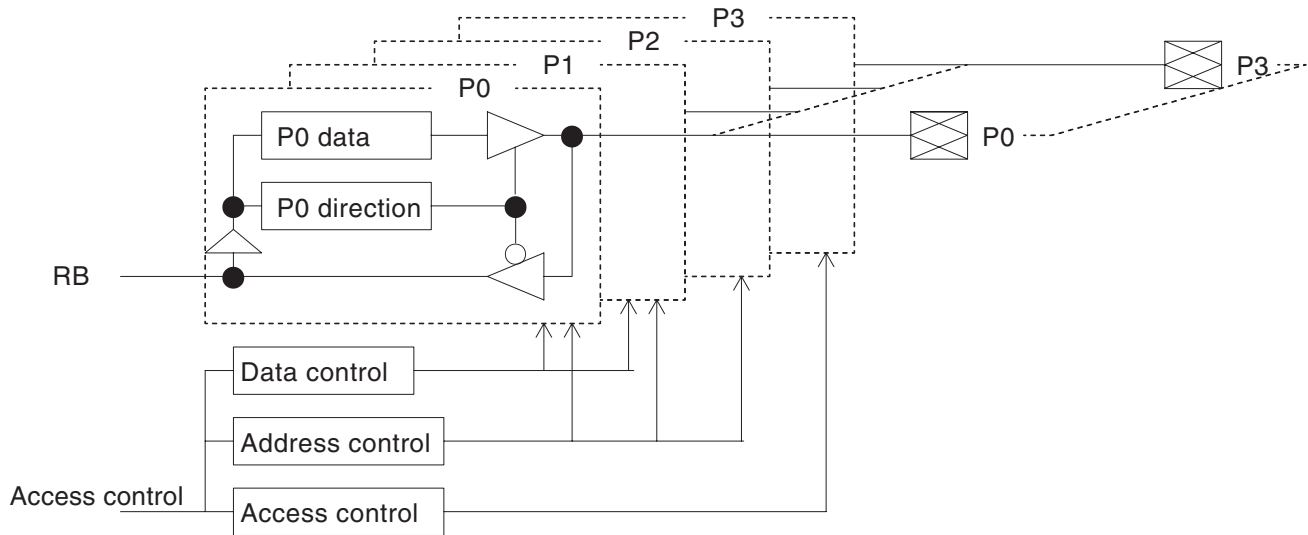
The following address, data, and control signals are used to access external memory and peripherals of the MB90340 device:

- CLK (P37): Machine cycle clock (KBP) output pin
- RDY (P36): External ready input pin
- $\overline{\text{WRH}}$ (P33): Write signal for upper 8 bits of data bus, Active only when 16 bit access
- $\overline{\text{WRL}}/\overline{\text{WR}}$ (P32): Write signal for lower 8 bits of data bus, Active both when 16 bit and 8 bit access
- $\overline{\text{RD}}$ (P31): Read signal
- ALE (P30): Address latch enable signal

The external bus pin control circuit is used to control the external bus pins to enable external expansion of the CPU address and data buses.

Block Diagram of External Memory Access

Figure 9.2-1 External Bus Controller



9.2.1 External Memory Access (External Bus Pin Control Circuit) Registers

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

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

■ External Memory Access Registers

Figure 9.2-2 External memory access (external bus pin control circuit) registers

Automatic ready function selection register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:0000A5H	IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write ⇒	(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
Initial value ⇒	(0)	(0)	(1)	(1)	(X)	(X)	(0)	(0)	

External address output control register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:0000A6H	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Bus control signal selection register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:0000A7H	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(X)	

9.2.2 Automatic Ready Function Selection Register (ARSR)

The automatic ready function selection register (ARSR) is used to set the automatic wait time for memory access for each area during external access.

■ Automatic Ready Function Selection Register (ARSR)

Figure 9.2-3 Automatic Ready Function Selection Register Configuration

Automatic ready function selection register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address:0000A5H	IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write	⇒ (W)	⇒ (W)	⇒ (W)	⇒ (W)	⇒ (-)	⇒ (-)	⇒ (W)	⇒ (W)	
Initial value	⇒ (0)	⇒ (0)	⇒ (1)	⇒ (1)	⇒ (X)	⇒ (X)	⇒ (0)	⇒ (0)	

[Bits 15 and 14] IOR1, IOR0

The IOR1 and IOR0 bits are used to specify the automatic wait function for external access to the area from 0000F0_H to 0000FF_H. Table 9.2-1 “IOR1 and IOR0 (Automatic Wait Function Specification Bit) Functions” lists the settings that can be specified by combining the IOR1 and IOR0 bits.

Table 9.2-1 IOR1 and IOR0 (Automatic Wait Function Specification Bit) Functions

IOR1	IOR0	Function
0	0	Automatic wait disabled [initial value]
0	1	Automatic wait of 1 cycle is inserted for external access
1	0	Automatic wait of 2 cycles is inserted for external access
1	1	Automatic wait of 3 cycles is inserted for external access

[Bits 13 and 12] HMR1, HMR0

The HMR1 and HMR0 bits are used to specify the automatic wait function for external access to the area from 800000_H to FFFFFFF_H. Table 9.2-2 “HMR1 and HMR0 (Automatic Wait Function Specification Bit) Functions” lists the settings that can be specified by combining the HMR1 and HMR0 bits.

Table 9.2-2 HMR1 and HMR0 (Automatic Wait Function Specification Bit) Functions

HMR1	HMR0	Function
0	0	Automatic wait disabled
0	1	Automatic wait of 1 cycle is inserted for external access
1	0	Automatic wait of 2 cycles is inserted for external access
1	1	Automatic wait of 3 cycles is inserted for external access [initial value]

[Bits 9 and 8] LMR1, LMR0

The LMR1 and LMR0 bits are used to specify the automatic wait function for external access to the areas between 008000_H and 7FFFFFF_H. Table 9.2-3 “LMR1 and LMR0 (Automatic Wait Function Specification Bit) Functions” lists the settings that can be specified by combining the LMR1 and LMR0 bits.

Table 9.2-3 LMR1 and LMR0 (Automatic Wait Function Specification Bit) Functions

LMR1	LMR0	Function
0	0	Automatic wait disabled [initial value]
0	1	Automatic wait of 1 cycle is inserted for external access
1	0	Automatic wait of 2 cycles is inserted for external access
1	1	Automatic wait of 3 cycles is inserted for external access

9.2.3 External Address Output Control Register (HACR)

The external address output control register (HACR) controls the external output of addresses (A23 to A16). The bits correspond to addresses A23 to A16, which control address output pins, as shown in Figure 9.2-4 “External Address Output Control Register Configuration”.

■ External Address Output Control Register (HACR)

Figure 9.2-4 External Address Output Control Register Configuration

External address output control register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address:0000A6H	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

The HACR register controls output of addresses (A23 to A16) to the external circuit. The address output pin is controlled as follows with the eight bits that correspond to address bits A23 to A16.

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

All bits of this register are write-only bits, and the value read from the bits is 1.

Table 9.2-4 External Address Output Control Register (E23 to E16 bits) Functions

0	The corresponding pin is used for address output (AXX) [initial value].
1	The corresponding pin is used as an I/O port (PXX).

9.2.4 Bus Control Signal Selection Register (ECSR)

The bus control signal selection register sets the bus operation control function in external bus mode. This register cannot be accessed when the device is in single-chip mode, since all pins function as I/O ports whatever the value of this register. All bits of the bus control signal selection register are write-only bits, and the value from the bits is 1.

■ Bus Control Signal Selection Register (ECSR)

Figure 9.2-5 Bus Control Signal Selection Register Configuration

Bus control signal selection register

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

[Bit 15] CKE

The CKE bit controls output of the external clock signal pin (CLK), as shown in Table 9.2-5 “CKE (External Clock (CLK) Output Control Bit) Functions”.

Table 9.2-5 CKE (External Clock (CLK) Output Control Bit) Functions

0	I/O port (P37) operation (clock output disabled) [initial value]
1	Clock signal (CLK) output enabled

[Bit 14] RYE

The RYE bit controls input of the external ready (RDY) signal pin, as shown in Table 9.2-6 “RYE (External Ready (RDY) Input Control Bit) Functions”.

Table 9.2-6 RYE (External Ready (RDY) Input Control Bit) Functions

0	I/O port (P36) operation (external RDY input disabled) [initial value]
1	External ready (RDY) input enabled

[Bit 13] HDE

The HDE bit specifies that input-output of hold signals is enabled. The hold request input signal (HRQ) and hold acknowledge output signal ($\overline{\text{HAK}}$) are controlled according to the setting of the HDE bit, as shown in Table 9.2-7 “HDE (Hold Signal Input-Output Enable Specification Bit) Functions”.

Table 9.2-7 HDE (Hold Signal Input-Output Enable Specification Bit) Functions

0	I/O port (P35, P34) operation (hold function input-output disabled) [initial value]
1	Hold request (HRQ) input/hold acknowledge ($\overline{\text{HAK}}$) output enabled

[Bit 12] IOBS

The IOBS bit is used to specify the bus width for external access to the area from 0000F0_H to 0000FF_H in external 16-bit data bus mode. Control is based on the setting of this bit, as shown in Table 9.2-8 "IOBS (Bus Width Specification Bit)".

Table 9.2-8 IOBS (Bus Width Specification Bit)

0	16-bit bus width access [initial value]
1	8-bit bus width access

[Bit 11] HMBS

The HMBS bit is used to specify the bus width for external access to the area from 800000_H to FFFFFFF_H in external 16-bit data bus mode. Control is based on the setting of this bit, as shown in Table 9.2-9 "HMBS (Bus Width Specification Bit)".

Table 9.2-9 HMBS (Bus Width Specification Bit)

0	16-bit bus width access [initial value]
1	8-bit bus width access

[Bit 10] WRE

The WRE bit controls output of external write signals (both $\overline{\text{WRH}}$ and $\overline{\text{WRL}}$ pins in external 16-bit data bus mode and $\overline{\text{WR}}$ pin in external 8-bit data bus mode), as shown in Table 9.2-10 "WRE (External Write Signal Output Control Bit) Functions".

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

Table 9.2-10 WRE (External Write Signal Output Control Bit) Functions

0	I/O port (P33, P32) operation (write signal output disabled) [initial value]
1	Write strobe signal ($\overline{\text{WRH}}/\overline{\text{WRL}}$ or $\overline{\text{WR}}$ only) output enabled

[Bit 9] LMBS

The LMBS bit is used to specify the bus width for external access to the area from 008000_H to 7FFFFFF_H in external 16-bit data bus mode. Control is based on the setting of this bit, as shown in Table 9.2-11 "LMBS (Bus Width Specification Bit)".

Table 9.2-11 LMBS (Bus Width Specification Bit)

0	16-bit bus width access [initial value]
1	8-bit bus width access

Notes:

- To use the WRE bit to enable the $\overline{\text{WRH}}$ and $\overline{\text{WRL}}$ functions in external 16-bit data bus mode, place P33 and P32 in input mode (set bits 3 and 2 of the DDR3 register to 0).
- To use the WRE bit to enable the $\overline{\text{WR}}$ function in external 8-bit data bus mode, place P32 in input mode (set bit 2 of the DDR3 register to 0).
- If the RYE and HDE bits are used to enable the RDY and HRQ signals, the I/O port function of the port is also enabled. Be sure to write 0 (input mode) to the DDR3 register that corresponds to the port.

9.3 External Memory Access Control Signal Operation

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

External Memory Access Control Signal

The HMBS, LMBS, and IOBS bits are used to specify whether 16-bit bus width access or 8-bit bus width access is to be used in external 16-bit data bus mode.

The ALE and address signals may be asserted even when no external bus access is performed. Do not use the ALE signal only to allow access to the external component. The RDX, WRLX and WRHX signals must be used together with the ALE signal to control the external access.

Figure 9.3-1 Access Timing Chart for External 8-bit Data Bus Mode)

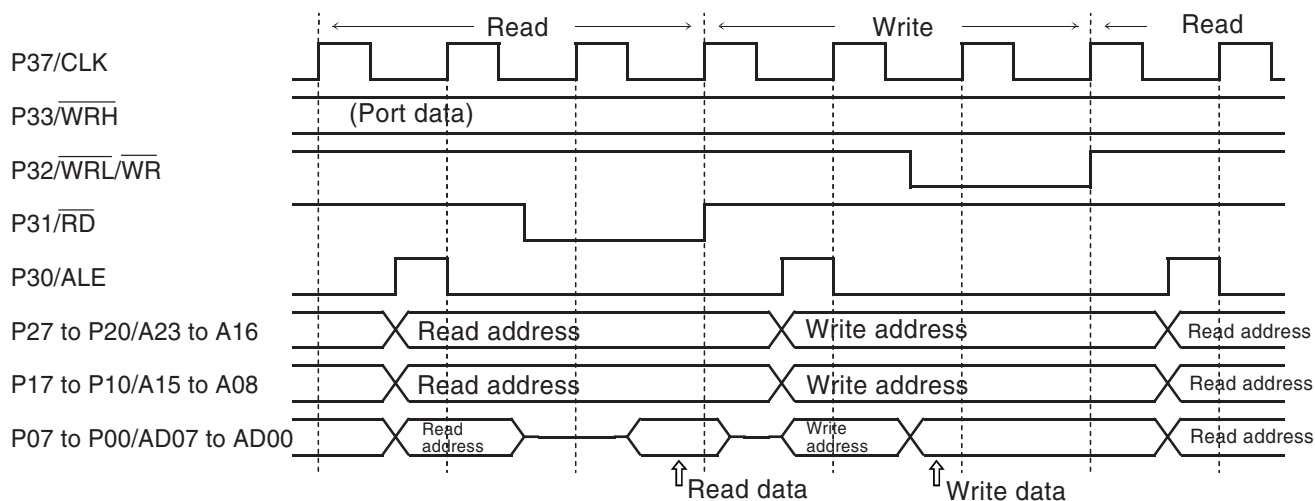
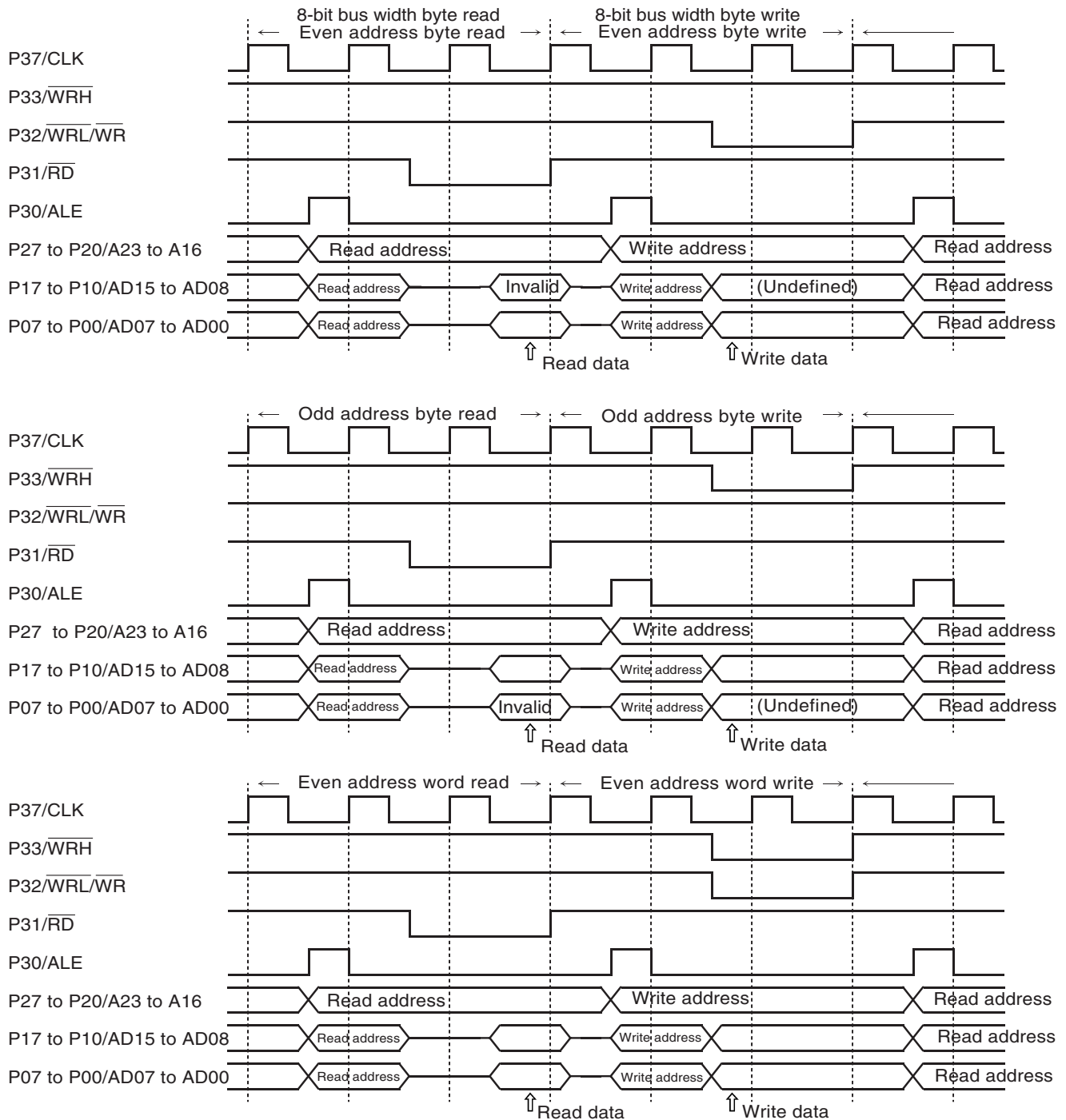


**Figure 9.3-2 Access Timing Chart for External 16-bit Data Bus Mode
(16-bit bus width access)**



Note:

Set the external circuit so that data is always read in word mode.

The setting of P36/RDY pin or the automatic ready function selection register (ARSR) enables access to low-speed memory and peripheral circuits.

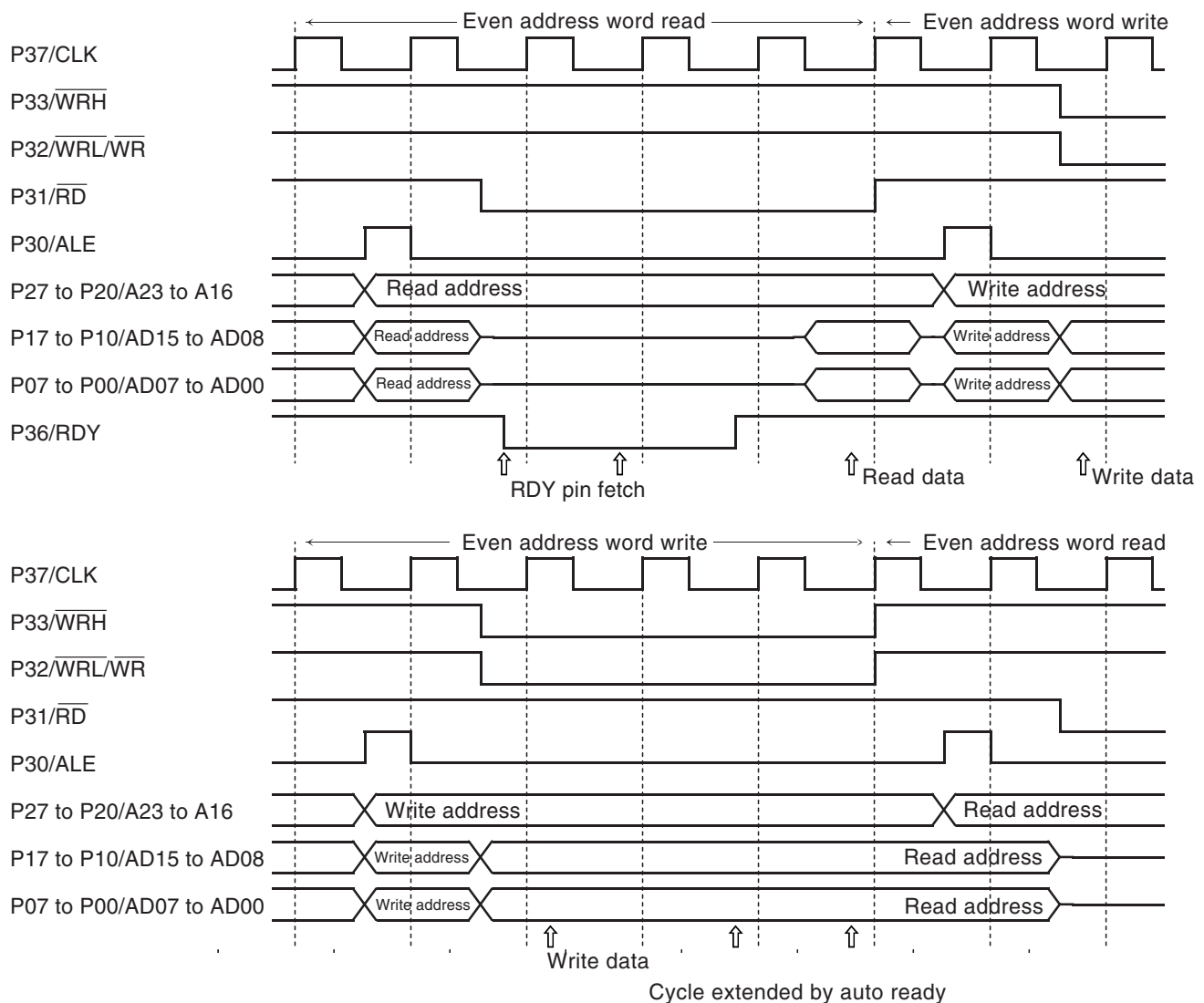
9.3.1 Ready Function

The setting of P36/RDY pin or the automatic ready function selection register (ARSR) enables access to the low-speed memory and peripheral circuits.

If the RYE bit of the bus control signal selection register (EPCR) is set to 1, the wait cycle is entered to enable extension of the access cycle while the L level is input to P36/RDY signal during access to the external circuit.

■ Ready Function

Figure 9.3-3 Timing Chart for Ready Function



The MB90340 series has two types of auto ready functions for external memory access. The auto ready function can automatically insert 1 to 3 wait cycles to extend the access cycle without an external circuit for access to the external areas at lower addresses 008000_H to 7FFFFF_H and at upper addresses 800000_H to FFFFFFF_H. This function is activated according to the setting of the LMR1 and LMR0 bits (external areas at lower addresses) of ARSR and the HMR1 and HMR0 bits (external area at upper addresses) of ARSR.

The MB90340 series also has an auto ready function for I/O that is independent of the auto ready function for memory. When the IOR1 and IOR0 bits of the ARSR register are set to 0, 1 to 3 wait cycles can be automatically inserted to extend the access cycle without an external circuit for access to the external area from addresses 0000C0_H to 0000FF_H.

If the RYE bit of the EPCR register is set to "1" and when the P36/RDY pin is signaled as "0", then the wait cycle is extended even after the wait cycle by the auto ready function is completed.

9.3.2 Hold Function

If the HDE bit in the bus control signal selection register (EPCR) is set to 1, the external bus hold function specified by the P34/HRQ and P35/ $\overline{\text{HAK}}$ pins is enabled.

■ Hold Function

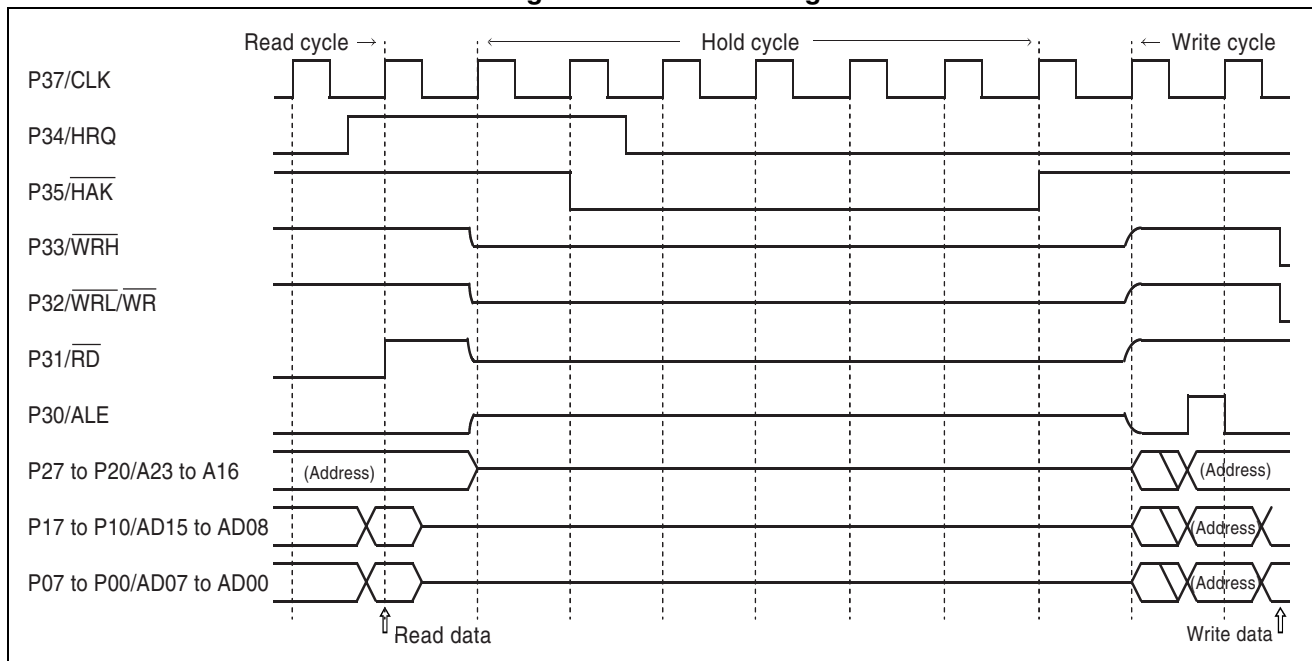
If the high level is applied to the P34/HRQ pin, the hold state is set up at termination of a CPU instruction (for a string instruction, at termination of 1-element data processing). The P35/ $\overline{\text{HAK}}$ pin outputs the low level to place the following pins in a high-impedance state:

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

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

Figure 9.3-4 "Hold Timing" shows the hold timing.

Figure 9.3-4 Hold Timing



CHAPTER 10

I/O PORTS

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

10.1 “I/O Port”

10.2 “I/O Port Registers”

10.1 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.

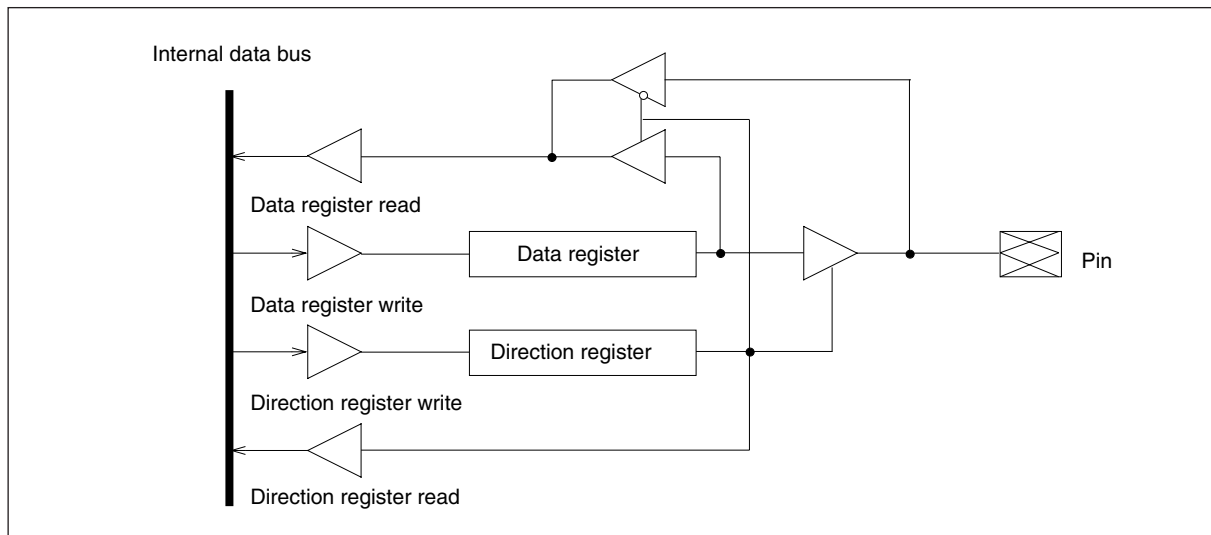
■ I/O ports

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 10.1-1 “I/O port block diagram” is a block diagram of the I/O ports.

Figure 10.1-1 I/O port block diagram



10.2 I/O Port Registers

There are four types of I/O port registers:

- Port data register (PDR0 to A)
- Port direction register (DDR0 to A)
- Pull-up control register (PUCR0 to 3)
- Analog input enable register (ADER)
- Input level select register (ILSR)

■ I/O port registers

Figure 10.2-1 “I/O port registers” shows the I/O port registers.

Figure 10.2-1 I/O port registers

Address:	Bit	7	6	5	4	3	2	1	0	
000000 _H	P07	P06	P05	P04	P03	P02	P01	P00		Port 0 data register (PDR0)
000001 _H	P17	P16	P15	P14	P13	P12	P11	P10		Port 1 data register (PDR1)
000002 _H	P27	P26	P25	P24	P23	P22	P21	P20		Port 2 data register (PDR2)
000003 _H	P37	P36	P35	P34	P33	P32	P31	P30		Port 3 data register (PDR3)
000004 _H	P47	P46	P45	P44	P43	P42	P41	P40		Port 4 data register (PDR4)
000005 _H	P57	P56	P55	P54	P53	P52	P51	P50		Port 5 data register (PDR5)
000006 _H	P67	P66	P65	P64	P63	P62	P61	P60		Port 6 data register (PDR6)
000007 _H	P77	P76	P75	P74	P73	P72	P71	P70		Port 7 data register (PDR7)
000008 _H	P87	P86	P85	P84	P83	P82	P81	P80		Port 8 data register (PDR8)
000009 _H	P97	P96	P95	P94	P93	P92	P91	P90		Port 9 data register (PDR9)
00000A _H	-	-	-	-	-	-	PA1	PA0		Port A data register (PDRA)
000010 _H	D07	D06	D05	D04	D03	D02	D01	D00		Port 0 direction register (DDR0)
000011 _H	D17	D16	D15	D14	D13	D12	D11	D10		Port 1 direction register (DDR1)
000012 _H	D27	D26	D25	D24	D23	D22	D21	D20		Port 2 direction register (DDR2)
000013 _H	D37	D36	D35	D34	D33	D32	D31	D30		Port 3 direction register (DDR3)
000014 _H	D47	D46	D45	D44	D43	D42	D41	D40		Port 4 direction register (DDR4)
000015 _H	D57	D56	D55	D54	D53	D52	D51	D50		Port 5 direction register (DDR5)
000016 _H	D67	D66	D65	D64	D63	D62	D61	D60		Port 6 direction register (DDR6)
000017 _H	D77	D76	D75	D74	D73	D72	D71	D70		Port 7 direction register (DDR7)
000018 _H	D87	D86	D85	D84	D83	D82	D81	D80		Port 8 direction register (DDR8)
000019 _H	D97	D96	D95	D94	D93	D92	D91	D90		Port 9 direction register (DDR9)
00001A _H	SIL4	SIL3	SIL2	SIL1	SIL0	-	DA1	DA0		Port A direction register (DDRA)
00001C _H	PU07	PU06	PU05	PU04	PU03	PU02	PU01	PU00		Port 0 pull-up control register (PUCR0)
00001D _H	PU17	PU16	PU15	PU14	PU13	PU12	PU11	PU10		Port 1 pull-up control register (PUCR1)
00001E _H	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20		Port 2 pull-up control register (PUCR2)
00001F _H	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30		Port 3 pull-up control register (PUCR3)
00000B _H	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8		Port 5 analog input enable register (ADER5)
00000C _H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0		Port 6 analog input enable register (ADER6)
00000D _H	ADE23	ADE22	ADE21	ADE20	ADE19	ADE18	ADE17	ADE16		Port B analog input enable register (ADER7)
00000E _H	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0		Input level select register0 (ILSR0)
00000F _H	ILT3	ILT2	ILT1	ILT0	-	ILA	IL9	IL8		Input level select register1 (ILSR1)

10.2.1 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 10.2-2 “Port data registers” shows the port data registers.

Figure 10.2-2 Port data registers

PDR0	7	6	5	4	3	2	1	0	Initial value	Access
Address: 000000 _H	P07	P06	P05	P04	P03	P02	P01	P00	Undefined	R/W
PDR1	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000001 _H	P17	P16	P15	P14	P13	P12	P11	P10	Undefined	R/W
PDR2	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000002 _H	P27	P26	P25	P24	P23	P22	P21	P20	Undefined	R/W
PDR3	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000003 _H	P37	P36	P35	P34	P33	P32	P31	P30	Undefined	R/W
PDR4	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000004 _H	P47	P46	P45	P44	P43	P42	P41	P40	Undefined	R/W
PDR5	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000005 _H	P57	P56	P55	P54	P53	P52	P51	P50	Undefined	R/W
PDR6	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000006 _H	P67	P66	P65	P64	P65	P62	P61	P60	Undefined	R/W
PDR7	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000007 _H	P77	P76	P75	P74	P73	P72	P71	P70	Undefined	R/W
PDR8	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000008 _H	P87	P86	P85	P84	P83	P82	P81	P80	Undefined	R/W
PDR9	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 000009 _H	P97	P96	P95	P94	P93	P92	P91	P90	Undefined	R/W
PDRA	7	6	5	4	3	2	1	0	Undefined	R/W
Address: 00000A _H	-	-	-	-	-	-	PA1	PA0	Undefined	R/W

R/W : Readable and writable
- : Undefined

● Reading the port data register

When a Port Data register is read, the value depends on the corresponding bit in the Data Direction Register and on the current status of the resource that is connected to the same pin (if applicable).

The following cases are possible:

DDR value	Resource	Read value
0 (input)	enabled	Resource value
1 (output)	enabled	Resource value
0 (input)	disabled	Pin value
1 (output)	disabled	PDR value

10.2.2 Data Direction Register

This register has following functions:

- Setting the data direction of each pin that is used as a port.
- Setting the input level of SIN -- Serial data input pin for LIN-UART.

■ Data direction register

Figure 10.2-3 “Data direction registers” shows the Data Direction Registers.

Figure 10.2-3 Data direction registers

DDR0	7	6	5	4	3	2	1	0	Initial value	Access
Address: 000010 _H	D07	D06	D05	D04	D03	D02	D01	D00	00000000 _B	R/W
DDR1	7	6	5	4	3	2	1	0		
Address: 000011 _H	D17	D16	D15	D14	D13	D12	D11	D10	00000000 _B	R/W
DDR2	7	6	5	4	3	2	1	0		
Address: 000012 _H	D27	D26	D25	D24	D23	D22	D21	D20	00000000 _B	R/W
DDR3	7	6	5	4	3	2	1	0		
Address: 000013 _H	D37	D36	D35	D34	D33	D32	D31	D30	00000000 _B	R/W
DDR4	7	6	5	4	3	2	1	0		
Address: 000014 _H	D47	D46	D45	D44	D43	D42	D41	D40	00000000 _B	R/W
DDR5	7	6	5	4	3	2	1	0		
Address: 000015 _H	D57	D56	D55	D54	D53	D52	D51	D50	00000000 _B	R/W
DDR6	7	6	5	4	3	2	1	0		
Address: 000016 _H	D67	D66	D65	D64	D63	D62	D61	D60	00000000 _B	R/W
DDR7	7	6	5	4	3	2	1	0		
Address: 000017 _H	D77	D76	D75	D74	D73	D72	D71	D70	00000000 _B	R/W
DDR8	7	6	5	4	3	2	1	0		
Address: 000018 _H	D87	D86	D85	D84	D83	D82	D81	D80	00000000 _B	R/W
DDR9	7	6	5	4	3	2	1	0		
Address: 000019 _H	D97	D96	D95	D94	D93	D92	D91	D90	00000000 _B	R/W
DDRA	7	6	5	4	3	2	1	0		
Address: 00001A _H	SIL4	SIL3	SIL2	SIL1	SIL0	-	DA1	DA0	00000100 _B	bit0 to 1: R/W bit3 to 7: W

R/W : Readable and writable
W : Write Only (Read always returns "1")
- : Undefined

Bits Dxx (DDR0 to DDRA)

These bits set the data direction of ports.

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

0	Input mode
1	Output mode

Note:

The Data Direction Register can be read independently from the status of the corresponding resource. However, the value of the DDR influences the result of a read access on the Port Data Register.

Bits SIL0 to SIL4 (DDRA bit3 to bit7)

These bits set the input level of the corresponding SIN (Serial Data Input for LIN-UART) signal to CMOS forcibly. SIL0 sets the input level of SIN0(LIN-UART0), SIL4 sets the input level of SIN4(LIN-UART4).

Setting these bits to “0” selects CMOS or Automotive or TTL input, depending of the setting of the corresponding ILx bit and ILTx bit of ILSR.

(About ILSR, please see 10.2.5 “Input Level Select Register”, also.)

Setting these bits to “1” selects CMOS input level, independent of the setting of the corresponding ILx bit and ILTx bit of ILSR. The initial value of these bits is “0”.

Table 10.2-1 The input level setting of SIN0/SIN1

SIL0/SIL1	IL8	input level of SIN0 (P82)/SIN1 (P85)
0	0	Automotive level
0	1	CMOS level
1	x	CMOS level

Table 10.2-2 The input level setting of SIN2

SIL2	IL5	input level of SIN2 (P50)
0	0	Automotive level
0	1	CMOS level
1	x	CMOS level

Table 10.2-3 The input level setting of SIN3/SIN4

SIL3/SIL4	ILT1	IL1	input level of SIN3 (P12)/SIN4 (P15)
0	0	0	Automotive level
0	0	1	CMOS level
0	1	x	TTL
1	x	x	CMOS level

Note:

SIL0 to SIL4 are write-only, and “1” is always read from these bits. Therefore, instructions that perform a read-modify-write (RMW) operation such as the INC/DEC instruction, cannot be used at DDRA.

10.2.3 Pull-up Control Register (PUCR)

Each pin of port0 to port3 has programmable pull-up resistor. Each bit of this register controls corresponding pull-up register whether to be used or not.

Figure 10.2-4 "Pull-up Control Register" shows the pull-up control register (PUCR), and Figure 10.2-5 "Block Diagram of Pull-up Control Register (PUCR)" is the block diagram.

■ Pull-up Control Register (PUCR)

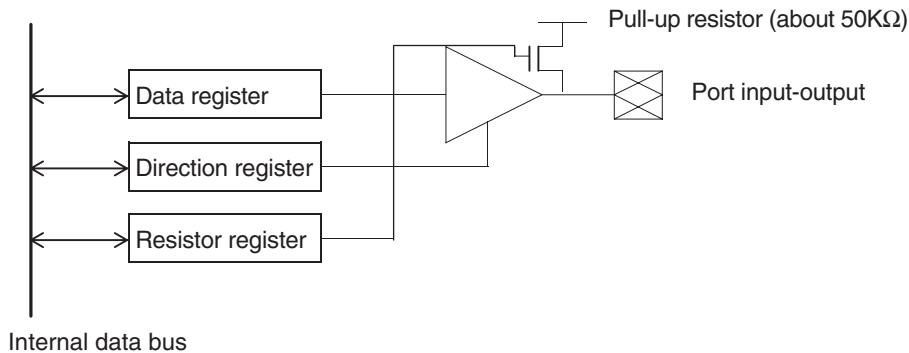
Figure 10.2-4 Pull-up Control Register

PUCR0	7	6	5	4	3	2	1	0	Initial value	Access
Address: 00001C _H	PU07	PU06	PU05	PU04	PU03	PU02	PU01	PU00	00000000 _B	R/W
PUCR1	15	14	13	12	11	10	9	8	00000000 _B	R/W
Address: 00001D _H	PU17	PU16	PU15	PU14	PU13	PU12	PU11	PU10	00000000 _B	R/W
PUCR2	7	6	5	4	3	2	1	0	00000000 _B	R/W
Address: 00001E _H	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20	00000000 _B	R/W
PUCR3	15	14	13	12	11	10	9	8	00000000 _B	R/W
Address: 00001F _H	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30	00000000 _B	R/W

R/W : Readable and writable

■ Block Diagram of Pull-up Control Register (RDR)

Figure 10.2-5 Block Diagram of Pull-up Control Register (PUCR)



Note:

- In input mode, the pull-up resistor is controlled.
- 0: No pull-up resistor in input mode
- 1: Pull-up resistor in input mode
- In output mode, this register has no meaning (no pull-up resistor).
- The direction register (DDR) determines the input-output mode.
- In stop mode (SPL=1), the state with no pull-up resistor is entered (high impedance).
- If the port is used as an external bus, this function is disabled and data is not written to the register.

10.2.4 Analog Input Enable Register

This register controls the port 5 to 7 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 registers

Figure 10.2-6 “Analog input enable registers” shows the analog input enable register.

Figure 10.2-6 Analog input enable registers (ADER7/6/5)

ADER7	15	14	13	12	11	10	9	8	Initial value	Access
Address: 00000D _H	ADE23	ADE22	ADE21	ADE20	ADE19	ADE18	ADE17	ADE16	11111111 _B	R/W
ADER6	7	6	5	4	3	2	1	0		
Address: 00000C _H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	11111111 _B	R/W
ADER5	15	14	13	12	11	10	9	8		
Address: 00000B _H	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8	11111111 _B	R/W

R/W : Readable and writable

Note:

Initial value of the ADE_x bits are "1". Therefore the corresponding pins can not be used as logical input/output as initial setting. If these are used as IO ports or peripheral function IOs, then write "0" to the ADE_x bits.

10.2.5 Input Level Select Register

The Input Level Select Register allows to switch from Automotive Hysteresis input levels to CMOS Hysteresis input levels or to TTL input levels.

■ Input level select register

The input level select register ILSR is located on addresses 0E_H and 0F_H.

Figure 10.2-7 Input level select register (ILSR)

Address		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILSR1 : 00000F _H		ILT3	ILT2	ILT1	ILT0	-	ILA	IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0
ILSR0 : 00000E _H		R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value:		0/1	0/1	0/1	0/1	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

R/W : Readable and writable
- : undefined

Bits 0 to 10: IL0 to ILA

These bits set the input level of the corresponding port. IL0 sets the input level of Port0, ILA sets the input level of Port A. Setting these bits to “0” selects the Automotive input level. Setting these bits to “1” selects the CMOS input level. The initial value of these bits depend on the mode pin setting:

- For the Flash memory mode, the initial value is “1” (CMOS).
- For all other modes, the initial value is “0” (Automotive).

Bit 11: undefined

Reading this bit always returns “0”.

Bit 12 to 15: ILT0 to ILT3

These bits set the input level of the corresponding port to TTL or CMOS/Automotive. ILT0 sets the input level of Port0, ILT3 sets the input level of Port3. Setting these bits to “0” selects the CMOS input level or the Automotive input level, depending of the setting of the corresponding ILx register. Setting these bits to “1” selects the TTL input level, independent of the setting of the corresponding ILx register. The initial value of these bits depend on the mode pin setting:

- For the external reset vector fetch modes, the initial value is “1” (TTL).
- For all other modes, the initial value is “0” (Automotive/CMOS).

■ Initialization of ILSR

Initial value of each bit of ILSR are determined when External Reset Signal is released depending on the value of MD2, MD1, MD0 pin input, as shown in following table.

About detail of each mode, please see CHAPTER 9 “Memory Access Modes”.

MD2	MD1	MD0	Mode Name	Initial value		Port Input level	
				ILTx	ILx	Port 0 to 3	Port 4 to A
0	0	0	External vector mode 0	1	0	TTL	Automotive
0	0	1	External vector mode 1	1	0	TTL	Automotive
0	1	0	Reserved				
0	1	1	Internal vector mode	0	0	Automotive	Automotive
1	0	0	Reserved				
1	0	1					
1	1	0	Flash memory serial programing	0	0	Automotive	Automotive
1	1	1	Flash memory	0	1	CMOS	CMOS

CHAPTER 11

TIMEBASE TIMER

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

11.1 “Outline of Timebase Timer”

11.2 “Timebase Timer Control Register”

11.3 “Operations of Timebase Timer”

11.1 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 main clock. The timebase timer issues interrupts at specified intervals based on carry signals of the timebase counter.

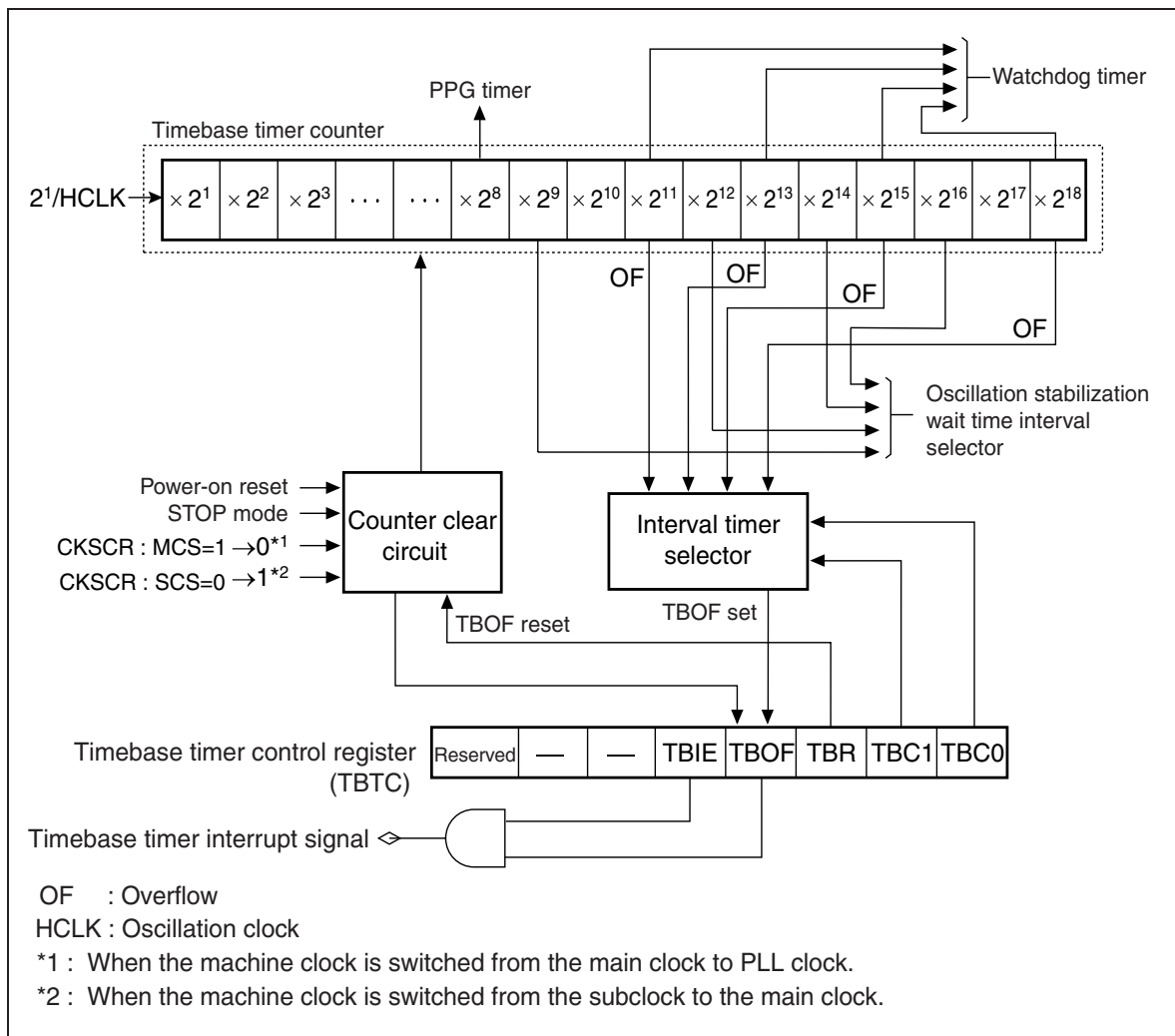
■ Outline of timebase timer

After the power on reset, the timebase timer is initialized to zero. The timebase timer is also initialized when transition to the stop mode or transition between different clock modes, or by writing "0" to the TBR bit of the TBTC register. The timebase timer continues counting as long as the main clock is supplied. The timebase timer can be used to generate clock signals to the watch-dog timer or for generating the oscillation stabilization wait time.

■ Block diagram of timebase timer

Figure 11.1-1 “Block diagram of timebase timer” shows a block diagram of the timebase timer.

Figure 11.1-1 Block diagram of timebase timer



11.2 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)

Figure 11.2-1 Configuration of the timebase timer control register (TBTC)

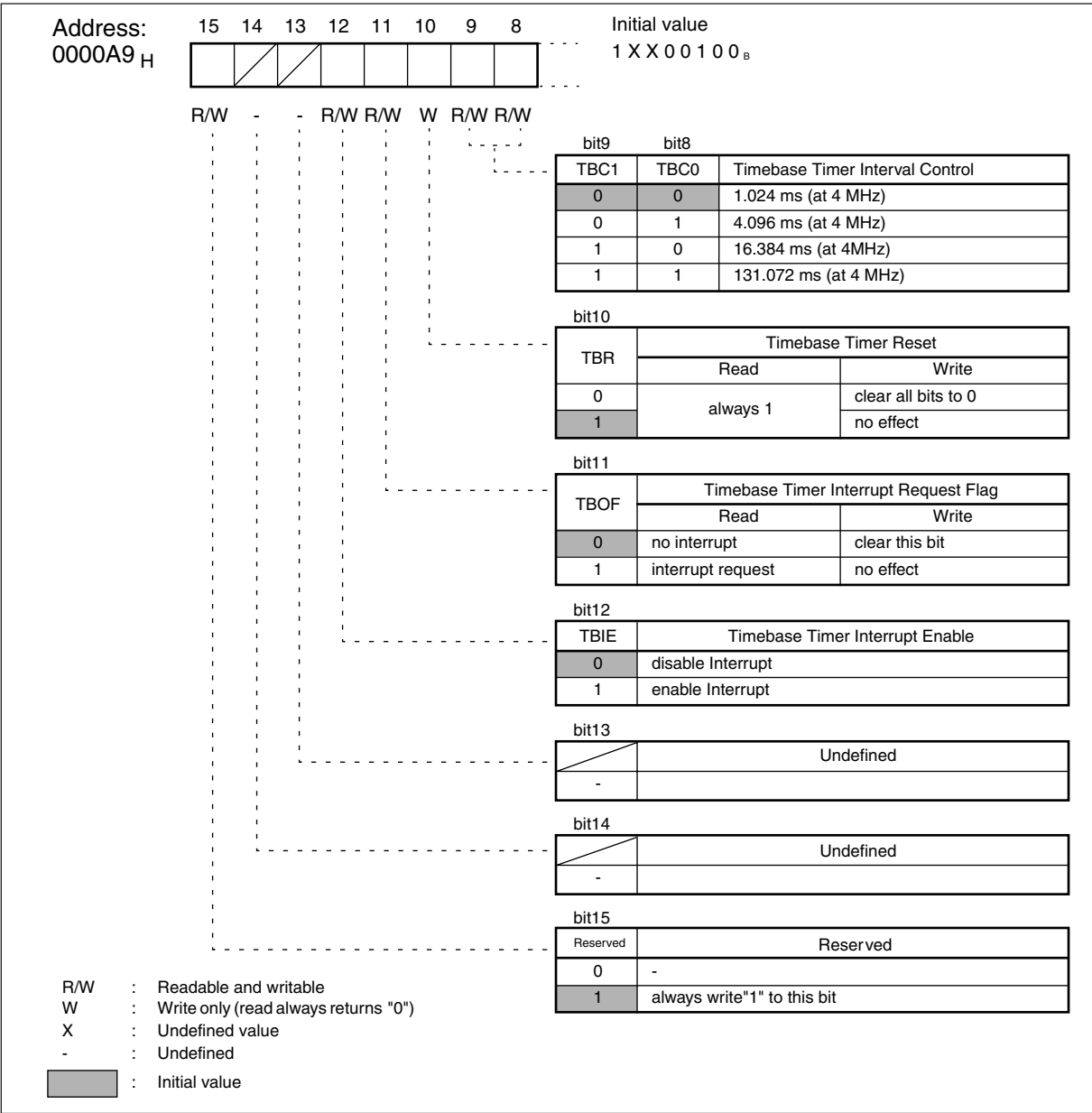


Table 11.2-2 shows the settings for TBC1 and TBC0:

Table 11.2-1 Function description of each bit of the timebase timer control register

Bit name		Function
bit 15	Reserved	This is a reserved bit. When writing data to the TBTC register ensure that "1" is written to this bit.
bit 14	Undefined	
bit 13	Undefined	
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". This bit is also cleared when transition to the stop mode or transition between different clock modes (from the subclock mode to the main clock mode, from the subclock mode to the PLL clock mode and from the main clock mode to the PLL clock mode), or by writing "0" to the TBR bit of the TBTC register and by any 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.
bit 9/8	TBC1/0	These bits are used to set the timebase timer interval. Table 11.2-2 "Selecting the timebase timer interval" lists the specifiable intervals.

Table 11.2-2 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

11.3 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 clear

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 timer counter is cleared when transition to the stop mode or transition between different clock modes (from the subclock mode to the main clock mode, from the subclock mode to PLL clock mode and from the main clock mode to the PLL clock mode), or by writing "0" to the TBR bit of the TBTC register and by the power on reset.

■ 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 set with reference to the time at which the timebase timer is cleared last time.

Upon transition from stop 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 the mode transition to the stop mode.

■ Timebase counter clear

The timebase timer counter is cleared when transition to the stop mode or transition between different clock modes (from the subclock mode to the main clock mode, from the subclock mode to PLL clock mode and from the main clock mode to the PLL clock mode), or by writing "0" to the TBR bit of the TBTC register and by the power on reset.

CHAPTER 12

WATCH-DOG TIMER

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

12.1 “Outline of Watch-Dog Timer”

12.2 “Watch-dog Timer Operations”

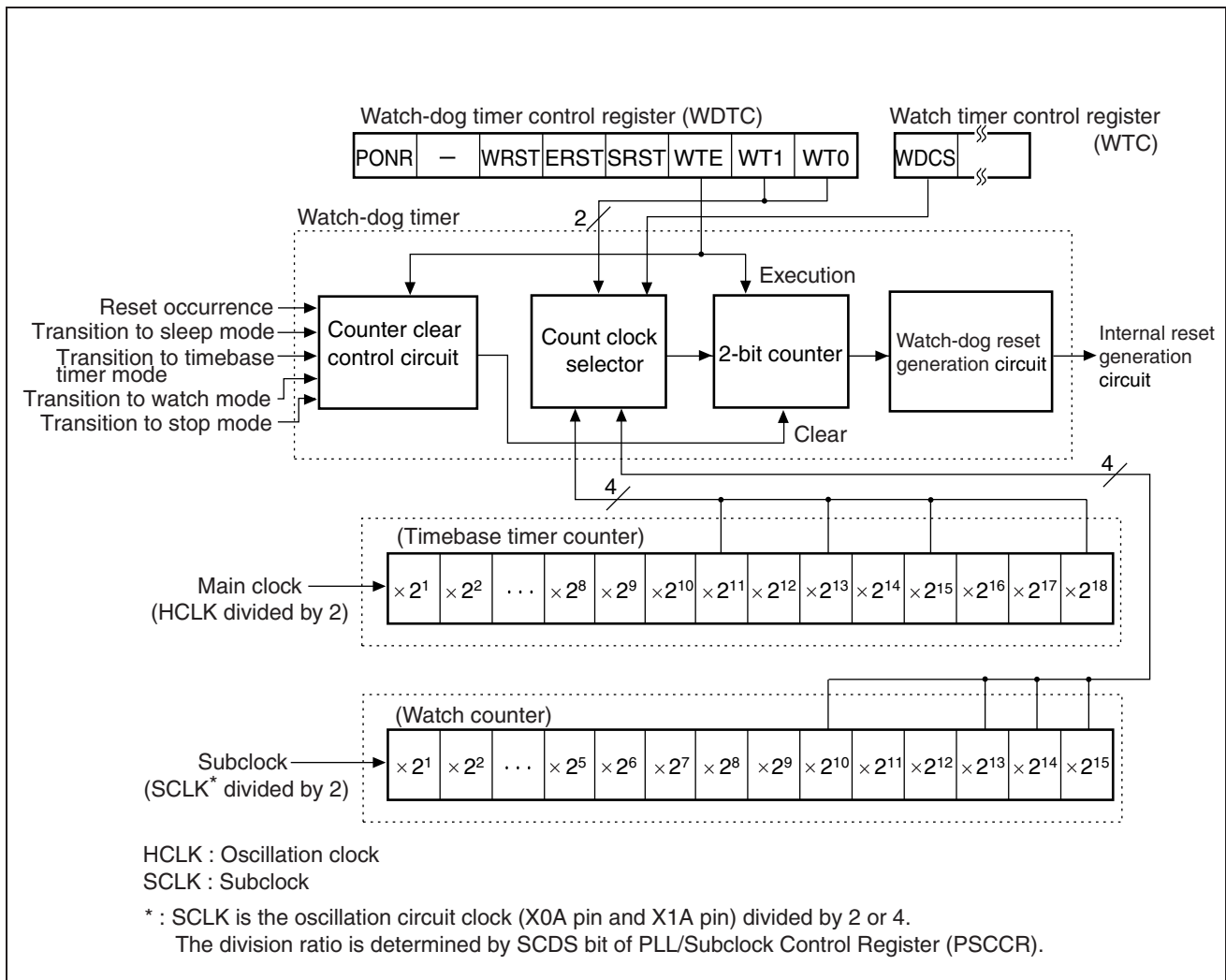
12.1 Outline of Watch-Dog Timer

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 a carry signal of the 18-bit timebase counter as a clock source.

■ Watch-dog timer block diagram

Figure 12.1-1 “Watch-dog timer block diagram” is a diagram of the configuration of the watch-dog timer.

Figure 12.1-1 Watch-dog timer block diagram



■ Watch-dog timer control register (WDTC)

WDTC Address : 0000A8 _H	7	6	5	4	3	2	1	0	Initial value XXXXXX11 _B
	PONR	-	WRST	ERST	SRST	WTE	WT1	WT0	
	R	-	R	R	R	W	W	W	

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

These flags indicate the reset causes. The flags are set upon a reset as described in Table 12.1-1 “Reset cause registers”.

All bits are cleared to "0" after the WDTC register is read. These bits are read-only bits.

Table 12.1-1 Reset cause registers

Reset cause	PONR	WRST	ERST	SRST
Power-on	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 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 12.1-2 “Watch-dog timer interval selection bit” lists the interval settings.

These bits are write-only bits.

Table 12.1-2 Watch-dog timer interval selection bit (WTC: WDSC = 0)

WT1	WT0	Interval *		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

*: For a source oscillation of 4 MHz.

Table 12.1-3 Watch-dog timer interval selection bit (WTC: WDSC = 1)

WT1	WT0	Interval *		Main clock cycle count
		Minimum	Maximum	
0	0	approx. 0.457 s	approx. 0.576 ms	2^{12} plus or minus 2^9 cycles
0	1	approx. 3.584 s	approx. 4.608 ms	2^{15} plus or minus 2^{12} cycles
1	0	approx. 7.168 s	approx. 9.216 ms	2^{16} plus or minus 2^{13} cycles
1	1	approx. 14.336 s	approx. 18.432 ms	2^{17} plus or minus 2^{14} cycles

*: For a source oscillation of 8.192 MHz.

Note:

The interval time uses a carry signal of the time-base timer or watch timer as a count clock. If the time-base timer or watch timer is cleared, the interval time of the watchdog timer may become longer.

The time-base timer is also cleared by writing zero to the TBR bit in the time-base timer control register (TBTC), transition from main clock mode to PLL clock mode, transition from sub-clock mode to main clock mode and transition from sub-clock mode to PLL clock mode.

12.2 Watch-dog Timer Operation

The watch-dog timer function enables detection of program surge.

If the watch-dog timer is not accessed within the specified time due to, for example, a program surge, 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.

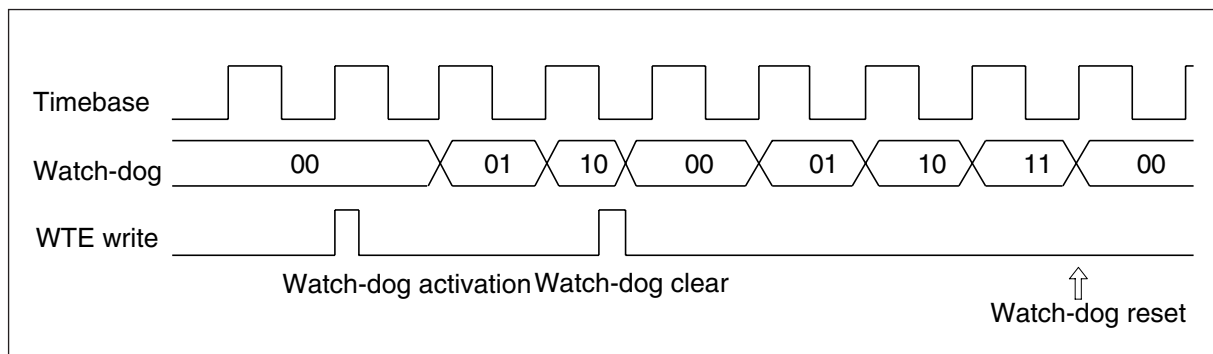
■ Watch-dog counter

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 a carry signals of the timebase counter or watch timer 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 any reset and transition to the standby (sleep, stop, watch, timebase timer modes) or stop mode.

Figure 12.2-1 “Watch-dog timer operation” is a diagram of the watch-dog timer operation.

Figure 12.2-1 Watch-dog timer operation



■ Watch-dog stop

Once activated, the watch-dog timer can be stopped only by reset.

CHAPTER 13

16-BIT I/O TIMER

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

13.1 “Outline of 16-Bit I/O Timer”

13.2 “16-Bit I/O Timer Registers”

13.3 “16-bit Free-running Timer”

13.4 “Output Compare”

13.5 “Input Capture”

13.1 Outline of 16-Bit I/O Timer

The MB90340 Series contains two 16-bit free-running timer modules, four output compare modules, and four input capture modules and supports eight input channels and eight output channels. The following sections describe the 16-bit free-running timer, Output Compare and Input Capture.

■ 16-bit free-running timer

The two 16-bit free-run timers consist of a 16-bit up counter, control register, and prescaler each. The values output from these timer counters are used as the base time for input capture and output compare.

- Eight counter clocks are available.

Internal clock: ϕ , $\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$, $\phi/32$, $\phi/64$, $\phi/128$ (ϕ : machine clock)

- An interrupt can be generated upon a counter overflow or a match with compare register 0 and 4.
- The counter value can be initialized to '0000_H' upon a reset, software clear, or match with compare register 0 for timer 0, resp. compare register 4 for timer 1.

■ Output compare (2 channels per one module)

The four output compare modules consist of two 16-bit compare registers, compare output latch, and control register each.

Output Compare 0 and 1 (channels OUT0, OUT1, OUT2 and OUT3) are assigned to Free-running Timer 0 and Output Compare 2 and 3 (channels OUT4, OUT5, OUT6 and OUT7) are assigned to Free-running Timer 1.

When a 16-bit free-running timer value matches the corresponding compare register value, the output level is toggled and an interrupt can be issued.

- The two compare registers can be used independently for each Output Compare.

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 toggled 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 four input capture modules consist of two 16-bit capture registers and control registers each corresponding to two independent external input pins. The 16-bit free-running timer values 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.

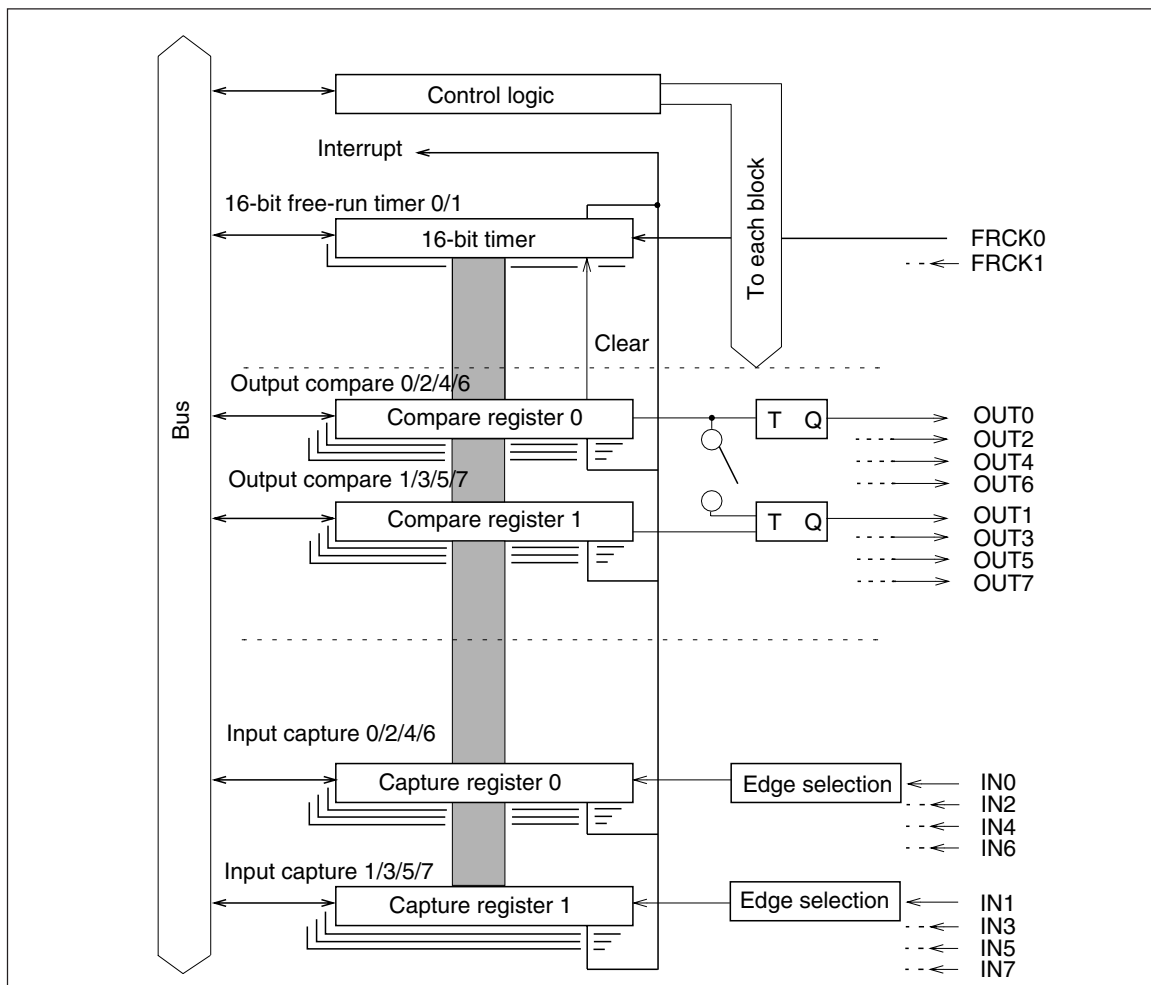
- An interrupt can be issued upon a valid edge of an external input signal.

The intelligent I/O service can be activated upon an input capture interrupt.

■ Block diagram of 16-bit I/O timer

Figure 13.1-1 “Block diagram of 16-bit I/O timer” shows a block diagram of the 16-bit I/O timer.

Figure 13.1-1 Block diagram of 16-bit I/O timer



13.2 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

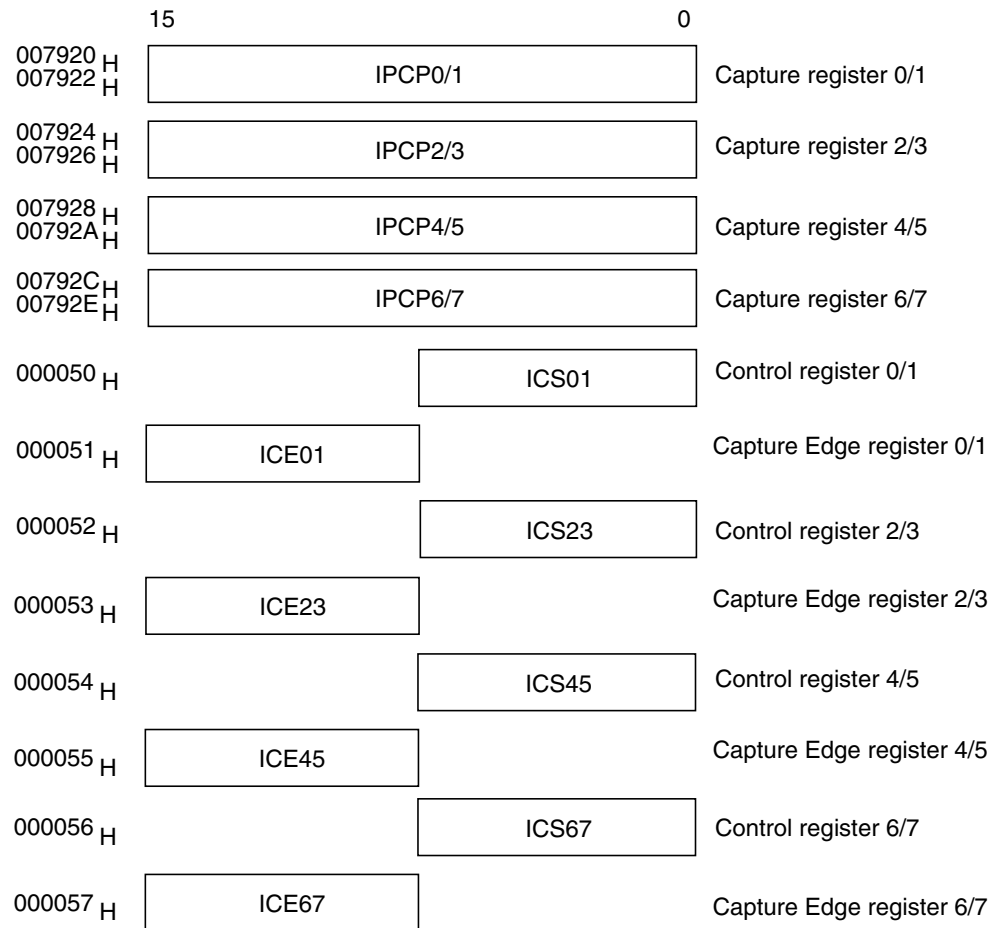
■ 16-bit free-running timer 0 and 1

	15		0	
007940 _H	TCDT0			Timer data register 0
007944 _H	TCDT1			Timer data register 1
007942 _H 007943 _H	TCCSH0		TCCSL0	Timer status register 0
007946 _H 007947 _H	TCCSH1		TCCSL1	Timer status register 1

■ 16-bit output compare

	15		0	
007930 _H 007932 _H	OCCP0/1			Compare register 0/1
007934 _H 007936 _H	OCCP2/3			Compare register 2/3
007938 _H 00793A _H	OCCP4/5			Compare register 4/5
00793C _H 00793E _H	OCCP6/7			Compare register 6/7
000058 _H 000059 _H	OCS1		OCS0	Control status register 0/1
00005A _H 00005B _H	OCS3		OCS2	Control status register 2/3
00005C _H 00005D _H	OCS5		OCS4	Control status register 4/5
00005E _H 00005F _H	OCS7		OCS6	Control status register 6/7

■ 16-bit input capture



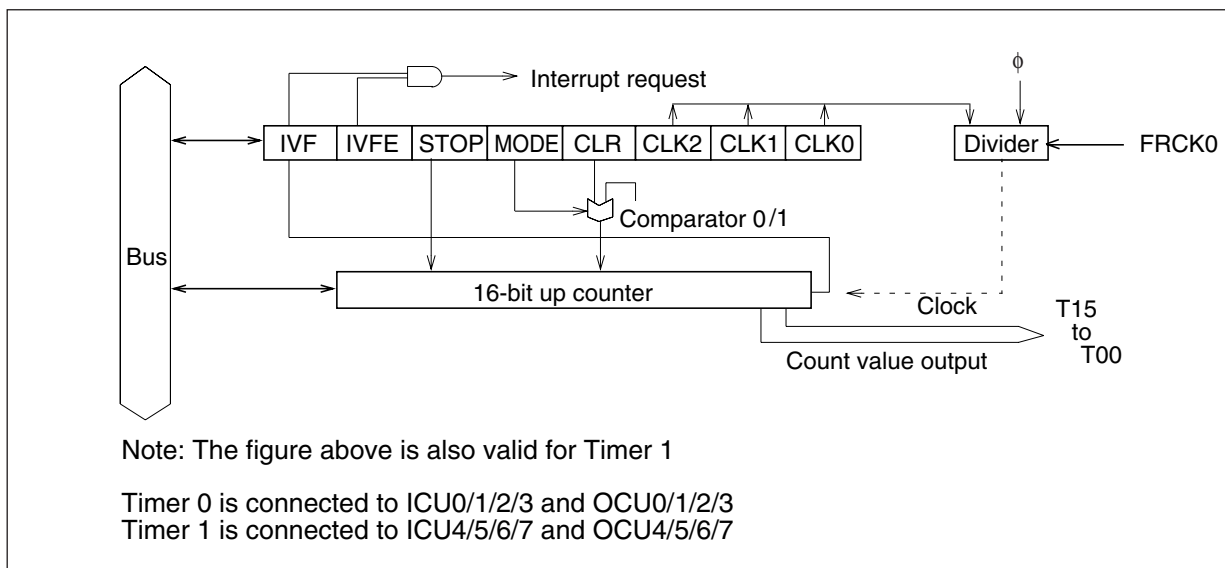
13.3 16-bit Free-running Timer

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 time for the output compares and input captures.

- Eight 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 and compare register 4, depending on the mode.
- Two separate timers are available on MB90340 series

■ 16-bit free-running timer block diagram

Figure 13.3-1 16-bit free-running timer block diagram



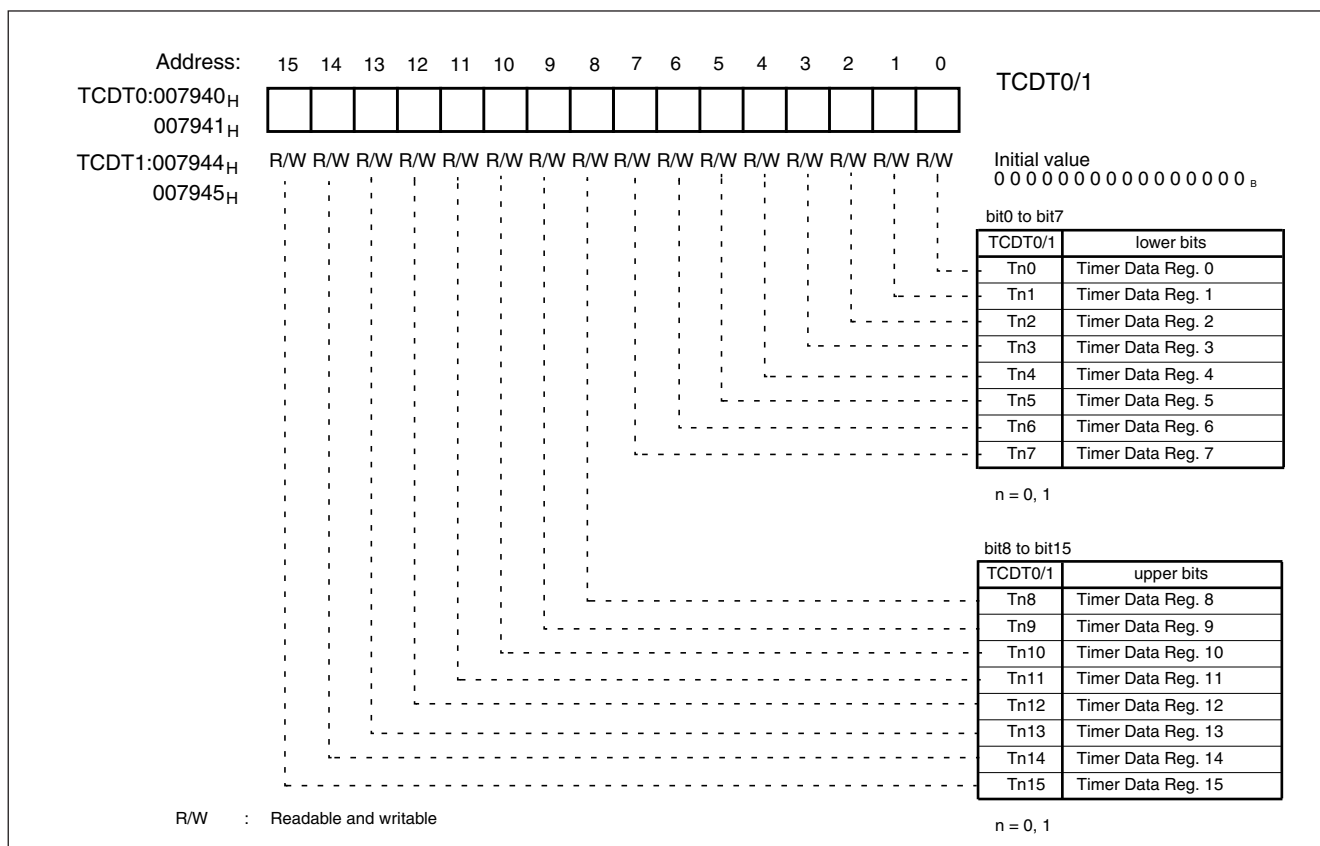
13.3.1 Data Register

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 of free-running timer

Figure 13.3-2 Data register of free-running timer (TCDT0/1)



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.

13.3.2 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 of free-running timer (lower)

Figure 13.3-3 Control status register of free-running timer (TCCSL0/1)

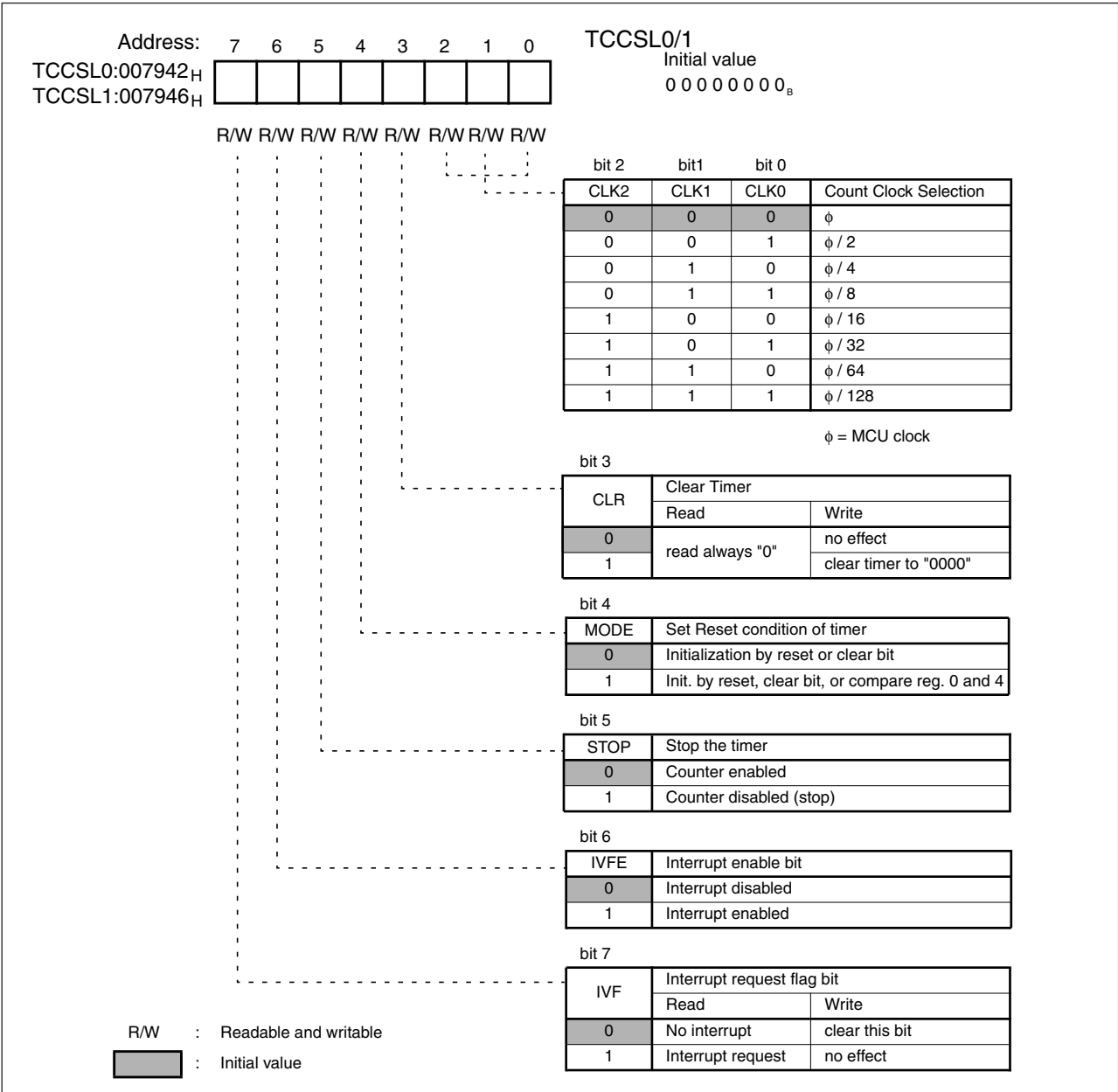


Table 13.3-1 Control status register of free-running-timer (lower)

Bit name		Function																																																																																	
bit 7	IVF	- This bit is the interrupt request flag bit and clear bit - Writing "0": A possible interrupt is cleared. - Writing "1": No effect. "1" is always read during a read-modify-write cycle. - This bit is set to "1" if the free running timer overflows or if the timer value matches with the compare register 0 and 4 when the MODE bit is set to "1".																																																																																	
bit 6	IVFE	- This bit enables the interrupt request - Writing "0": Interrupt disabled. - Writing "1": Interrupt enabled.																																																																																	
bit 5	STOP	- The STOP bit is used to stop the timer. - Writing "0": Counter enabled (operation). - Writing "1": Counter disabled (stop).																																																																																	
bit 4	MODE	"0": Initialization by reset or clear bit "1": Initialization by reset, clear bit, or compare register 0 and 4																																																																																	
bit 3	CLR	- The CLR bit initializes the operating free-running timer to the value "0000" - Writing "0": no effect. - Writing "1": Counter is initialized. Note: To initialize the counter value while the timer is stopped (STOP = 1), write "0000" to the data register (TCDT).																																																																																	
bit 2 to 0	CLK2, CLK1, CLK0	These bits 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 input capture and output compare operations are stopped before a value is written to these bits. <table><tr><th>CLK2</th><th>CLK1</th><th>CLK0</th><th>Count clock</th><th>Φ = 20 MHz</th><th>Φ = 16 MHz</th><th>Φ = 8 MHz</th><th>Φ = 4 MHz</th><th>Φ = 1 MHz</th></tr><tr><td>0</td><td>0</td><td>0</td><td>Φ</td><td>50 ns</td><td>62.5 ns</td><td>125 ns</td><td>0.25 μs</td><td>1 μs</td></tr><tr><td>0</td><td>0</td><td>1</td><td>Φ / 2</td><td>100 ns</td><td>125 ns</td><td>0.25 μs</td><td>0.5 μs</td><td>2 μs</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Φ / 4</td><td>0.2 μs</td><td>0.25 μs</td><td>0.5 μs</td><td>1 μs</td><td>4 μs</td></tr><tr><td>0</td><td>1</td><td>1</td><td>Φ / 8</td><td>0.4 μs</td><td>0.5 μs</td><td>1 μs</td><td>2 μs</td><td>8 μs</td></tr><tr><td>1</td><td>0</td><td>0</td><td>Φ / 16</td><td>0.8 μs</td><td>1 μs</td><td>2 μs</td><td>4 μs</td><td>16 μs</td></tr><tr><td>1</td><td>0</td><td>1</td><td>Φ / 32</td><td>1.6 μs</td><td>2 μs</td><td>4 μs</td><td>8 μs</td><td>32 μs</td></tr><tr><td>1</td><td>1</td><td>0</td><td>Φ / 64</td><td>3.2 μs</td><td>4 μs</td><td>8 μs</td><td>16 μs</td><td>64 μs</td></tr><tr><td>1</td><td>1</td><td>1</td><td>Φ / 128</td><td>6.4 μs</td><td>8 μs</td><td>16 μs</td><td>32 μs</td><td>128 μs</td></tr></table>	CLK2	CLK1	CLK0	Count clock	Φ = 20 MHz	Φ = 16 MHz	Φ = 8 MHz	Φ = 4 MHz	Φ = 1 MHz	0	0	0	Φ	50 ns	62.5 ns	125 ns	0.25 μs	1 μs	0	0	1	Φ / 2	100 ns	125 ns	0.25 μs	0.5 μs	2 μs	0	1	0	Φ / 4	0.2 μs	0.25 μs	0.5 μs	1 μs	4 μs	0	1	1	Φ / 8	0.4 μs	0.5 μs	1 μs	2 μs	8 μs	1	0	0	Φ / 16	0.8 μs	1 μs	2 μs	4 μs	16 μs	1	0	1	Φ / 32	1.6 μs	2 μs	4 μs	8 μs	32 μs	1	1	0	Φ / 64	3.2 μs	4 μs	8 μs	16 μs	64 μs	1	1	1	Φ / 128	6.4 μs	8 μs	16 μs	32 μs	128 μs
CLK2	CLK1	CLK0	Count clock	Φ = 20 MHz	Φ = 16 MHz	Φ = 8 MHz	Φ = 4 MHz	Φ = 1 MHz																																																																											
0	0	0	Φ	50 ns	62.5 ns	125 ns	0.25 μs	1 μs																																																																											
0	0	1	Φ / 2	100 ns	125 ns	0.25 μs	0.5 μs	2 μs																																																																											
0	1	0	Φ / 4	0.2 μs	0.25 μs	0.5 μs	1 μs	4 μs																																																																											
0	1	1	Φ / 8	0.4 μs	0.5 μs	1 μs	2 μs	8 μs																																																																											
1	0	0	Φ / 16	0.8 μs	1 μs	2 μs	4 μs	16 μs																																																																											
1	0	1	Φ / 32	1.6 μs	2 μs	4 μs	8 μs	32 μs																																																																											
1	1	0	Φ / 64	3.2 μs	4 μs	8 μs	16 μs	64 μs																																																																											
1	1	1	Φ / 128	6.4 μs	8 μs	16 μs	32 μs	128 μs																																																																											

■ Control status register of free-running timer (upper)

Figure 13.3-4 Control status register of free-running timer (TCCSH0/1)

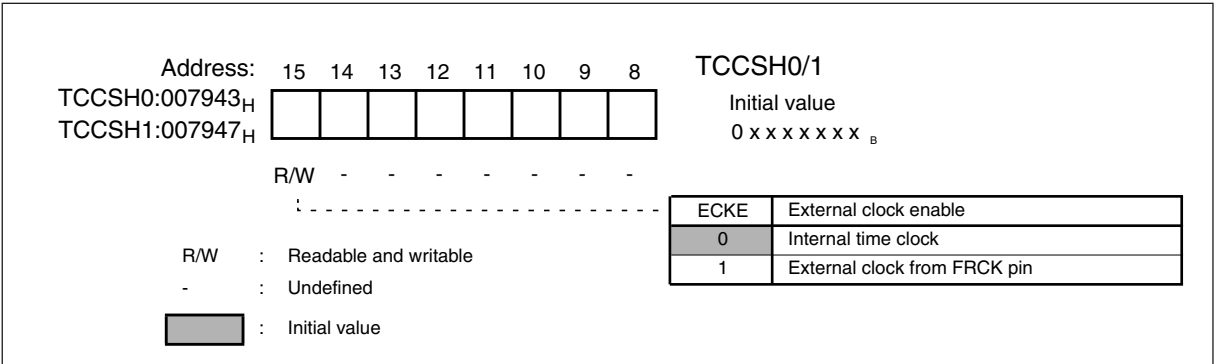


Table 13.3-2 Control status register of free-running-timer (upper)

Bit name		Function
bit 15	ECKE	• This bit chose between internal time clock and external clock from FRCK • Writing “0”: Internal clock selected. • Writing “1”: External clock selected.
bit 14 to 8	Undefined	

13.3.3 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 base time for the 16-bit output compare and 16-bit input capture operations.

■ 16-bit free-running timer operation

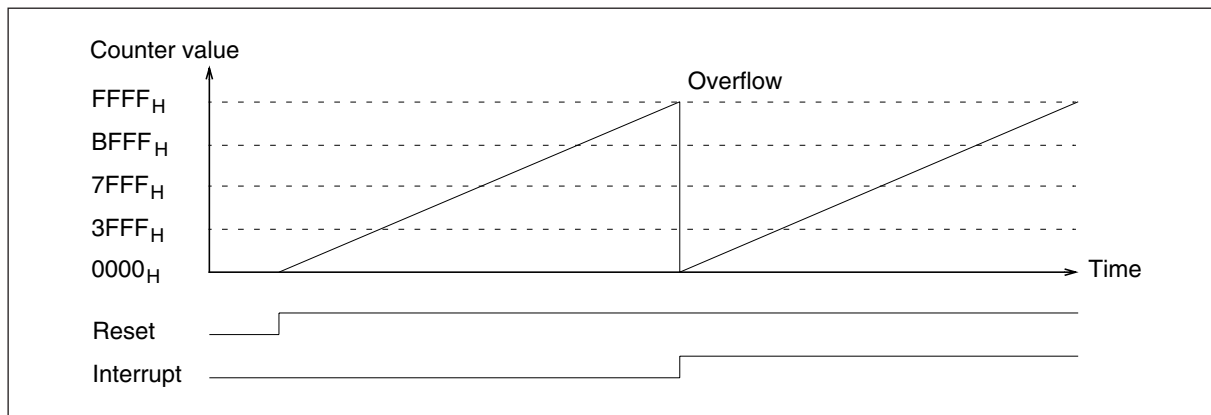
The counter value is cleared in the following conditions:

- When an overflow occurs.
- When a match with the output compare register 0 and 4 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 TCDT 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 and 4. (Compare match interrupts can be used only in an appropriate mode.)

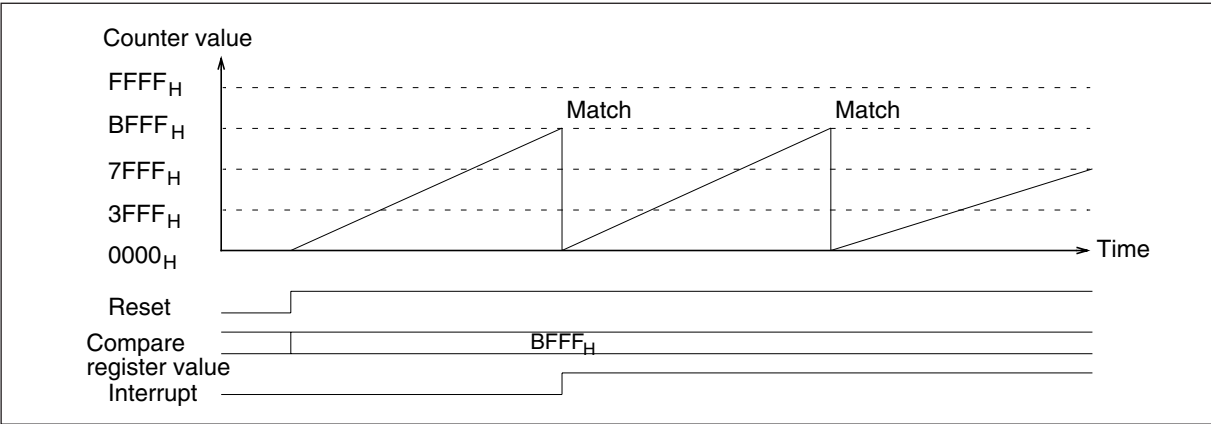
■ Clearing the counter by an overflow

Figure 13.3-5 Clearing the counter by an overflow



■ Clearing the counter upon a match with output compare register 0

Figure 13.3-6 Clearing the counter upon a match with output compare register 0

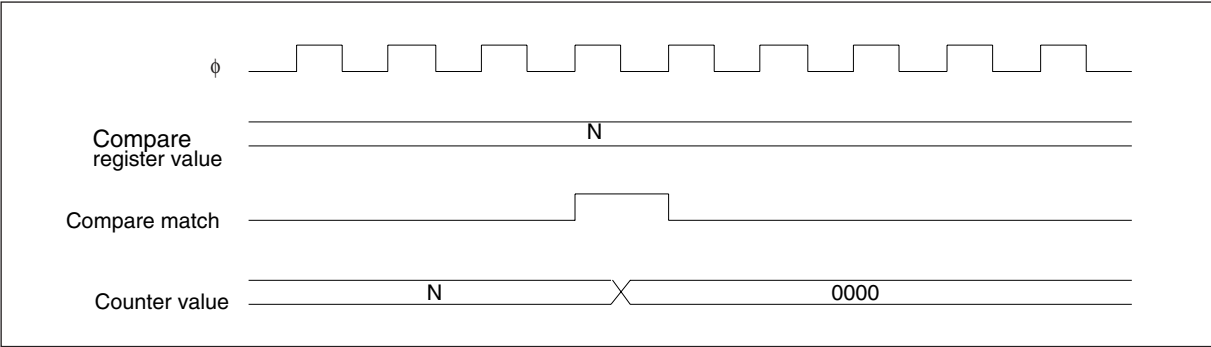


■ 16-bit free-running timer timing

● 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.

Figure 13.3-7 16-bit free-running timer clear timing (match with the compare register 0)



13.4 Output Compare

The output compare module consists of two 16-bit compare registers, two compare output pins, and one 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 toggled and an interrupt can be issued.

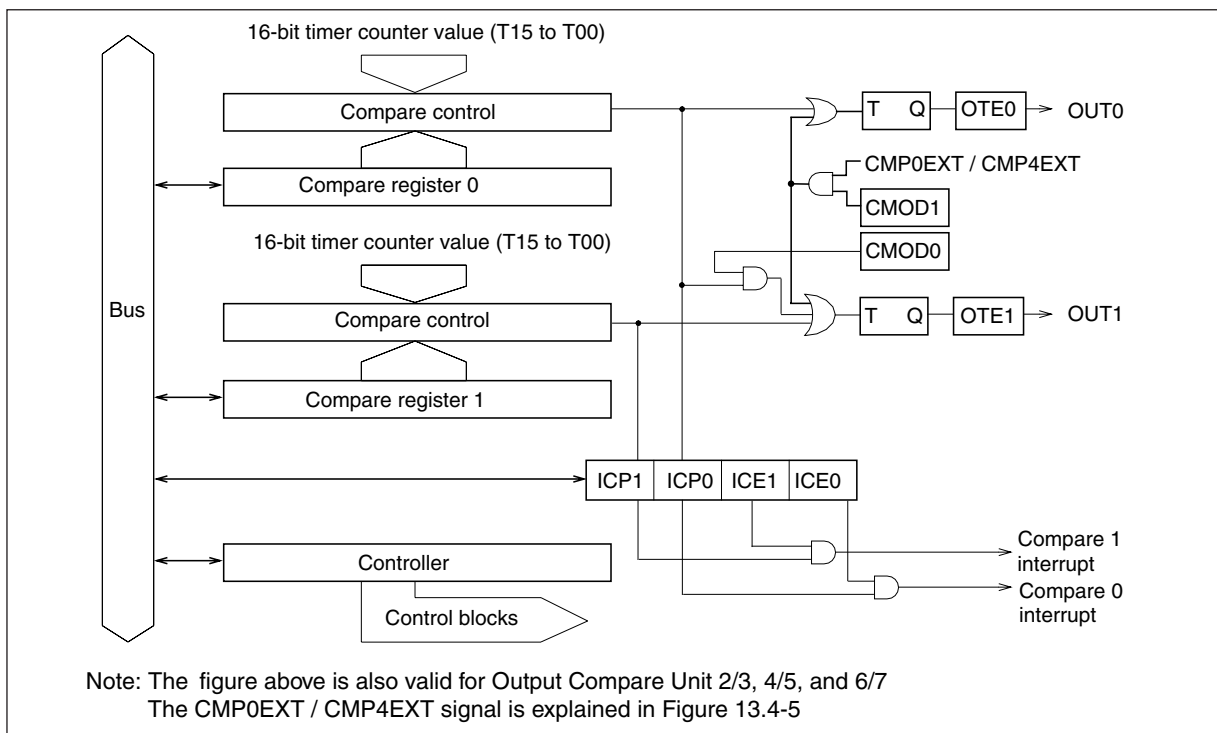
■ Output compare

- Four separate Output Compare Modules are available on MB90340 series.
- For each module, two compare registers exist which can be used independently. Depending on the mode setting, the two compare registers can be used to control pin outputs.
- The initial value for each pin output can be specified separately.
- An interrupt can be issued upon a match as a result of comparison.
- Each module can generate one PWM signal using two compare registers. (Except for OUT0 and OUT4 pins)
- Two PWM signals can be generated by one output compare module using three compare registers. (OUT2/3 and OUT6/7 pins)
- There compare registers can be used to generate one cyclic waveform signal. (OUT3 and OUT7 pins)

■ Output compare block diagram

Figure 13.4-1 “Output compare block diagram” shows a block diagram of output compare.

Figure 13.4-1 Output compare block diagram



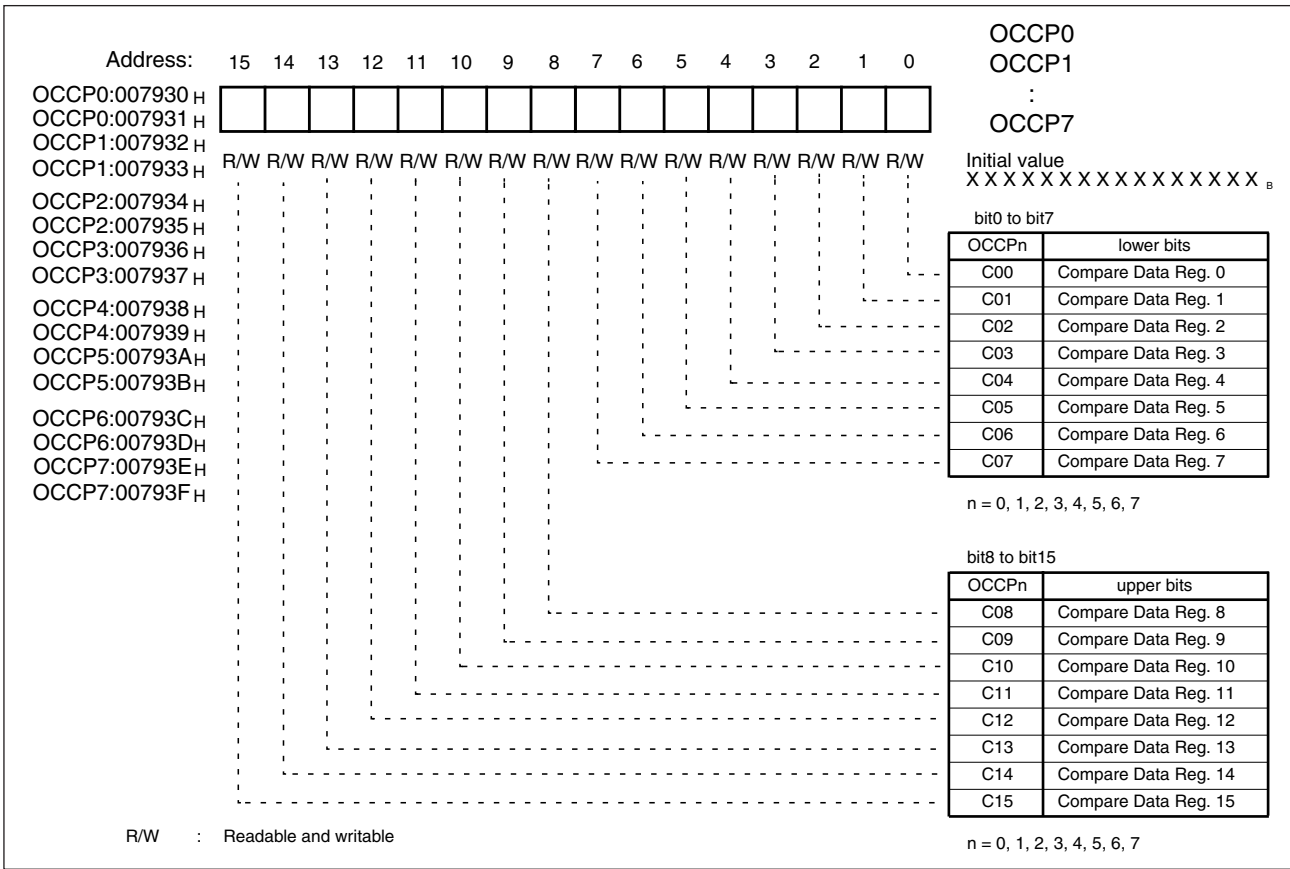
13.4.1 Output Compare Register

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.

If a compare match coincides with write operation to this register, resulting compare operation is not predictable. Therefore make sure to evaluate the counter value of the free running timer or to initialize the timer before writing to this register.

■ Output compare register

Figure 13.4-2 Output compare register (OCCP)



13.4.2 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 (lower byte)

Figure 13.4-3 Control status register of output compare (OCS0/2/4/6)

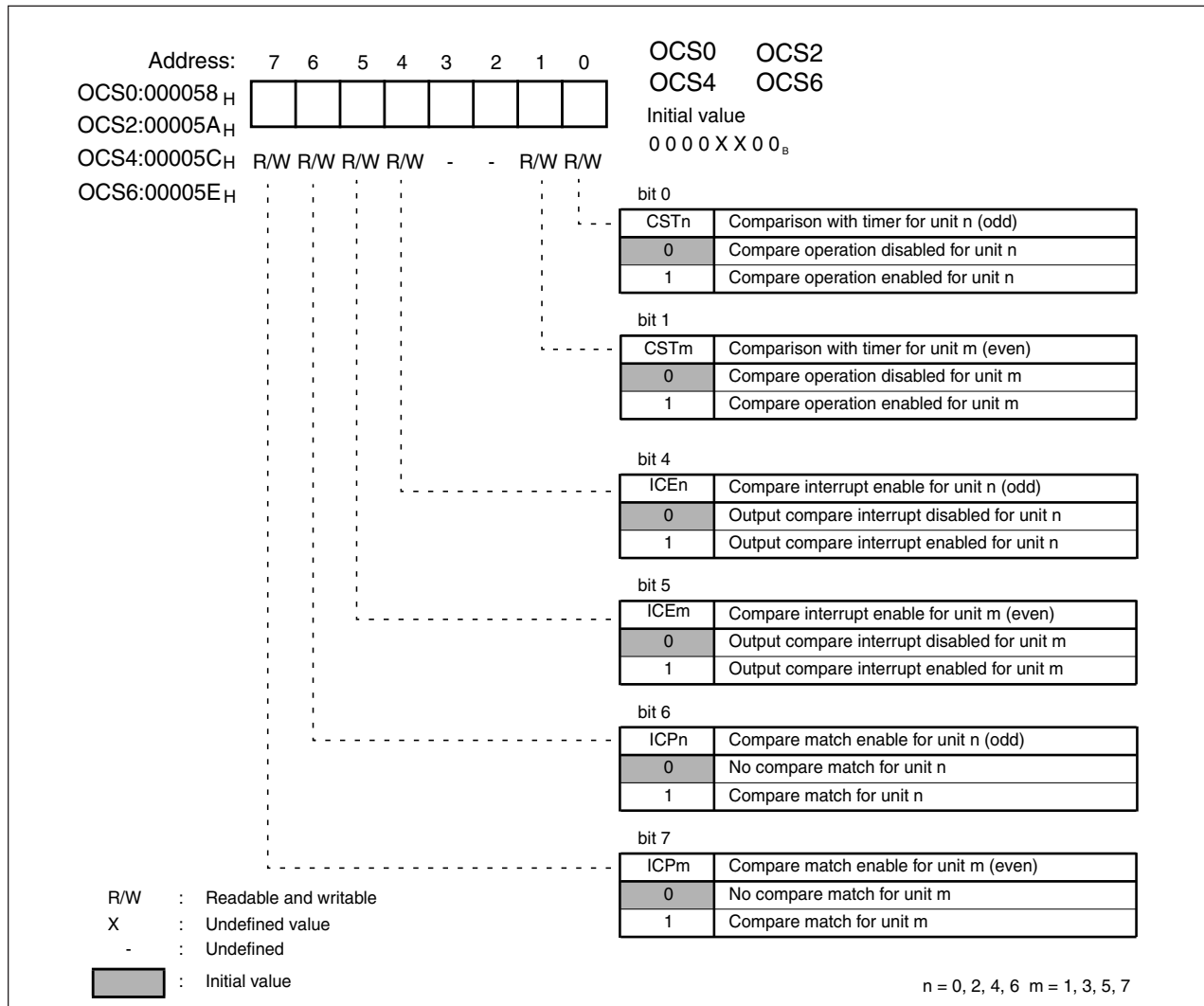


Table 13.4-1 Control status register of output compare (lower)

Bit name		Function
bit 7	ICPm	- 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-run timer value. While the interrupt request bits (ICEm and ICEn) are enabled, an output compare interrupt occurs when the ICPm and ICPn bits are set. These bits are cleared by writing "0". "0": No compare match. "1": Compare match. - Writing "1" has no effect. "1" is always read by a read-modify-write instruction. Note: ICPn: Corresponds to output compare n (odd). ICPm: Corresponds to output compare m (even).
bit 6	ICPn	
bit 5	ICEm	- 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 (ICPm or ICPn) is set. - Writing "0": Output compare interrupt disabled. - Writing "1": Output compare interrupt enabled. Note: ICEn: Corresponds to output compare unit n (odd). ICEm: Corresponds to output compare unit m (even).
bit 4	ICEn	
bit 3 to 2	Undefined	
bit 1	CSTm	- These bits are used to enable the compare operation - Writing "0": Compare operation disabled. - Writing "1": Compare operation enabled. Note: Ensure that a value is written to the compare register before the compare operation is enabled. CSTn: Corresponds to output compare n (odd). CSTm: Corresponds to output compare m (even). Since output compare is synchronized with the 16-bit free-running timer clock, stopping the 16-bit free-running timer stops compare operation.
bit 0	CSTn	

n = 0, 2, 4, 6 m = 1, 3, 5, 7

■ Control status register of output compare (upper byte)

Figure 13.4-4 Control status register of output compare (OCS1/3/5/7)

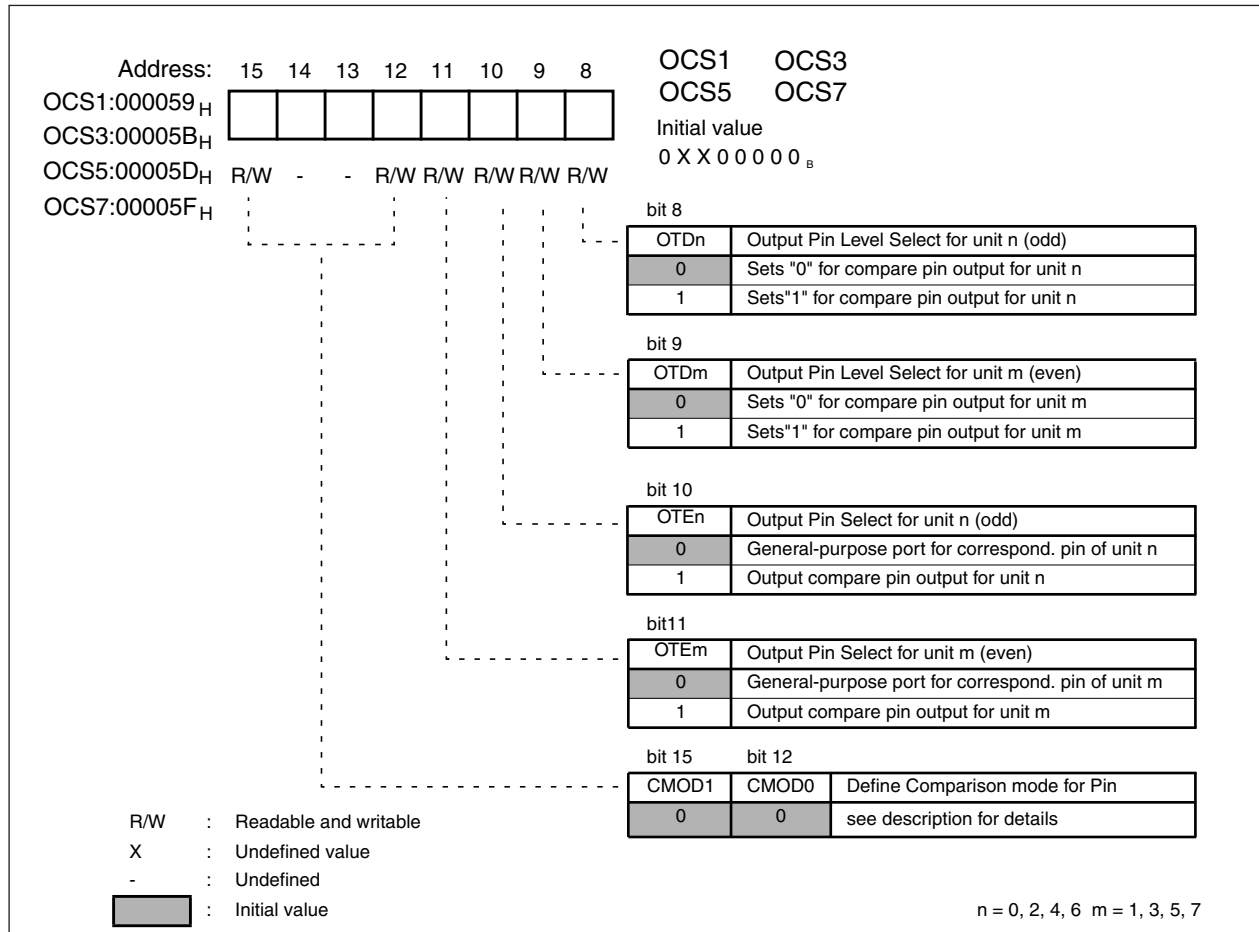


Table 13.4-2 Control status register of output compare (upper) (1/2)

Bit name		Function
bit 15 and 12	CMOD0, CMOD1	These bits define the operation mode for the pin output signals. Depending on the defined mode, the level is toggled upon a match with different compare registers. See Table 13.4-3 "Function of CMOD1 and 0 bits" and Section 13.4.3 "16-bit Output Compare Operation" for details.
bit 14, 13	Undefined	
bit 11	OTEm	These bits are used to enable the output compare output pins. The initial value for these bits is "0". "0": General-purpose port. "1": Output compare pin output. Note: OTE _n : Corresponds to OUT _n (n is a odd number). OTE _m : Corresponds to OUT _m (m is an even number). When they are specified as outputs, the corresponding bits of the Port Direction Registers should also be set to "1".
bit 10	OTEn	

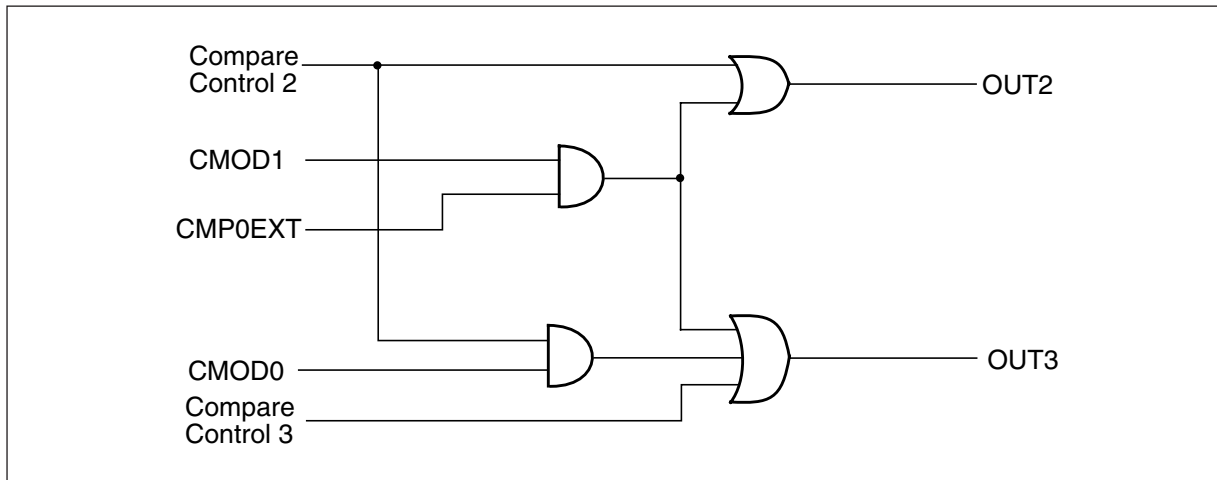
Table 13.4-2 Control status register of output compare (upper) (2/2)

Bit name		Function
bit 9	OTDm	These bits are used to change the pin output level when the 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. - Writing "0": Sets "0" for compare pin output. - Writing "1": Sets "1" for compare pin output. Note: OTDn: Corresponds to output compare n (odd). OTDm: Corresponds to output compare m (even).
bit 8	OTDn	

n = 0, 2, 4, 6 m = 1, 3, 5, 7

Table 13.4-3 Function of CMOD1 and 0 bits

Pin output value toggled upon match with compare register number.			
OCS1		Register OCCPx	
CMOD1	CMOD0	OUT0	OUT1
x	0	0	1
x	1	0	0/1
OCS3		Register OCCPx	
CMOD1	CMOD0	OUT2	OUT3
0	0	2	3
0	1	2	2/3
1	0	0/2	0/3
1	1	0/2	0/2/3
OCS5		Register OCCPx	
CMOD1	CMOD0	OUT4	OUT5
x	0	4	5
x	1	4	4/5
OCS7		Register OCCPx	
CMOD1	CMOD0	OUT6	OUT7
0	0	6	7
0	1	6	6/7
1	0	4/6	4/7
1	1	4/6	4/6/7

Figure 13.4-5 Block diagram of output selection (OCU module 1)

For OCU module 1, which requires a match with Output Compare Register 0 if $CMOD[1:0] = "10_B"$ the comparison result from module 0 is carried inside by the $CMP0EXT$ signal. Of course, this does not apply to module 0 itself. Here, no other register can be used but $OCCP0$ and $OCCP1$.

The equivalent situation applies to OCU module 3, where the result from module 2 is needed as $CMP4EXT$.

13.4.3 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 toggled when the specified compare register value matches the 16-bit free-run timer value. The CMOD0 and CMOD1 bits can be used to define the corresponding compare registers for each pin.

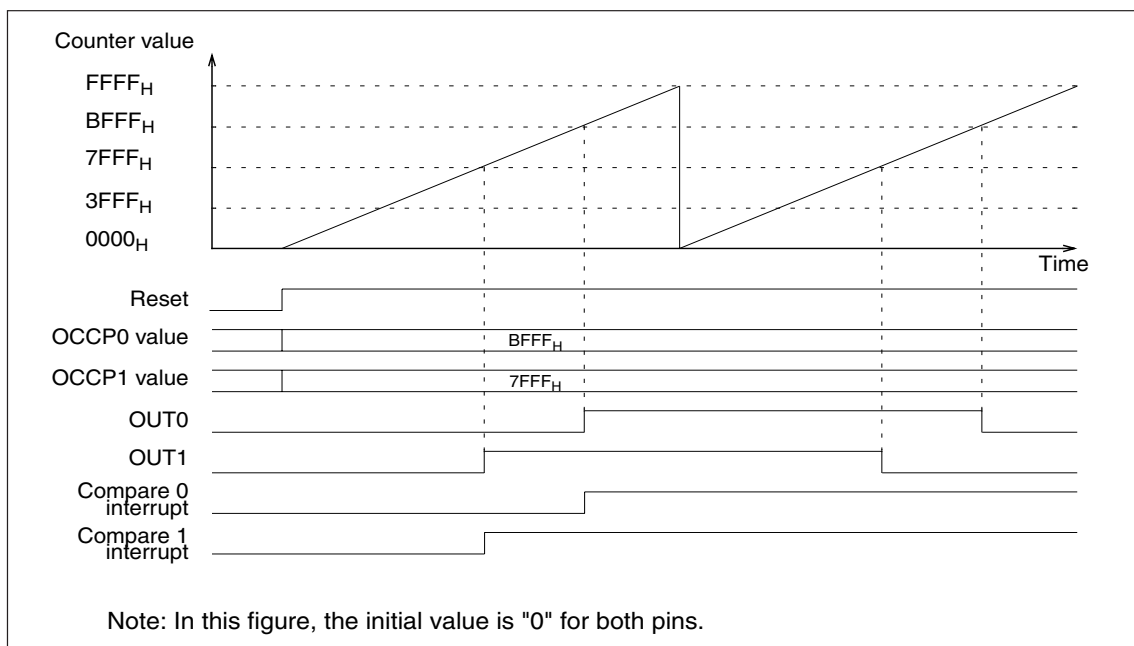
■ Sample output waveform when CMOD[1:0] = "00_B"

When CMOD[1:0] = "00_B", the output level of the pin corresponding to the compare register is reversed on every match with the register value. Each output value is controlled by one compare register.

OUT0: The level is only reversed by a match with compare register 0.

OUT1: The level is only reversed by a match with compare register 1.

Figure 13.4-6 Sample of output waveform when CMOD[1:0] = "00_B"



■ Sample output waveform with two compare registers when CMOD[1:0] = "01_B"

When CMOD[1:0] = "01_B", the output level of the pin corresponding to compare register 0 (2) is reversed upon every match with the register value. This is identical to the behavior for CMOD[1:0] = "00_B". However, the output level of the second pin is reversed upon a match with either compare register 0 or compare register 1 (3). This allows to define a pulsed signal with one edge defined by the value in compare register 0 and the other edge defined by compare register 1 (3) or vice versa. If both compare registers have the same value, the operation is identical to the case for CMOD[1:0] = "00_B".

A pulse width modulated signal with differing frequency can be defined by using this mode together with the reset option by compare register match for the Free-running timer (MODE-bit in TCCSL0/1 registers).

OUT0 (2): The level is only reversed by a match with compare register 0 (2).

OUT1 (3): The level is reversed by a match with compare register 0 (2) or with compare register 1 (3).

For OUT4, OUT5, OUT6 and OUT7, compare register 4 plays the same role as compare register 0 above.

Figure 13.4-7 Sample of a output waveform when CMOD[1:0] = "01_B" (no timer reset by match)

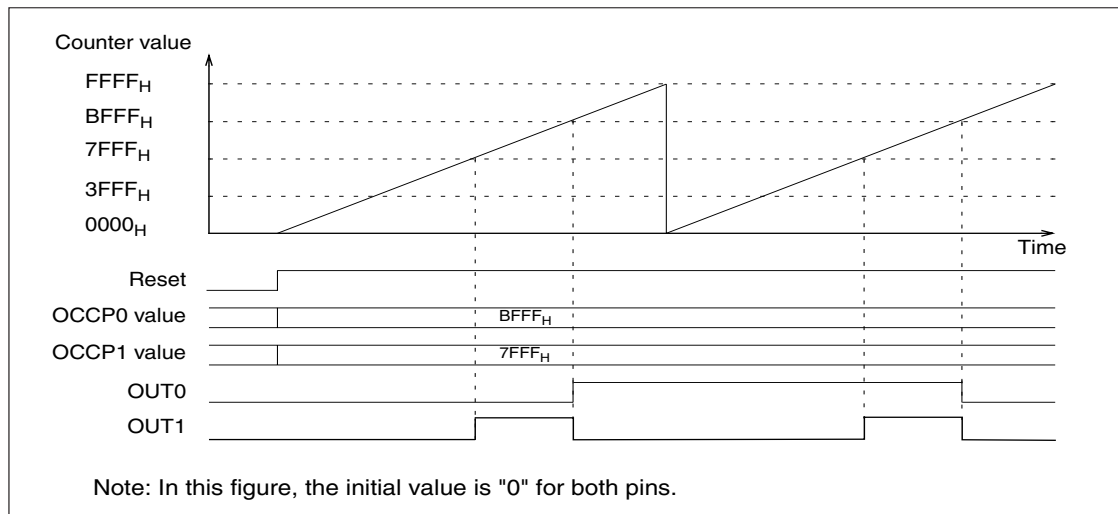
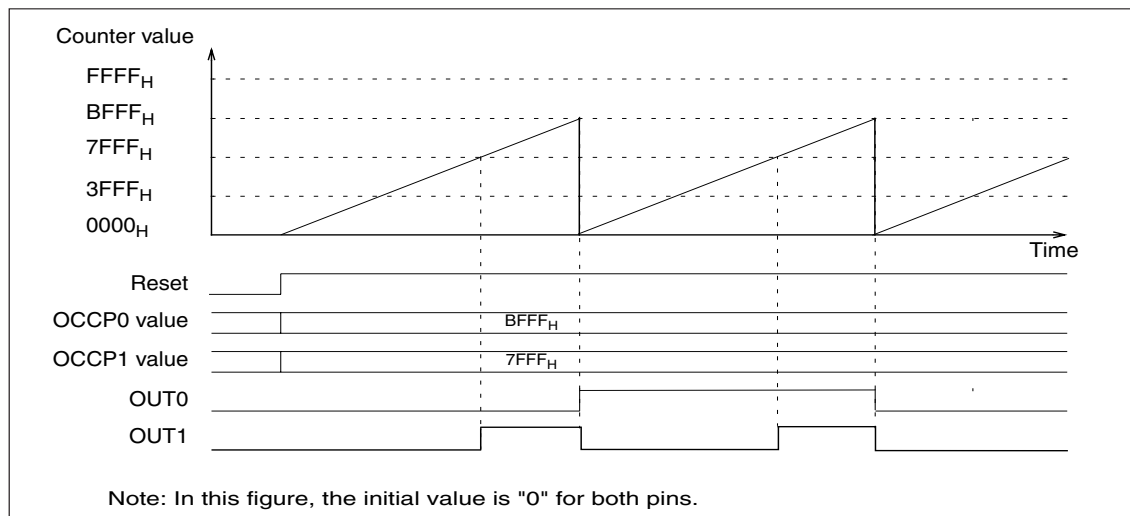


Figure 13.4-8 Sample of a output waveform when CMOD[1:0] = "01_B" (with timer reset by match)



■ Sample output waveform when CMOD[1:0] = "10_B"

The operation mode defined by CMOD[1:0] = "10_B" is intended for the use of three pulse width modulated signals for each free-running timer instead of two. If this mode is set to OCU module 1, a match of the timer value with compare register 0 reverses both OUT2 and OUT3. For the third pulsed signal, the CMOD[1:0] bits of OCU module 0 should be set to "01_B".

In register OCS1: CMOD[1:0] = "01_B"

OUT0: The level is only reversed by a match with compare register 0.

OUT1: The level is reversed by a match with compare register 0 or with compare register 1.

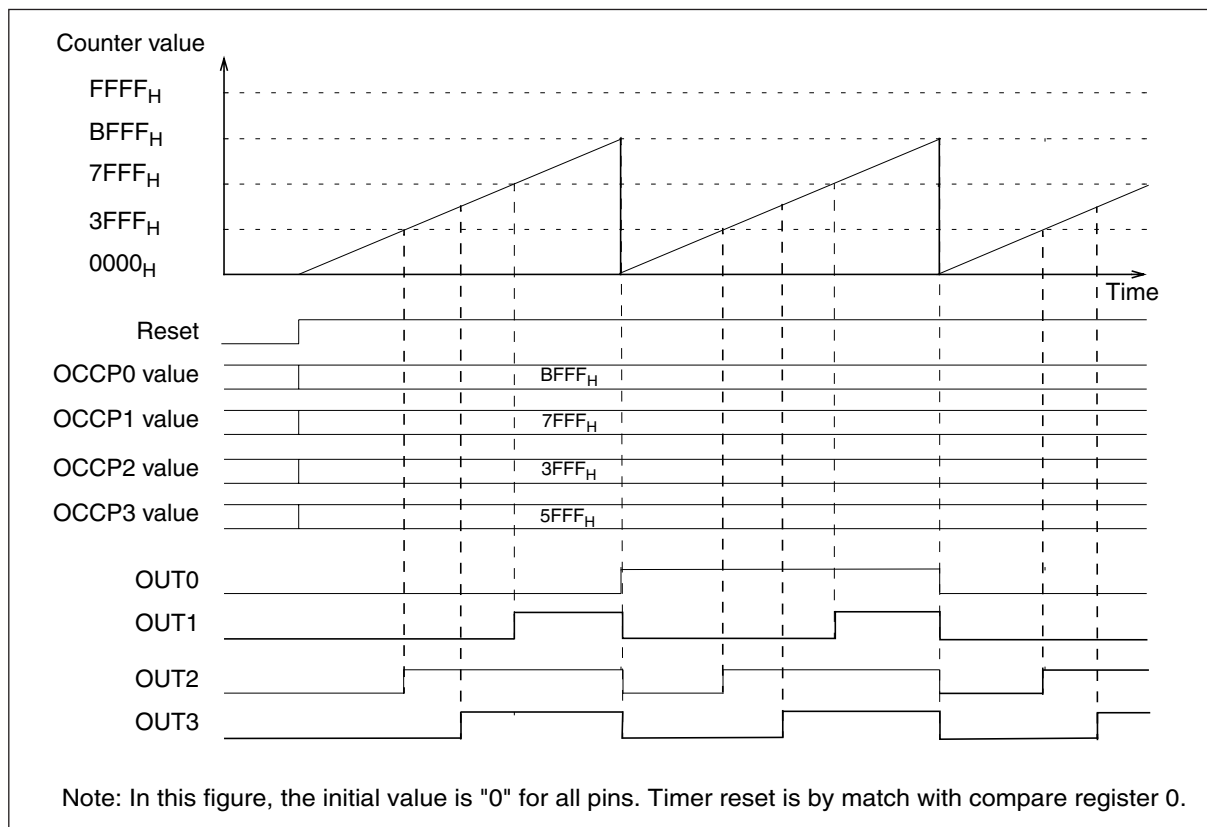
In register OCS3: CMOD[1:0] = "10_B"

OUT2: The level is reversed by a match with compare register 0 or with compare register 2.

OUT3: The level is reversed by a match with compare register 0 or with compare register 3.

For OUT4, OUT5, OUT6 and OUT7, compare register 4 plays the same role as compare register 0 above.

Figure 13.4-9 Output waveform when OCS1.CMOD[1:0] = "01_B" and OCS3.CMOD[1:0] = "10_B"



■ Sample output waveform when CMOD[1:0] = "11_B"

When CMOD[1:0] = "11_B", the output level of the OUT3 (OUT7) pin is reversed by the compare registers 0, 2 or 3 (4, 6 or 7). For the pin OUT1 (OUT5), this setting is identical to CMOD[1:0] = "01_B" (see also Table 13.4-3 "Function of CMOD1 and 0 bits")

OUT0: The level is only reversed by a match with compare register 0.

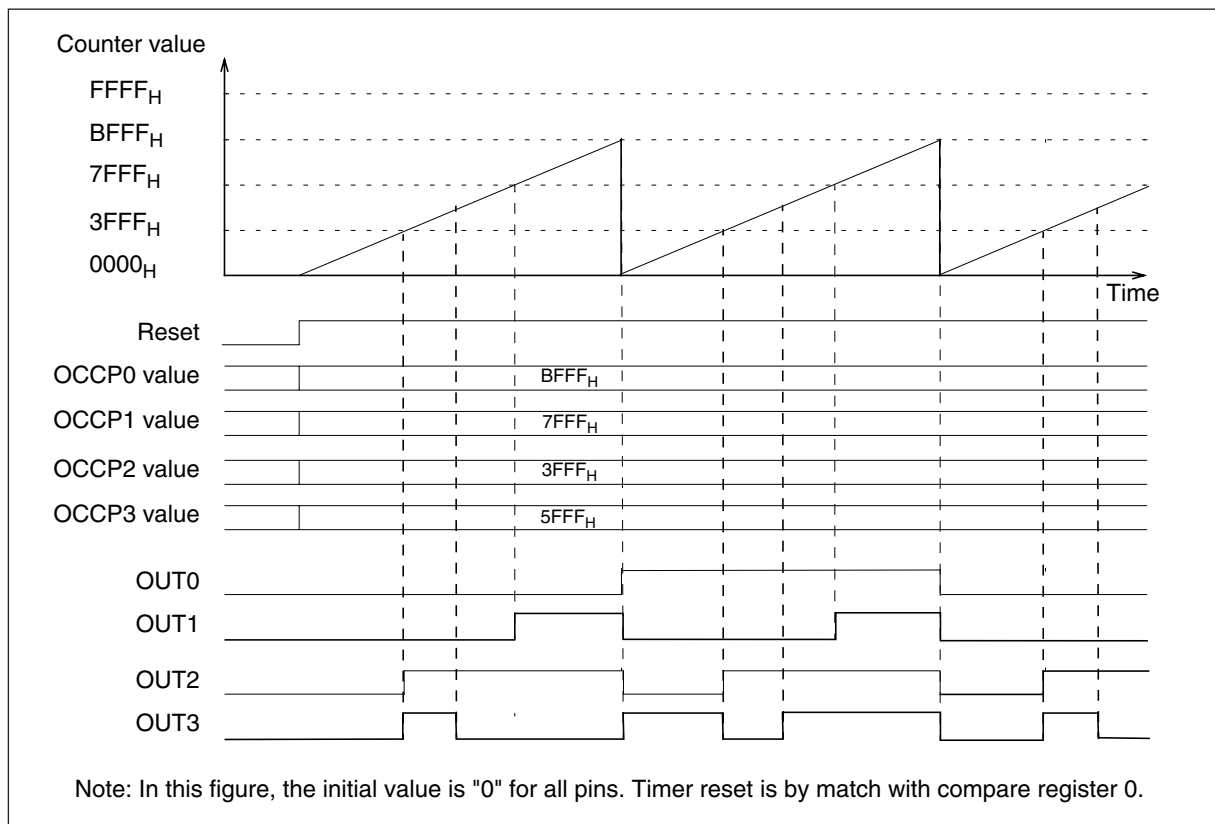
OUT1: The level is reversed by a match with compare register 0 or with compare register 1.

OUT2: The level is reversed by a match with compare register 0 or with compare register 2.

OUT3: The level is reversed by a match with compare register 0, compare register 2 or with compare register 3.

For OUT4, OUT5, OUT6 and OUT7, compare register 4 plays the same role as compare register 0 above.

Figure 13.4-10 Output waveform when OCS1.CMOD[1:0] = "11_B" and OCS3.CMOD[1:0] = "11_B"



■ Output compare timing

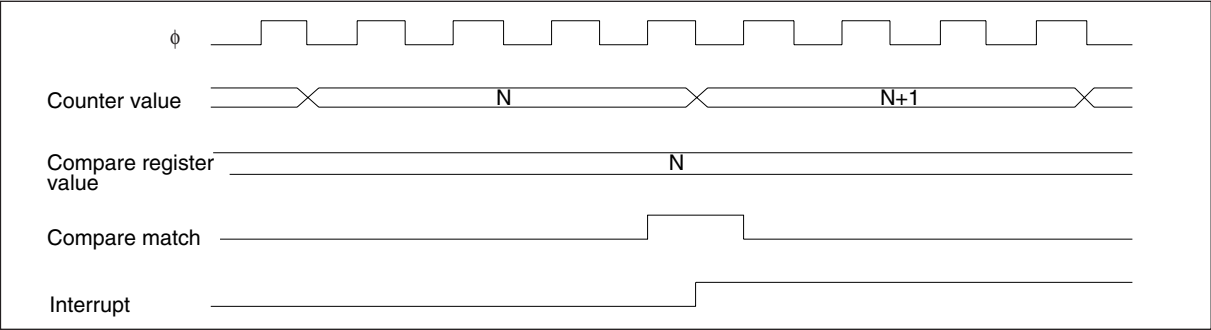
In output compare operation, a compare match signal is generated when the free-running timer value matches the specified compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing upon a compare match is synchronized with the counter timing.

● Compare operation upon update of compare register

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

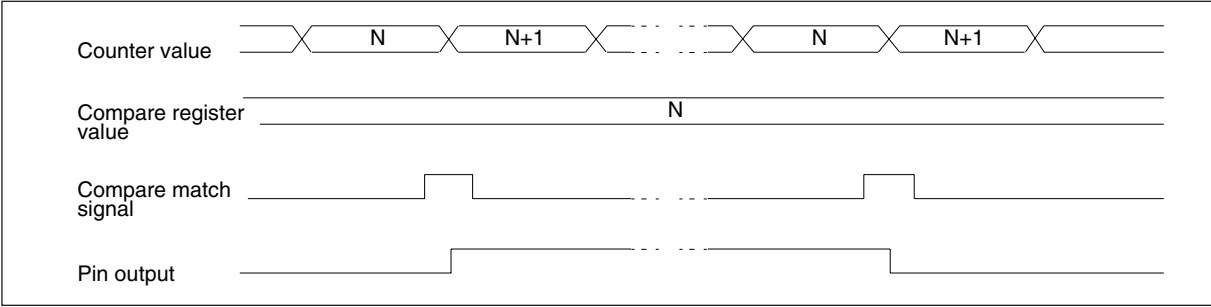
● Interrupt timing

Figure 13.4-11 Interrupt timing



● Output pin change timing

Figure 13.4-12 Output pin change timing



13.5 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. One input capture module consists of two input capture registers and one control register.

■ Input capture

MB90340 Series has four input capture modules.

Each input capture data register has a corresponding external input pin.

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

Table 13.5-1 Types of external input edges

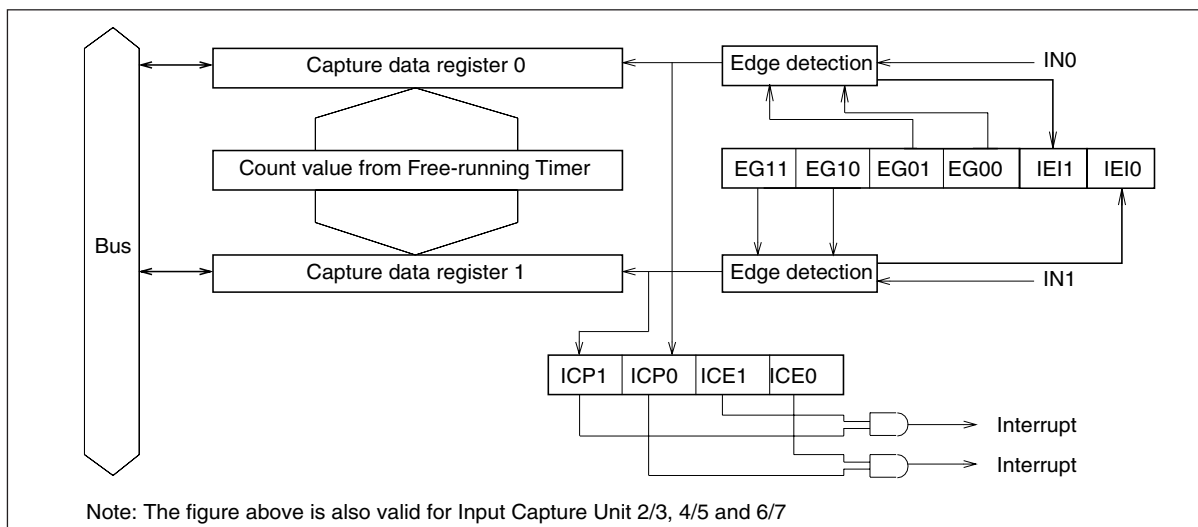
Rising edge	↑
Falling edge	↓
Both edges	↑↓

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

■ Input capture block diagram

Figure 13.5-1 “Input Capture Block Diagram” shows a block diagram of input capture.

Figure 13.5-1 Input capture block diagram



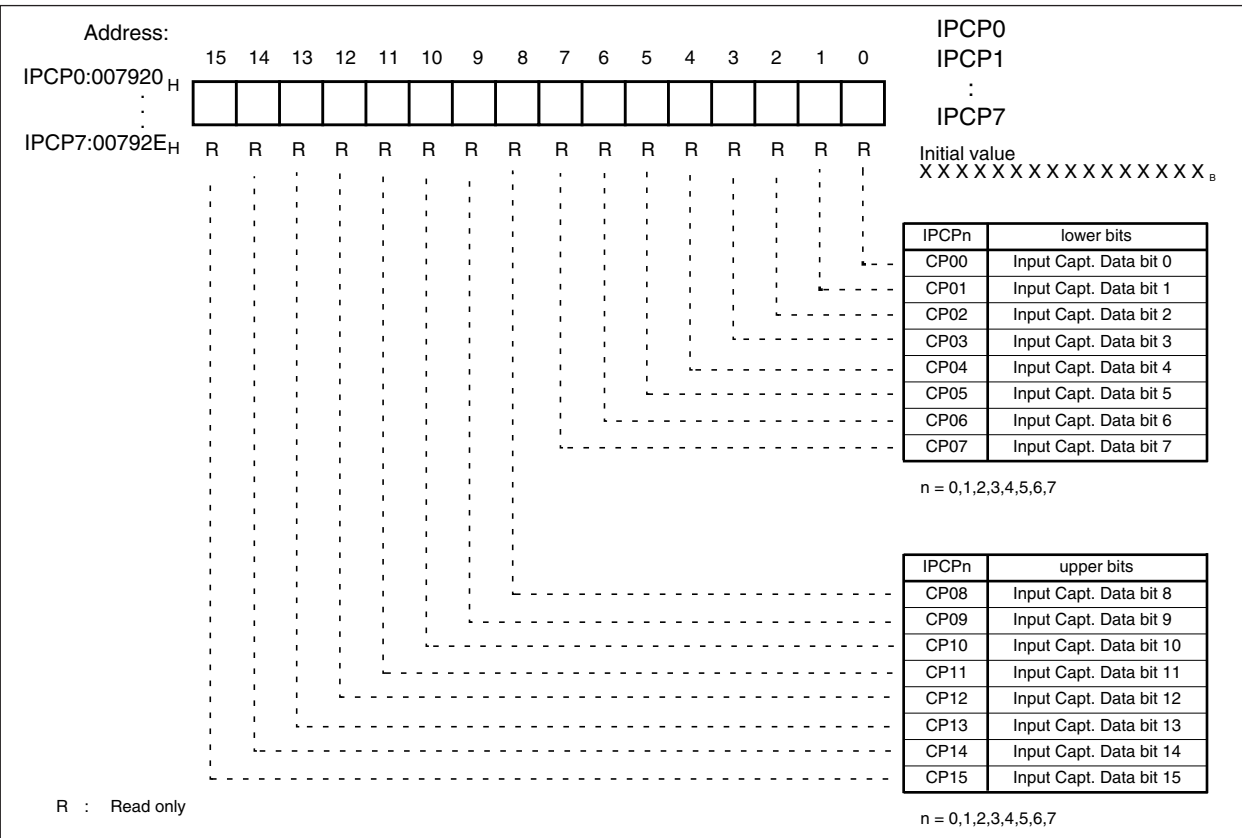
13.5.1 Input Capture Register Details

Input capture module has the two data 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

Figure 13.5-2 Input capture data register (IPCP)



■ Control status register

Figure 13.5-3 Control status register (ICS01,ICS23,ICS45,ICS67)

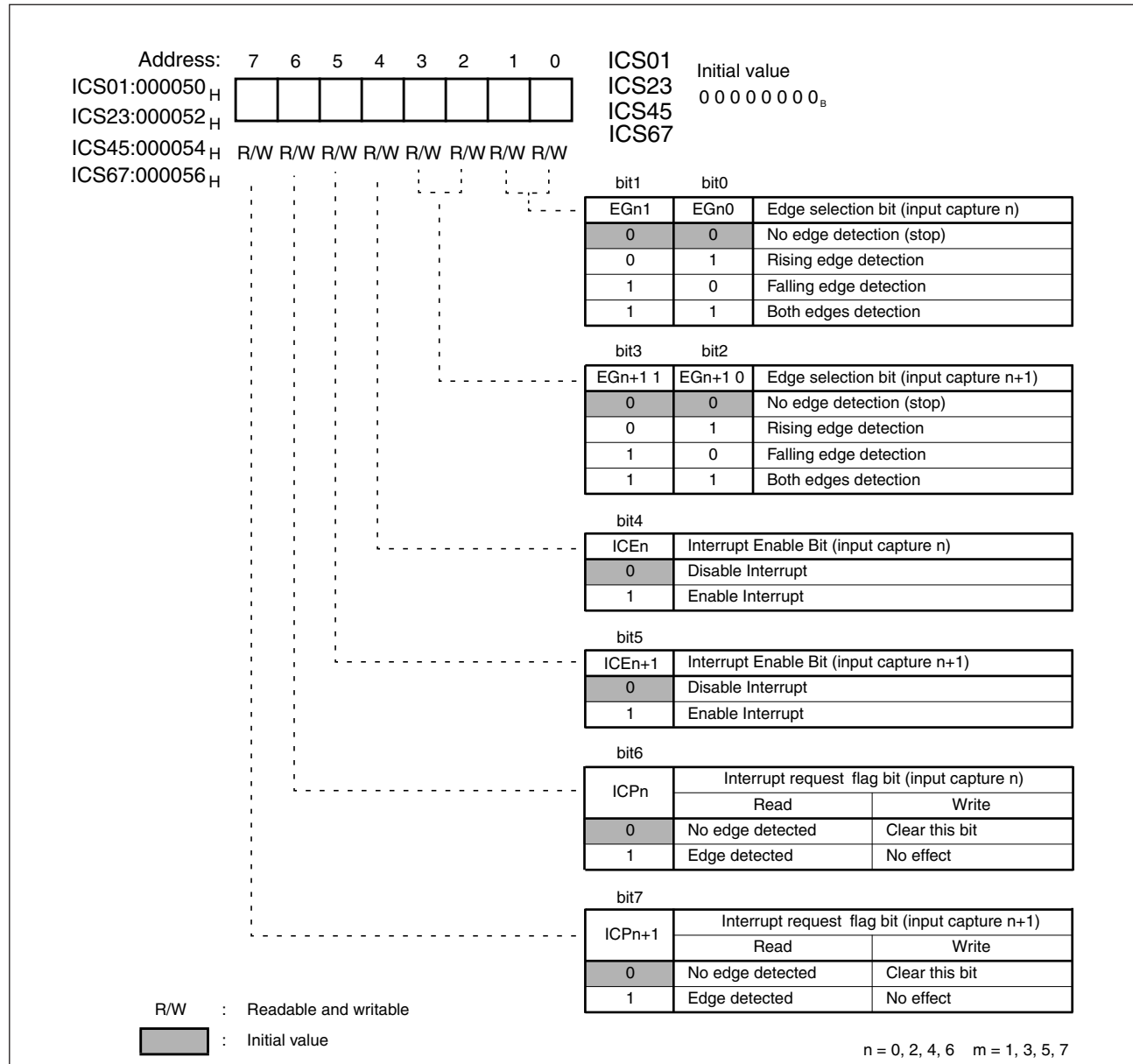


Table 13.5-2 Input capture control status register bits

Bit name		Function
bit7	ICPn+1: Interrupt request flag bit (Input capture n+1)	• This bit is used as interrupt request flag for input capture n. • "1" is set to this bit upon detection of a valid edge of an external input pin. • While the interrupt enable bit (ICEn+1) is set, an interrupt can be generated upon detection of a valid edge. • Writing "0" will clear this bit. • Writing "1" has no effect. • In read-modify-write operation, "1" is always read.
bit6	ICPn: Interrupt request flag bit (Input capture n)	• This bit is used as interrupt request flag for input capture n. • "1" is set to this bit upon detection of a valid edge of an external input pin. • While the interrupt enable bit (ICEn) is set, an interrupt can be generated upon detection of a valid edge. • Writing "0" will clear this bit. • Writing "1" has no effect. • In read-modify-write operation, "1" is always read.
bit5	ICEn+1: Interrupt request enable bit (Input capture n+1)	• This bit is used to enable input capture interrupt request for input capture n+1. • While "1" is written to this bit, an input capture interrupt is generated when the interrupt flag (ICPn+1) is set.
bit4	ICEn: Interrupt request enable bit (Input capture n)	• This bit is used to enable input capture interrupt request for input capture n. • While "1" is written to this bit, an input capture interrupt is generated when the interrupt flag (ICPn) is set.
bit3/2	EG[n+1]1, EG[n+1]0	• These bits are used to specify the valid edge polarity of an external input for input capture n+1 • These bits are also used to enable input capture operation
bit1/0	EGn1, Egn0	• These bits are used to specify the valid edge polarity of an external input for input capture n • These bits are also used to enable input capture operation

n = 0, 2, 4, 6

■ Input capture edge register (ICE01, ICE23, ICE45, ICE67)

Figure 13.5-4 Input capture edge register (ICE01,23,45,67)

Address	15	14	13	12	11	10	9	8	Initial value
ICE01 : 000051 _H	-	-	-	ICUS1	-	ICUS0	IEI1	IEI0	XXX0X0XX _B
	-	-	-	R/W	-	R/W	R	R	
ICE23 : 000053 _H	-	-	-	-	-	-	IEI3	IEI2	XXXXXXXX _B
	-	-	-	-	-	-	R	R	
ICE45 : 000055 _H	-	-	-	-	-	-	IEI5	IEI4	XXXXXXXX _B
	-	-	-	-	-	-	R	R	
ICE67 : 000057 _H	-	-	-	ICUS7	ICUS61	ICUS60	IEI7	IEI6	XXX000XX _B
	-	-	-	R/W	R/W	R/W	R	R	

R/W : Readable and writable
X : Undefined

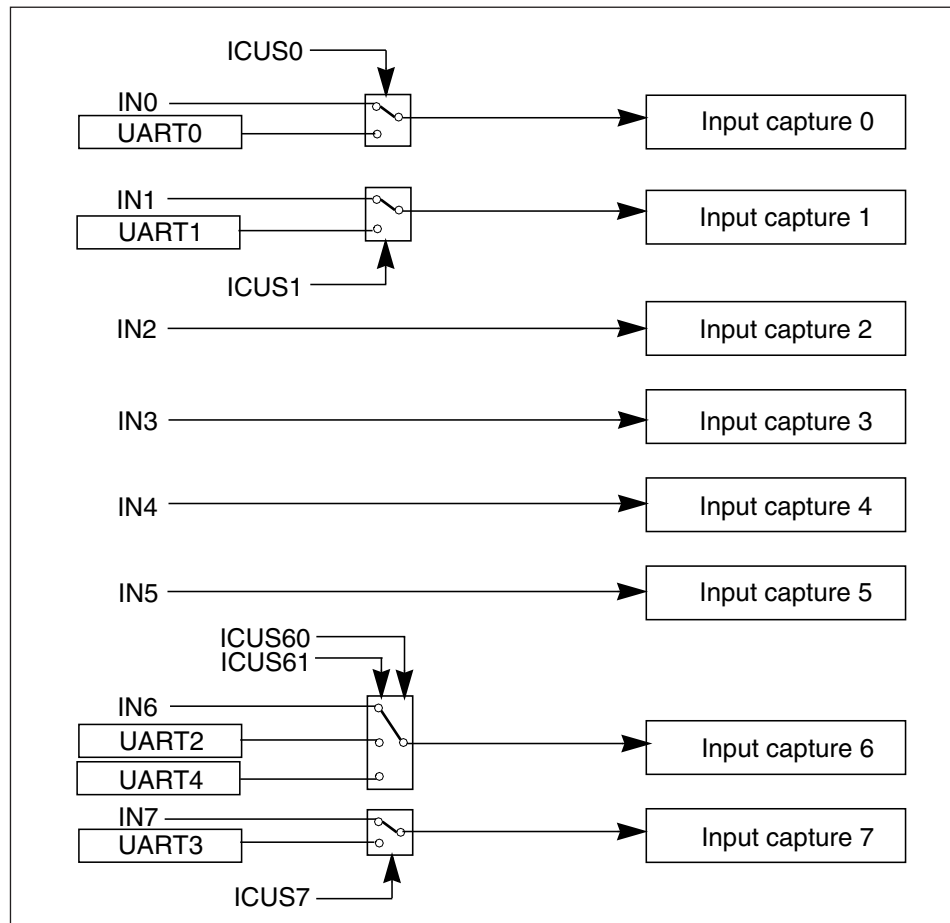
Table 13.5-3 Input capture edge register bits (upper and lower)

Bit name		Function
bit15 to 13	Undefined	
bit12	for ICE01: ICUS1 for ICE67: ICUS7	This bit defines the input signal for Input Capture1 or 7, and is used by UART1 or 3 for LIN operation: - Writing "0": The Input Capture source is external. - Writing "1": The Input Capture source is UART1 or 3.
bit11,10	for ICE67: ICUS61, ICUS60	This bit defines the input signal for Input Capture6: - ICUS61 / 60 = "00_B": Pin IN6 input signal used as ICU6 input signal. - ICUS61 / 60 = "01_B": ICU6 connected to UART2. - ICUS61 / 60 = "10_B" or "11_B": ICU6 connected to UART4.
bit10	for ICE01: ICUS0	This bit defines the input signal for Input Capture0: - ICUS0 = "0_B": Pin IN0 input signal used as ICU0 input signal. - ICUS0 = "1_B": ICU0 connected to UART0.
bit9	IEIm: Valid edge indication bit	- This bit is a valid edge indication bit for capture register IPCP1, IPCP3, IPCP5 and IPCP7 to indicate that a rising or falling edge is detected "0": falling edge detected. "1": rising edge detected. - This bit is read only. Note: The read value is meaningless, if EG [n+1] 1, EG [n+1] 0 = "00".
bit8	IEIn: Valid edge indication bit	- This bit is a valid edge indication bit for capture register IPCP0, IPCP2, IPCP4 and IPCP6 to indicate that a rising or falling edge is detected "0": falling edge detected. "1": rising edge detected. - This bit is read only. Note: The read value is meaningless, if EGn1, EGn0 = "00".

n = 0, 2, 4, 6 m = 1, 3, 5, 7

Figure 13.5-5 “Block diagram of Input Capture connection” shows input signal connection of Input Captures.

Figure 13.5-5 Block diagram of Input Capture connection



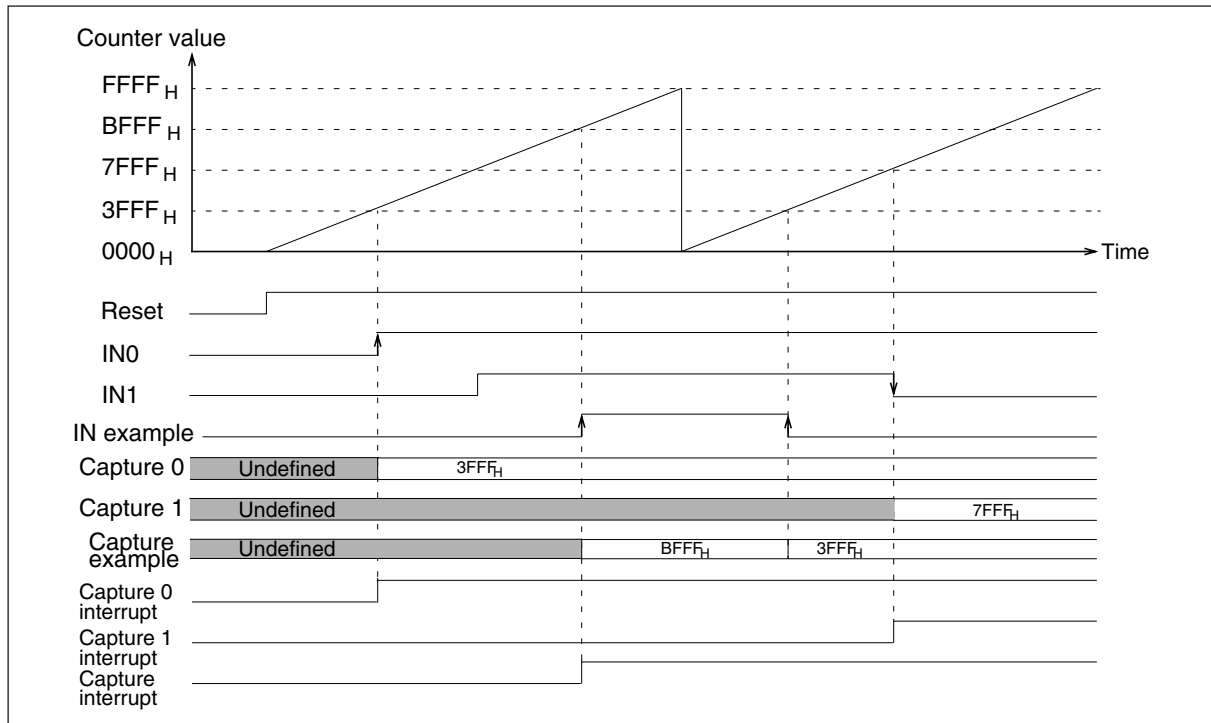
13.5.2 16-bit Input Capture Operation

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

■ Sample of input capture fetch timing

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

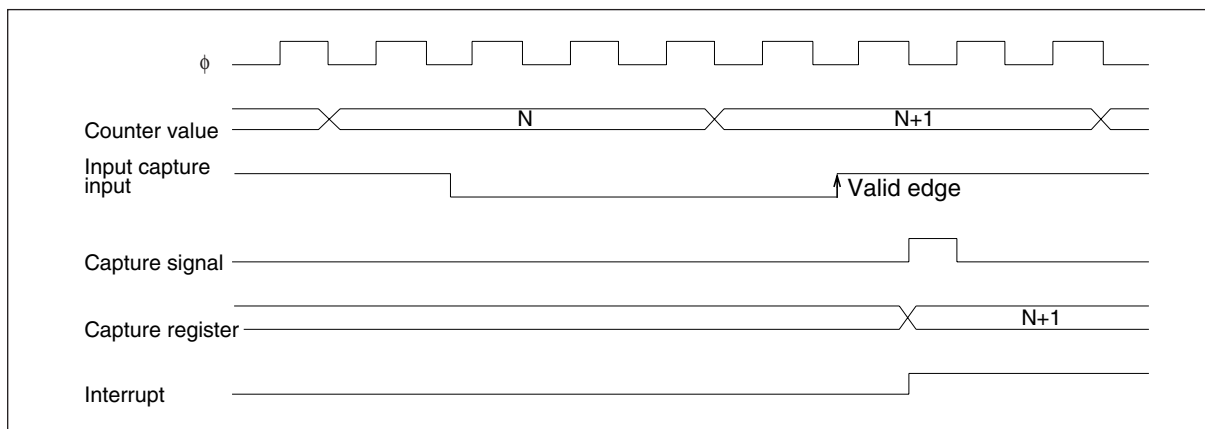
Figure 13.5-6 Sample of input capture fetch timing



■ Input capture input timing

- Capture timing for input signals

Figure 13.5-7 Capture timing for input signals



CHAPTER 14

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).

- 14.1 “Outline of 16-bit Reload Timer (with Event Count Function)”
- 14.2 “16-bit Reload Timer (with Event Count Function)”
- 14.3 “Internal Clock and External Clock Operations of 16-bit Reload Timer”
- 14.4 “Underflow Operation of 16-bit Reload Timer”
- 14.5 “Output Pin Functions of 16-bit Reload Timer”
- 14.6 “Counter Operation State”

14.1 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 MB90340 Series has four 16-bit reload timers.

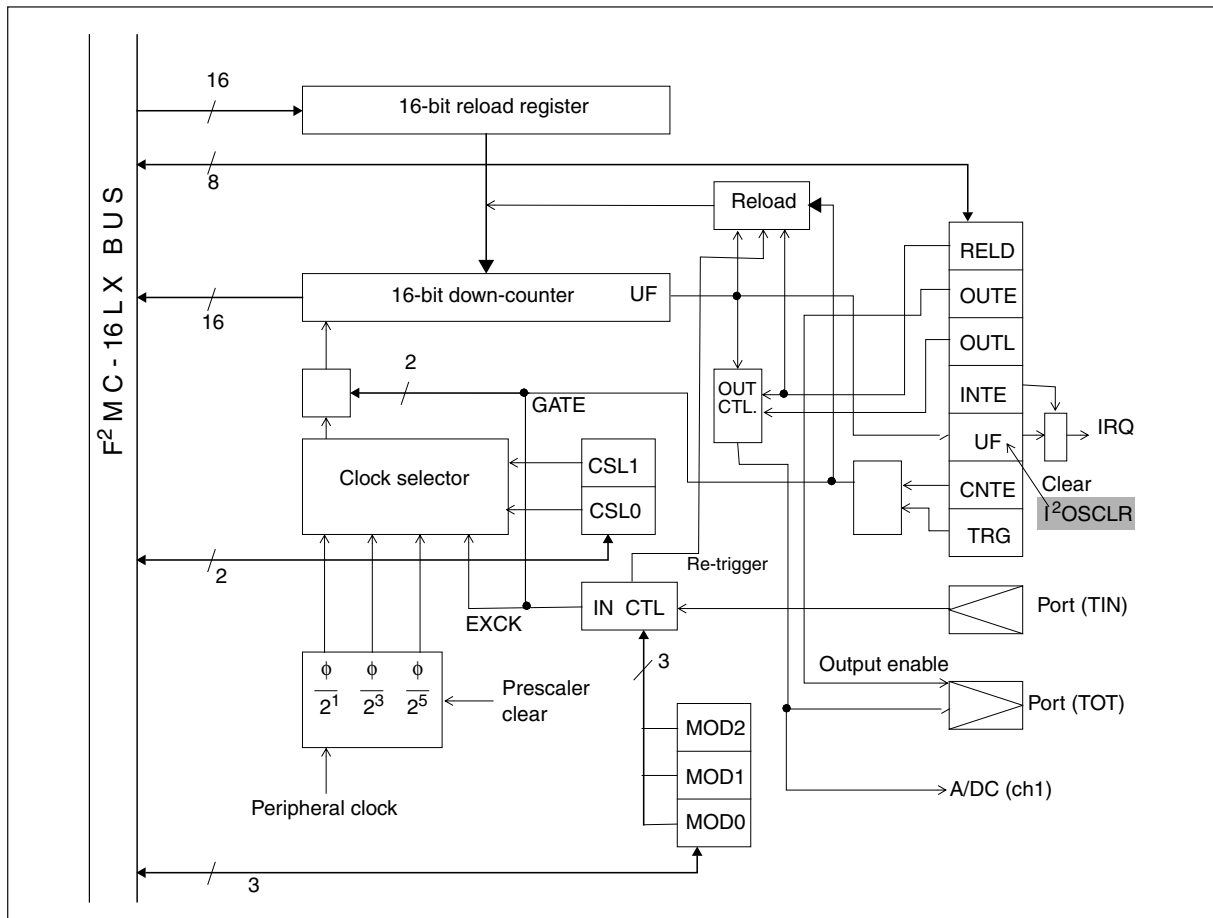
■ Intelligent I/O service (EI²OS) function, DMA and interrupts

The timer includes a circuit that supports EI²OS. The timer can activate EI²OS when an underflow occurs. EI²OS can be used with all four reload timers on this product. However, as two timers (ch0 and ch1, ch2 and ch3) are connected to the same interrupt control register (ICRx) in the interrupt controller, ch0 and ch1, ch2 and ch3 cannot be assigned to different EI²OS services. Also, as the four timers have different interrupt vectors, they can be assigned to four different interrupt services. However, as ch0 and ch1, ch2 and ch3 share an interrupt control register as described above, the same interrupt level applies to both pair of channels.

■ Block diagram of 16-bit reload timer

Figure 14.1-1 “Block diagram of 16-bit reload timer” shows a block diagram of the 16-bit reload timer.

Figure 14.1-1 Block diagram of 16-bit reload timer



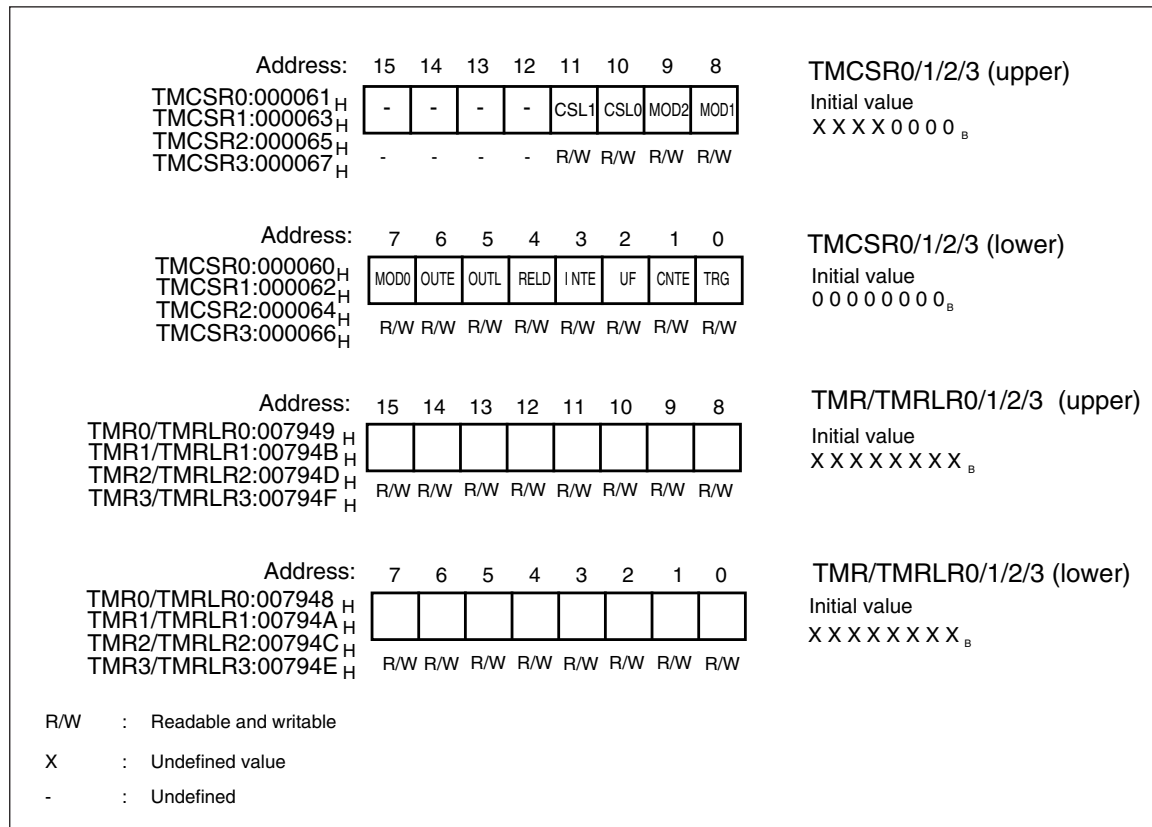
14.2 16-Bit Reload Timer (with Event Count Function)

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

Figure 14.2-1 16-bit reload timer register

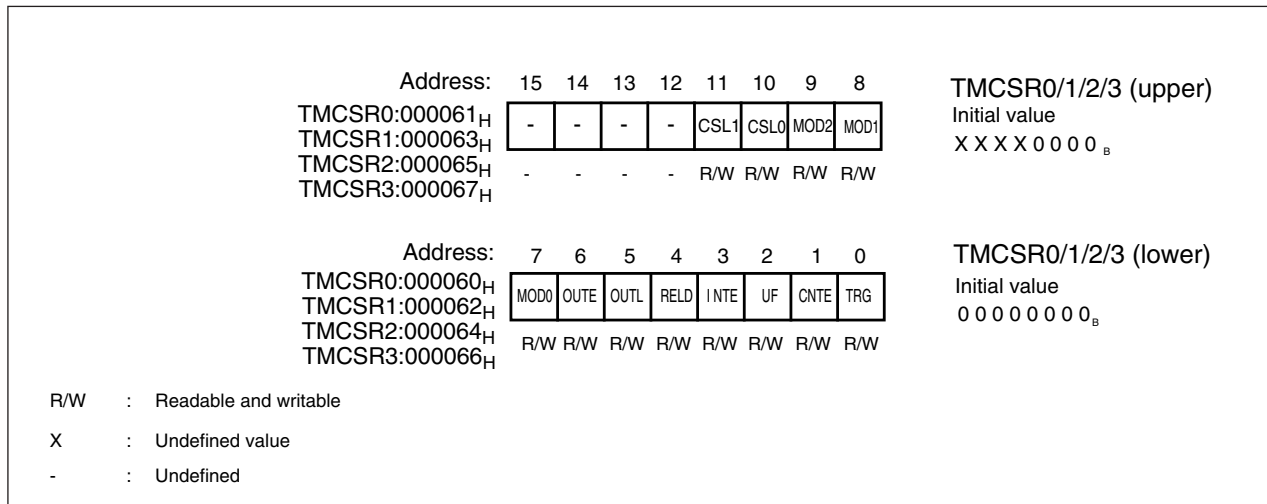


14.2.1 Timer Control Status Register (TMCSR)

Controls the operation mode and interrupts for the 16-bit timer.

■ Register layout of timer control register (TMCSR)

Figure 14.2-2 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 14.2-1 “Clock sources for CSL bit settings” lists the selected clock sources.

Table 14.2-1 Clock sources for CSL bit settings

CSL1	CSL0	Clock Source (Machine cycle ϕ = 24 MHz)
0	0	$\phi/2^1$ (0.083 μ s)
0	1	$\phi/2^3$ (0.33 μ s)
1	0	$\phi/2^5$ (1.3 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 14.2-2 "MOD2, 1, 0 bit settings (1)" and Table 14.2-3 "MOD2, 1, 0 bit settings (2)" list the MOD2, 1, 0 bit settings.

Table 14.2-2 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	x	0	Gate input	"L" level
1	x	1	↑	"H" level

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

Table 14.2-3 MOD2, 1, 0 bit settings (2)

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

- Event counter mode (CSL0, 1 = "11_B")
- Bits marked as x 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 14.2-4 OUTE, RELD, and OUTL settings

OUTE	RELD	OUTL	Output Waveform
0	x	x	General-purpose port
1	0	0	Output an "H" level pulse during counting.
1	0	1	Output an "L" level pulse 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".

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

- TMR contents (for reading)

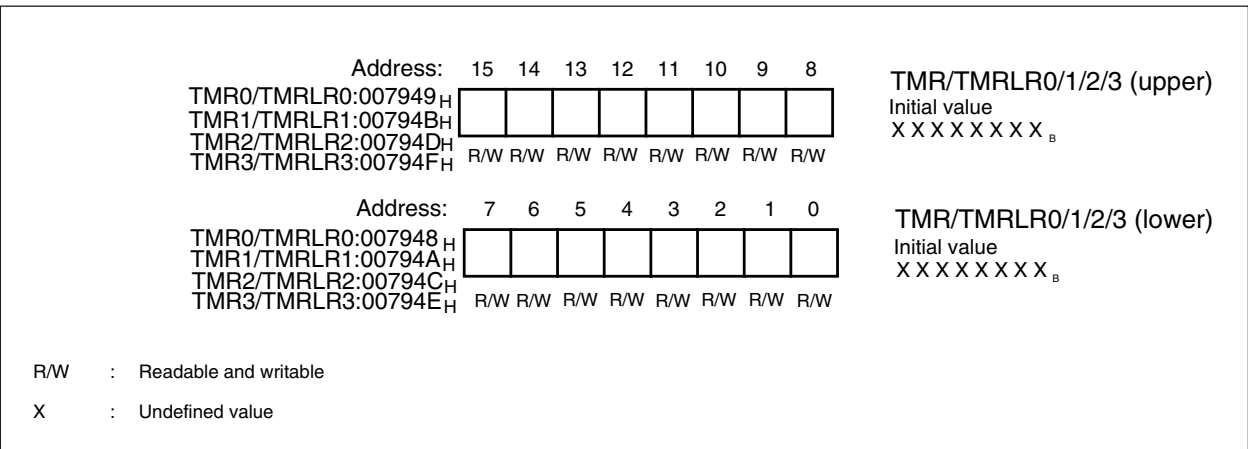
Reading this register reads the count value of the 16-bit timer. The initial value is undefined. Always read this register using the word access instructions.

- TMRLR contents (for writing)

The 16-bit reload register holds the reload 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)

Figure 14.2-3 Register layout of 16-bit timer register (TMR)/16-bit reload register (TMRLR)



14.3 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.

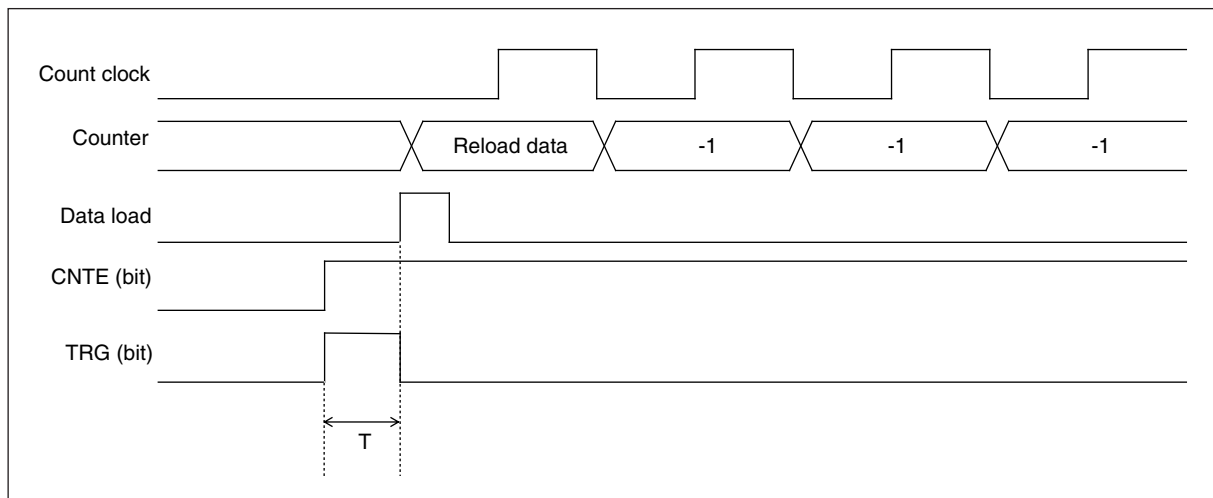
If an external clock is selected, the TIN pin functions as an external event input pin to count the number of valid edges set in the register.

■ 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 14.3-1 "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 14.3-1 Activation and operation of 16-bit reload timer counter

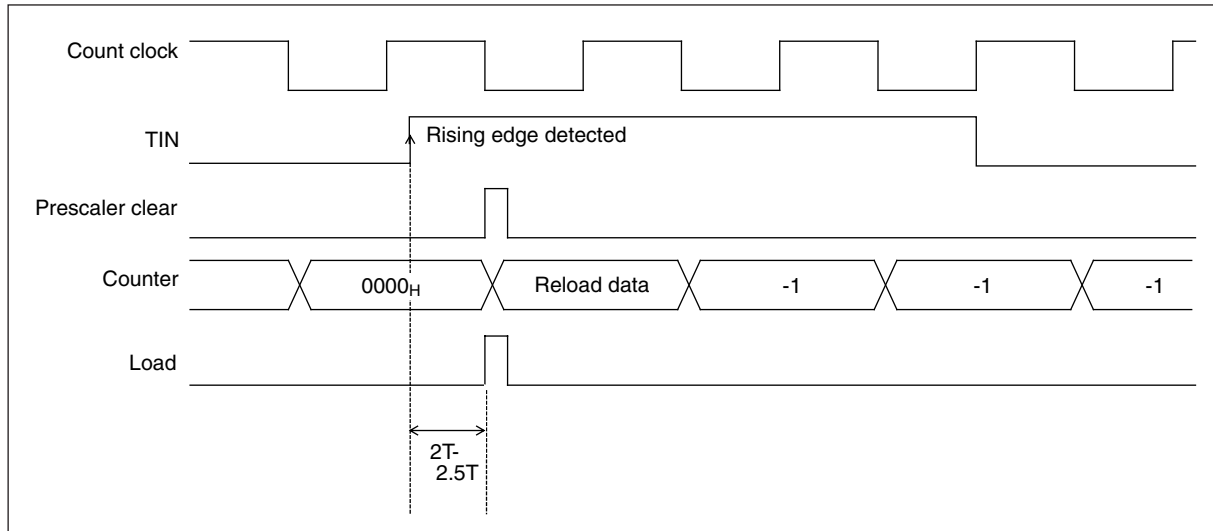


■ Input pin functions of 16-bit reload timer (in internal clock mode)

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.

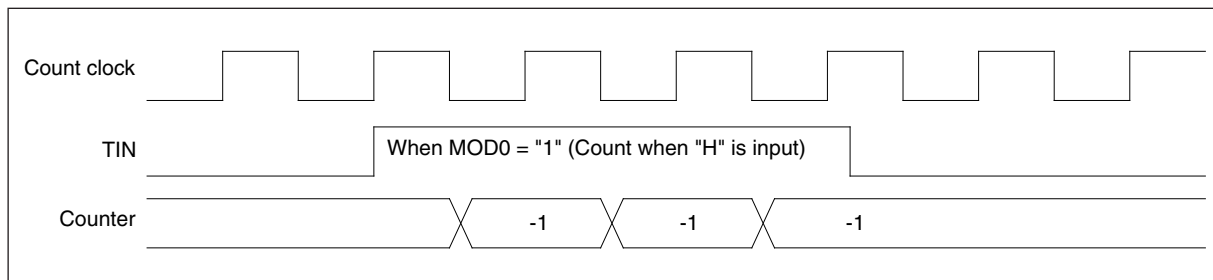
Figure 14.3-2 “Trigger input operation of 16-bit reload timer” shows the operation of trigger input.

Figure 14.3-2 Trigger input operation of 16-bit reload timer



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 14.3-3 “Gate input operation of 16-bit reload timer” shows the operation of gate input.

Figure 14.3-3 Gate input operation of 16-bit reload timer



■ 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.

14.4 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.

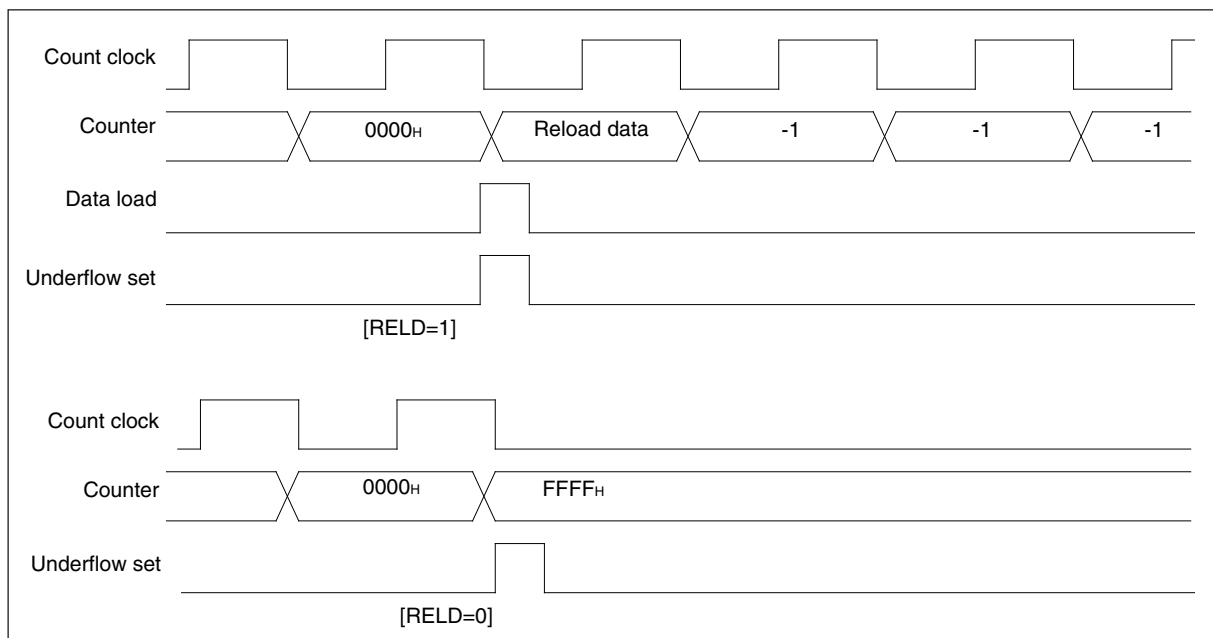
■ Underflow operation of 16-bit reload timer

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 14.4-1 "Underflow operation of 16-bit reload timer" shows the operation when an underflow occurs.

Figure 14.4-1 Underflow operation of 16-bit reload timer



14.5 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 shows 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 14.5-1 "Output pin function of 16-bit reload timer (1)"

and Figure 14.5-2 "Output pin function of 16-bit reload timer (2)" show the output pin functions.

Figure 14.5-1 Output pin function of 16-bit reload timer (1)

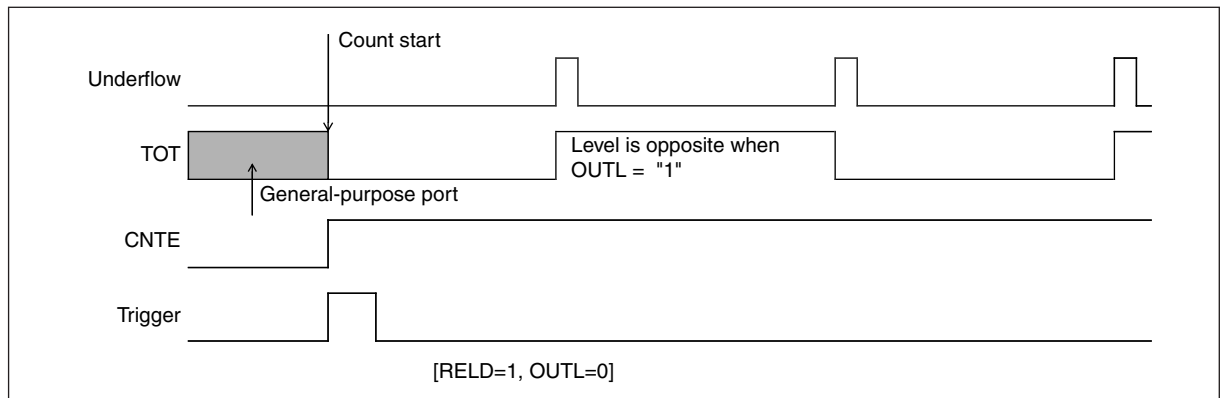
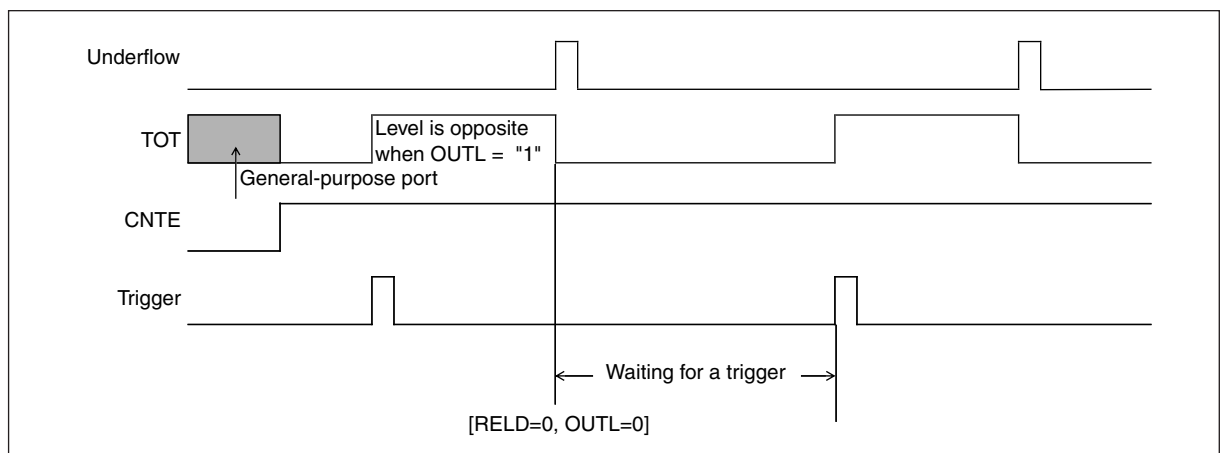


Figure 14.5-2 Output pin function of 16-bit reload timer (2)



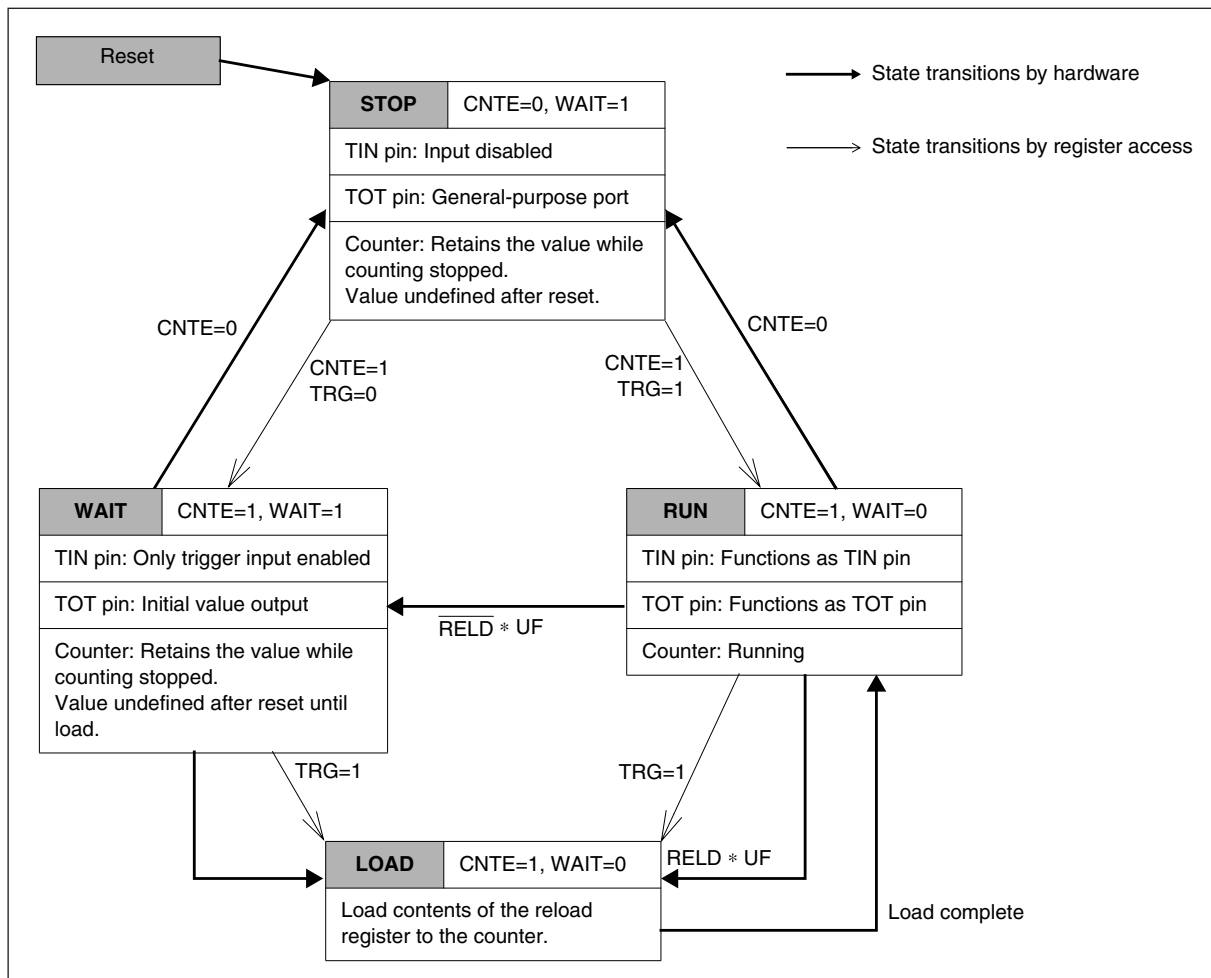
14.6 Counter Operation State

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 14.6-1 "Counter state transitions" shows the transitions between each state.

Figure 14.6-1 Counter state transitions



CHAPTER 15

WATCH TIMER

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

15.1 “Outline of Watch Timer”

15.2 “Watch Timer Control Register (WTC)”

15.3 “Watch Timer Operation”

15.1 Outline of Watch Timer

The watch timer consists of a 15-bit timer and a circuit that controls an interval interrupt. The watch timer uses subclock signals regardless of the MCS bit and SCS bit of the clock selection register (CKSCR).

■ Watch Timer Register

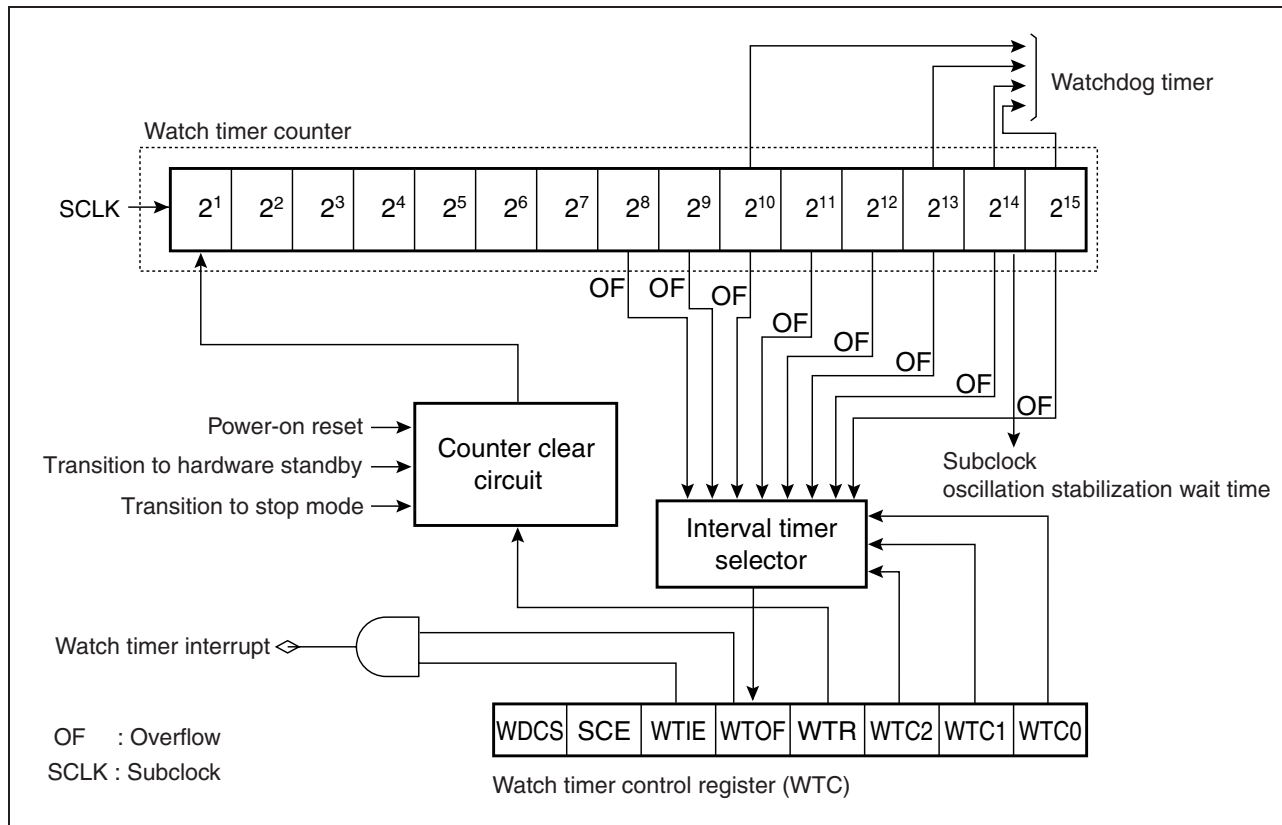
Figure 15.1-1 Watch Timer Control Register (WTC)

Watch timer control register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 0000AAH	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	WTC
Read/write ⇨	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(X)	(0)	(0)	(1)	(0)	(0)	(0)	

■ Block Diagram of Watch Timer

Figure 15.1-2 Block Diagram of Watch Timer



15.2 Watch Timer Control Register (WTC)

The watch timer control register (WTC) controls the operation of the watch timer and the interval interrupt time.

■ Watch Timer Control Register (WTC)

Figure 15.2-1 Watch Timer Control Register (WTC)

Watch timer control register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 0000AAH	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	WTC
Read/write ⇨	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(X)	(0)	(0)	(1)	(0)	(0)	(0)	

[Bit 7] WDCS

The WDCS bit is used to specify whether the clock signals from the watch timer or timebase timer is used as the input clock of the watch-dog timer when the main clock or PLL clock is selected as the machine clock. When WDCS=0, the clock signal from the watch timer is selected. When WDCS=1, the clock signal from the timebase timer is selected. Write "0" to this bit when switching to the sub-clock mode.

This bit is initialized to 1 by reset.

Note:

If WDCS is set to 1, the watch-dog timer counter may be incremented because the timebase timer output and watch timer output are asynchronous. If WDCS is set to 1, the watch-dog timer must be cleared before and after the WDCS bit is changed.

[Bit 6] SCE

The SCE bit indicates that the subclock oscillation stabilization wait time has elapsed. When this bit is 0, the oscillation stabilization is currently in progress. The oscillation stabilization time is fixed at 2^{14} cycles (subclock). This bit is initialized to 0 by a power-on reset and stop.

[Bit 5] WTIE

The WTIE bit enables an interval interrupt by the watch timer. When this bit is 1, the interrupt is enabled. When this bit is 0, the interrupt is disabled. This bit is initialized to 0 by a reset. This bit can be read and written to.

[Bit 4] WTOF

The WTOF bit is the watch timer interrupt flag. When the WTIE bit is 1 and WTOF is set to 1, an interrupt request is issued. This bit is set to 1 at each interval specified by the WTC1 and WTC0 bits. This bit is cleared by writing 0, transition to stop mode or a reset. Writing 1 has no effect.

During read operation using a read-modify-write instruction, 1 is always read from this bit.

[Bit 3] WTR

The WTR bit clears all bits of the watch timer counter to 0. Writing 0 to this bit clears the timer counter. Writing 1 has no effect. The value read from this bit is always 1.

[Bits 2, 1, and 0] WTC2, WTC1, WTC0

The WTC2, WTC1, and WTC0 bits set the watch timer interval. Table 15.2-1 “Settings for the Watch Timer Interval” lists the settings for the interval. These bits are initialized to 000 by a reset. These bits can be read and written to.

When data is written to these bits, bit 4 (WTOF) should be cleared at the same time.

Table 15.2-1 Settings for the Watch Timer Interval

WTC2	WTC1	WTC0	Interval (when SCLK*=8.192kHz)
0	0	0	$2^8 / \text{SCLK}$ (31.25 ms)
0	0	1	$2^9 / \text{SCLK}$ (62.5 ms)
0	1	0	$2^{10} / \text{SCLK}$ (125 ms)
0	1	1	$2^{11} / \text{SCLK}$ (250 ms)
1	0	0	$2^{12} / \text{SCLK}$ (500 ms)
1	0	1	$2^{13} / \text{SCLK}$ (1.000 s)
1	1	0	$2^{14} / \text{SCLK}$ (2.000 s)
1	1	1	$2^{15} / \text{SCLK}$ (4.000 s)

*: SCLK is Sub Clock, whose frequency is the oscillation clock frequency divided by 2 or 4.

The subclock division ratio is determined by SCDS bit of PLL/Subclock Control Register (PSCCR).

15.3 Watch Timer Operation

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

■ Watch Timer

The watch timer consists of a 15-bit counter that counts clock pulses originated from the subclock. As long as the subclock is input, the watch timer keeps counting. The watch timer is cleared by a power-on reset, transition to stop mode or writing 0 to the WTR bit of the watch timer control register (WTC).

Notes:

- Clearing the clock counter affects the watchdog counter and interval interrupts that use watch timer output.
- To clear the watch timer by writing “0” to the WTR bit in the watch timer control register (WTC), set the WTIE bit to “0” and set the watch timer to interrupt inhibited state. Before permitting an interrupt, clear the interrupt request issued by writing “0” to the WTOF flag.

■ Interval Interrupt Function of Watch Timer

The interval interrupt function generates interrupts at specified intervals according to the carry signals of the timer counter. The WTOF flag is set at the intervals specified by the WTC1 and WTC0 bits of the watch timer control register (WTC). This flag is set with reference to the last time when the watch timer was cleared.

On transition to stop, the watch timer is used as a timer for subclock oscillation to stabilize upon recovery, and the WTOF flag is immediately cleared upon mode transition.

■ Setting Operation Clock of Watch-dog Timer

The watch-dog clock select bit (WDCS) in the watch timer control register (WTC) can be used to set the clock input source of the watch-dog timer.

When using the subclock as the machine clock, always set the WDCS bit to "0" and select the output of the watch timer.

CHAPTER 16

8/16-BIT PPG

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

- 16.1 “Outline of 8/16-bit PPG”
- 16.2 “Block Diagram of 8/16-bit PPG”
- 16.3 “8/16-bit PPG Registers”
- 16.4 “Operations of 8/16-bit PPG”
- 16.5 “Selecting a Count Clock for 8/16-bit PPG”
- 16.6 “Controlling Pin Output of 8/16-bit PPG Pulses”
- 16.7 “8/16-bit PPG Interrupts”
- 16.8 “Initial Values of 8/16-bit PPG Hardware”

16.1 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:

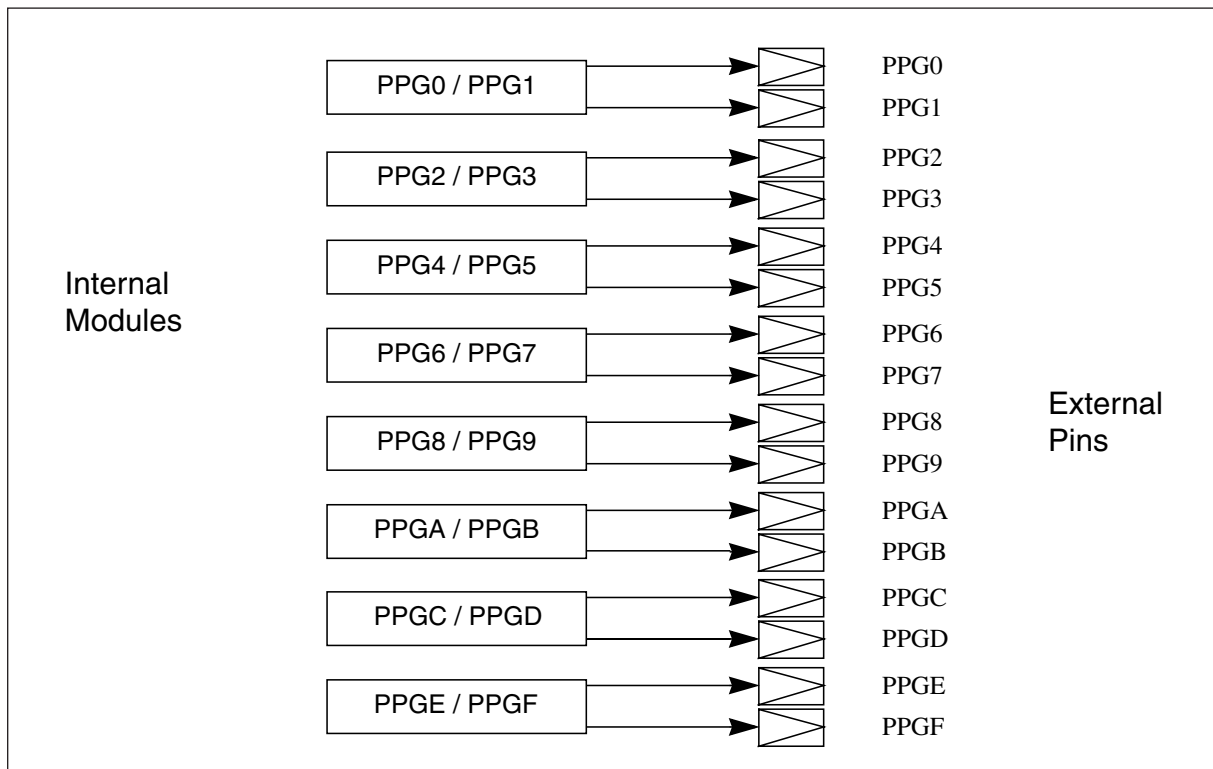
8-bit PPG output operation is implemented at specifies intervals, using channel 0 output as channel 1 clock input.

- 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 MB90340 Series contains eight 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.

Figure 16.1-1 "Relationship between PPG modules and external pins" shows the connection of internal PPG modules and external pins.

Figure 16.1-1 Relationship between PPG modules and external pins

16.2 Block Diagram of 8/16-bit PPG

Figure 16.2-1 “8-bit PPG ch0 block diagram” shows a block diagram of the 8/16-bit PPG (ch0). Figure 16.2-2 “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 16.2-1 8-bit PPG ch0 block diagram

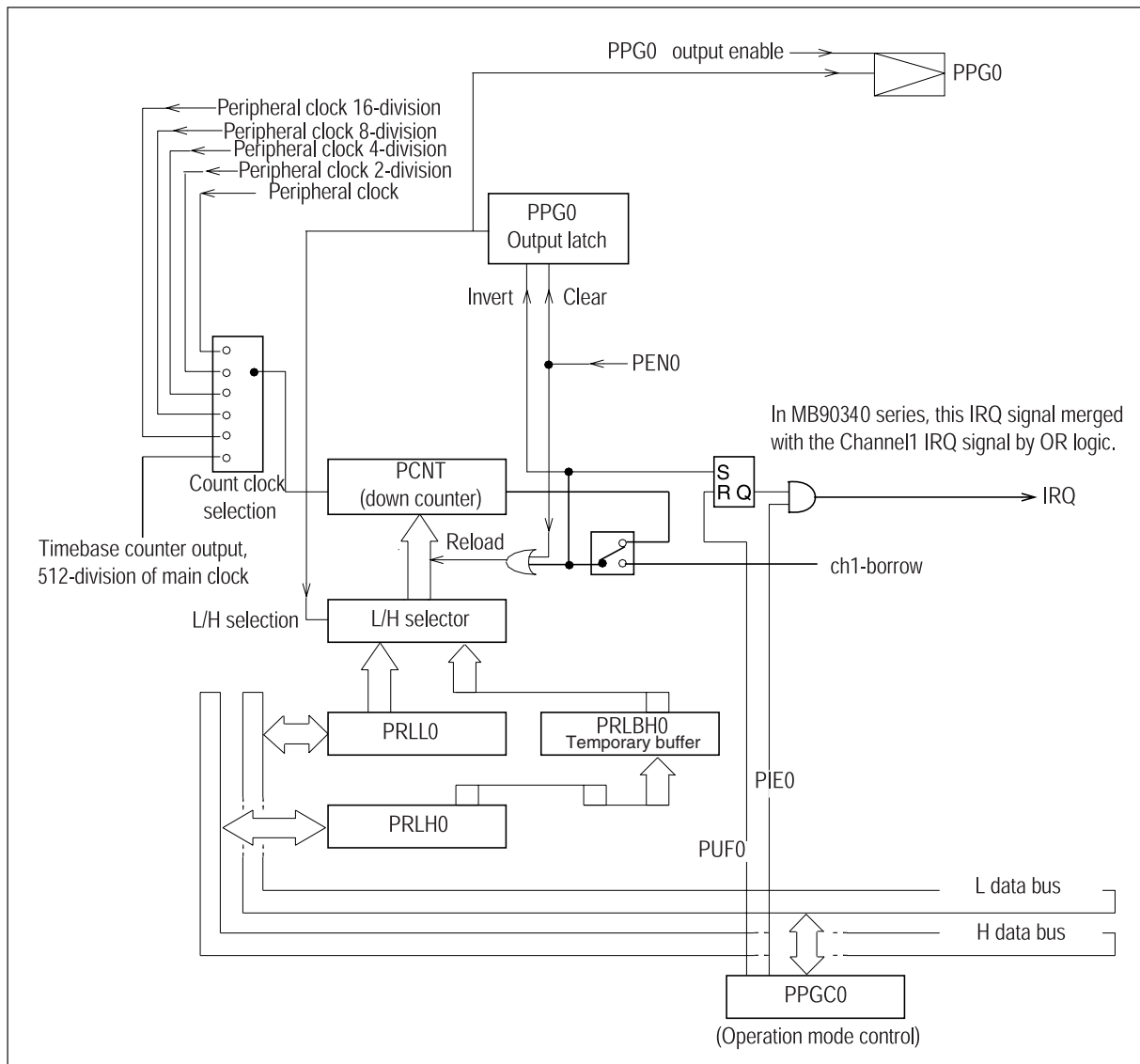
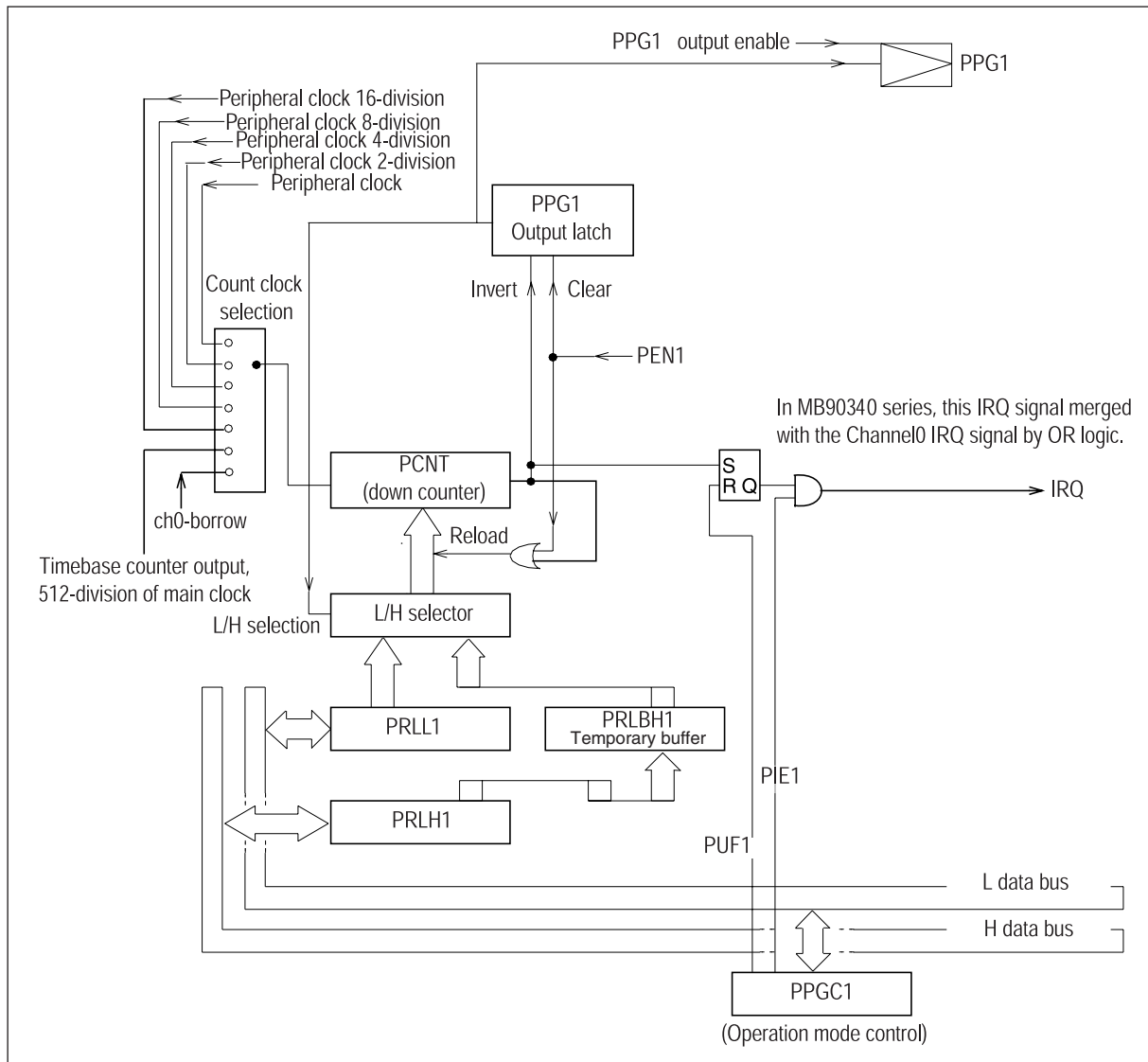


Figure 16.2-2 8-bit PPG ch1 block diagram

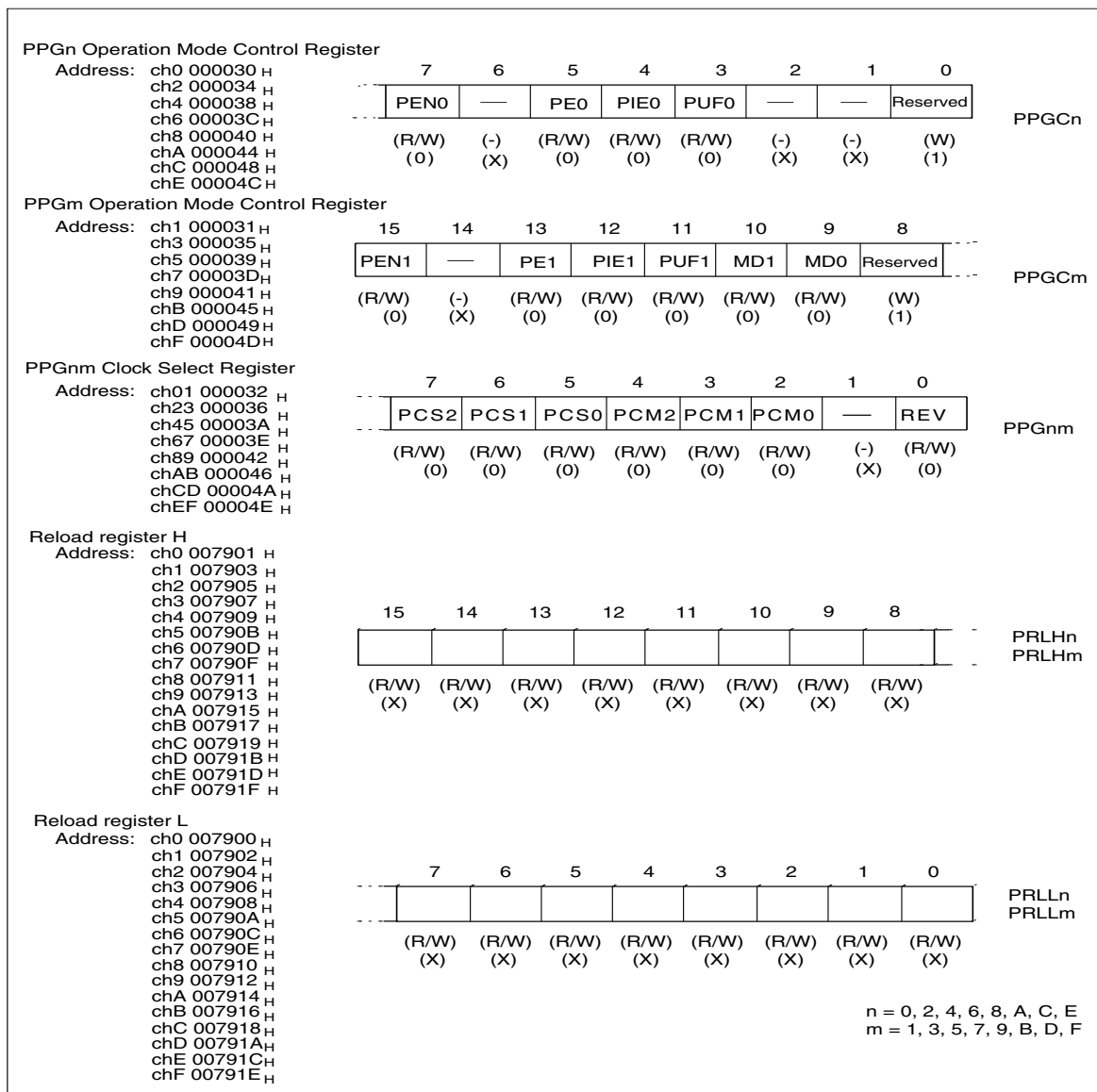
16.3 8/16-bit PPG Registers

The 8/16-bit PPG has the following five types of registers:

- PPGC0 (2, 4, 6, 8, A, C, E) Operation Mode Control Register
- PPGC1 (3, 5, 7, 9, B, D, F) Operation Mode Control Register
- PPG01 (23, 45, 67, 89, AB CD, EF) Clock Select Register
- Reload register H
- Reload register L

■ 8/16-bit PPG registers

Figure 16.3-1 8/16-bit PPG register



16.3.1 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)

Figure 16.3-2 Configuration of the PPG0 operation mode control register

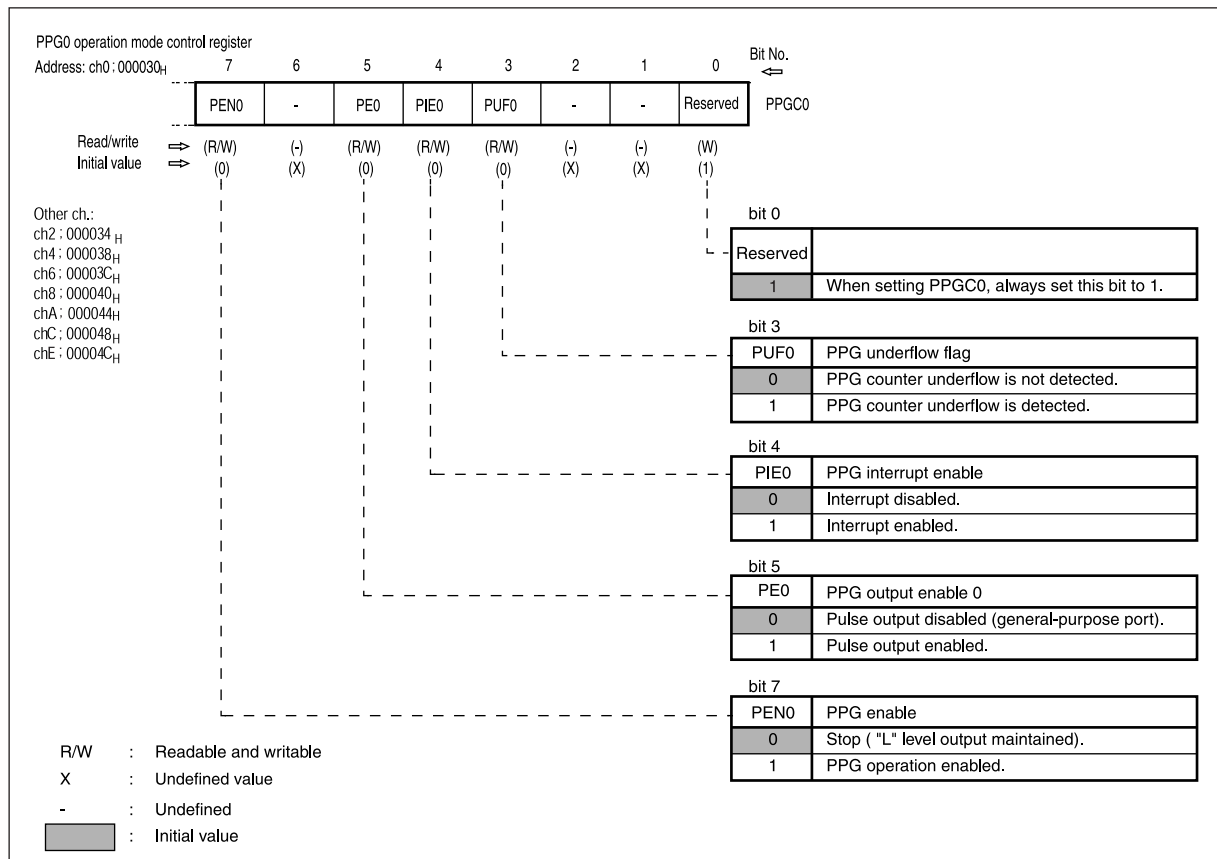


Table 16.3-1 Bit function description of the PPG0 operation mode control register

Bit name		Function
bit 7	PEN0: Operation enable bit	When set to "1", this bit enables the counter operation of the PPG. When operation is disabled but output is enabled (bit 5), a low level is maintained at the output.
bit 5	PE0: PPG0 pin output enable bit	When set to "1", this bit enables the pulse output. For MB90340 Series, the pulse signal is output to the "PPG0" external pin. When disabled, the pin can be used as general-purpose port.
bit 4	PIE0: PPG interrupt enable bit	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".
bit 3	PUF0: PPG counter underflow bit	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.
bit 0	Reserved bit.	This is a reserved bit. When setting PPGC0, always set this bit to 1.

16.3.2 PPG1 Operation Mode Control Register (PPGC1)

PPGC1 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)

Figure 16.3-3 Configuration of the PPG1 operation mode control register

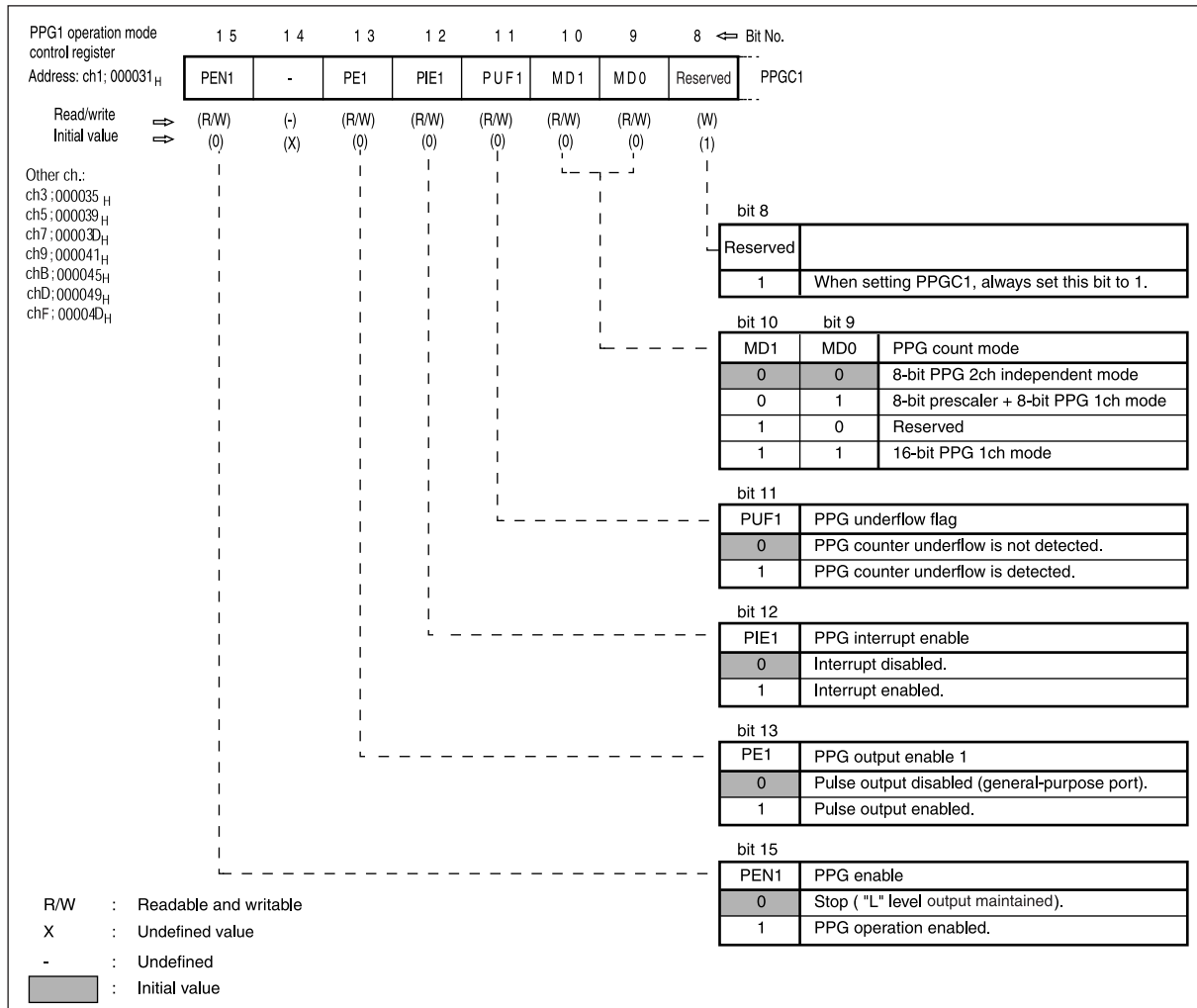


Table 16.3-2 Bit function description of the PPG1 operation mode control register

Bit name		Function
bit 15	PEN1: Operation enable bit	When set to "1", this bit enables the counter operation of the PPG. When operation is disabled but output is enabled (bit 13), a low level is maintained at the output.
bit 13	PE1: PPG1 pin output enable bit	When set to "1", this bit enables the pulse output. For MB90340 Series, the pulse signal is output to the "PPG1" external pin. When disabled, the pin can be used as general-purpose port.
bit 12	PIE1: PPG interrupt enable bit	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".
bit 11	PUF1: PPG counter underflow bit	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 ch1 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.
bit 10, 9	MD1, MD0: PPG count mode	These bits select the PPG timer operation mode as described in Figure 16.3-3 "Configuration of the PPG1 operation mode control register". 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	Reserved bit.	This is a reserved bit. When setting PPGC1, always write "1" to this bit.

16.3.3 PPG0/1 Clock Select Register (PPG01)

The PPG0/1 Clock Select Register (PPG01) is an 8-bit control register that controls the counter clock of the 8/16-bit PPG.

■ PPG0/1 clock select register (PPG01)

Figure 16.3-4 Configuration of the PPG0/1 clock select register (PPG01)

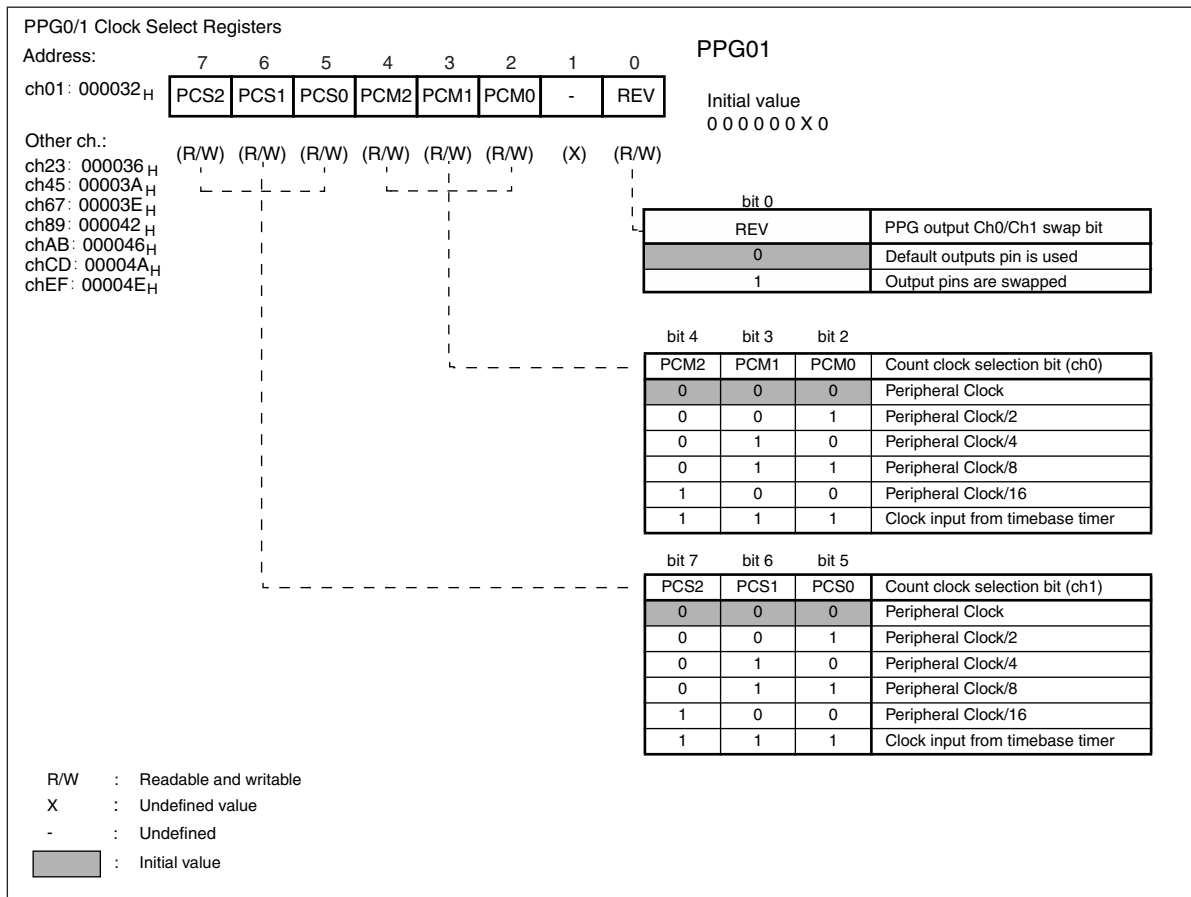


Table 16.3-3 Bit function description of the clock select register (PPG01)

Bit name		Function			
bits 7 to 5	PCS2 to PCS0: Count clock selection bit	These bits select the operation clock for the down counter of Channel 1 as described below. 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.			
		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)
		bits 4 to 2	PCM2 to PCM0: 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 μs machine clock, 16 MHz)
1	0			1	Clock input from the timebase timer (128 μs, 4 MHz source oscillation)
bit 0	REV: PPG output ch0/ch1 swap bit			Set this bit to "1" , swap PPG channel1 output pin and PPG channel0 output pin. Set this bit to "0"(default value), both PPG channel0 and channel1 signals output from default output pins.	

16.3.4 Reload Register (PRLH/PRLH)

The reload registers (PRLH and PRLH) are 8-bit registers that store reload values for the PCNT down counters. The PRLH and PRLH registers are readable and writable.

■ Reload register (PRLH/PRLH)

Figure 16.3-5 Reload register (PRLH/PRLH)

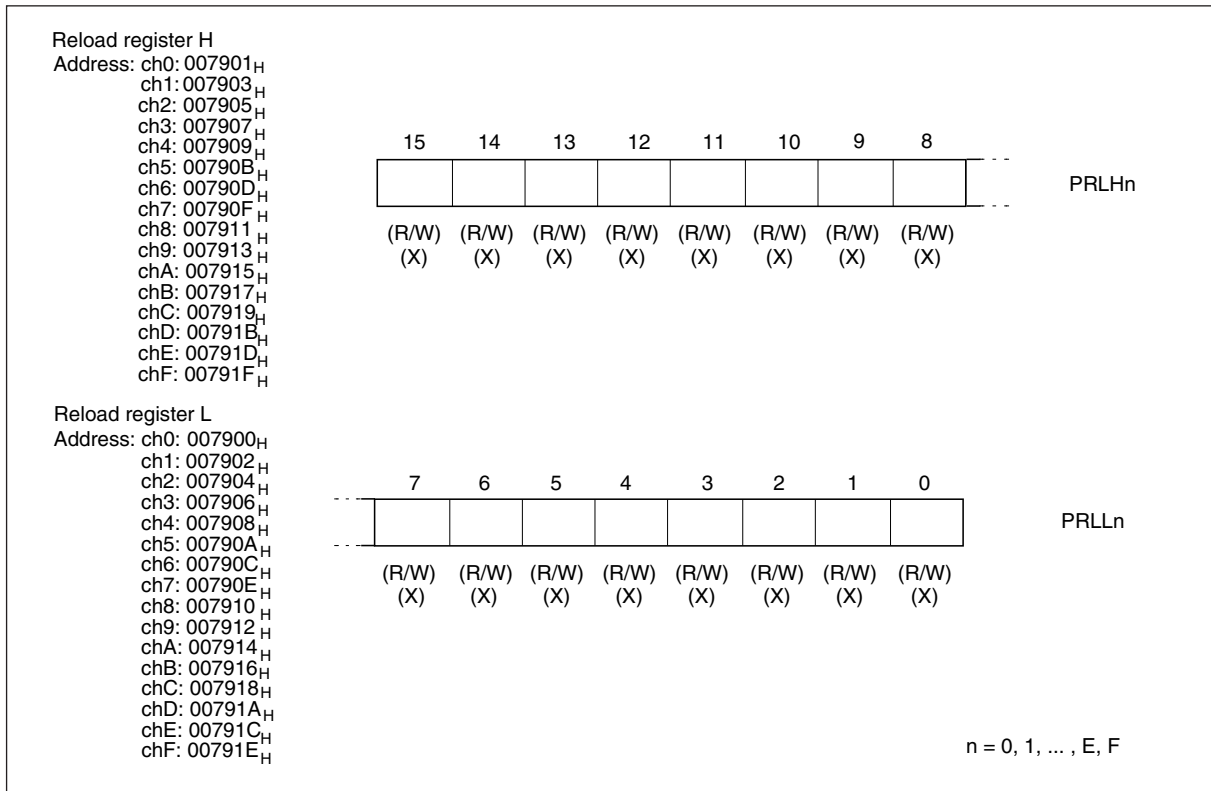


Table 16.3-4 Register function of the reload registers

Register name	Function
PRLLn	Holds the L side reload value.
PRLHn	Holds the H side reload value.

Note:

In 8-bit prescaler + 8-bit PPG mode, different values in PRLH and PRLH of Channel 0 may cause the PPG waveform of ch1 to vary in each cycle. Write the same value to PRLH and PRLH of ch0.

16.4 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 16.4-1 "Reload operation and pulse output" lists the relationship between the reload operation and pulse outputs.

Table 16.4-1 Reload operation and pulse output

Reload operation	Pin output change		
PRLH --> PCNT	PPG0/1 [0 --> 1]	↑	Rise
PRL --> PCNT	PPG0/1 [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 PPG0 pin is connected to the ch0 PPG output, while the PPG1 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 PPG0 pin is connected to the ch0 prescaler output, while the PPG1 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 PPG0 and PPG1 pins are connected to the 16-bit PPG output.

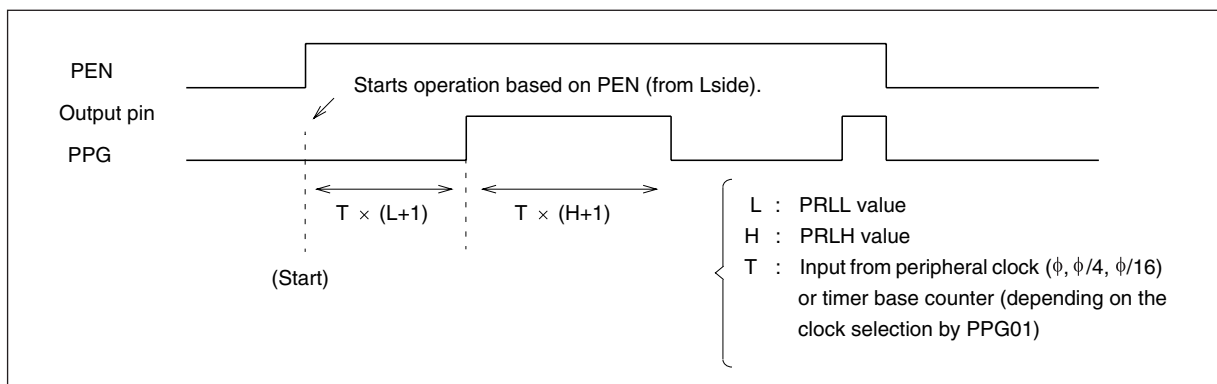
■ 8/16-bit PPG output operation

In this block, the ch0 PPG is activated to start counting when "1" is written to bit 7 (PEN0) of the PPGC0 (PWM 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.

In 8-bit prescaler + 8-bit PPG mode, do not set ch1 to be in operation while ch0 operation is stopped.

In 16-bit PPG mode, ensure that bit 7 (PEN0) of PPGC0 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 to the L-level period). PPG continues operation until stop is specified explicitly.

Figure 16.4-1 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.

$$\begin{aligned} P1 &= T \times (L+1) \\ Ph &= T \times (H+1) \end{aligned} \quad \left\{ \begin{array}{l} L : \text{PRL value} \\ H : \text{PRLH value} \\ T : \text{Input clock cycle} \\ Ph : \text{High pulse width} \\ Pl : \text{Low pulse width} \end{array} \right.$$

16.5 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, the first count cycle may be shifted.

16.6 Controlling Pin Output of 8/16-bit PPG Pulses

The pulses generated by this module can be output from external pins PPG0 and PPG1.

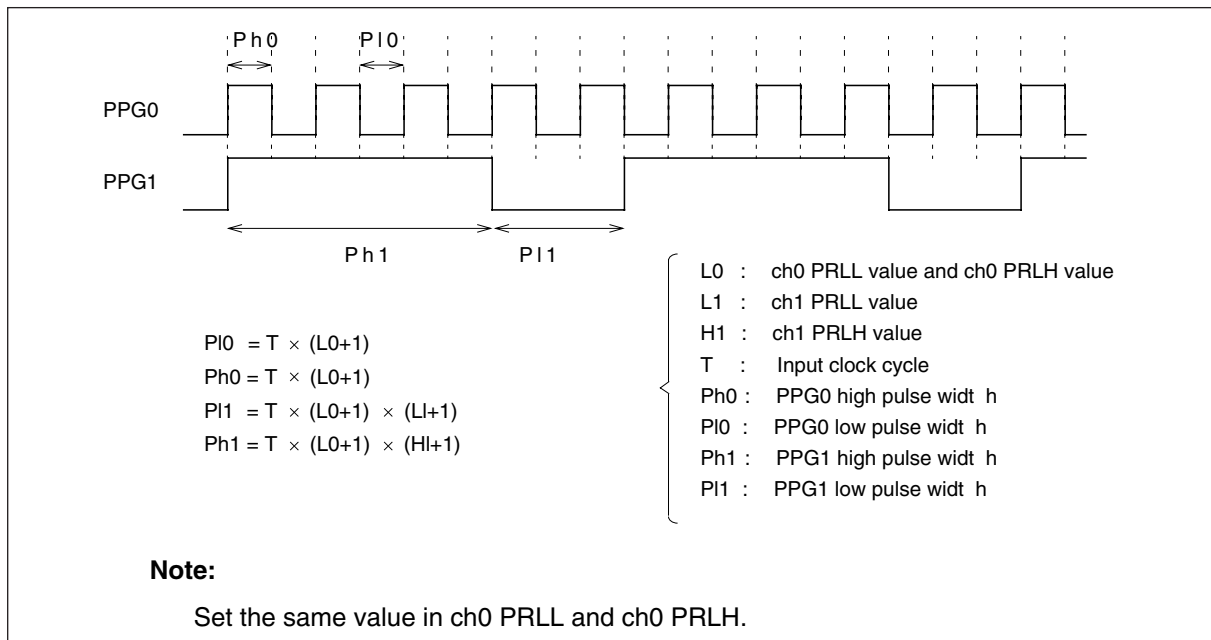
■ Controlling pin output of 8/16-bit PPG pulses

To output the pulses from an external pin, write "1" to the bit corresponding to each pin (PPGC0: PE0, PPGC1: PE1). 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 PPG0 and PPG1. Thus, the same output can be obtained by enabling both external pin.

In 8-bit prescaler + 8-bit PPG mode, the 8-bit prescaler square waveform is output from PPG0, while the 8-bit PPG waveform is output from PPG1. Figure 16.6-1 "8+8 PPG output operation waveform" is a diagram of output waveforms in this mode.

Figure 16.6-1 8+8 PPG output operation waveform



Note:

Set the same value in ch0 PRLH and ch0 PRLH.

16.7 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, PUF0 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.

16.8 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
- PPG01 --> 000000X0_B

● Pulse outputs

- PPG0 --> "L"
- PPG1 --> "L"
- PE0 --> PPG0 output disabled
- PE1 --> PPG1 output disabled

● Interrupt requests

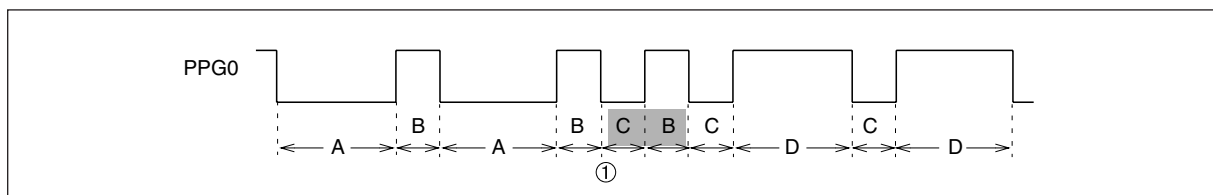
- IRQ0 --> "L"
- IRQ1 --> "L"

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 PRLH 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 16.8-1 Write timing for 8/16-bit PPG reload registers (PRLH and PRLH)

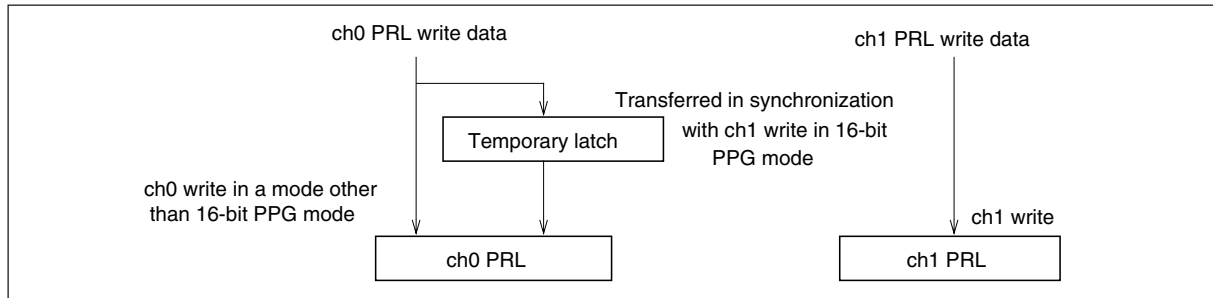


Assume that PRL_L is updated from A to C before point 1 in the time chart above, and PRL_H is updated from B to D after point 1. Since the PRL values at point 1 are PRL_L=C and PRL_H=B, a pulse of L side count value C and H side count value B is output only once.

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 16.8-2 PRL write operation block diagram



CHAPTER 17

DTP/EXTERNAL INTERRUPTS

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

- 17.1 “Outline of DTP/External Interrupts”
- 17.2 “DTP/External Interrupt Registers”
- 17.3 “Operations of DTP/External Interrupts”
- 17.4 “Switching Between External Interrupt and DTP Requests”
- 17.5 “Notes on using DTP/External Interrupt function”

17.1 Outline of DTP/External Interrupts

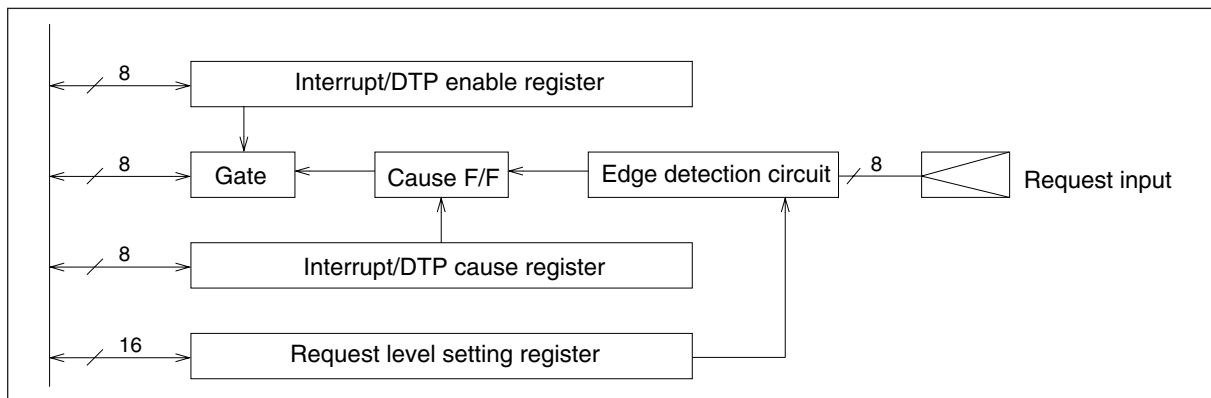
The data transfer peripheral (DTP) is located between an external peripheral and the F²MC-16LX CPU. The DTP receives a DMA request or interrupt request from the external peripheral, transfers the request to the F²MC-16LX CPU to activate the intelligent I/O service or interrupt processing.

■ 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.

■ Block diagram of DTP/external interrupts

Figure 17.1-1 Block diagram of DTP/external interrupts



■ DTP/External interrupts registers

ENIR0	bit	7	6	5	4	3	2	1	0	Initial Value
Address : 0000C6 _H		EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	00000000 _B
ENIR1	bit	7	6	5	4	3	2	1	0	
Address : 0000CA _H		EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	00000000 _B
EIRR0	bit	15	14	13	12	11	10	9	8	
Address : 0000C7 _H		ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	XXXXXXXX _B
EIRR1	bit	15	14	13	12	11	10	9	8	
Address : 0000CB _H		ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8	XXXXXXXX _B
ELVR0	bit	7	6	5	4	3	2	1	0	
Address : 0000C8 _H		LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	00000000 _B
ELVR0	bit	15	14	13	12	11	10	9	8	
Address : 0000C9 _H		LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	00000000 _B
ELVR1	bit	7	6	5	4	3	2	1	0	
Address : 0000CC _H		LB11	LA11	LB10	LA10	LB9	LA9	LB8	LA8	00000000 _B
ELVR1	bit	15	14	13	12	11	10	9	8	
Address : 0000CD _H		LB15	LA15	LB14	LA14	LB13	LA13	LB12	LA12	00000000 _B
EISSR	bit	7	6	5	4	3	2	1	0	
Address : 0000CE _H		INT15R	INT14R	INT13R	INT12R	INT11R	INT10R	INT9R	INT8R	00000000 _B

17.2 DTP/External Interrupt Registers

The DTP/external interrupts has the following three types of registers:

- Interrupt/DTP enable register (ENIR: External interrupt request enable register)
- Interrupt/DTP flag (EIRR: External interrupt request register)
- Request level setting register (ELVR: External level register)
- External interrupt source select register (EISSR)

■ Interrupt/DTP enable register (ENIR: External interrupt request enable register)

Figure 17.2-1 External interrupt request enable register (ENIR)

ENIR0 Address : 0000C6 _H		7	6	5	4	3	2	1	0	Initial value 00000000 _B
		EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ENIR1 Address : 0000CA _H		7	6	5	4	3	2	1	0	00000000 _B
		EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W : Readable and writable										

The ENIR register contains the interrupt enable bits of the DTP/external interrupt module. When an EN_x bit is enabled, an interrupt request is signaled if the specified interrupt event is detected at the corresponding pin. Writing "1" to these bits enables and writing "0" disables interrupt requests.

■ Interrupt/DTP flags (EIRR: External interrupt request register)

Figure 17.2-2 External interrupt request register (EIRR)

EIRR Address : 0000C7 _H	15	14	13	12	11	10	9	8	Initial value XXXXXXXX _B
	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
EIRR Address : 0000CB _H	15	14	13	12	11	10	9	8	Initial value XXXXXXXX _B
	ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W : Readable and writable									
X : Undefined									

The EIRR register contains the interrupt flags.

If the specified interrupt event is detected at a pin, the corresponding ER_x bit is set and if the EN_x bit is "1", then an interrupt request is signaled to the interrupt controller. Writing "1" to these bits has no effect and writing "0" clears the flags. "1" is always read from these bits by the read-modify-write instructions.

Note:

When clearing the interrupt flag, make sure to clear the flag which caused the interrupt alone but not others.

■ Request level setting register (ELVR: External Interrupt level register)

Figure 17.2-3 External interrupt level register (ELVR)

ELVR0	7	6	5	4	3	2	1	0	Initial value
Address : 0000C8 _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	
	15	14	13	12	11	10	9	8	
Address : 0000C9 _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	
ELVR1	7	6	5	4	3	2	1	0	
Address : 0000CC _H	LB11	LA11	LB10	LA10	LB9	LA9	LB8	LA8	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	15	14	13	12	11	10	9	8	
Address : 0000CD _H	LB15	LA15	LB14	LA14	LB13	LA13	LB12	LA12	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Readable and writable

ELVR defines the request event at the external pin. Each pin is assigned with two bits as described in Table 17.2-1 “Interrupt request detection factor for external 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 17.2-1 Interrupt request detection factor for external 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

■ External interrupt source select register

Figure 17.2-4 External interrupt source select register (EISSR)

EISSR	7	6	5	4	3	2	1	0	Initial value
Address : 0000CE _H	INT15R	INT14R	INT13R	INT12R	INT11R	INT10R	INT9R	INT8R	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W : Readable and writable									

The 8 external interrupt inputs INT8 to INT15 can be relocated to different pins to offer a “wake up by resource input” function like CAN wakeup. The standard location of INT8 to INT15 is Port0 (External bus). some interrupt inputs are moved to other pins to use the interrupt function even when the external bus is used.

The source for the interrupt function will be defined by this EISSR register according to the following table.

Table 17.2-2 The selection of the interrupt function

EISSR bit	“0” (Initial value)	“1”
INT8R	P00	PA0 (RX0)
INT9R	P01	P42(RX1/IN6)
INT10R	P02	P32(RX2)
INT11R	P03	P12(SIN3)
INT12R	P04	P80(TIN0/ADTG)
INT13R	P05	P81(TOT0/CKOT)
INT14R	P06	P82(SIN0/TIN2)
INT15R	P07	P84(SCK0)

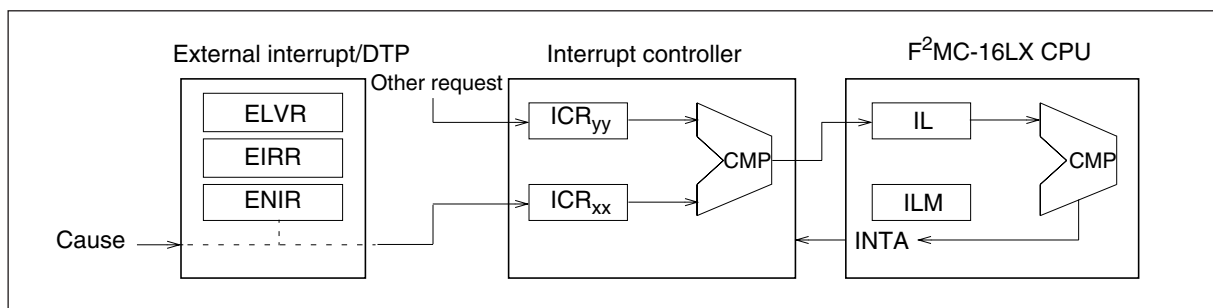
17.3 Operations of DTP/External Interrupts

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 17.3-1 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 17.3-2 Timing to cancel the external interrupt at the end of DTP operation

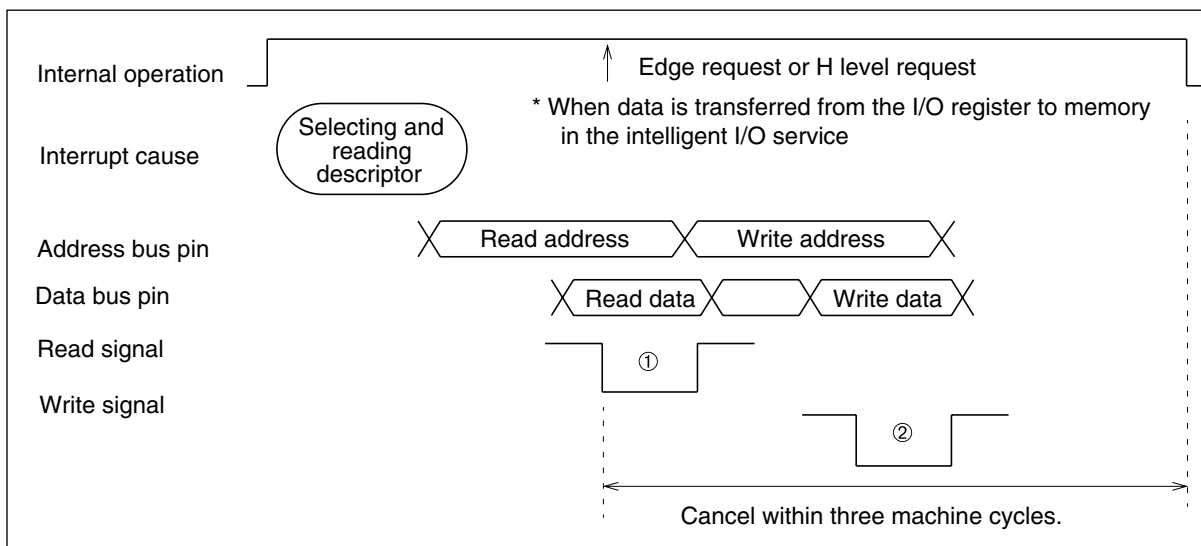
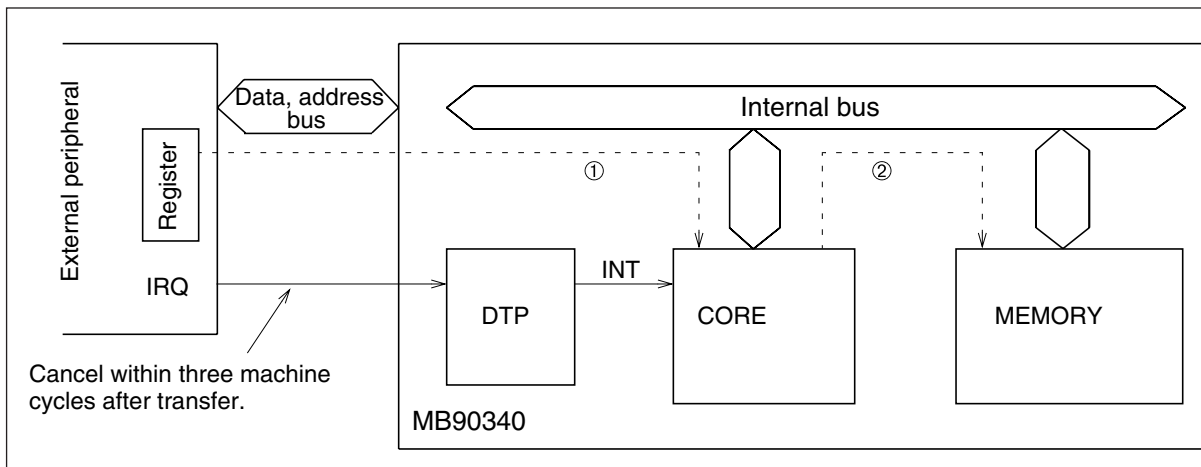


Figure 17.3-3 Sample interface to the external peripheral

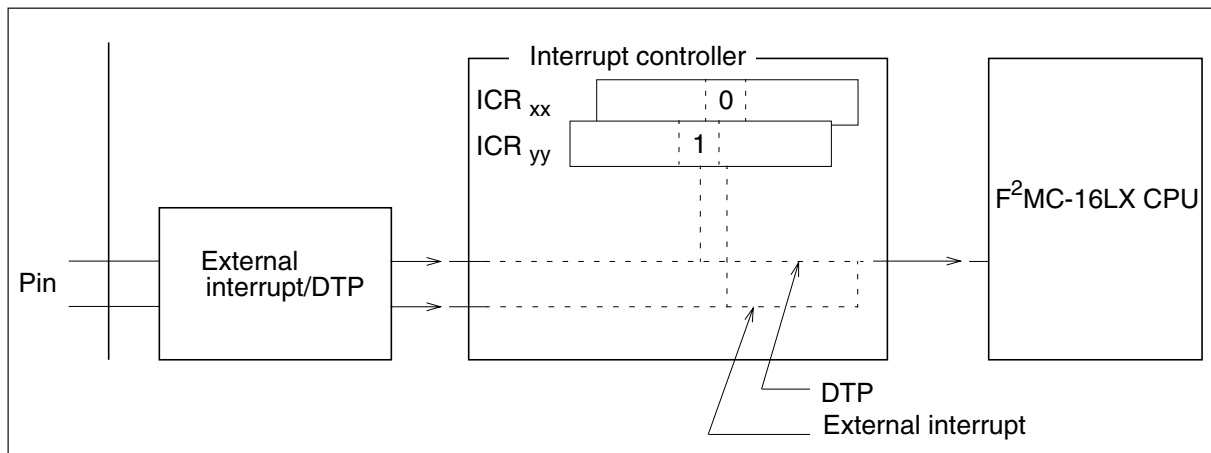


17.4 Switching between External Interrupt and DTP Requests

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 17.4-1 Switching between external interrupt and DTP requests



17.5 Notes on using DTP/External Interrupt functions

The following points must be considered to use the DTP/External function.

- Conditions on the behavior of external circuit for the DTP function
- Clearing interrupt flag
- External interrupt request level

■ Conditions on the behavior of external circuit for the DTP function

- The external circuit which uses the DTP function must be able to inactivate the request signal when the requested DTP transfer is performed.
- If the DTP request should be withdrawn, the request signal must be inactivated within 3 machine clock cycles after the transfer is initiated (the RDX signal is activated). Otherwise it is regarded as a valid request and the transfer will be performed.

■ Clearing interrupt flag

- When it is used as an external interrupt the interrupt flag must be cleared within the interrupt service routine. Otherwise the same service is performed again after the completion of the first interrupt service.
- When level detection is specified as the event input, the interrupt flag is set again even after it is cleared as long as the active level is kept at the input pin. In this case, the external cause of the request should be cleared or the interrupt enable bit should be cleared.

■ External interrupt request level

- When edge detection is specified as the event input, the pulse width of the input signal must be greater than equal to 3 machine clock cycles to recognize the input edge.
- When level detection is specified as the event input, the interrupt flag keeps active status once the specified level is input even after the input signal changes to the inactive level as shown in figure 17.5-1. In order to clear the request, the interrupt flag must be cleared.

Figure 17.5-1 Clearing interrupt/DTP cause register upon level set

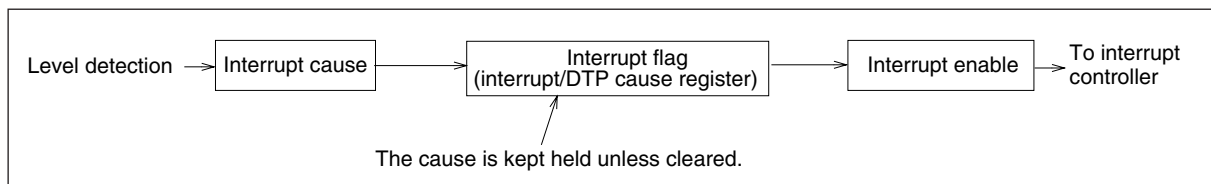
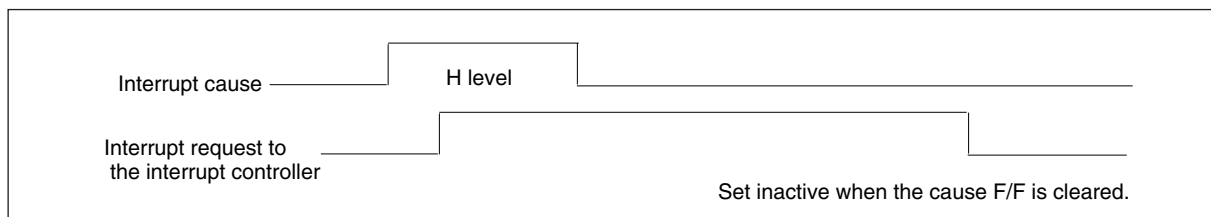


Figure 17.5-2 Interrupt cause and interrupt request to the interrupt controller while interrupts are enabled



CHAPTER 18

A/D CONVERTER

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

- 18.1 “Features of A/D Converter”
- 18.2 “Block Diagram of A/D Converter”
- 18.3 “Registers for A/D Converter”
- 18.4 “Operation of the A/D Converter”
- 18.5 “Conversion Using EI²OS”
- 18.6 “Convert Data Protection Function”

18.1 Features of A/D Converter

The A/D converter converts analog input voltages to digital values.

■ Features of A/D converter

- Conversion time: 1.9 μ s min. per channel (with 24-MHz machine clock)
- RC successive approximation conversion with sample and hold circuit
- Resolution: 10 bits or 8 bits
- Analog input pin selected from up to 24* channels by program
 - Single-channel conversion: 1 channel is selected and converted.
 - Scan conversion: Sequential channels (up to 24* channels) are converted.
- Mode setting
 - Single mode: Converts the specified channels once. Up to 24* channels can be specified.
 - Continuous mode: Converts the specified channels repeatedly. Up to 24* channels can be specified.
 - Stop mode: Converts one channel, then the converter stops and stands by for the next activation. (starting of conversion can be synchronized)
- Interrupt requests

At the end of A/D conversion, the interrupt request for an A/D conversion can be signaled to the interrupt controller. This interrupt can start EI²OS or μ DMA, which can transfer A/D conversion result data to the memory.

- Selectable activation cause

The activation can be done by software, reload timer 1, or external trigger (falling edge).

* 24channel Analog inputs are only supported by the products with 'C'-suffix. The products without 'C'-suffix have 16 channel Analog inputs.

■ Analog input enable register

The A/D converter's analog input pins are assigned to ports 5 to 7.

Set pins to be used as analog input to “analog input enable (default)” by setting “1” to corresponding bit of analog input enable register.

For details, see Section 10.2.4 Analog enable register.

Figure 18.1-1**Analog Input Enable register for Port5**

	15	14	13	12	11	10	9	8
ADER5								
OB_H	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8
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)

Analog Input Enable register for Port6

	7	6	5	4	3	2	1	0
ADER6								
OC_H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0
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)

Analog Input Enable register for Port7

	15	14	13	12	11	10	9	8
ADER7								
OD_H	ADE23	ADE22	ADE21	ADE20	ADE19	ADE18	ADE17	ADE16
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)

Pins of port 5,6 and 7 are controlled as described below.

0: Port input/output mode

1: Analog input mode

“1” is set upon a reset

Table 18.1-1 Analog input pins (1/2)

Port	pin name		Analog input channel name	corresponding ADE register	corresponding ADE bit
	LQFP	QFP			
Port6	34	36	AN0	ADER6	ADE0
	35	37	AN1		ADE1
	36	38	AN2		ADE2
	37	39	AN3		ADE3
	38	40	AN4		ADE4
	39	41	AN5		ADE5
	40	42	AN6		ADE6
	41	43	AN7		ADE7

Table 18.1-1 Analog input pins (2/2)

Port	pin name		Analog input channel name	corresponding ADE register	corresponding ADE bit
	LQFP	QFP			
Port5	22	24	AN8	ADER5	ADE8
	23	25	AN9		ADE9
	24	26	AN10		ADE10
	25	27	AN11		ADE11
	26	28	AN12		ADE12
	27	29	AN13		ADE13
	28	30	AN14		ADE14
	29	31	AN15		ADE15
Port7	43	45	AN16	ADER7	ADE16
	44	46	AN17		ADE17
	45	47	AN18		ADE18
	46	48	AN19		ADE19
	47	49	AN20		ADE20
	48	50	AN21		ADE21
	53	55	AN22		ADE22
	54	56	AN23		ADE23

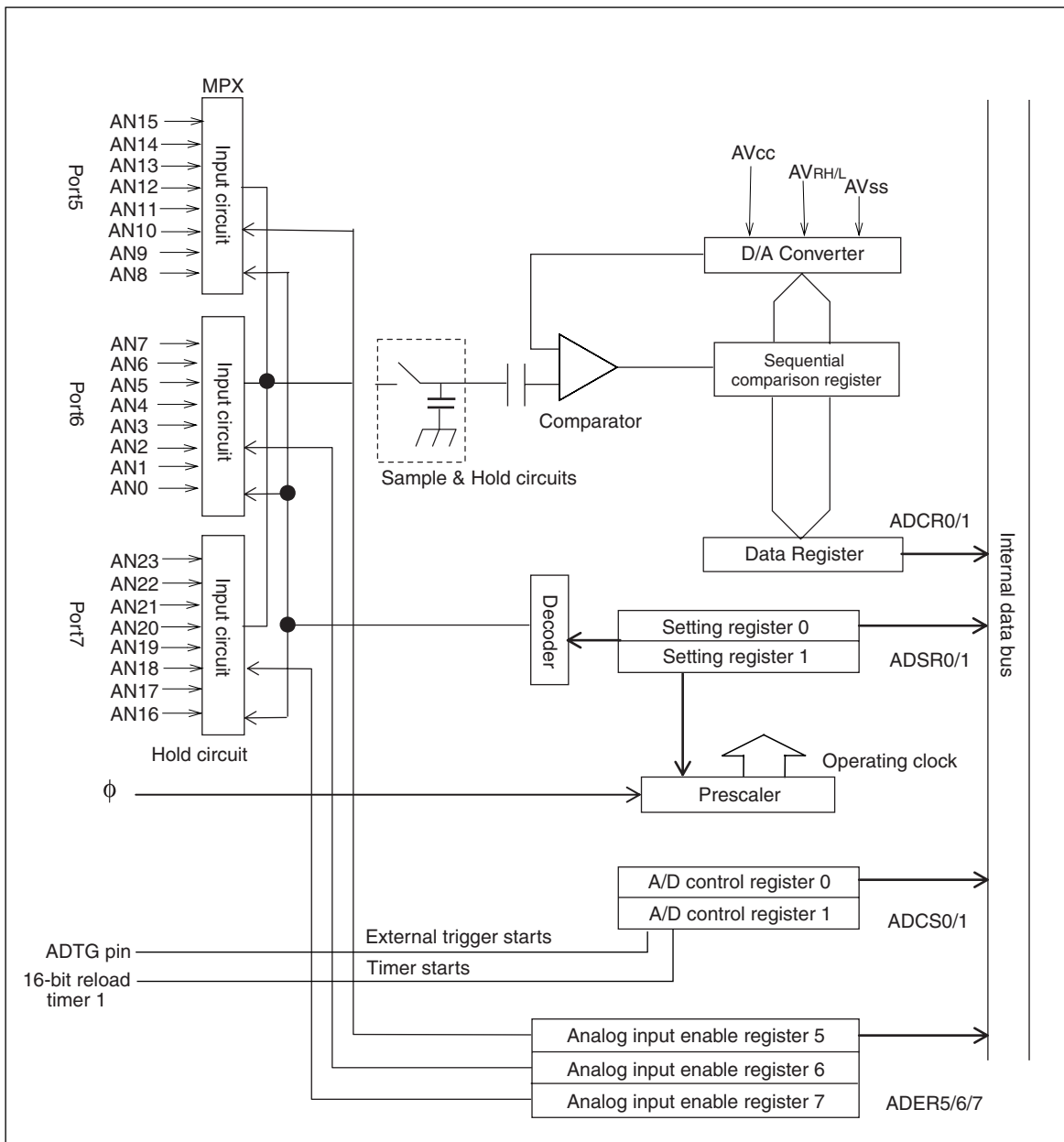
*Analog input channel AN16 to AN23 are only supported by products with 'C'-suffix.

18.2 Block Diagram of A/D Converter

Figure 18.2-1 shows the block diagram of A/D converter.

■ Block diagram of A/D converter

Figure 18.2-1 Block diagram of A/D converter



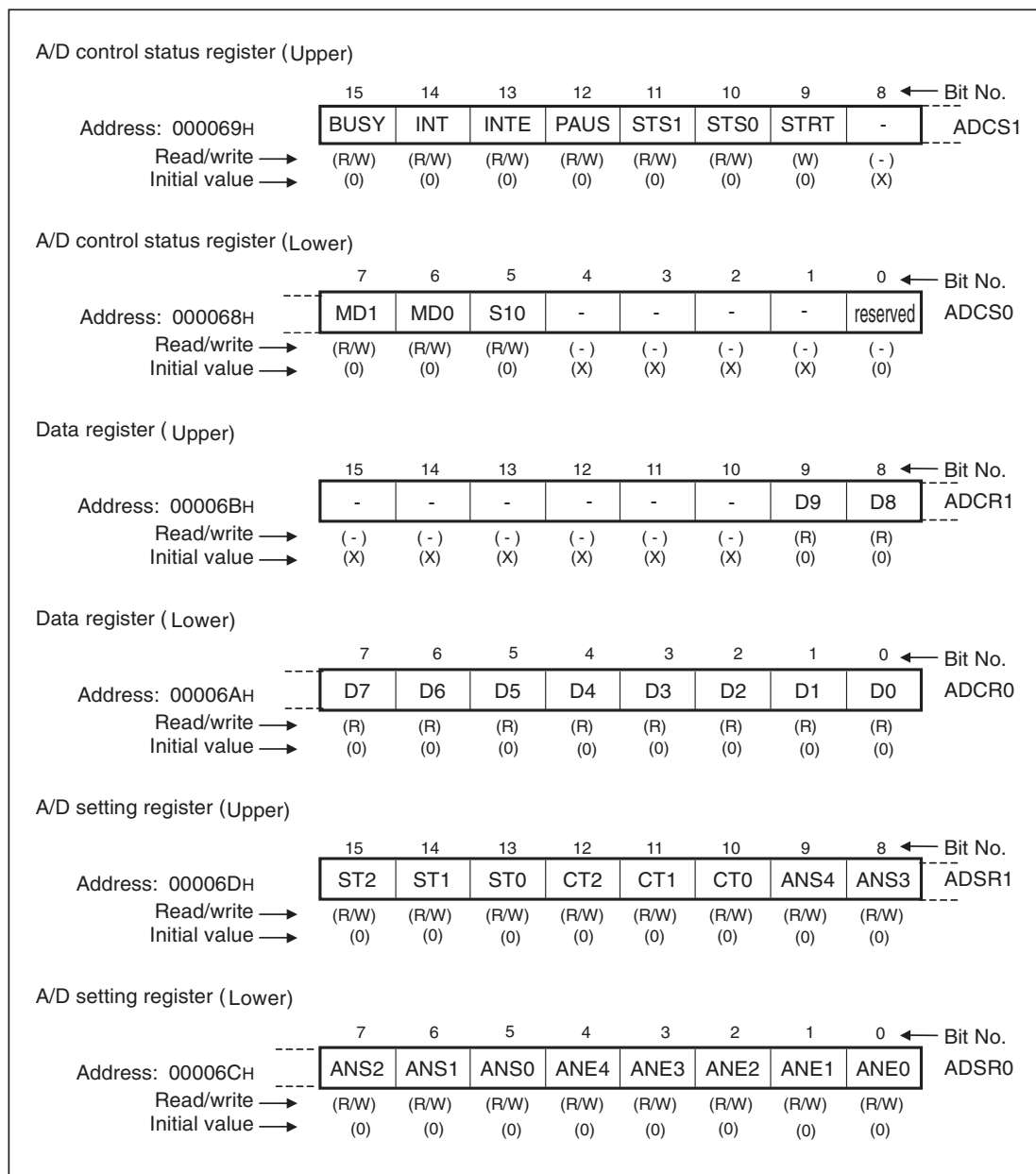
18.3 Registers for A/D Converter

The A/D converter has the following three registers:

- Control status register: ADCS0/1
- Data register: ADCR0/1
- Setting register: ADSR0/1

■ Registers for A/D converter

Figure 18.3-1 Registers for A/D Converter



18.3.1 Control status register (ADCS0)

The control status register (ADCS0) controls the A/D converter and indicates its status. Do not rewrite ADCS0 during A/D conversion.

■ Control Status Register (ADCS0)

Figure 18.3-2 A/D control status register (ADCS0)

A/D control status register (Lower)								
	7	6	5	4	3	2	1	0 ← Bit No.
Address: 000068H	MD1	MD0	S10	-	-	-	-	reserved
Read/write →	(R/W)	(R/W)	(R/W)	(-)	(-)	(-)	(-)	(-)
Initial value →	(0)	(0)	(0)	(X)	(X)	(X)	(X)	(0)

[bit 7 and bit 6] MD1 and MD0 (A/D converter mode set)

Table 18.3-1 Operation Mode Settings

MD1	MD0	Operation mode
0	0	Single mode 1 (Reactivation during A/D conversion is allowed.)
0	1	Single mode 2 (Reactivation during A/D conversion is not allowed.)
1	0	Continuous mode (Reactivation during A/D conversion is not allowed.)
1	1	Stop mode (Reactivation during A/D conversion is not allowed.)

- **Single mode**
A/D conversion is continuously performed from the channel specified with ANS4 to ANS0 to the channel specified with ANE4 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 ANS4 to ANS0 to the channel specified with ANE4 to ANE0 in a row.
- **Stop mode**
A/D conversion is performed from the channel specified with ANS4 to ANS0 to the channel specified with ANE4 to ANE0, pausing for each channel. The A/D conversion is resumed upon an activation.

Note:

- The A/D conversion in the continuous or stop mode continues until it is stopped by the BUSY bit.
- Write 0 to the BUSY bit to stop the A/D conversion.
- Reactivation disabled in single mode2, continuous mode, and stop mode. This applies to all kinds of activation.

[bit 5] S10

This bit specifies the resolution of conversion. When “0” is written to this bit, 10-bit A/D conversion is performed. When “1” is written to this bit, 8-bit A/D conversion is performed and the conversion result is stored in D7 to D0.

[bits 4 to bit 1] Unused bits

Writing to these bits has no effect. Reading these bits always returns “1”.

[bit 0] reserved (reserved bit)

This bit is a reserved bit. Always write “0” to this bit.

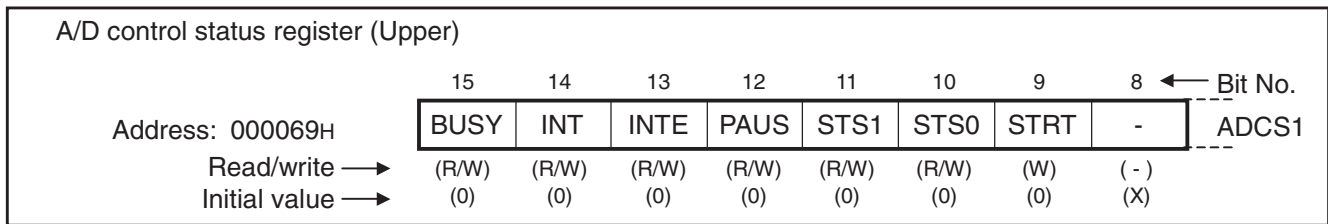
Reading this bit always returns “0”.

18.3.2 Control status register (ADCS1)

The control status register (ADCS1) controls the A/D converter and indicates its status.

■ Control Status Register (ADCS1)

Figure 18.3-3 A/D control status register (ADCS1)



[bit 15] BUSY (busy flag and stop)

- Read operation:
 - This bit indicates an operation of the A/D converter.
 - The bit is set by starting of A/D conversion, and cleared by end of A/D conversion.
- Write operation:
 - When “0” is written to this bit during the A/D conversion, the conversion is forced to stop.
 - Writing “1” to this bit has no effect.

Notes:

- “1” is read from this bit when an read-modify-write instruction is used.
- In the single mode, this bit is cleared when A/D conversion ends.
- In the continuous or stop mode, the A/D conversion does not stop until writing “0” to this bit.
- This bit is initialized to 0 by reset.
- Do not perform the forced stop and the activation concurrently (using software (BUSY = 0, STRT = 1), external trigger, or timer).

[bit 14] INT (interrupt)

This bit is set when converted data is written to ADCR. When this bit is set with bit5 (INTE) set to “1”, an interrupt request is generated. In addition, the EI²OS or μ DMA is activated if it is enabled. Writing “1” to this bit has no effect. This bit is cleared by writing “0” or by the interrupt clear signal from the EI²OS / μ DMA.

Notes:

- To clear this bit by writing “0”, ensure that A/D conversion is not in progress.
- This bit is initialized to “0” at reset.
- Using an read-modify-write instruction, “1” is read from this bit.

[bit 13] INTE (Interrupt enable)

This bit enables and disables the interrupt.

- 0: Interrupt disabled
- 1: Interrupt enabled

When using EI²OS or μ DMA, set this bit (EI²OS or μ DMA is started by an interrupt request).

This bit is initialized to 0 at reset.

[bit 12] PAUS (A/D converter pause)

This bit is set when A/D conversion pauses.

Only one register is available for storing the A/D conversion result. Therefore, unless the conversion results are transferred by the EI²OS (or μ DMA), the result data would be continuously updated and destroyed in continuous conversion.

To prevent the above condition, the system is so designed as not to store the next converted-data when a conversion result is not transferred yet (INT = “1”), with INTE = “1”. A/D conversion pauses during that period. When transfer by EI²OS (or μ DMA) ends or when “0” is set to INT bit, A/D conversion is resumed. Once this bit is set, it is not cleared by itself. Write “0” to clear this bit.

This bit is valid only when EI²OS (or μ DMA) is used.

Notes:

- Regarding the convert data protection function, see Section 18.6.
- This bit is initialized to “0” at reset.

[bit 11 and bit 10] STS1 and STS0 (Start source select)

These bits are initialized to “00_B” by reset.

Select an A/D conversion activation cause by setting these bits.

Table 18.3-2 Function Settings

STS1	STS0	Function
0	0	Activation A/D conversion by software
0	1	Activation A/D conversion by external pin trigger and by software
1	0	Activation A/D conversion by timer and by software
1	1	Activation A/D conversion by external pin trigger, timer, and by software

In a mode allowing two or more activation causes, A/D conversion is activated by the source that occurs first. When changing the setting of these bits during A/D conversion, the result is immediately reflected. Therefore it is not a recommended practice.

Notes:

- An external pin trigger is detected by the falling edge.
When the external trigger input level is Low, setting external pin triggers starting may start A/D converter.
- When the timer is selected, the 16-bit reload timer 1 is selected and the conversion is activated when the output from the timer becomes “1”.

[bit 9] STRT (Start)

A/D conversion is started by writing “1” to this bit.

To reactivate A/D conversion, write “1” to this bit again.

At reset, this bit is initialized to “0”.

Reading this bit always returns “1”. Using an read-modify-write instruction, “0” is read from this bit. Reactivation during the operation is only available in the single mode 1, but not supported in the single mode2, continuous mode and the stop mode. Check the BUSY bit before writing “1” to this bit in the latter three modes.

Do not force to stop A/D conversion concurrently with activation of A/D conversion by software (BUSY=0, STRT=1).

[bit 8] Unused bit

Writing to this bit has no effect.

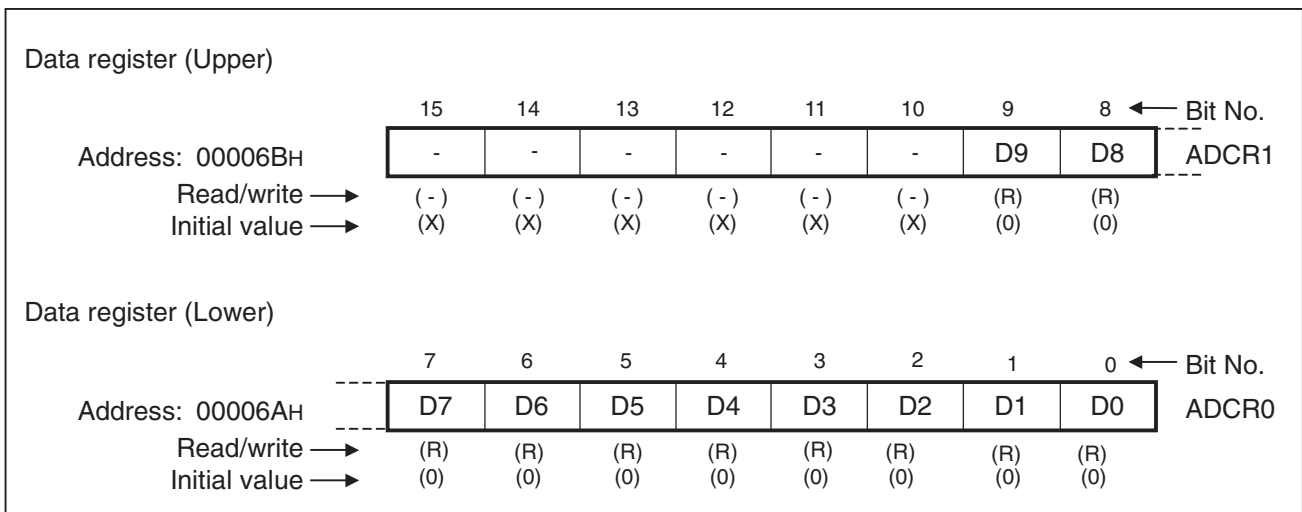
Reading this bit always returns “1”.

18.3.3 Data Register (ADCR0, ADCR1)

The data register (ADCR0, ADCR1) is used to store digital value generated as a result of conversion. ADCR0 stores lower 8 bits; ADCR1 stores most significant 2 bits of the conversion result. These register's values are rewritten every time conversion ends. Normally, the last converted value is stored in these register's bits.

■ Data register (ADCR0, ADCR1)

Figure 18.3-4 Data register (ADCR0, ADCR1)



Reading bit 10 to bit 15 of ADCR1 always returns “1”.

When S10 bit of ADCS0 is “1”, the 8-bit mode is established, storing converted data in bit7 to bit0. In this case, Reading bit9 to bit8 always returns “1”.

For the use of the conversion data protection function, see Section 18.4.

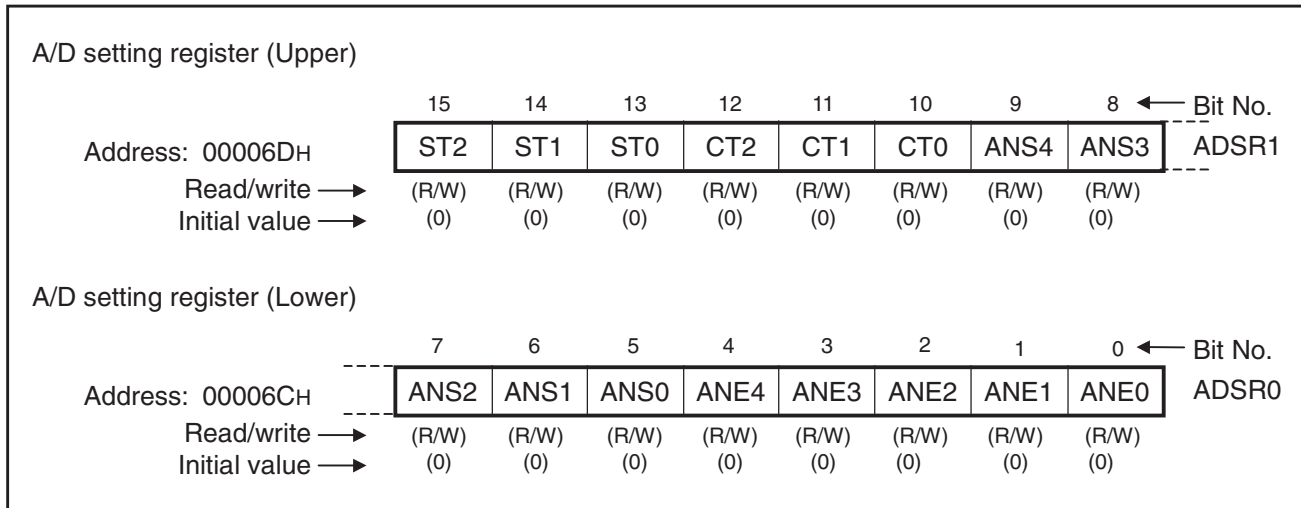
Do not write to this register.

18.3.4 Setting Register (ADSR0, ADSR1)

The setting register (ADSR) is used to set the A/D conversion time and the sampling channels and to indicate the current sampling channel.

■ Setting Register (ADSR)

Figure 18.3-5 Setting register (ADSR)



[bit 15 to bit 13] ST2, ST1, ST0 (Sampling time)

Table 18.3-3 Function Settings

ST2	ST1	ST0	Function
0	0	0	4-machine cycle (8 MHz, 0.5 μ s)
0	0	1	6-machine cycle (8 MHz, 0.75 μ s)
0	1	0	8-machine cycle (16 MHz, 0.5 μ s)
0	1	1	12-machine cycle (24 MHz, 0.5 μ s)
1	0	0	24-machine cycle (8 MHz, 3 μ s)
1	0	1	36-machine cycle (16 MHz, 2.25 μ s)
1	1	0	48-machine cycle (16 MHz, 3.0 μ s)
1	1	1	128-machine cycle (24 MHz, 5.3 μ s)

These bits define the sampling time for the analog input signal. Setting for these bits must be carefully evaluated with the following conditions. Otherwise, the result may not provide the guaranteed conversion accuracy.

If the driving impedance, R_{ext} is 1.5 k Ω or less, the sampling time must be greater than equal to;

0.5 μ s; For 4.5V $\leq$ Avcc $\leq$ 5.5V

1.2μs; For 4.0V ≤ Avcc < 4.5V

If the driving impedance, R_{ext} is greater than 1.5 kΩ, the sampling time must be greater than Tsamp given by the following formula;

$$T_{\text{samp}} = (2.25 \text{ k}\Omega + R_{\text{ext}}) \times 10.7 \text{ pF} \times 7; \text{ For } 4.5\text{V} \leq A_{\text{vcc}} \leq 5.5\text{V}$$

$$T_{\text{samp}} = (13.6 \text{ k}\Omega + R_{\text{ext}}) \times 10.7 \text{ pF} \times 7; \text{ For } 4.0\text{V} \leq A_{\text{vcc}} < 4.5\text{V}$$

[bit 12 and bit 10] CT2, CT1, CT0 (Compare time)

Table 18.3-4 Function Settings

CT2	CT1	CT0	Function
0	0	0	22-machine cycle (8 MHz, 2.8 μs)
0	0	1	33-machine cycle (16 MHz, 2.1 μs)
0	1	0	44-machine cycle (20 MHz, 2.2 μs)
0	1	1	66-machine cycle (24 MHz, 2.8 μs)
1	0	0	88-machine cycle (8 MHz, 11.0 μs)
1	0	1	132-machine cycle (16 MHz, 8.3 μs)
1	1	0	176-machine cycle (20 MHz, 8.8 μs)
1	1	1	264-machine cycle (24 MHz, 11.0 μs)

These bits define the conversion time of the successive approximation. Setting for these bits must meet the following conditions. Otherwise, the result may not provide the guaranteed conversion accuracy.

$$\geq 1.0\mu\text{s}; \text{ For } 4.5\text{V} \leq A_{\text{vcc}} \leq 5.5\text{V}$$

$$\geq 2.0\mu\text{s}; \text{ For } 4.0\text{V} \leq A_{\text{vcc}} < 4.5\text{V}$$

[bit 9 to bit 5] ANS4, ANS3, ANS2, ANS1, ANS0 (Analog start channel set)

These bits are used to set the starting channel for A/D conversion, and to indicate the current analog input channel.

When the A/D converter is activated, A/D conversion is started from the channel selected by these bits.

Table 18.3-5 Starting Channel Settings (1/2)

ANS4	ANS3	ANS2	ANS1	ANS0	Starting channel
0	0	0	0	0	AN0
0	0	0	0	1	AN1
0	0	0	1	0	AN2
0	0	0	1	1	AN3
• • •					
1	1	1	0	0	AN28

Table 18.3-5 Starting Channel Settings (Continued) (2/2)

ANS4	ANS3	ANS2	ANS1	ANS0	Starting channel
1	1	1	0	1	AN29
1	1	1	1	0	AN30
1	1	1	1	1	AN31

- Read operation

These bits indicate the current analog input channel for the conversion operation. When the conversion operation is stopped, the last channel is indicated.

And before A/D conversion starts, the previous conversion channel will be read even if these bits have already been set to the new value.

- At reset: These bits are initialized to “00000_B”.

[bit 4 to bit 0] ANE4, ANE3, ANE2, ANE1, ANE0 (Analog end channel set)

These bits are used to set the ending channel for A/D conversion.

Table 18.3-6 Ending Channel Settings

ANE4	ANE3	ANE2	ANE1	ANE0	Ending channel
0	0	0	0	0	AN0
0	0	0	0	1	AN1
0	0	0	1	0	AN2
0	0	0	1	1	AN3
• • •					
1	1	1	0	0	AN28
1	1	1	0	1	AN29
1	1	1	1	0	AN30
1	1	1	1	1	AN31

- At reset, these bits are initialized to “00000_B”.

Note:

- When writing to this register, always use word access. When byte write or read-modify-write is performed for this register, A/D conversion may be started from an unintended channel.
 - When the same channel is written to the ANE4 to ANE0 bits and to the ANS4 to ANS0 bits, conversion is performed for only 1 channel (single channel conversion).
 - When conversion of the Analog end channel set by ANE4 to ANE0 is ended, with the continuous mode or the stop mode, control returns to the Analog start channel set by the ANS4 to ANS0 bits.
 - When the ANS > ANE is set for channel, conversion starts from the Analog start channel and when conversion of AN31 ends, and then returns to the AN0 and ends in the Analog end channel. In this case, the conversion results could be incorrect, because analog input AN16 to AN31(for the products without 'C'-suffix) or AN24 to AN31(for the products with 'C'-suffix) do not exist. To prevent this operation, do not set ANS larger than ANE.
 - Do not set number of channel that doesn't exist on the device to ANEx bits or ANSx bits.
 - the products without 'C'-suffix: 0 to 15 can be set to ANEx bits or ANSx bits
 - the products with 'C'-suffix: 0 to 23 can be set to ANEx bits or ANSx bits
-

18.4 Operation of A/D Converter

The A/D converter operates using the sequential-comparison converter system; and 10 bits or 8 bits can be selected for the A/D converter's resolution.

Since this A/D converter has only one register (10 bits; conversion result data register ADCR0 and ADCR1) for storing conversion results, this register is rewritten every time conversion is ended. So, this A/D converter alone is not suitable for continuous conversion; and therefore conversion be performed with transferring converted data to the memory using the extended intelligent I/O function (EI²OS) or using μ DMA is recommended.

■ Single mode 1 / 2

In the single mode, analog inputs from the starting channel set by the ANS bits to the ending channel set by the ANE bits are sequentially converted; and when conversion of the ending channel is ended, A/D conversion is stopped. When the starting channel and the ending channel are the same (ANS = ANE), conversion is performed for only the channel set by the ANS bits.

Example:

ANS = 00000_B, ANE = 00011_B:

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

ANS = 00010_B, ANE = 00010_B:

Start --> AN2 --> End

■ Continuous mode

In the continuous mode, analog inputs from the starting channel set by the ANS bits to the ending channel set by the ANE bits are sequentially converted; and when conversion of the ending channel is ended, control returns to the starting channel to continue A/D conversion. When the starting channel and the ending channel are the same (ANS = ANE), conversion is continued for only the channel set by the ANS bits.

Example:

ANS = 00000_B, ANE = 00011_B:

Start --> AN0 --> AN1 --> AN2 --> AN3 --> AN0 -->-->--> Repetition of sequence

ANS = 00010_B, ANE = 00010_B:

Start --> AN2 --> AN2 --> AN2 -->-->--> Repetition of sequence

In the continuous mode, A/D conversion is repeated until "0" is written to the BUSY bit (when "0" is written to the BUSY bit, operation of the A/D converter stops forcibly).

Note that when operation of the A/D converter stops forcibly, A/D conversion stops halfway. In this case, the conversion result register contains the previous data for which conversion completed.

■ Stop mode

In the stop mode, analog inputs from the starting channel set by the ANS bits to the ending channel set by the ANE bits are sequentially converted, but every time one channel is converted, conversion stops.

When conversion of the ending channel is ended, control returns to the starting channel to continue A/D

conversion. When the starting channel and the ending channel are the same ($ANS = ANE$), conversion is performed for only one channel.

Example:

$ANS = 00000_B$, $ANE = 00011_B$:

Start --> AN0 --> Stop --> Start --> AN1 --> Stop --> Start --> AN2 --> Stop --> Start --> AN3 --> Stop
--> Start --> AN0 -->-->--> Repetition of sequence

$ANS = 00010_B$, $ANE = 00010_B$:

Start --> AN2 --> Stop --> Start --> AN2 --> Stop --> Start --> AN2 -->-->--> Repetition of sequence

In the above sequences, only A/D activation causes set using the STS1 and STS0 bits are valid.

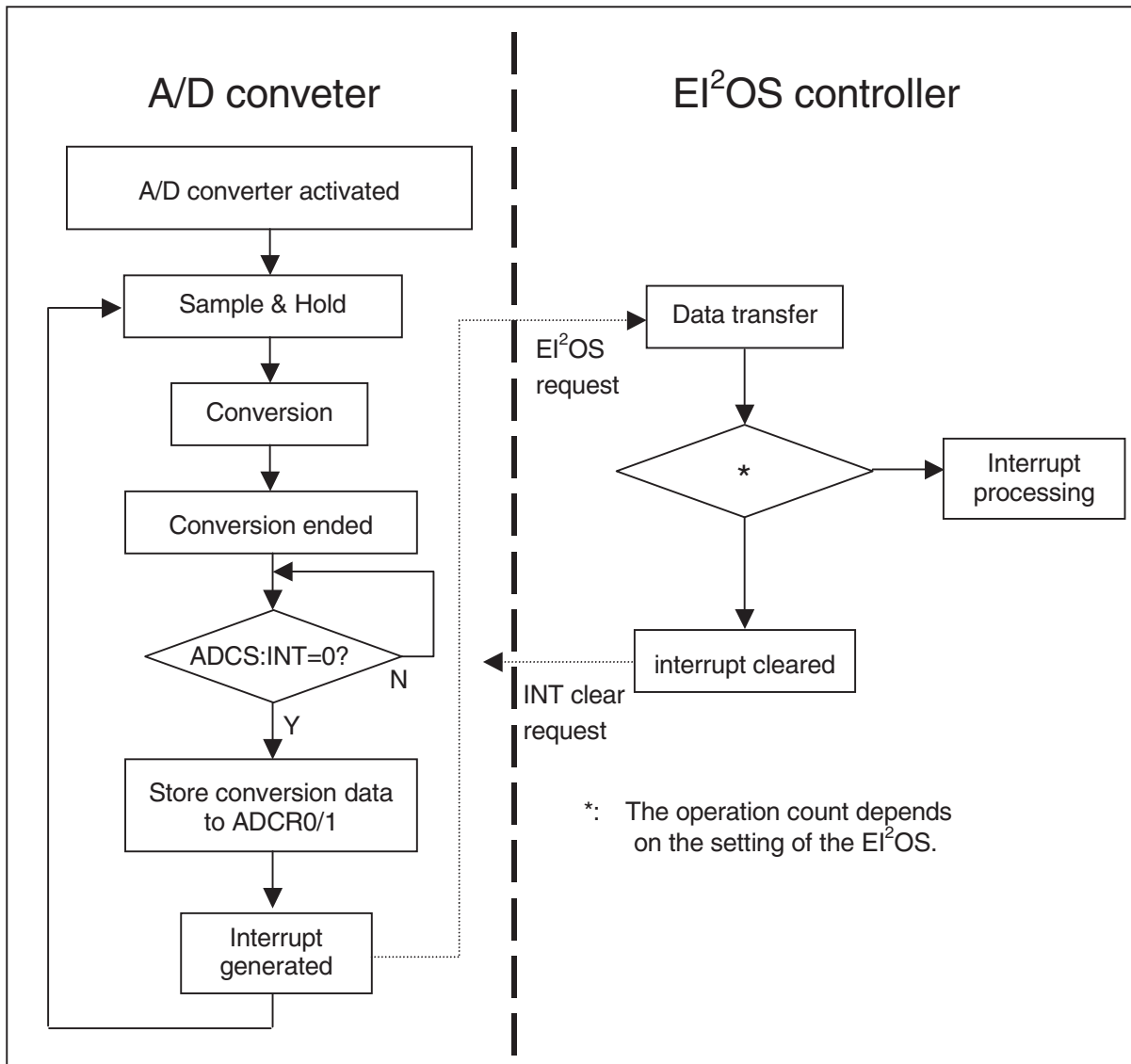
The start of conversion can be synchronized by this mode.

18.5 Conversion Using EI²OS

Figure 18.5-1 gives an example of flow (continuous mode) from the activation of A/D conversion to the transfer of converted data.

■ Conversion using EI²OS

Figure 18.5-1 Example of flow (continuous mode) from activation of A/D conversion to transfer of converted data



18.5.1 Example of activating of EI²OS in single mode

In the single mode, EI²OS is activated in the following procedure:

- Convert analog inputs AN1 to AN3 and then end the conversion.
- Transfer sequentially converted data to the 200_H to 205_H addresses.
- Activate conversion using software.
- Use the highest interrupt level.

■ Example of starting of EI²OS in single mode

Table 18.5-1 Operation Description

Setting item	Sample program	Operation description
Setting of EI ² OS	MOV ICR03, #08 _H	Set the highest interrupt level, start EI ² OS at the interrupt, and set the descriptor address.
	MOV BAPL, #00 _H	Transfer destination address for converted data
	MOV BAPM, #02 _H	
	MOV BAPH, #00 _H	
	MOV ISCS, #18 _H	Transfer word data, and then increment the transfer destination address. Transfer converted data from I/O to the memory. Don't terminate transfer due to a request from the resource.
	MOV IOAL, #6A _H	Set the A/D converter result register
	MOV IOAH, #00 _H	
	MOV DCTL, #03 _H	Performs EI ² OS transfer three times (the number of conversion times)
	MOV DCTH, #00 _H	
Setting of A/D converter	MOV ADCS0, #00 _H	Single mode1
	MOVW ADSR0, #6823 _H	The sampling time is 0.6 μs; the comparison operation time is 2.2 μs (as machine CLK is 20 MHz); the starting channel is AN1; the ending channel is AN3.
	MOV ADCS1, #A2 _H	Start A/D conversion using software.
Interrupt sequence	MOV ADCS1, #80 _H	
	RETI	Return from the interrupt.

ICR03: Interrupt control register

BAPL: Buffer address pointer (lower)

BAPM: Buffer address pointer (middle)

BAPH: Buffer address pointer (upper)

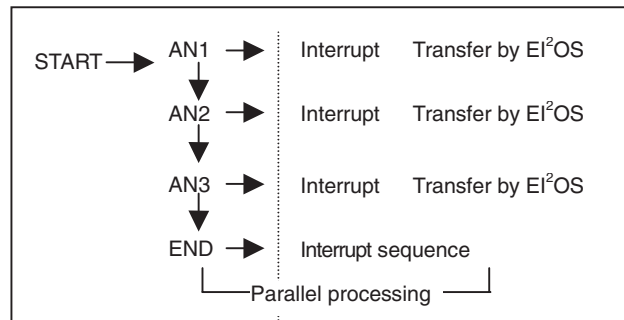
ISCS: EI²OS status register

IOAL: I/O address counter (lower)

DCTL: Data counter (lower)

IOAH: I/O address counter (upper)

DCTH: Data counter (upper)

Figure 18.5-2 Example of starting of EI²OS in single mode

18.5.2 Example of activating of EI²OS in continuous mode

An example of the activating of EI²OS in the continuous mode is given below.

- Convert analog inputs AN3 to AN5 and obtain two converted data for each channel.
- Transfer sequentially converted data to the 600_H to 60B_H addresses.
- Start conversion using an external edge input.
- Use the highest interrupt level.

■ Example of starting of EI²OS in continuous mode

Table 18.5-2 Operation Description

Setting item	Sample program	Operation description
Setting of EI ² OS	MOV ICR03, #08 _H	Set the highest interrupt level, start EI ² OS at the interrupt, and set the descriptor address.
	MOV BAPL, #00 _H	Transfer destination address for converted data
	MOV BAPM, #06 _H	
	MOV BAPH, #00 _H	
	MOV ISCS, #18 _H	Transfer word data, and then increment the transfer destination address. Transfer converted data from I/O to the memory. Don't terminate transfer due to a request from the resource.
	MOV IOAL, #6A _H	Transfer source address
	MOV IOAH, #00 _H	
	MOV DCTL, #06 _H	Transfer converted data by EI ² OS six times data for '3 channels × 2'.
	MOV DCTH, #00 _H	
Setting of A/D converter	MOV ADCS0, #80 _H	Continuous mode
	MOVW ADSR0, #6865 _H	The sampling time is 0.6 μs; the comparison operation time is 2.2 μs (with machine CLK is 20 MHz); the starting channel is AN3; the ending channel is AN5.
	MOV ADCS1, #A4 _H	Start A/D conversion using external edge.
EI ² OS end interrupt sequence	MOV ADCS1, #84 _H	
	RETI	Return from the interrupt.

ICR03: Interrupt control register

BAPL: Buffer address pointer (lower)

BAPM: Buffer address pointer (middle)

BAPH: Buffer address pointer (upper)

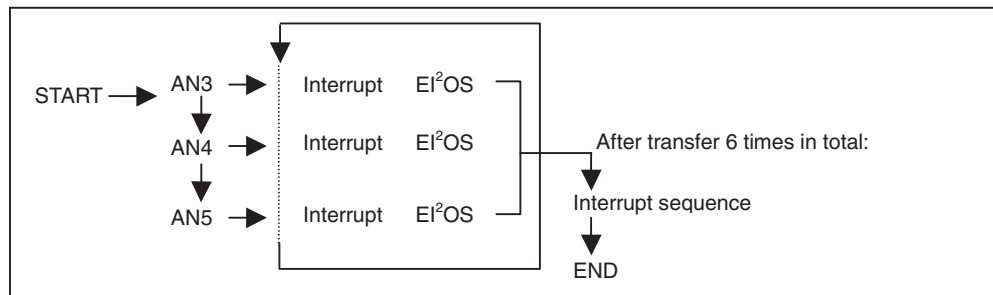
ISCS: EI²OS status register

IOAL: I/O address counter (lower)

DCTL: Data counter (lower)

IOAH: I/O address counter (upper)

DCTH: Data counter (upper)

Figure 18.5-3 Example of starting of EI²OS in continuous mode

18.5.3 Example of activating of EI²OS in stop mode

An example of the activating of EI²OS in the stop mode is given below.

- Convert the analog input AN3 12 times at regular intervals.
- Transfer sequentially converted data to the 600_H to 617_H addresses.
- Start conversion using an external edge input.
- Use the highest interrupt level.

■ Example of starting of EI²OS in stop mode

Table 18.5-3 Operation Description

Setting item	Sample program	Operation description
Setting of EI ² OS	MOV ICR03, #08 _H	Set the highest interrupt level, start EI ² OS at the interrupt time, and set the descriptor address.
	MOV BAPL, #00 _H	Transfer destination address for converted data
	MOV BAPM, #06 _H	
	MOV BAPH, #00 _H	
	MOV ISCS, #18 _H	Transfer word data, and then increment the transfer destination address. Transfer converted data from I/O to the memory. Don't terminate transfer due to a request from the resource.
	MOV IOAL, #6A _H	Transfer source address
	MOV IOAH, #00 _H	
	MOV DCTL, #0C _H	Transfer converted data by EI ² OS 12 times.
	MOV DCTH, #00 _H	
Setting of A/D converter	MOV ADCS0, #C0 _H	Stop mode
	MOVW ADSR0, #6863 _H	The sampling time is 0.6 μs; the comparison operation time is 2.2 μs (with machine CLK is 20 MHz); the starting channel is AN3; the ending channel is AN3 (1 channel is converted).
	MOV ADCS1, #A4 _H	Start A/D conversion using the external edge.
EI ² OS end interrupt sequence	MOV ADCS1, #84 _H	
	RETI	Return from the interrupt.

ICR03: Interrupt control register

BAPL: Buffer address pointer (lower)

BAPM: Buffer address pointer (middle)

BAPH: Buffer address pointer (upper)

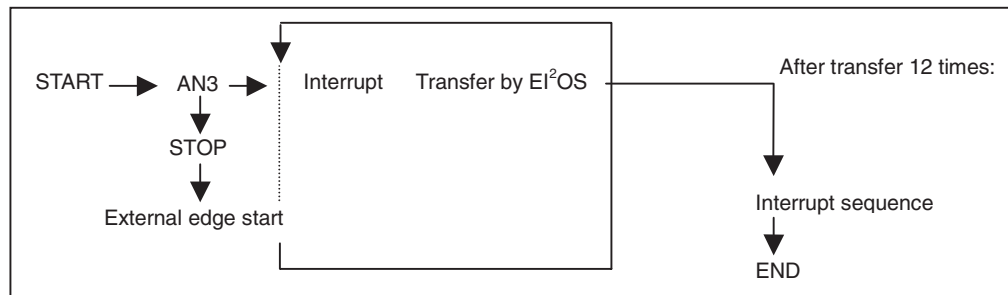
ISCS: EI²OS status register

IOAL: I/O address counter (lower)

DCTL: Data counter (lower)

IOAH: I/O address counter (upper)

DCTH: Data counter (upper)

Figure 18.5-4 Example of starting of EI²OS in stop mode

18.6 Convert Data Protection Function

The A/D converter has the converted-data protection function, is featured by continuous conversion and securing two or more data using EI²OS (or μ DMA). Only one conversion data register (ADCR0/1) is provided, so when A/D conversion is continuously performed, converted data is stored every time one conversion is ended, destroying the previous data. To prevent this, the A/D converter has a function that conversion data is not stored in the register and the A/D conversion pauses even when conversion is ended unless the previous data is transferred to the memory using EI²OS (or μ DMA).

■ Converted-data protection function

The pause operation is cancelled after converted data is transferred to the memory via EI²OS (or μ DMA).

When the previous data is already transferred to the memory, A/D conversion is continuously performed without a pause.

Note:

This function is related to the INT and INTE bits of ADCS1 register.

The data protection function operates only in the interrupt enabled state (INTE = 1).

In the interrupt disabled state (INTE = 0), this function does not operate. In this state when A/D conversion is continuously performed, converted data is successively stored in the register, destroying old data.

When EI²OS (or μ DMA) is not used in the interrupt enabled state (INTE = 1), the INT bit is not cleared automatically, so the data protection functions works, placing A/D conversion in a pause. In this case, when the INT bit is cleared (by writing “0” to the bit) in an interrupt sequence, the pause is cancelled.

When interrupts are disabled (by writing “0” to INTE bit), with EI²OS (or μ DMA) operating and with the A/D converter pausing, A/D converter resume conversion and the contents of the converted-data register may change before it is transferred to memory.

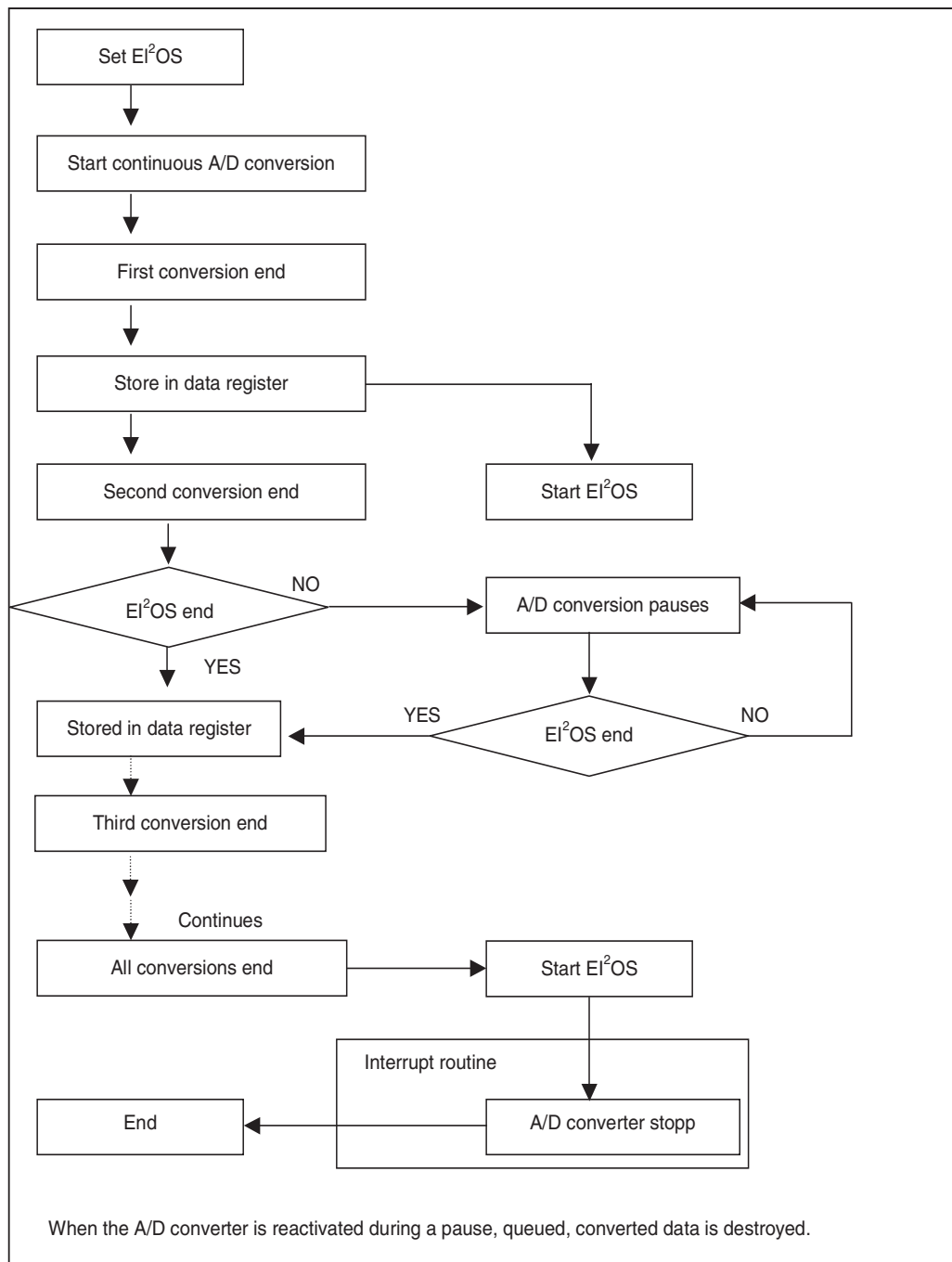
When A/D conversion is reactivated during a pause (by writing “1” to STRT bit in the single mode1), pending conversion result is destroyed.

When the data protection function works to cause a pause, the PAUS bit is set.

The PAUS bit is not cleared by itself. Write “0” to clear this bit.

■ Example of flow of data protection function (when EI²OS is used)

Figure 18.6-1 Example of flow of data protection function (when EI²OS is used)



■ Cautions

To select the external trigger or the internal timer as the activation causes of A/D converted, the A/D start source select bits (STS1 and STS0 bits) of ADCS1 register are used. In this case, ensure that the input value of an external trigger or of the internal timer are “inactive”. If the values are “active”, A/D conversion may start operating immediately.

When setting the STS1 bit and STS0 bit, always set the ADTG pin to “1” (input) and set the internal

timer (reload timer 1) to “0” (output).

CHAPTER 19

CLOCK MONITOR FUNCTION

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

19.1 “Overview of the Clock Monitor Function”

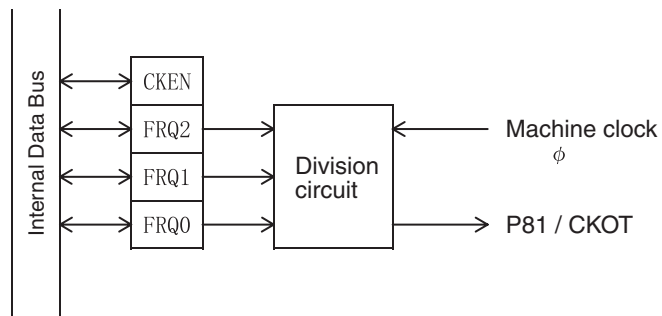
19.2 “Clock Output Enable Register (CLKR)”

19.1 Clock Monitor Function

The clock monitor function is output a division clock (a clock for monitoring) from the clock monitor CKOT pin.

■ Block Diagram of the Clock Monitor

Figure 19.1-1 Block Diagram of the Clock Monitor



19.2 Clock Output Enable Register (CLKR)

The clock output enable register (CLKR) selects the CKOT output enable and the clock output frequency.

■ Clock Output Enable Register (CLKR)

Figure 19.2-1 Clock Output Enable Register (CLKR)

Clock output enable register	7	6	5	4	3	2	1	0	↔ Bit No.
Address: 796CH	—	—	—	—	CKEN	FRQ2	FRQ1	FRQ0	CLKR
Read/write ↔	(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↔	(X)	(X)	(X)	(X)	(0)	(0)	(0)	(0)	

[bit 7 to 4] Undefined bit

Reading these bits always returns “1”.

[bit 3] CKEN

This is the CKOT output enable bit.

Table 19.2-1 CKEN Bit Function

CKEN	Function of P81/CKOT pin
0	Normal port
1	CKOT output

[bit 2 to 0] FRQ2, FRQ1, FRQ0

This bit is used to select the clock output frequency.

Table 19.2-2 FRQ2, FRQ1, FRQ0 (Clock Output Frequency Selection Bit) Function (1/2)

FRQ2	FRQ1	FRQ0	Output clock	$\phi=24$ MHz	$\phi=16$ MHz	$\phi=8$ MHz
0	0	0	$\phi/2^1$	83 ns	125 ns	250 ns
0	0	1	$\phi/2^2$	167 ns	250 ns	500 ns
0	1	0	$\phi/2^3$	333 ns	500 ns	1 μ s
0	1	1	$\phi/2^4$	667 ns	1 μ s	2 μ s
1	0	0	$\phi/2^5$	1.3 μ s	2 μ s	4 μ s
1	0	1	$\phi/2^6$	2.7 μ s	4 μ s	8 μ s

Table 19.2-2 FRQ2, FRQ1, FRQ0 (Clock Output Frequency Selection Bit) Function (2/2)

FRQ2	FRQ1	FRQ0	Output clock	$\phi=24$ MHz	$\phi=16$ MHz	$\phi=8$ MHz
1	1	0	$\phi/2^7$	5.3 μ s	8 μ s	16 μ s
1	1	1	$\phi/2^8$	10.7 μ s	16 μ s	32 μ s

CHAPTER 20

UART

This chapter explains the functions and operation of UART.

- 20.1 “Overview of UART”
- 20.2 “Configuration of UART”
- 20.3 “UART Pins”
- 20.4 “UART Registers”
- 20.5 “UART Interrupts”
- 20.6 “UART Baud Rates”
- 20.7 “Operation of UART”
- 20.8 “Notes on Using UART”

20.1 Overview of UART

The UART with LIN (Local Interconnect Network) - Function is a general-purpose serial data communication interface for performing synchronous or asynchronous communication with external devices. UART provides bidirectional communication function (normal mode), master-slave communication function (multiprocessor mode in master/slave systems), and special features for LIN-bus systems (working both as master or as slave device).

Please note that UART is not software compatible to the other UARTs.

The MB90340 series contain up to five UARTs.

■ UART functions

● UART functions

UART is a general-purpose serial data communication interface for transmitting serial data to and receiving data from another CPU and peripheral devices. It has the functions listed in Table 20.1-1 “UART functions”.

Table 20.1-1 UART functions (1/2)

Item	Function
Data buffer	Full-duplex
Serial Input	5 times oversampling in asynchronous mode
Transfer mode	• Clock synchronous (start-stop synchronization and start-stop-bit-option) • Clock asynchronous (using start-, stop-bits)
Baud rate	• A dedicated baud rate generator is provided, which consists of a 15-bit-reload counter • An external clock can be input and also be adjusted by the reload counter
Data length	• 7 bits (not in synchronous or LIN mode) • 8 bits
Signal mode	Non-return to zero (NRZ)
Start bit timing	Clock synchronization to the falling edge of the start bit in asynchronous mode
Reception error detection	• Framing error • Overrun error • Parity error (Not supported in Mode 1)

Table 20.1-1 UART functions (2/2)

Item	Function
Interrupt request	• Reception interrupt (reception complete, reception error detect, LIN-Synch-break detect) • Transmission interrupt (transmission data empty) • Interrupt request to ICU (LIN synch field detection: LSYN) • Both transmission and reception support for extended intelligent I/O service (EI²OS) and DMA function.
Master-slave communication function (multiprocessor mode)	One-to-n communication (one master to n slaves) (This function is supported both for master and slave system).
Synchronous mode	Function as Master- or Slave-UART
Transceiving pins	Direct access possible
LIN bus options	• Operation as master device • Operation as slave device • Generation of LIN-Synch-break • Detection of LIN-Synch-break • Detection of start/stop edges in LIN-Synch-field connected to ICU 0, 1, 6 and 7
Synchronous serial clock	The synchronous serial clock can be output continuously on the SCK pin for synchronous communication with start & stop bits
Clock delay option	Special synchronous Clock Mode for delaying clock (useful for SPI)

■ UART operation modes

The UART operates in four different modes, which are determined by the MD0- and the MD1-bit of the Serial mode register (SMR). Mode 0 and 2 are used for bidirectional serial communication, mode 1 for master/slave communication and mode 3 for LIN master/slave communication.

Table 20.1-2 UART operation modes

Operation mode		Data length		Synchronization of mode	Length of stop bit	data bit direction ^{*1}
		parity disabled	parity enabled			
0	normal mode	7 or 8		asynchronous	1 or 2	L/M
1	multiprocessor	7 or 8 + 1 ^{*2}	--	asynchronous	1 or 2	L/M
2	normal mode	8		synchronous	0, 1 or 2	L/M
3	LIN mode	8	--	asynchronous	1	L

*1: means the data bit transfer format: LSB or MSB first.

*2: "+1" means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

Note:

Mode 1 operation is supported both for master or slave operation of UART in a master-slave connection system. In Mode 3 the UART function is locked to 8N1-Format, LSB first.

If the mode is changed, UART cuts off all possible transmission or reception and awaits then new action.

The MD1 and MD0 bit of the Serial Mode Register (SMR) determine the operation mode of UART as shown in the following table:

Table 20.1-3 Mode bit setting

MD1	MD0	Mode	Description
0	0	0	Asynchronous (normal mode)
0	1	1	Asynchronous (multiprocessor mode)
1	0	2	Synchronous (normal mode)
1	1	3	Asynchronous (LIN mode)

■ UART interrupt and EI²OS

Table 20.1-4 UART interrupt and EI²OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI ² OS	DMA channel number
		Register name	Address	Lower	Upper	Bank		
UART0 reception interrupt	#35(23 _H)	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	*1	DRQ10
UART0 transmission interrupt	#36(24 _H)	ICR12	0000BC _H	FFFF6C _H	FFFF6D _H	FFFF6E _H	*2	DRQ11
UART1/UART3 reception interrupt	#37(25 _H)	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	*1	DRQ12
UART1/UART3 transmission interrupt	#38(26 _H)	ICR13	0000BD _H	FFFF64 _H	FFFF65 _H	FFFF66 _H	*2	DRQ13
UART2/UART4 reception interrupt	#39(27 _H)	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	*1	DRQ14
UART2/UART4 transmission interrupt	#40(28 _H)	ICR14	0000BE _H	FFFF5C _H	FFFF5D _H	FFFF5E _H	*2	DRQ15

*1: EI²OS service is usable if the other interrupt (s) which shares the same interrupt vector is (are) not enabled. Detection of receive errors is possible and stop request for EI²OS service is supported.

*2: EI²OS service is usable if the other interrupt (s) which shares the same interrupt vector is (are) not enabled.

20.2 Configuration of UART

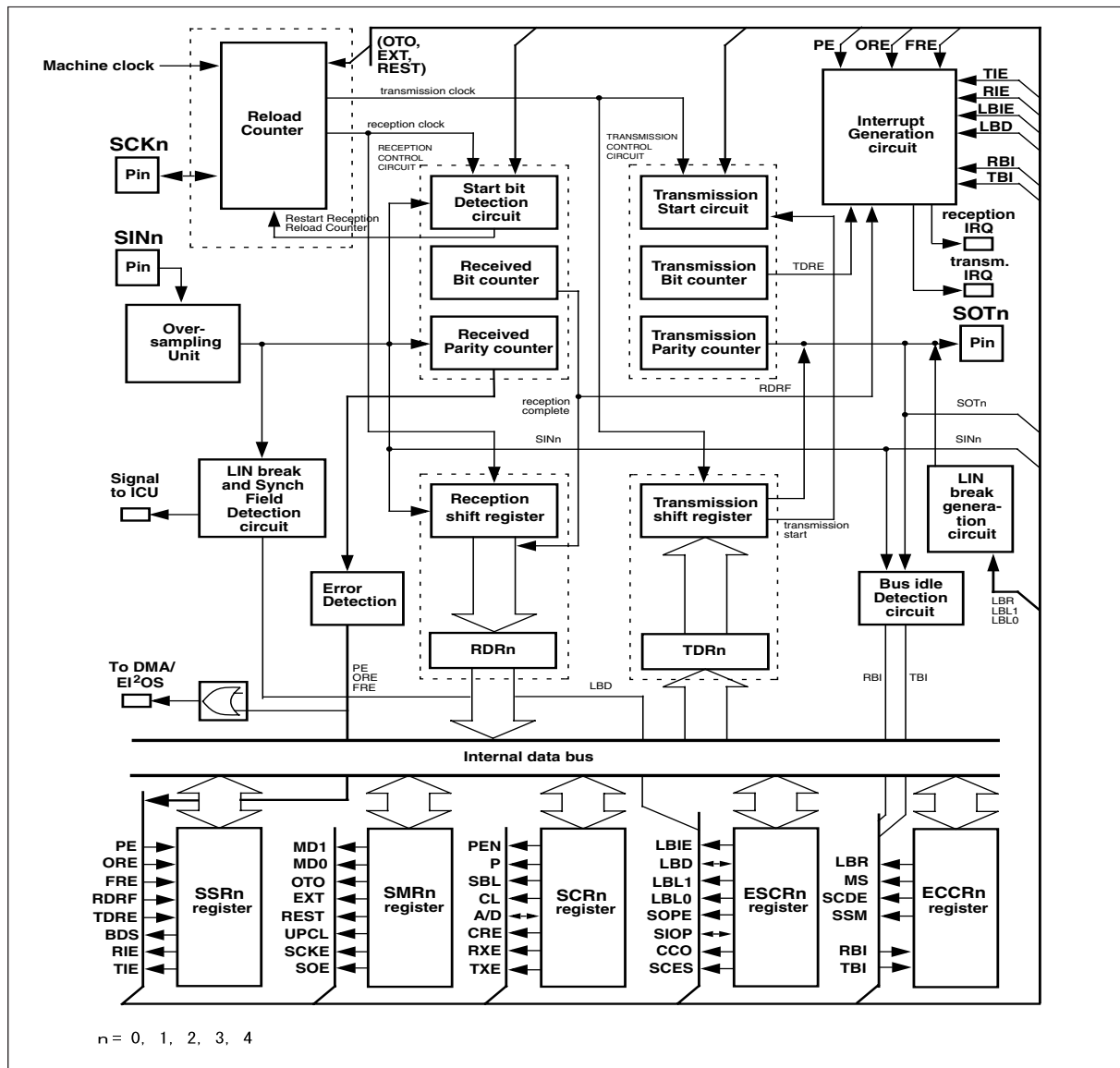
This section provides a short overview on the building blocks of UART.

■ Block diagram of UART

UART consists of the following blocks:

- Reload Counter
- Reception Control Circuit
- Reception Shift Register
- Reception Data Register (RDR)
- Transmission Control Circuit
- Transmission Shift Register
- Transmission Data Register (TDR)
- Error Detection Circuit
- Oversampling Unit
- Interrupt Generation Circuit
- LIN Synch Break/Synch Field Detection
- Bus Idle Detection Circuit
- LIN-UART Serial Mode Register (SMR)
- Serial Control Register (SCR)
- Serial Status Register (SSR)
- Extended Com. Contr. Reg. (ECCR)
- Extended Status/Contr. Reg. (ESCR)

Figure 20.2-1 Block diagram of UART



■ Explanation of the different blocks

- Reload Counter

The reload counter functions as the dedicated baud rate generator. It can select external input clock or internal clock for the transmitting and receiving clocks. The reload counter has a 15 bit register for the reload value. The actual count of the transmission reload counter can be read via the BGRn0/n1.

- Reception Control Circuit

The reception control circuit consists of a received bit counter, start bit detection circuit, and received parity counter. The received bit counter counts reception data bits. When reception of one data item for the specified data length is complete, the received bit counter sets the Reception data register full flag. The start bit detection circuit detects start bits from the serial input signal and sends a signal to the reload counter to synchronize it to the falling edge of these start bits. The reception parity counter calculates the parity of the reception data.

● **Reception Shift Register**

The reception shift register fetches reception data input from the SINn pin, shifting the data bit by bit. When reception is complete, the reception shift register transfers receive data to the RDR register.

● **Reception Data Register**

This register retains reception data. Serial input data is converted and stored in this register.

● **Transmission Control Circuit**

The transmission control circuit consists of a transmission bit counter, transmission start circuit, and transmission parity counter. The transmission bit counter counts transmission data bits. When the transmission of one data item of the specified data length is complete, the transmission bit counter sets the Transmission data register full flag. The transmission start circuit starts transmission when data is written to TDR. The transmission parity counter generates a parity bit for data to be transmitted if parity is enabled.

● **Transmission Shift Register**

The transmission shift register transfers data written to the TDR register to itself and outputs the data to the SOTn pin, shifting the data bit by bit.

● **Transmission Data Register (TDR)**

This register sets transmission data. Data written to this register is converted to serial data and output.

● **Error Detection Circuit**

The error detection circuit checks if there was any error during the last reception. If an error has occurred it sets the corresponding error flags.

● **Oversampling Unit**

The oversampling unit oversamples the incoming data at the SINn pin for five times. It is switched off in synchronous operation mode.

● **Interrupt Generation Circuit**

The interrupt generation circuit administers all cases of generating a reception or transmission interrupt. If a corresponding enable flag is set and an interrupt case occurs the interrupt will be generated immediately.

● **LIN synch Break and Synchronization Field Detection Circuit**

The LIN break and LIN synchronization field detection circuit detects a LIN break, if a LIN master node is sending a message header. If a LIN break is detected a special flag bit is generated. The first and the fifth falling edge of the synchronization field is recognized by this circuit by generating an internal signal for the Input Capture Unit to measure the actual serial clock time of the transmitting master node.

● **LIN Synch Break Generation Circuit**

The LIN break generation circuit generates a LIN break of a determined length.

● Bus Idle Detection circuit

The bus idle detection circuit recognizes if neither reception nor transmission is going on. In this case, the circuit generates the special flag bits TBI and RBI.

● LIN-UART Serial Mode Register (SMR)

This register performs the following operations:

- Selecting the LIN-UART operation mode
- Selecting a clock input source
- Selecting if an external clock is connected “one-to-one” or connected to the reload counter
- Resetting dedicated reload timer
- Resetting the LIN-UART (preserving the settings of the registers)
- Specifying whether to enable serial data output to the corresponding pin
- Specifying whether to enable clock output to the corresponding pin

● Serial Control Register (SCR)

This register performs the following operations:

- Specifying whether to provide parity bits
- Selecting parity bits
- Specifying a stop bit length
- Specifying a data length
- Selecting a frame data format in mode 1
- Clearing the error flags
- Specifying whether to enable transmission
- Specifying whether to enable reception

● Serial Status Register (SSR)

This register performs the following functions

- Indicating status of receive/transmit operations and errors
- Specifying LSB first or MSB first
- Receive interrupt enable/disable
- Transmit interrupt enable/disable

● Extended Status/Control Register (ESCR)

This register performs the following functions

- LIN synch break interrupt enable/disable
- Indicating LIN synch break detection
- Specifying LIN synch break length
- Directly accessing SINn and SOTn pins
- Specifying continuous clock output operation
- Specifying sampling clock edge

● Extended Communication Control Register (ECCR)

This register performs the following functions

- Indicating bus idle state
- Specifying synchronous clock
- Specifying LIN synch break generation

20.3 UART Pins

This section describes the UART pins and provides a pin block diagram.

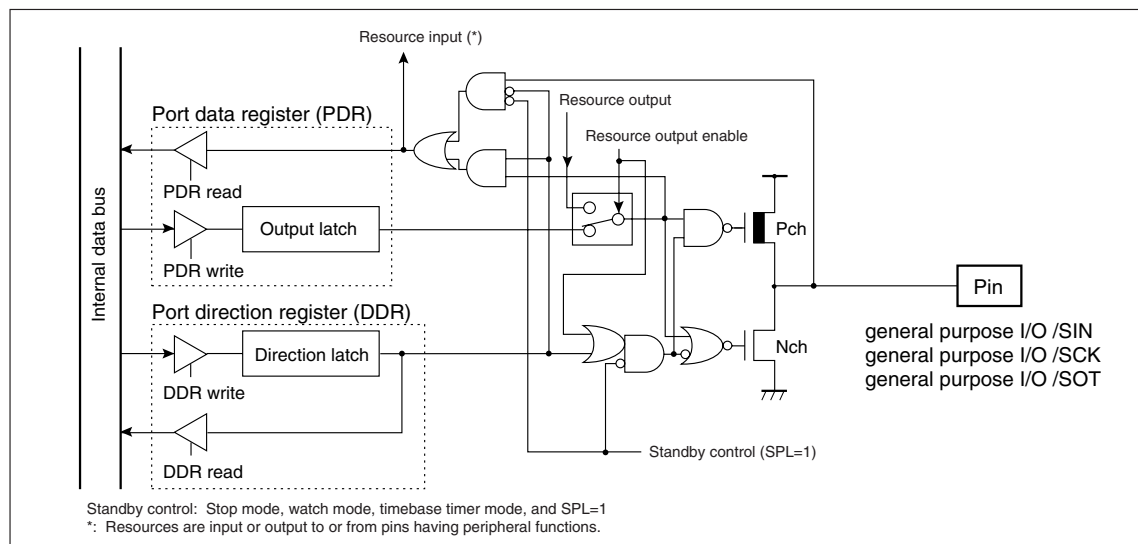
■ UART pins

The UART pins also serve as general ports. Table 20.3-1 “UART pins” lists the pin functions, I/O formats, and settings required to use UART.

Table 20.3-1 UART pins

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
P82/SIN0 P85/SIN1 P50/SIN2 P12/SIN3 P15/SIN4	Port I/O or serial data input	CMOS output and CMOS input	P12 to P17 have programmable pull-up register. Pins other than above do not have pull-up register.	Provided	Set as an input port (DDR: corresponding bit = 0)
P83/SOT0 P86/SOT1 P51/SOT2 P13/SOT3 P16/SOT4	Port I/O or serial data output				Set to output enable mode (SMRn: SOE = 1)
P84/SCK0 P87/SCK1 P52/SCK2 P14/SCK3 P17/SCK4	Port I/O or serial clock input/output				Set as an input port when a clock is input (DDR: corresponding bit = 0) Set to output enable mode when a clock is output (SMRn: SCKE = 1)

Figure 20.3-1 Block diagram of UART pins



20.4 UART Registers

The following figure shows the UART registers.

■ UART registers

Figure 20.4-1 UART registers

- UART0 registers		bit 15	bit 8	bit 7	bit 0
Address:		SCR0 (Serial Control Register)		SMR0 (Serial Mode Register)	
000021H, 000020H		SSR0 (Serial Status Register)		RDR0/TDR0 (Rx, Tx Data Register)	
000023H, 000022H		ESCR0 (Extended Status/Control Reg.)		ECCR0 (Extended Comm. Contr. Reg.)	
000025H, 000024H		BGR01 (Baud Rate Generator Reg. 01)		BGR00 (Baud Rate Generator Reg. 00)	
000027H, 000026H					
- UART1 registers		bit 15	bit 8	bit 7	bit 0
Address:		SCR1 (Serial Control Register)		SMR1 (Serial Mode Register)	
000029H, 000028H		SSR1 (Serial Status Register)		RDR1/TDR1 (Rx, Tx Data Register)	
00002BH, 00002AH		ESCR1 (Extended Status/Control Reg.)		ECCR1 (Extended Comm. Contr. Reg.)	
00002DH, 00002CH		BGR11 (Baud Rate Generator Reg. 11)		BGR10 (Baud Rate Generator Reg. 10)	
00002FH, 00002EH					
- UART2 registers		bit 15	bit 8	bit 7	bit 0
Address:		SCR2 (Serial Control Register)		SMR2 (Serial Mode Register)	
0000D9H, 0000D8H		SSR2 (Serial Status Register)		RDR2/TDR2 (Rx, Tx Data Register)	
0000DBH, 0000DAH		ESCR2 (Extended Status/Control Reg.)		ECCR2 (Extended Comm. Contr. Reg.)	
0000DDH, 0000DCH		BGR21 (Baud Rate Generator Reg. 21)		BGR20 (Baud Rate Generator Reg. 20)	
0000DFH, 0000DEH					
- UART3 registers		bit 15	bit 8	bit 7	bit 0
Address:		SCR3 (Serial Control Register)		SMR3 (Serial Mode Register)	
007951H, 007950H		SSR3 (Serial Status Register)		RDR3/TDR3 (Rx, Tx Data Register)	
007953H, 007952H		ESCR3 (Extended Status/Control Reg.)		ECCR3 (Extended Comm. Contr. Reg.)	
007955H, 007954H		BGR31 (Baud Rate Generator Reg. 31)		BGR30 (Baud Rate Generator Reg. 30)	
007957H, 007956H					
- UART4 registers		bit 15	bit 8	bit 7	bit 0
Address:		SCR4 (Serial Control Register)		SMR4 (Serial Mode Register)	
007959H, 007958H		SSR4 (Serial Status Register)		RDR4/TDR4 (Rx, Tx Data Register)	
00795BH, 00795AH		ESCR4 (Extended Status/Control Reg.)		ECCR4 (Extended Comm. Contr. Reg.)	
00795DH, 00795CH		BGR41 (Baud Rate Generator Reg. 41)		BGR40 (Baud Rate Generator Reg. 40)	
00795FH, 00795EH					

20.4.1 Serial Control Register (SCR)

This register specifies parity bits, selects the stop bit and data lengths, selects a frame data format in mode 1, clears the reception error flag, and specifies whether to enable transmission and reception.

Serial control register (SCR)

Figure 20.4-2 Configuration of the serial control register (SCR)

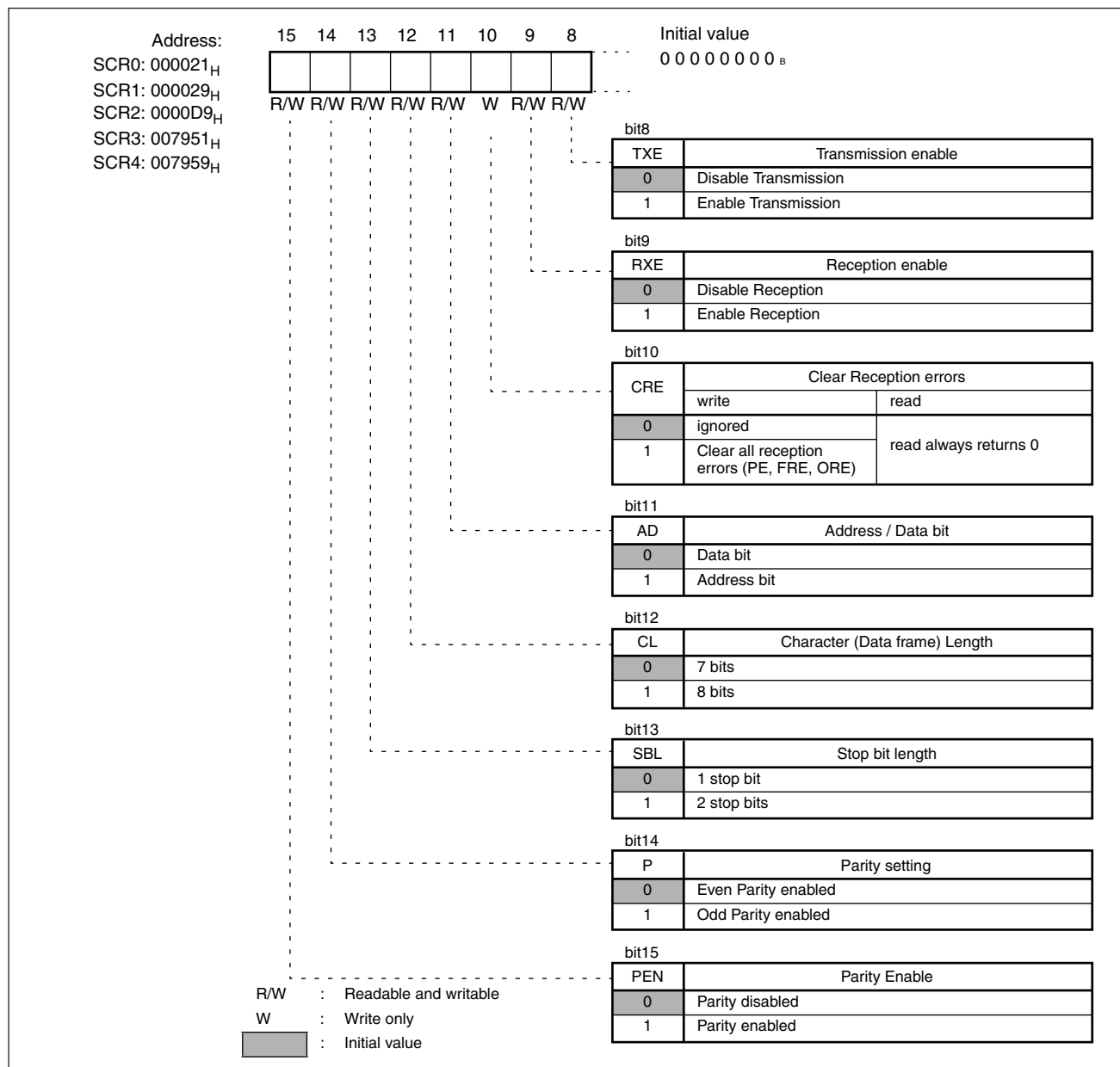


Table 20.4-1 Functions of each bit of control register (SCR)

Bit name		Function
bit15	PEN: Parity enable bit	This bit selects whether to add a parity bit during transmission or detect it during reception. Parity is only provided in mode 0 and in mode 2 if SSM of the ECCR is selected. This bit is fixed to 0 (no parity) in mode 3 (LIN).
bit14	P: Parity selection bit	When parity is provided and enabled this bit selects even (0) or odd (1) parity
bit13	SBL: Stop bit length selection bit	This bit selects the length of the stop bit of an asynchronous data frame or a synchronous frame if SSM of the ECCR is selected. This bit is fixed to 0 (1 stop bit) in mode 3 (LIN).
bit12	CL: Data length selection bit	This bit specifies the length of transmission or reception data. This bit is fixed to 1 (8 bits) in mode 2 and 3.
bit11	AD: Address/Data selection bit	This bit specifies the data format in multiprocessor mode 1. Writing to this bit is provided for a master CPU, reading from it for slave CPU. A 1 indicates an address frame, a 0 indicates a usual data frame. Note: Please read the hints about using this bit in Section 20.8 “Notes on Using UART”.
bit10	CRE: Clear reception error flags bit	This bit clears the FRE, ORE, and PE flag of the Serial Status Register (SSR). Writing a 1 to it clears the error flag. Writing a 0 has no effect. Reading from it always returns 0. Note: Clear reception error flags after the receive operation.
bit9	RXE: Reception enable bit	This bit enables/disables LIN-UART reception. If this bit is set to 0, UART disables the reception of data frames. If this bit is set to 1, UART enables the reception of data frames. The LIN synch break detection in mode 3 remains unaffected. Note: If reception is disabled (RXE=0) during receiving, it is stopped immediately. In this case, data is not guaranteed.
bit8	TXE: Transmission enable bit	This bit enables/disables LIN-UART transmission. If this bit is set to 0, UART disables the transmission of data frames. If this bit is set to 1, UART enables the transmission of data frames. Note: If transmission is disabled (TXE=0) during transmitting, it is stopped immediately. In this case, data is not guaranteed.

20.4.2 Serial Mode Register (SMR)

This register selects an operation mode and baud rate clock and specifies whether to enable output of serial data and clocks to the corresponding pin.

■ Serial mode register (SMR)

Figure 20.4-3 Configuration of the serial mode register (SMR)

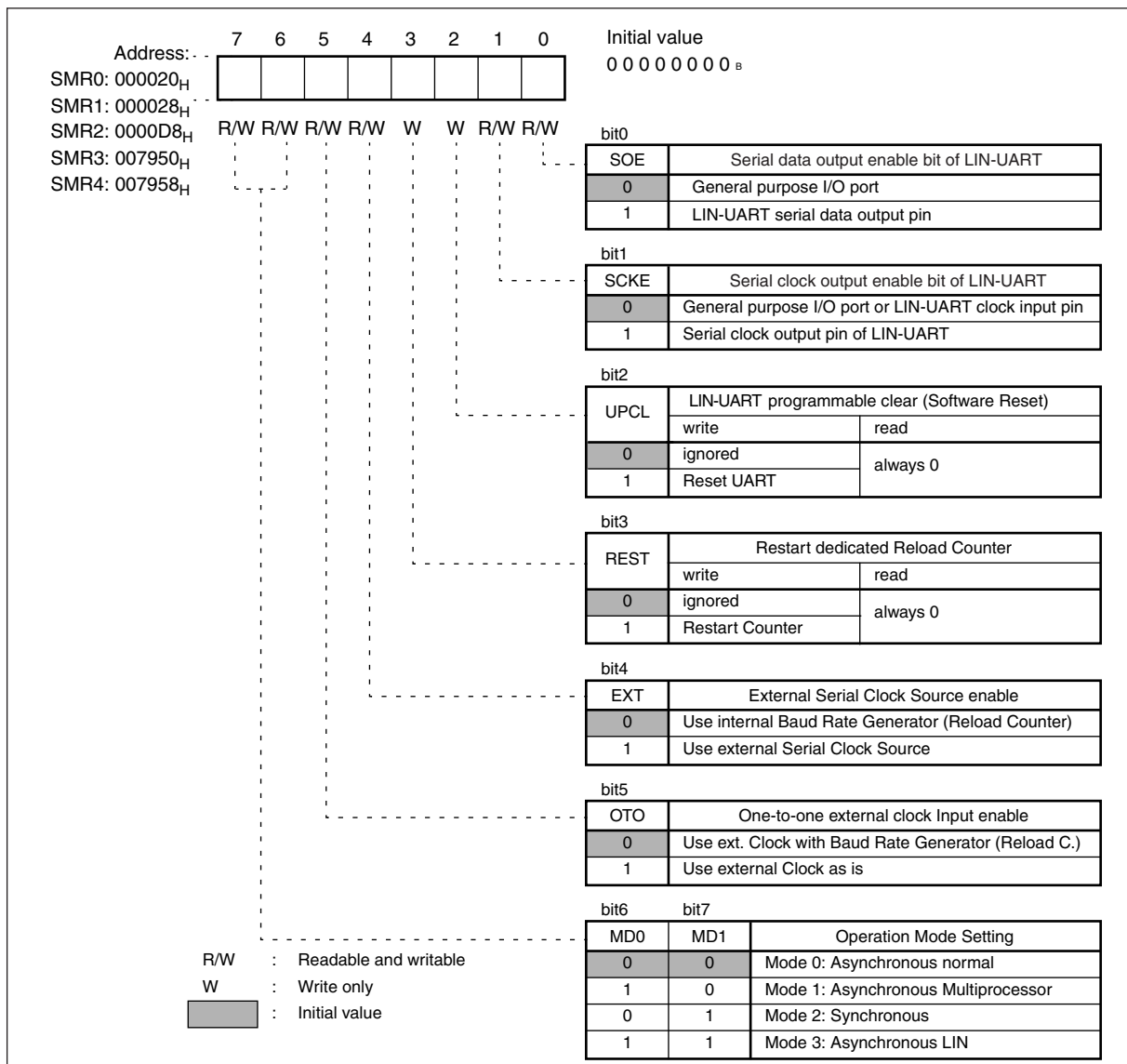


Table 20.4-2 Bit function of the serial mode register (SMR)

Bit name		Function
bit7 bit6	MD1 and MD0: Operation mode selection bits	These two bits sets the UART operation mode.
bit5	OTO: One-to-one external clock selection bit	This bit sets an external clock directly to the LIN-UART's serial clock. This function is used for operating mode 2 (synchronous) slave mode operation.
bit4	EXT: External clock selection bit	This bit executes internal or external clock source for the reload counter
bit3	REST: Restart of transmission reload counter bit	If a 1 is written to this bit the reload counter is restarted. Writing 0 to it has no effect. Reading from this bit always returns 0.
bit2	UPCL: UARTprogrammable clear bit (Software reset)	Writing a 1 to this bit resets LIN-UART immediately. The register settings are preserved. Possible reception or transmission will cut off. All flags (TDRE, RDRF, LBD, PE, ORE, FRE) are cleared and the Reception Data Register (RDR) contains 00 _H . Writing 0 to this bit has no effect. Reading from it always returns 0. LIN-UART reset should performed after disabling the interrupt enable bits.
bit1	SCKE: Serial clock output enable	- This bit controls the serial clock I/O ports. - When this bit is 0, SCKn pin operate as general purpose I/O port or serial clock input pin. When this bit is 1, the pin operates as serial clock output pin and outputs clock in operating mode 2 (synchronous). Note: When using SCKn pin as serial clock input (SCKE=0) pin, set the corresponding bit of DDR as input port. Also, select external clock (EXT = 1) using the external clock selection bit. Reference: When the SCKn pin is assigned to serial clock output (SCKE=1), it functions as the serial clock output pin regardless of the status of the general purpose I/O ports.
bit0	SOE: Serial data output enable bit	- This bit enables or disables the output of serial data. - When this bit is 0, SOTn pin operates as general purpose I/O pin. When this bit is 1, SOTn pin operates as serial data output pins (SOT). Reference: When the output of serial data is enabled (SOE=1), SOTn pin functions as serial data output pin (SOT) regardless of the status of general input-output ports.

20.4.3 Serial Status Register (SSR)

This register checks the transmission and reception status and error status, and enables and disables the transmission and reception interrupts.

Serial status register (SSR)

Figure 20.4-4 Configuration of the serial status register (SSR)

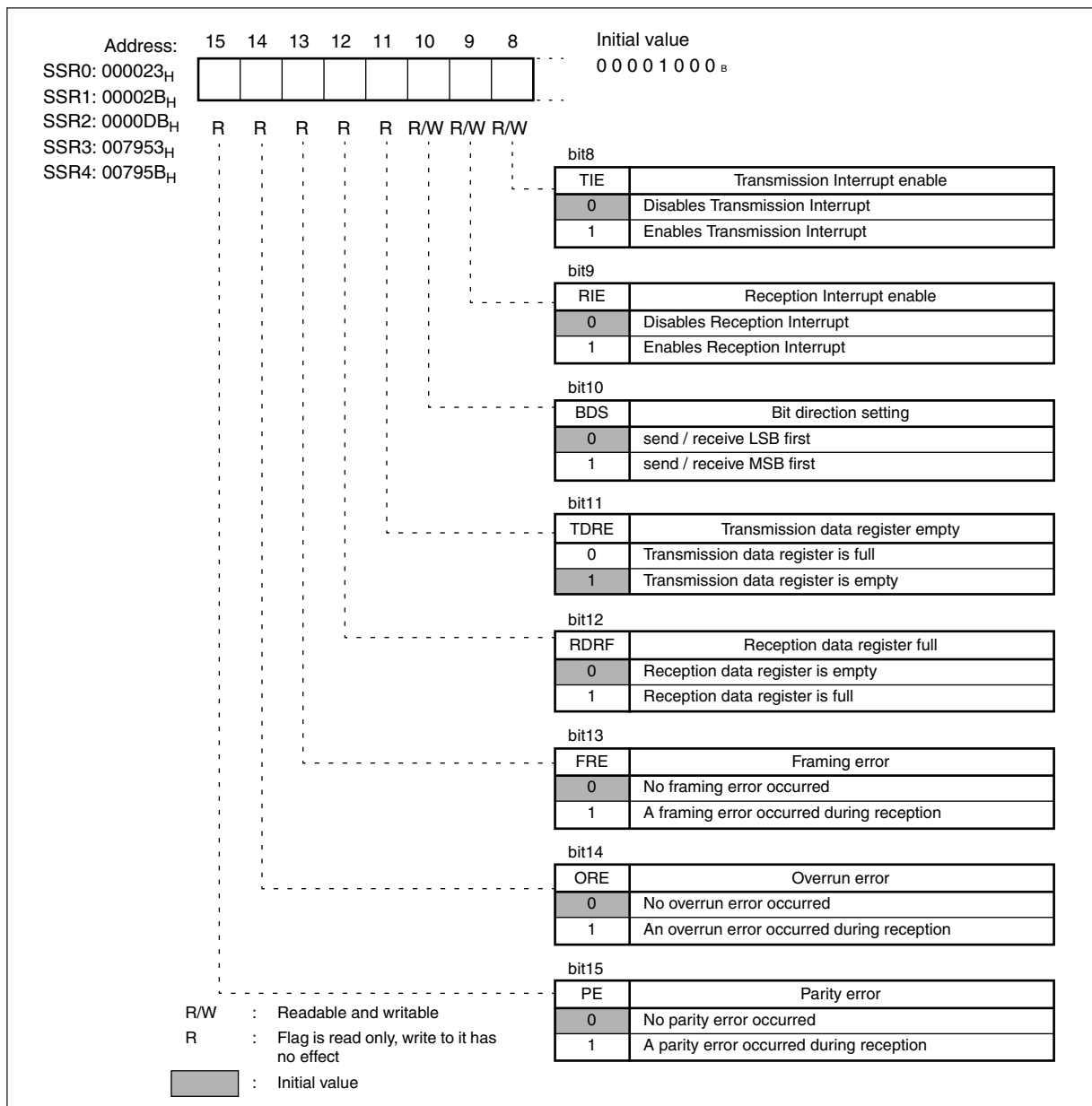


Table 20.4-3 Functions of each bit of status register (SSR)

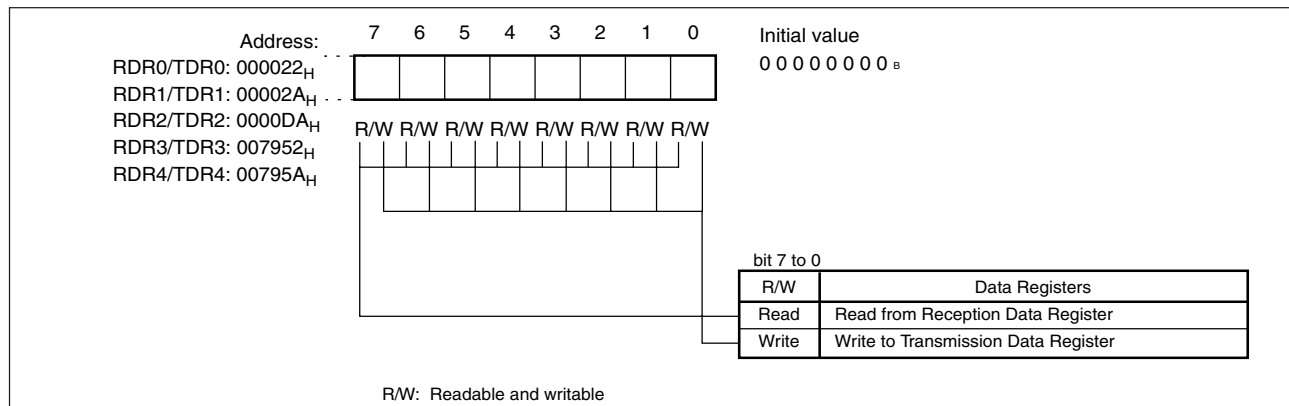
Bit name		Function
bit15	PE: Parity error flag bit	- This bit is set to 1 when a parity error occurs during reception at PE=1 and is cleared when 1 is written to the CRE bit of the serial mode register (SMR). - A reception interrupt request is output when this bit and the RIE bit are 1. - Data in the reception data register (RDR) is invalid when this flag is set.
bit14	ORE: Overrun error flag bit	- This bit is set to 1 when an overrun error occurs during reception and is cleared when 0 is written to the CRE bit of the serial mode register (SMR). - A reception interrupt request is output when this bit and the RIE bit are 1. - Data in the reception data register (RDR) is invalid when this flag is set.
bit13	FRE: Framing error flag bit	- This bit is set to 1 when a framing error occurs during reception and is cleared when 0 is written to the CRE bit of the serial mode register 1 (SMR). - A reception interrupt request is output when this bit and the RIE bit are 1. - Data in the reception data register (RDR) is invalid when this flag is set.
bit12	RDRF: Receive data full flag bit	- This flag indicates the status of the reception data register (RDR). - This bit is set to 1 when reception data is loaded into RDR and can only be cleared to 0 when the reception data register (RDR) is read. - A reception interrupt request is output when this bit and the RIE bit are 1.
bit11	TDRE: Transmission data empty flag bit	- This flag indicates the status of the transmission data register (TDR). - This bit is cleared to 0 when transmission data is written to TDR and is set to 1 when data is loaded into the transmission shift register and transmission starts. - A transmission interrupt request is generated if both this bit and the TIE bit are 1. - If the LBR bit in the ECCR register is set to "1" while the TDRE bit is "1", then this bit once changes to "0". After the completion of LIN synch break generator, the TDRE bit charges back to "1". Note: This bit is set to 1 (TDR empty) as its initial value.
bit10	BDS: Transfer direction selection bit	This bit selects whether to transfer serial data from the least significant bit (LSB first, BDS=0) or the most significant bit (MSB first, BDS=1). Note: The high-order and low-order sides of serial data are interchanged with each other during reading from or writing to the serial data register. If this bit is set to another value after the data is written to the RDR register, the data becomes invalid. This bit is fixed to 0 in mode 3 (LIN)
bit9	RIE: Reception interrupt request enable bit	This bit enables/disables the reception interrupt. If any of the RDRF, PE, ORE and FRE bits is set and this bit is "1", then a reception interrupt is signaled to the interrupt controller.
bit8	TIE: Transmission interrupt request enable bit	- This bit enables or disables the transmission interrupt. - A transmission interrupt request is output when this bit and the TDRE bit are 1.

20.4.4 Reception and Transmission Data Register (RDR/TDR)

The reception data register (RDR) holds the received data. The transmission data register (TDR) holds the transmission data. Both RDR and TDR registers are located at the same address.

■ Reception and transmission data registers (RDR/TDR)

Figure 20.4-5 Transmission and reception data registers (RDR/TDR)



● Reception:

RDR is the register that contains reception data. The serial data signal transmitted to the SIN_n pin is converted in the shift register and stored there. When the data length is 7 bits, the uppermost bit (D7) contains 0. When reception is complete the data is stored in this register and the reception data full flag bit (SSR: RDRF) is set to 1. If a reception interrupt request is enabled at this point, a reception interrupt occurs.

Read RDR when the RDRF bit of the status register (SSR) is 1. The RDRF bit is cleared automatically to 0 when RDR is read. Also the reception interrupt is cleared if it is enabled and no error has occurred.

Data in RDR is invalid when a reception error occurs (SSR: PE, ORE, or FRE = 1).

● Transmission:

When data to be transmitted is written to the transmission data register in transmission enable state, it is transferred to the transmission shift register, then converted to serial data, and transmitted from the serial data output terminal (SOTn pin). If the data length is 7 bits, the uppermost bit (D7) is not sent.

When transmission data is written to this register, the transmission data empty flag bit (SSR: TDRE) is cleared to 0. When transfer to the transmission shift register is complete and starts, the bit is set to 1. When the TDRE bit is 1, the next part of transmission data can be written. If output transmission interrupt requests have been enabled, a transmission interrupt is generated. Write the next part of transmission data when a transmission interrupt is generated or the TDRE bit is 1.

Note:

TDR is a write-only register and RDR is a read-only register. These registers are located in the same address, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

20.4.5 Extended Status/Control Register (ESCR)

This register provides several LIN functions, direct access to the SINn and SOTn pin and setting for UART synchronous clock mode.

Extended status/control register (ESCR)

Figure 20.4-6 Configuration of the extended status/control register (ESCR)

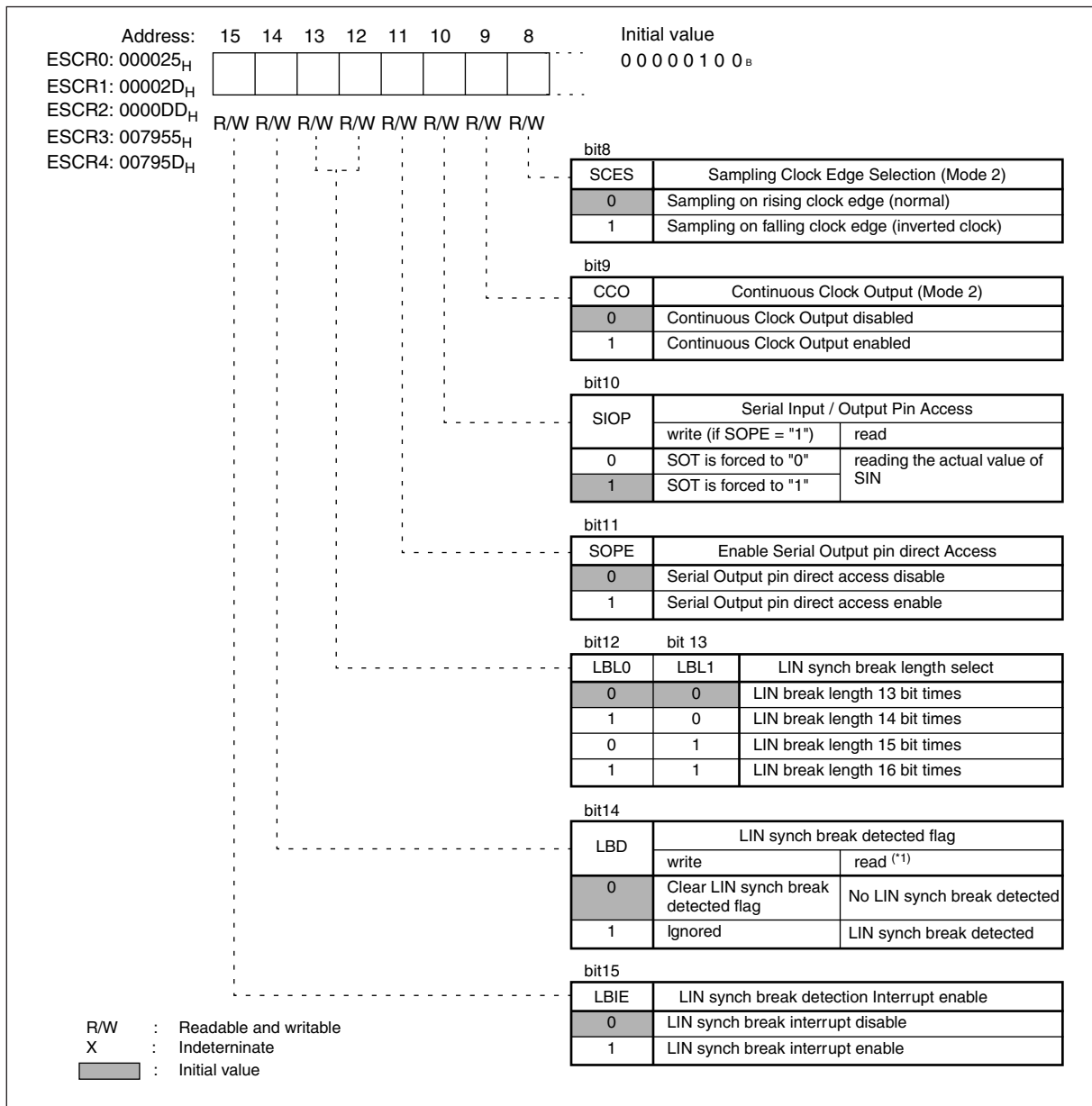


Table 20.4-4 Function of each bit of the extended status/control register (ESCR)

Bit name		Function
bit15	LBIE: LIN synch break detection interrupt enable bit	This bit enables/disables LIN synch break interrupt LIN synch break interrupt is connected to the reception interrupt. When the LBD bit is set and this bit is "1", a reception interrupt is signaled to the interrupt controller. This bit is fixed to "0" in operation mode 1 and 2.
bit14	LBD: LIN synch break detected flag	This bit goes 1 if a LIN synch break was detected in operating mode 3. Writing a 0 to it clears this bit and the corresponding interrupt, if it is enabled. It is recommended to write "0" to the RXE bit in the SCR register before using this bit. Read-modify-write instructions always return 1. Note that this does not indicate a LIN synch break.
bit13 bit12	LBL1/0: LIN synch break length selection	These two bits determine how many serial bit times the LIN synch break is generated by UART. Receiving a LIN synch break is always fixed to 11 bit times.
bit11	SOPE: Serial Output pin direct access enable *1	Setting this bit to 1 enables the direct write to the SOTn pin, if SOE = 1 (SMR). *1
bit10	SIOP: Serial Input/Output Pin direct access *1	Normal read instructions always return the actual value of the SINn pin. Writing to it sets the bit value to the SOTn pin, if SOPE = 1. During a Read-Modify-Write instruction the bit returns the SOTn value in the read cycle. *1
bit9	CCO: Continuous Clock Output enable bit	This bit enables a continuous serial clock at the SCKn pin if UART operates in master mode 2 (synchronous) and the SCKn pin is configured as a clock output. <Note> When CCO bit is "1", use SSM bit of ECCR as setting to "1".
bit8	SCES: Serial clock edge selection bit	This bit inverts the serial clock signal in operation mode 2 (synchronous communication). Receiving data is sampled at the falling edge of the internal clock. If the MS bit of the ECCR register is "0" (master mode) and the SCKE bit of the SMR register is "1" (clock output enabled), the output clock signal is also inverted. During operation mode 0,1,3, please set this bit to 0.

*1: See Table 20.4-5 "Description of the interaction of SOPE and SIOP".

Table 20.4-5 Description of the interaction of SOPE and SIOP

SOPE	SIOP	Writing to SIOP	Reading from SIOP
0	R/W	has no effect on SOTn, but holds the written value	returns current value of SINn
1	R/W	write "0" or "1" to SOTn	returns current value of SINn
1	RMW	reads current value of SOTn and write it back	

20.4.6 Extended Communication Control Register (ECCR)

The extended communication control register provides bus idle recognition interrupt settings, synchronous clock settings, and the LIN break generation.

Extended communication control register (ECCR)

Figure 20.4-7 Configuration of the extended communication control register (ECCR)

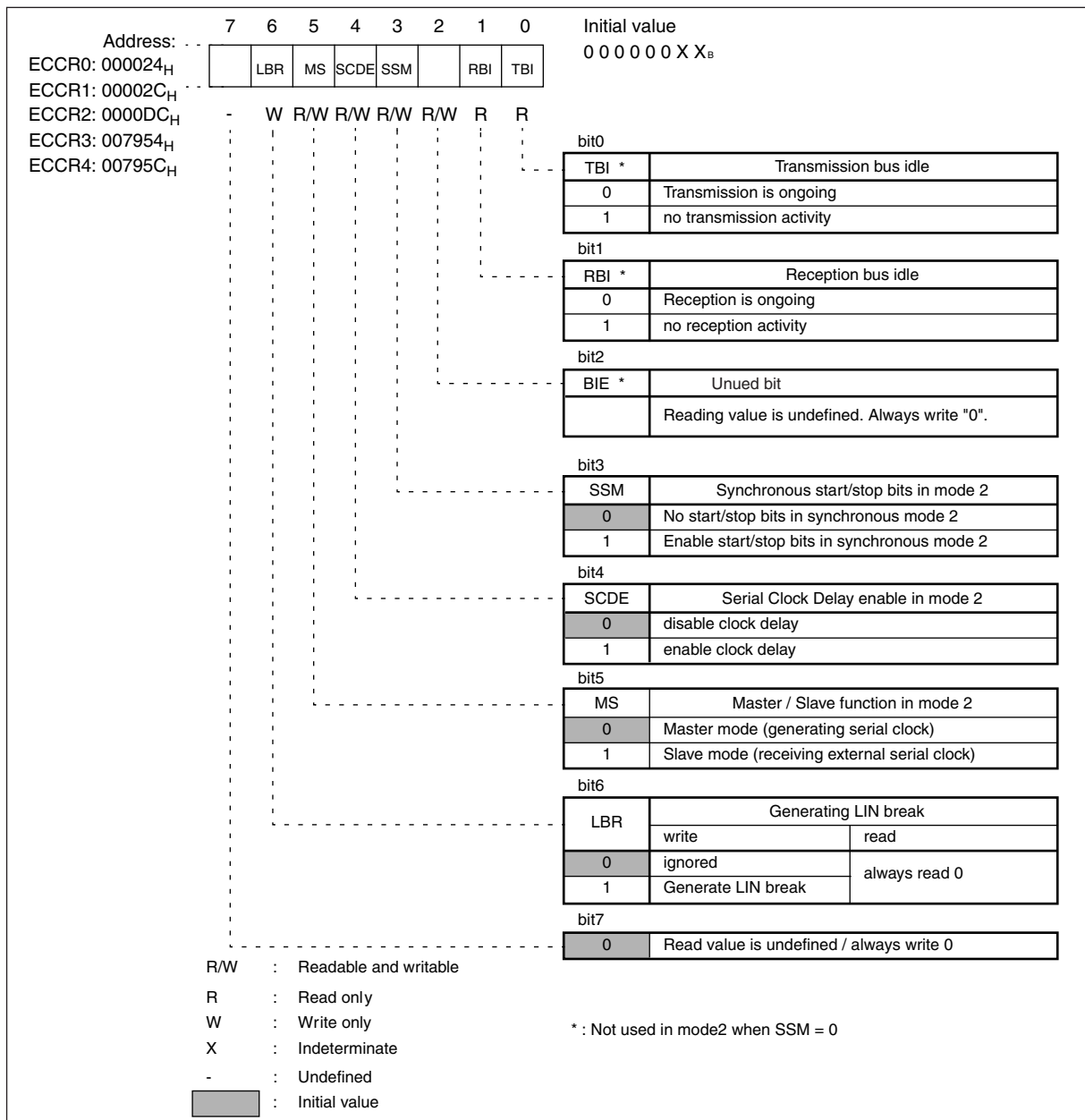


Table 20.4-6 Function of each bit of the extended communication control register (ECCR)

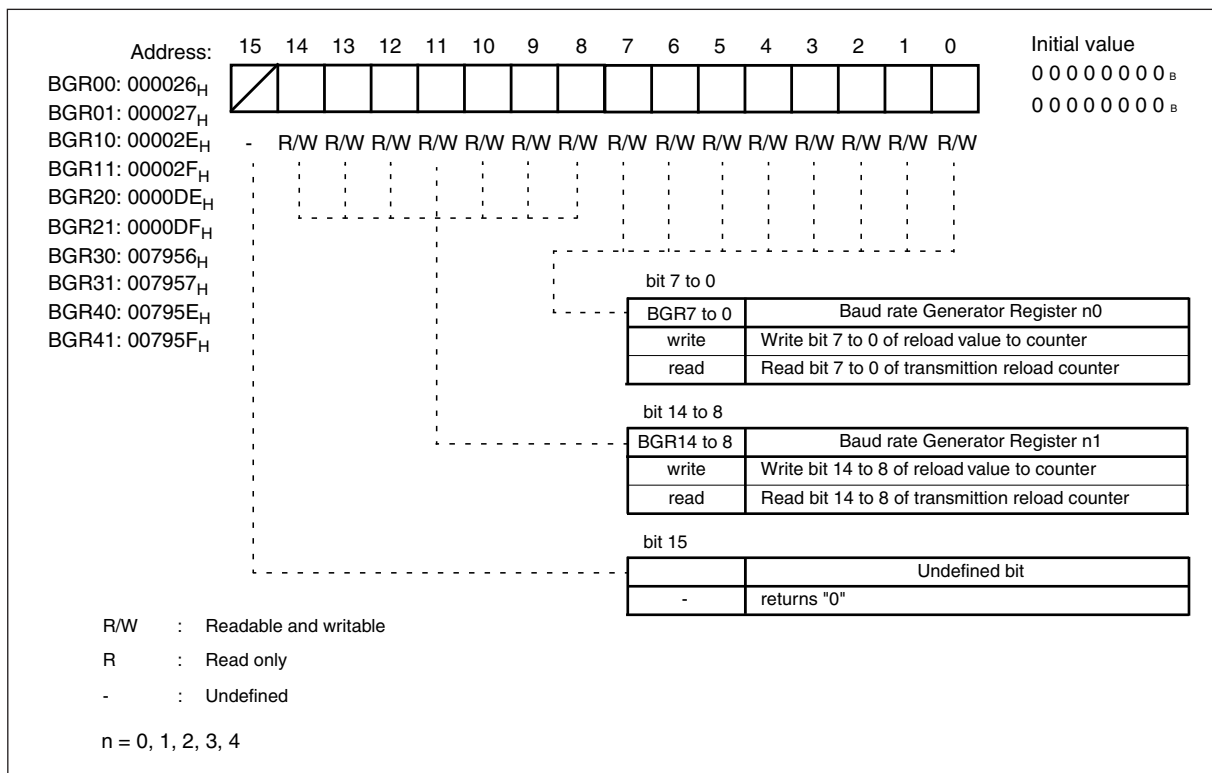
Bit name		Function
bit7	-	This bit is undefined. Always write “0”.
bit6	LBR: Generating LIN synch break bit	Writing a 1 to this bit generates a LIN synch break of the length selected by the LBL0/1 bits of the ESCR, if operation mode 3 is selected. Setting to “0” in operation mode 0.
bit5	MS: Master/Slave mode selection bit	This bit selects master or slave mode of UART in synchronous mode 2. If master is selected UART generates the synchronous clock by itself. If slave mode is selected, UART receives external serial clock. This bit is fixed to “0” in operation mode 0, 1 and 3. Note: If slave mode is selected, the clock source must be external and set to “One-to-One” (SMR: SCKE = 0, EXT = 1, OTO = 1).
bit4	SCDE: Serial clock delay enable bit	If this bit is set the serial output clock is delayed as shown in figure 20.7-4 if UART operates in master mode 2.
bit3	SSM: Start/Stop bit mode enable	This bit adds start and stop bits to the synchronous data format in operation mode 2. It is ignored in mode 0, 1, and 3.
bit2	Unused bit	Unused bit. Reading value is undefined. Always write to “ ϕ ”
bit1	RBI: Reception bus idle flag bit	This bit is “1” if there is no reception activity on the SINn pin and it is kept at “1”. Do not use this bit in mode 2 when SSM=0.
bit0	TBI: Transmission bus idle flag bit	This bit is “1” if there is no transmission activity on the SOTn pin. Do not use this bit in mode 2 when SSM=0.

20.4.7 Baud Rate/Reload Counter Register 0 and 1 (BGR0/1)

The baud rate/reload counter registers set the division ratio for the serial clock. Also the actual count of the transmission reload counter can be read.

■ Baud rate generator register (BGRn0/n1)

Figure 20.4-8 Baud rate generator register (BGRn0/n1)



■ Baud rate/reload counter register

The baud rate/reload counter registers determine the division ratio for the serial clock.

Both registers can be read or written via byte or word access.

20.5 UART Interrupts

UART uses both reception and transmission interrupts. An interrupt request can be generated for either of the following causes:

- Receive data is set in the Reception Data Register (RDR), or a reception error occurs.
- Transmission data is transferred from the Transmission Data Register (TDR) to the transmission shift register and started.
- A LIN break is detected

The extended intelligent I-O service (EI²OS) and DMA is available for these interrupts.

■ LIN-UART interrupts

Table 20.5-1 Interrupt control bits and interrupt causes of LIN-UART

Reception/ transmission/ ICU	Interrupt request flag bit	Flag Register	Operation mode				Interrupt cause	Interrupt cause enable bit	How to clear the Interrupt Request
			0	1	2	3			
Reception	RDRF	SSR	○	○	○	○	receive data is written to RDR	SSR: RIE	Receive data is read
	ORE	SSR	○	○	○	○	Overrun error		"1" is written to clear rec. error bit (SCR: CRE)
	FRE	SSR	○	○	*	○	Framing error		
	PE	SSR	○	x	*	x	Parity error		
	LBD	ESCR	x	x	x	○	LIN synch break detected	ESCR: LBIE	"0" is written to ESCR: LBD
Transmission	TDRE	SSR	○	○	○	○	TDR empty	SSR: TIE	Write data to TDR
Input Capture Unit	ICP0/1/6/7	ICS01/67	x	x	x	○	1st falling edge of LIN synch field	ICS01/45: ICE0/1/6/7	disable ICP0/1/6/7 temporary
	ICP0/1/6/7	ICS01/67	x	x	x	○	5th falling edge of LIN synch field	ICS01/45: ICE0/1/6/7	disable ICP0/1/6/7

○: Used

x: Unused

*: Only available if ECCR/SSM = 1

● Reception Interrupt

If one of the following events occurs in reception mode, the corresponding flag bit of the Serial Status Register (SSR) is set to “1”:

- Data reception is complete, i. e. the received data was transferred from the serial input shift register to the Reception Data Register (RDR) and data can be read: RDRF
- Overrun error, i. e. RDRF = 1 and RDR was not read by the CPU and received next serial data: ORE
- Framing error, i. e. a stop bit was expected, but a "0"-bit was received: FRE
- Parity error, i. e. a wrong parity bit was detected: PE

If at least one of these flag bits above go “1” and the reception interrupt is enabled (SSR: RIE = 1), a reception interrupt request is generated.

If the Reception Data Register (RDR) is read, the RDRF flag is automatically cleared to “0”. Note that this is the only way to reset the RDRF flag. The error flags are cleared to “0”, if a “1” is written to the Clear Reception Error (CRE) flag bit of the Serial Control Register (SCR). The RDR contains only valid data if the RDRF flag is “1” and no error bits are set.

Note, that the CRE flag is "write only" and by writing a "1" to it, it is internally held to "1" for one machine clock cycle.

● Transmission Interrupt

If transmission data is transferred from the Transmission Data Register (TDR) to the transfer shift register and transfer is started, the Transmission Data Register Empty flag bit (TDRE) of the Serial Status Register (SSR) is set to “1”. In this case an interrupt request is generated, if the Transmission Interrupt Enable (TIE) bit of the SSR was set to “1” before.

Note, that the initial value of TDRE (after hardware or software reset) is “1”. So an interrupt is generated immediately then, if the TIE flag is set to “1”. Also note, that the only way to reset the TDRE flag is writing data to the Transmission Data Register (TDR).

● LIN Synchronization Break Interrupt

This paragraph is only relevant, if UART operates in mode 3 as a LIN slave.

If the bus (serial input) goes “0” (dominant) for more than 11 bit times, the LIN Break Detected (LBD) flag bit of the Extended Status/Control Register (ESCR) is set to “1”. Note, that in this case after 9 bit times the reception error flags are set to “1”, therefore the RXE flag has to set to “0”, if only a LIN synch break detect is desired.

The interrupt and the LBD flag are cleared after writing a "1" to the LBD flag. This has to be performed before input capture interrupt for LIN sync field.

● LIN Synchronization Field Edge Detection Interrupts

This paragraph is only relevant, if UART operates in mode 3 as a LIN slave. After a LIN break detection the next falling edge of the reception bus is indicated by UART. Simultaneously an internal signal connected to the ICU1/5 is set to “1”. This signal is reset to “0” after the fifth falling edge of the LIN Synchronization Field. In both cases the ICU1/5 generates an interrupt, if “both edge detection” and the ICU1/5 interrupt are enabled. The difference of the ICU1/5 counter values is the serial clock multiplied by 8. Dividing it by 8 results in the new detected and calculated baud rate for the dedicated reload counter. This value - 1 has then to be written to the Baud Rate Generator Registers (BGR1/0). There is no need to restart the reload counter, because it is automatically reset if a falling edge of a start bit is detected.

■ LIN-UART interrupts and EI²OS

Table 20.5-2 UART interrupt and EI²OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI ² OS	DMA channel number
		Register name	Address	Lower	Upper	Bank		
UART0 reception interrupt	#35(23 _H)	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	*1	DRQ10
UART0 transmission interrupt	#36(24 _H)	ICR12	0000BC _H	FFFF6C _H	FFFF6D _H	FFFF6E _H	*2	DRQ11
UART1/UART3 reception interrupt	#37(25 _H)	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	*1	DRQ12
UART1/UART3 transmission interrupt	#38(26 _H)	ICR13	0000BD _H	FFFF64 _H	FFFF65 _H	FFFF66 _H	*2	DRQ13
UART2/UART4 reception interrupt	#39(27 _H)	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	*1	DRQ14
UART2/UART4 transmission interrupt	#40(28 _H)	ICR14	0000BE _H	FFFF5C _H	FFFF5D _H	FFFF5E _H	*2	DRQ15

*1: Provided with a function that detects a UART reception error and stops EI²OS.

*2: Usable when ICR12, ICR13 and ICR14 or interrupt causes that share an interrupt vector are not used.

■ UART EI²OS functions

UART has a circuit for operating EI²OS, which can be started up for either reception or transmission interrupts.

● For Reception

EI²OS can be used regardless of the status of other resources.

● For Transmission

UART shares the interrupt registers with the UART reception interrupts. Therefore, EI²OS can be started up only when no UART reception interrupts are used.

20.5.1 Reception Interrupt Generation and Flag Set Timing

The following are the reception interrupt causes: completion of reception (SSR: RDRF) and occurrence of a reception error (SSR: PE, ORE, or FRE).

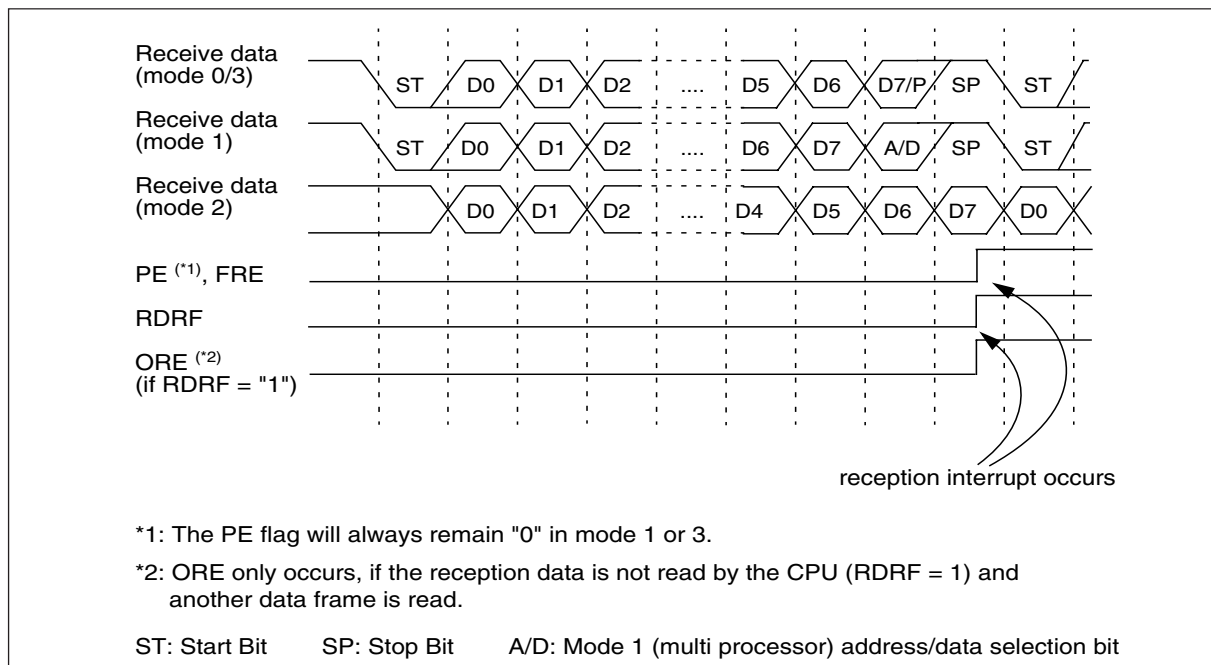
■ Reception interrupt generation and flag set timing

Generally a reception interrupt is generated, if the received data is complete (RDRF = 1) and the Reception Interrupt Enable (RIE) flag bit of the Serial Status Register (SSR) was set to “1”. This interrupt is generated if the first stop bit is detected in mode 0, 1, 2 (if SSM = 1), 3, or the last data bit was read in mode 2 (if SSM = 0).

Note:

If a reception error has occurred, the Reception Data Register (RDR) contains invalid data in each mode.

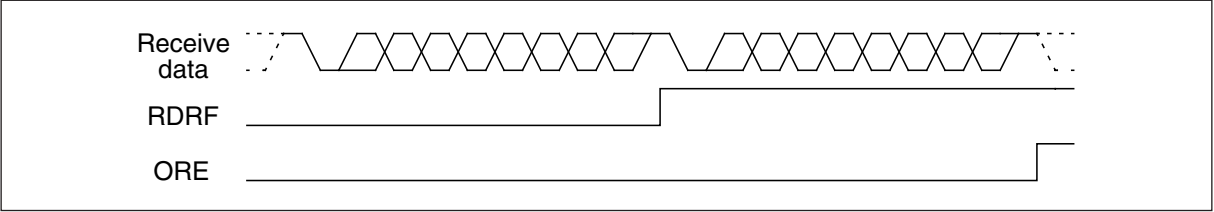
Figure 20.5-1 Reception operation and flag set timing



Note:

The example in Figure 20.5-1 “Reception operation and flag set timing” does not show all possible reception options for mode 0 and 3. Here it is: “7p1” and “8N1” (p = “E” [even] or “O” [odd]).

Figure 20.5-2 ORE set timing:



20.5.2 Transmission Interrupt Generation and Flag Set Timing

A transmission interrupt is generated when the transmission data is transferred from transmission data register (TDR) to transmission shift register and started.

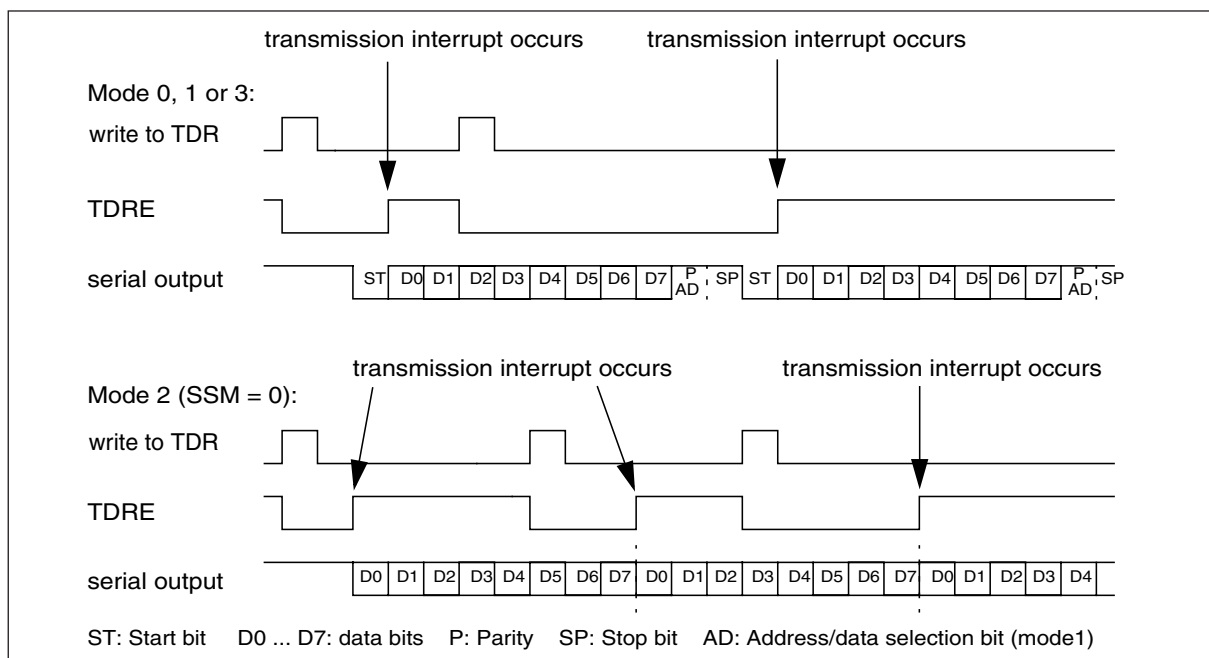
■ Transmission interrupt generation and flag set timing

A transmission interrupt is generated, when the next data to be sent is ready to be written to the Transmission Data Register (TDR), i. e. the TDR is empty, and the transmission interrupt is enabled by setting the Transmission Interrupt Enable (TIE) bit of the Serial Status Register (SSR) to “1”.

The Transmission Data Register Empty (TDRE) flag bit of the SSR indicates an empty TDR. Because the TDRE bit is “read only”, it only can be cleared by writing data into TDR.

The following figure demonstrates the transmission operation and flag set timing for the four modes of UART.

Figure 20.5-3 Transmission operation and flag set timing



Note:

The example in Figure 20.5-3 “Transmission operation and flag set timing” does not show all possible transmission options for mode 0. Here it is: “8p1” (p = “E” [even] or “O” [odd]). Parity is not provided in mode 3 or 2, if SSM = 0.

■ Transmission interrupt request generation timing

If the TDRE flag is set to “1” when a transmission interrupt is enabled (SSR: TIE=1), transmission interrupt request is generated.

Note:

A transmission completion interrupt is generated immediately after the transmission interrupt is enabled (TIE=1) because the TDRE bit is set to 1 as its initial value. TDRE is a read-only bit that can be cleared only by writing new data to the output data register (TDR). Carefully specify the transmission interrupt enable timing.

20.6 UART Baud Rates

One of the following can be selected for the UART serial clock source:

- Dedicated baud rate generator (Reload Counter)
 - External clock as it is (clock input to the SCKn pin)
 - External clock connected to the baud rate generator (Reload Counter)
-

■ UART baud rate selection

The baud rate selection circuit is designed as shown below. One of the following three types of baud rates can be selected:

● Baud rates determined using the dedicated baud rate generator (reload counter)

UART has two independent internal reload counters for transmission and reception serial clock. The baud rate can be selected via the 15-bit reload value determined by the Baud Rate Generator Register 0 and 1 (BGR0/1).

The reload counter divides the machine clock by the value set in the Baud Rate Generator Register 0 and 1.

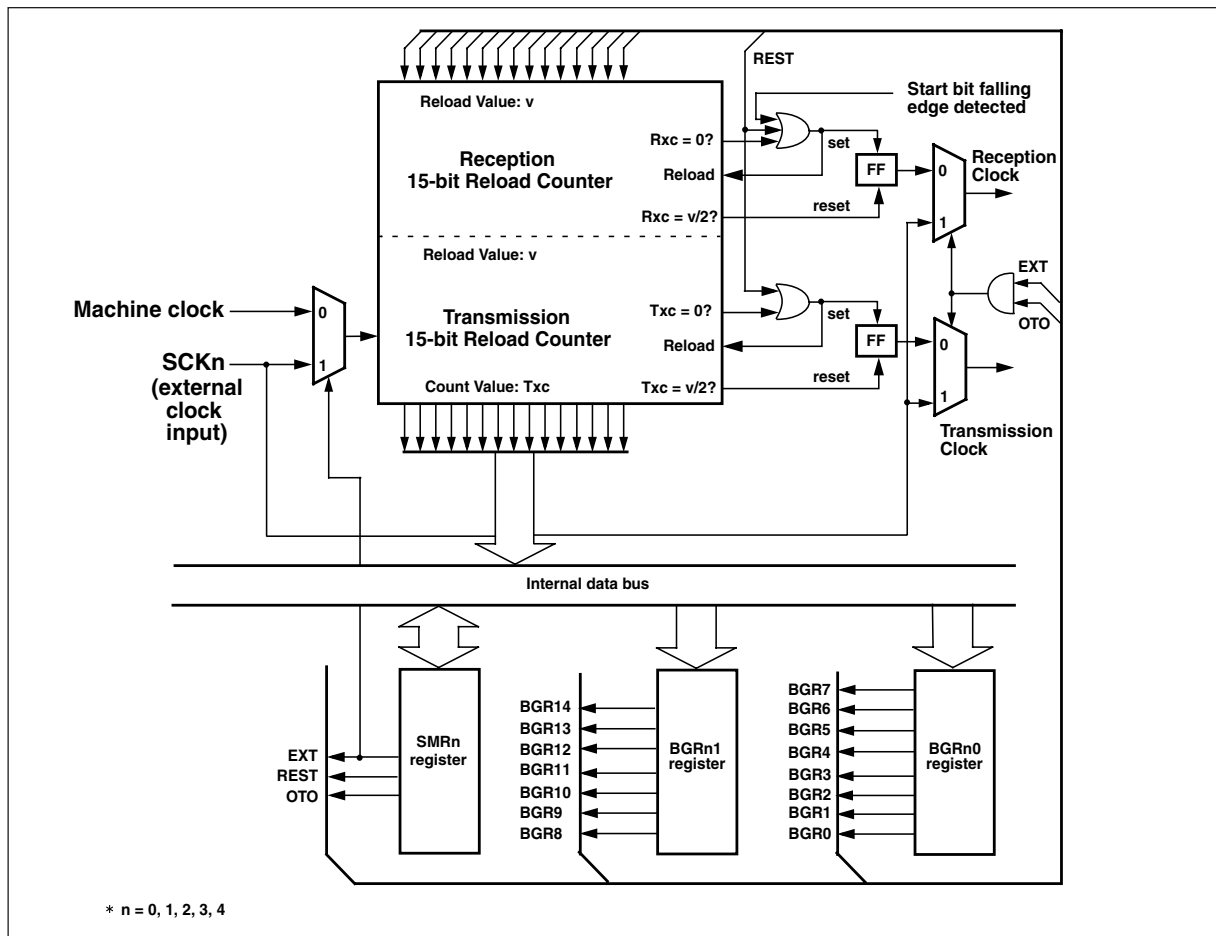
● Baud rates determined using external clock (one-to-one mode)

The clock input from UART clock pulse input pins (SCK) is used as it is (synchronous). Any baud rate less than the machine clock divided by 4 and is divisible can be set externally.

● Baud rates determined using the dedicated baud rate generator with external clock

An external clock source can also be connected internally to the reload counter. In this mode it is used instead of the internal machine clock. This was designed to use quartz oscillators with special frequencies and having the possibility to divide them.

Figure 20.6-1 Baud rate selection circuit (reload counter)



20.6.1 Setting the Baud Rate

This section describes how the baud rates are set and the resulting serial clock frequency is calculated.

■ Calculating the baud rate

The both 15-bit reload counters are programmed by the baud rate generator registers 1, 0 (BGR). The following calculation formula should be used to set the desired baud rate:

Reload Value:

$$v = [\Phi / b] - 1,$$

where Φ is the machine clock, b the baud rate and $[\]$ gaussian brackets (mathematical rounding function).

● Example of calculation

If the CPU clock is 16 MHz and the desired baud rate is 19200 bps baud then the reload value v is:

$$v = [16 \cdot 10^6 / 19200] - 1 = 832$$

The exact baud rate can then be recalculated: $b_{\text{exact}} = \Phi / (v + 1)$, here it is: $16 \cdot 10^6 / 833 = 19207.6831$

Note:

Setting the reload value to 0 stops the reload counter. For this reason the minimum division ratio is 2. For asynchronous communication, the reload value must be greater than equal to 4 because 5 times over-sampling is performed internally.

■ Suggested division ratios for different machine speeds and baud rates

The following settings are suggested for different MCU clock speeds and baud rates:

Table 20.6-1 Suggested baud rates and reload values at different machine speeds.

Baud rate	8 MHz		10 MHz		16 MHz		20 MHz		24 MHz	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
4M	-	-	-	-	-	-	4	0	5	0
2M	-	-	4	0	7	0	9	0	11	0
1M	7	0	9	0	15	0	19	0	23	0
500000	15	0	19	0	31	0	39	0	47	0
460800	-	-	-	-	-	-	-	-	51	-0.16
250000	31	0	39	0	63	0	79	0	95	0
230400	-	-	-	-	-	-	-	-	103	-0.16
153600	51	-0.16	64	-0.16	103	-0.16	129	-0.16	155	-0.16
125000	63	0	79	0	127	0	159	0	191	0
115200	68	-0.64	86	0.22	138	0.08	173	0.22	207	-0.16
76800	103	-0.16	129	-0.16	207	-0.16	259	-0.16	311	-0.16
57600	138	0.08	173	0.22	277	0.08	346	-0.06	416	0.08
38400	207	-0.16	259	-0.16	416	0.08	520	0.03	624	0
28800	277	0.08	346	<0.01	554	-0.01	693	-0.06	832	-0.03
19200	416	0.08	520	0.03	832	-0.03	1041	0.03	1249	0
10417	767	<0.01	959	<0.01	1535	<0.01	1919	<0.01	2303	<0.01
9600	832	0.04	1041	0.03	1666	0.02	2083	0.03	2499	0
7200	1110	<0.01	1388	<0.01	2221	<0.01	2777	<0.01	3332	<0.01
4800	1666	0.02	2082	-0.02	3332	<0.01	4166	<0.01	4999	0
2400	3332	<0.01	4166	<0.01	6666	<0.01	8332	<0.01	9999	0
1200	6666	<0.01	8334	0.02	13332	<0.01	16666	<0.01	19999	0
600	13332	<0.01	16666	<0.01	26666	<0.01	-	-	-	-
300	26666	<0.01	-	-	-	-	-	-	-	-

Note:

- 1) Deviations are given in%.
- 2) Maximum Synchronous Baud Rate: MCU-Clock div. by 5.

■ Using external clock

If the EXT bit of the SMR is set, an external clock is selected, which has to be connected to the SCKn pin. The external clock is used in the same way as the machine clock to the baud rate reload counter.

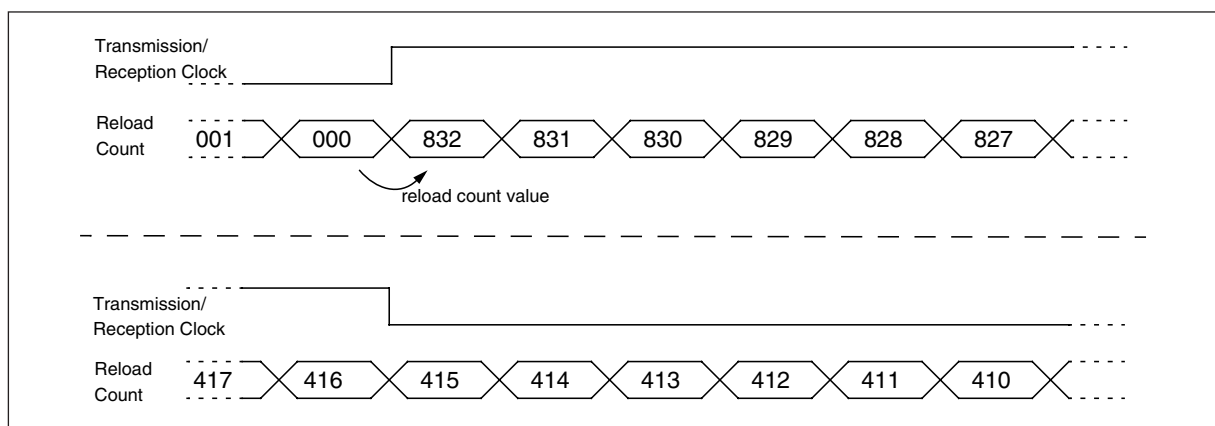
If One-to-one External Clock Input Mode (SMR: OTO) is selected the SCKn signal is directly connected to the UART serial clock inputs. This is needed for the UART synchronous mode 2 operating as slave device.

Note, that in any case the resulting clock signal is synchronized to the machine clock in the UART module. This means that indivisible clock rates will result in phase unstable signals.

■ Counting example

Assume the reload value is 832. The Figure 20.6-2 “Counting example of the reload counters” demonstrates the behavior of both Reload Counters:

Figure 20.6-2 Counting example of the reload counters



Note:

The falling edge of the Serial Clock Signal always occurs after $\lfloor (v + 1) / 2 \rfloor$.

20.6.2 Restarting the Reload Counter

The Reload Counters can be restarted of the following reasons:

Transmission and reception reload counter:

- Global MCU reset
- UART programmable clear (SMR:UPCL bit)
- User programmable restart (SMR: REST bit)

Reception reload counter:

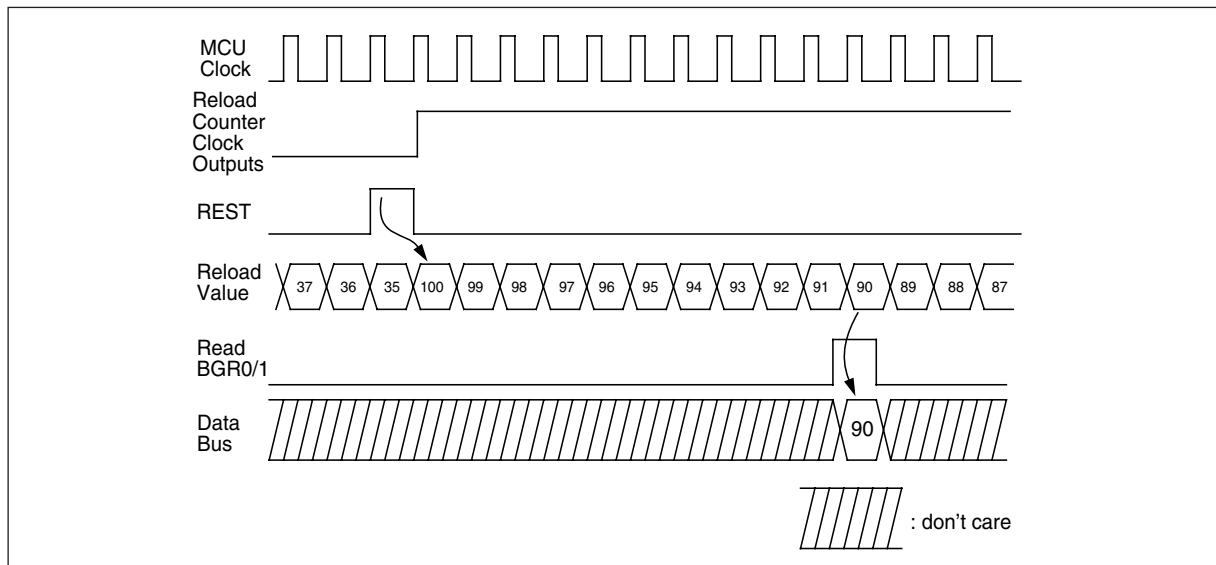
- Start bit falling edge detection in asynchronous mode

■ Programmable restart

If the REST bit of the Serial Mode Register (SMR) is set by the user, both Reload Counters are restarted at the next clock cycle. This feature is intended to use the Transmission Reload Counter as a small timer.

The following figure illustrates a possible usage of this feature (assume that the reload value is 100.)

Figure 20.6-3 Reload counter restart example



In this example the number of MCU clock cycles (cyc) after REST is then:

$$\text{cyc} = v - c + 1 = 100 - 90 + 1 = 11,$$

where v is the reload value and c is the read counter value.

Note:

If UART is reset by setting SMR:UPCL, the Reload Counters will restart too.

- Automatic restart

In asynchronous UART mode if a falling edge of a start bit is detected the Reception Reload Counter is restarted. This is intended to synchronize the serial input shifter to the incoming serial data stream.

● Clearing reload counters

The baud rate reload/counter register (BGRn1 and BGRn0) and the baud rate reload counters are cleared to "0" by the MCU global reset and the counters stops. The reload counters are cleared to "0" by writing "1" to the UPCL bit in the SMR register. However the value stored in the reload register is kept unchanged and the counters start from reload value immediately. Writing "0" to the REST bit does not clear the counters and they restart from reload value immediately.

20.7 Operation of UART

UART operates in operation mode 0 for normal bidirectional serial communication, in mode 2 and 3 in bidirectional communication as master or slave, and in mode 1 as master or slave in multiprocessor communication.

■ Operation of UART

● Operation modes

There are four UART operation modes: modes 0 to 3. As listed in Table 20.7-1 “UART operation mode”, an operation mode can be selected according to the communication method.

Table 20.7-1 UART operation mode

Operation mode		Data length		Synchronization of mode	Length of stop bit	data bit direction *1
		parity disabled	parity enabled			
0	normal mode	7 or 8		asynchronous	1 or 2	L/M
1	multiprocessor	7 or 8 + 1*2	-	asynchronous	1 or 2	L/M
2	normal mode	8		synchronous	0, 1 or 2	L/M
3	LIN mode	8	-	asynchronous	1	L

*1: means the data bit transfer format: LSB or MSB first

*2: "+1" means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

Note:

Mode 1 operation is supported both for master or slave operation of UART in a master-slave connection system. In Mode 3 the UART function is locked to 8N1-Format, LSB first.

If the mode is changed, UART cuts off all possible transmission or reception and awaits then new action.

■ Inter-CPU connection method

External Clock One-to-one connection (normal mode) and master-slave connection (multiprocessor mode) can be selected. For either connection method, the data length, whether to enable parity, and the synchronization method must be common to all CPUs. Select an operation mode as follows:

- In the one-to-one connection method, operation mode 0 or 2 must be used in the two CPUs. Select operation mode 0 for asynchronous transfer mode and operation mode 2 for synchronous transfer mode. Note, that one CPU has to set to the master and the other to the slave in synchronous mode 2.
- Select operation mode 1 for the master-slave connection method and use it either for the master or slave system.

■ Synchronization methods

In asynchronous operation UART reception clock is automatically synchronized to the falling edge of a received start bit.

In synchronous mode the synchronization is performed either by the clock signal of the master device or by UART itself if operating as master.

■ Signal mode

UART can treat data only in non-return to zero (NRZ) format.

■ Operation enable bit

UART controls both transmission and reception using the operation enable bit for transmission (SCR: TXE) and reception (SCR: RXE).

- If reception operation is disabled during reception (data is input to the reception shift register), finish frame reception and read the received data of the reception data register (RDR). Then stop the reception operation.
- If the transmission operation is disabled during transmission (data is output from the transmission shift register), wait until there is no data in the transmission data register (TDR) before stopping the transmission operation.

20.7.1 Operation in Asynchronous Mode (Op. Modes 0 and 1)

When UART is used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode), the asynchronous transfer mode is selected.

■ Operation in asynchronous mode

● Transfer data format

Generally each data transfer in the asynchronous mode operation begins with the start bit (low-level on bus) and ends with at least one stop bit (high-level). The direction of the bit stream (LSB first or MSB first) is determined by the BDS bit of the Serial Status Register (SSR). The parity bit (if enabled) is always placed between the last data bit and the (first) stop bit.

In operation mode 0 the length of the data frame can be 7 or 8 bits, with or without parity, and 1 or 2 stop bits.

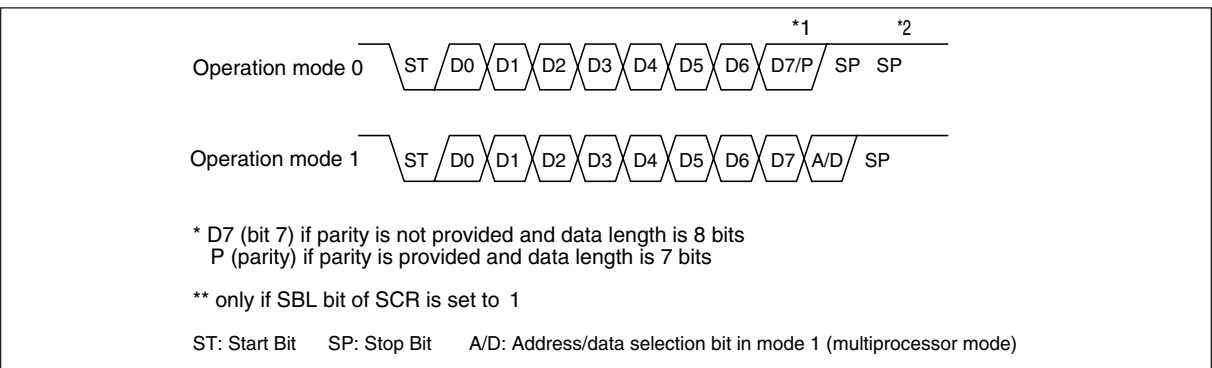
In operation mode 1 the length of the data frame can be 7 or 8 bits with a following address-/data-selection bit instead of a parity bit. 1 or 2 stop bits can be selected.

The calculation formula for the bit length of a transfer frame is:

$$\text{Length} = 1 + d + p + s$$

(d = number of data bits [7 or 8], p = parity [0 or 1], s = number of stop bits [1 or 2])

Figure 20.7-1 Transfer data format (operation modes 0 and 1))



Note:

If BDS bit of the Serial Status Register (SSR) is set to “1” (MSB first), the bit stream processes as: D7, D6, ..., D1, D0, (P).

During Reception both stop bits are detected, if selected. But the Reception data register full (RDRF) flag will go “1” at the first stop bit. The bus idle flag (RBI of ECCR) goes “1” after the second stop bit if no further start bit is detected. (The second stop bit belongs to “bus activity”, although it is just mark level.)

● Transmission operation

If the Transmission Data Register Empty (TDRE) flag bit of the Serial Status Register (SSR) is “1”, transmission data is allowed to be written to the Transmission Data Register (TDR). When data is written, the TDRE flag goes “0”. If the transmission operation is enabled by the TXE-Bit (“1”) of the Serial Control Register (SCR), the data is written next to the transmission shift register and the transmission starts at the next clock cycle of the serial clock, beginning with the start bit. Thereby the TDRE flag goes “1”, so that new data can be written to the TDR.

If transmission interrupt is enabled (TIE = 1), the interrupt is generated by the TDRE flag. Note, that the initial value of the TDRE flag is “1”, so that in this case if TIE is set to “1” an interrupt will occur immediately.

When the character length is set to 7 bits (CL=0), the unused bit of the TDR is always the MSB, independently from the bit direction setting in the BDS bit (LSB first or MSB first).

● Reception operation

Reception operation is performed when it is enabled by the Reception Enable (RXE) flag bit of the SCR. If a start bit is detected, a data frame is received according to the format specified by the SCR. In case of errors, the corresponding error flags are set (PE, ORE, FRE). After the reception of the data frame the data is transferred from the serial shift register to the Reception Data Register (RDR) and the Receive Data Register Full (RDRF) flag bit of the SSR is set. The data then has to be read by the CPU. By doing so, the RDRF flag is cleared. If reception interrupt is enabled (RIE = 1), the interrupt is simply generated by the RDRF.

When the character length is set to 7 bits (CL=0), the unused bit of the RDR is always the MSB, independently from the bit direction setting in the BDS bit (LSB first or MSB first).

Note:

Only when the RDRF flag bit is set and no errors have occurred the Reception Data Register (RDR) contains valid data.

● Used clock

Use the internal clock or external clock. Select the baudrate generator (SMR: EXT = 0 or 1, OTO = 0) for desired baudrate.

● Stop bit, error detection, and parity:

Number of stop bit, 1 or 2 can be specified by the SBL bit of the SCR register. When receiving and 2-bit is specified as the stop bit, the second stop bit is checked in addition to the first stop bit. The RBI (bus idle) flag is set after the second stop bit. However the RDRF flag is set when the first stop bit is received. In mode 0, parity error, overrun error and framing error are checked. In mode 1, parity check is not supported and overrun error and framing error are checked. The PEN bit of the SCR register enables/disables the parity bit and the P bit specifies even or odd parity in mode 0.

20.7.2 Operation in Synchronous Mode (Operation Mode 2)

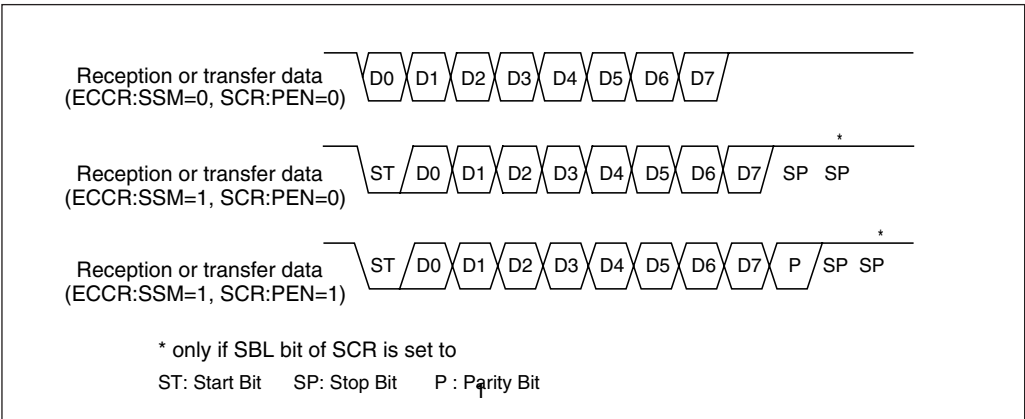
The clock synchronous transfer method is used for UART operation mode 2 (normal mode).

■ Operation in synchronous mode (operation mode 2)

● Transfer data format

In the synchronous mode, 8-bit data is transferred without start or stop bits if the SSM bit of the Extended Communication Control Register (ECCR) is 0. The figure below illustrates the data format during a transmission in the synchronous operation mode.

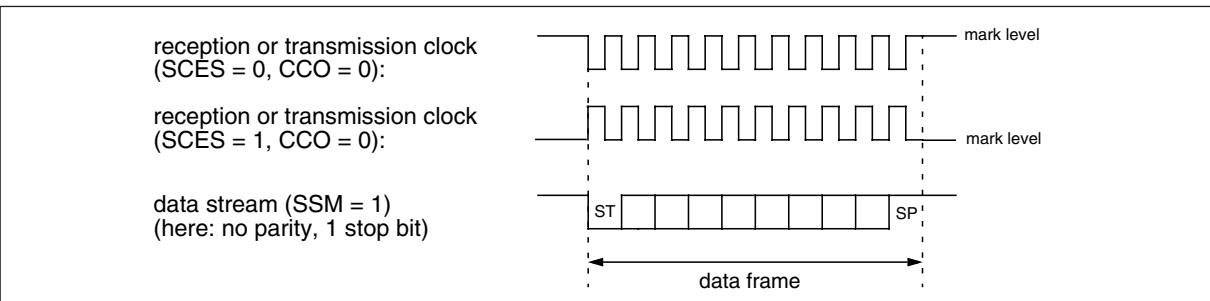
Figure 20.7-2 Transfer data format (operation mode 2)



● Clock inversion and start/stop bits in mode 2

If the SCES bit of the Extended Status/Control Register (ESCR) is set the serial clock is inverted. Therefore in slave mode UART samples the data bits at the falling edge of the received serial clock. Note, that in master mode if SCES is set the clock signal's mark level is "0". If the SSM bit of the Extended Communication Control Register (ECCR) is set the data format gets additional start and stop bits like in asynchronous mode.

Figure 20.7-3 Transfer data format with clock inversion

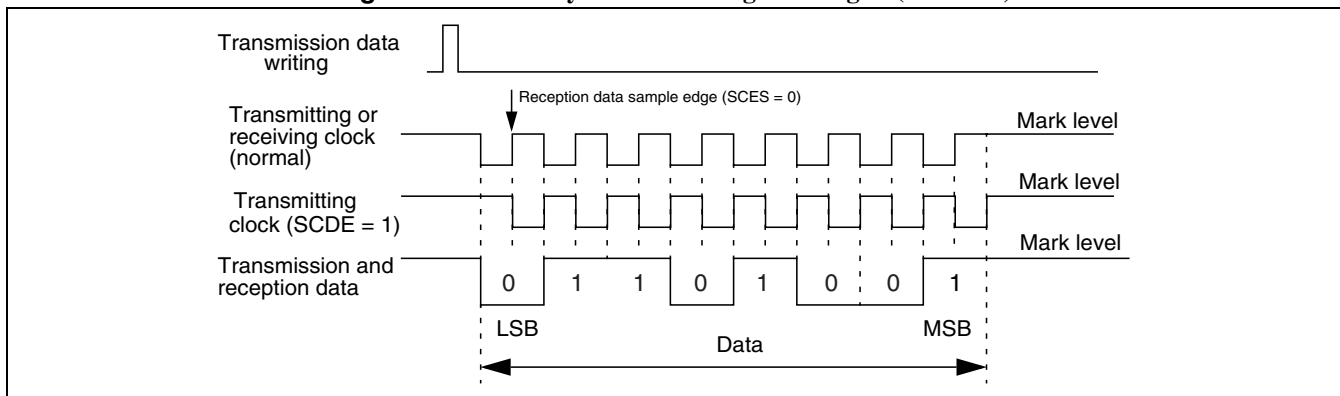


● Clock supply:

In operation mode 2, the number of clock cycles for the clock signal must be the same as the number of bits for the data including start and stop bits. If the MS bit of the ECCR register is "0" (master mode) and the SCKE bit of the SMR register is "1" (clock output enabled), the consistent clock cycles are generated automatically. If the MS bit of the ECCR register is "1" (slave mode), make sure that correct clock cycles are generated by the other communication device. While there is no communication, the clock signal must be kept at "1" as the mark level

If the SCDE bit of the ECCR register is "1", the clock output signal is delayed by the half of the serial clock cycle as shown in figure 20.7-4. The operation is prepared for communication devices which use the falling edge of the serial clock signal for the data sampling.

Figure 20.7-4 Delayed transmitting clock signal(SCDE=1)



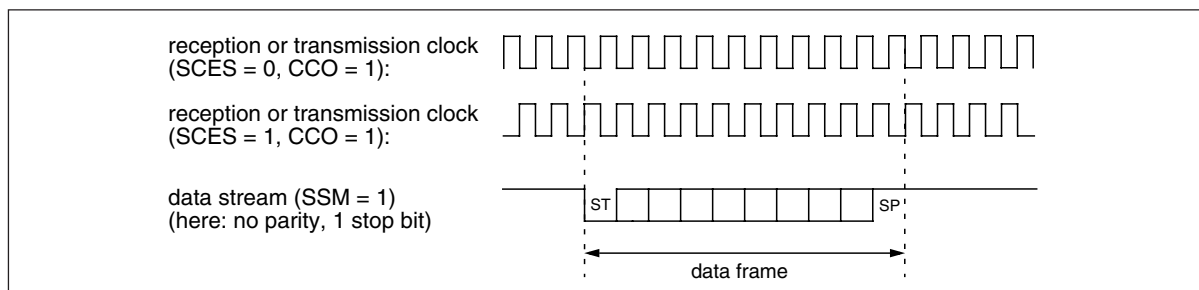
If the SCES bit of the ESCR register is "1", the serial clock signal is inverted. Receiving data is sampled at the falling edge of the serial clock.

If the MS bit of the ECCR register is "0" (master mode) and the SCKE bit of the SMR register is "1" (clock output enabled), the output clock signal is also inverted.

While there is no communication, the clock signal must be kept at "0" as the mark level.

If the CCO bit of the ESCR register is "1", the serial clock is signaled even while there is no data communication. Therefore it is recommended to specify the start/stop bits as shown in figure 20.7-5.

Figure 20.7-5 Continuous clock output in mode 2



● Error detection:

If no start/stop bits are selected (ECCR: SSM = 0) only overrun errors are detected.

● Communication:

For initialization of the synchronous mode, following settings have to be done:

Baud rate generator registers (BGR0/1):

Set the desired reload value for the dedicated baud rate reload counter.

Serial mode control register (SMR):

MD1, MD0: “10_B” (Mode 2)

SCKE: “1” for the dedicated Baud Rate Reload Counter

“0” for external clock input

SOE: “1” for transmission and reception

“0” for reception only

Serial control register (SCR):

RXE, TXE: set one or both of these flags to “1”

A/D: no Address/Data selection - don’t care

CL: automatically fixed to 8-bit data - don’t care

CRE: “1” to clear receive error flags.

-- when SSM=0 (default):

PEN, P, SBL: don’t care

-- when SSM=1:

PEN: “1” if parity bit is added/detected, “0” if not

P: “0” for even parity, “1” odd parity

SBL: “1” for 2 stop bits, “0” for 1 stop bit.

Serial status register (SSR):

BDS: “0” for LSB first, “1” for MSB first

RIE: “1” if interrupts are used; “0” reception interrupts are disabled.

TIE: “1” if interrupts are used; “0” transmission interrupts are disabled.

Extended communication control register (ECCR):

SSM: “0” if no start/stop bits are desired (normal); “1” for adding start/stop bits (special)

MS: “0” for master mode (UART generates the serial clock); “1” for slave mode (UART receives serial clock from the master device)

20.7.3 Operation with LIN Function (Operation Mode 3)

UART can be used either as LIN-Master or LIN-Slave. For this LIN function a special mode is provided. Setting the UART to mode 3 configures the data format to 8N1-LSB-first format.

■ Operation in asynchronous LIN mode (operation mode 3)

● UART as LIN master

In LIN master mode the master determines the baud rate of the whole sub bus, therefore slaves devices have to synchronize to the master. Therefore the desired baud rate remains fixed in master operation after initialization.

Writing a “1” into the LBR bit of the Extended Communication Control Register (ECCR) generates a 13 - 16 bit time low-level on the SOT pin, which is the LIN synchronization break and the start of a LIN message. Thereby the TDRE flag of the Serial Status Register (SSR) goes “0” and is reset to “1” after the break, and generates a transmission interrupt for the CPU (if TIE of SSR is “1”).

The length of the Synchronization break to be sent can be determined by the LBL1/0 bits of the ESCR as follows:

Table 20.7-2 LIN break length

LBL1	LBL0	Length of Break
0	0	13 Bit times
1	0	14 Bit times
0	1	15 Bit times
1	1	16 Bit times

The Synch Field is sent as byte data of 0x55 after the LIN break. To prevent a transmission interrupt, the 0x55 can be written to the TDR just after writing the “1” to the LBR bit, although the TDRE flag is “0”. The internal transmission shifter waits until the LIN break has finished and shifts the TDR value out afterwards. In this case no interrupt is generated after the LIN break and before the start bit.

● UART as LIN slave

In LIN slave mode UART has to synchronize to the master’s baud rate. If Reception is disabled (RXE = 0) but LIN break Interrupt is enabled (LBIE = 1) UART will generate a reception interrupt, if a synchronization break from the LIN master is detected, and indicates it with the LBD flag of the ESCR. Writing “0” to this bit clears the reception interrupt request. The LIN slave may need to calculate the baud rate from the synch field. In this case, the time between the first falling edge to the fifth falling edge of the synch field is measured by the input capture module. For this purpose, the input capture module is connected to the LIN-UART with an internal signal. This internal signal changes from “0” to “1” at the first falling edge then “1” to “0” at the fifth falling edge. Therefore the input capture module should be set to detect both rising and falling edge. Also the input signal from the LIN-UART should be selected. The time

measured by the input capture module represents 8 times of the baud rate clock cycle.

Therefore, baud rate setting value is summarized as follows:

without timer overflow : BGR value = $(b-a)/8$

with timer overflow : BGR value = $(\text{max} + b-a)/8$

where max is the timer maximum value at the overflow occurs.

where a is the value of the ICU counter register after the first Interrupt

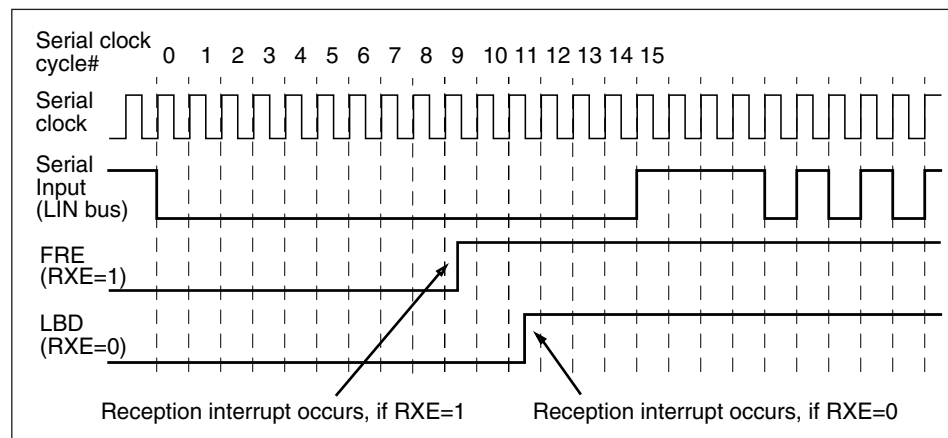
where b is the value of the ICU counter register after the second Interrupt

For the correspondence between other UARTs and ICUs, see 13.3 “16-bit Free-running Timer”.

● LIN Synch Break Detection Interrupt and Flags

If a LIN Synch synchronization break is detected in the slave mode, the LIN Break Detected (LBD) Flag of the ESCR is set to “1”. This causes an interrupt, if the LIN Break Interrupt Enable (LBIE) bit is set

Figure 20.7-6 LIN synch break detection and flag set timing.

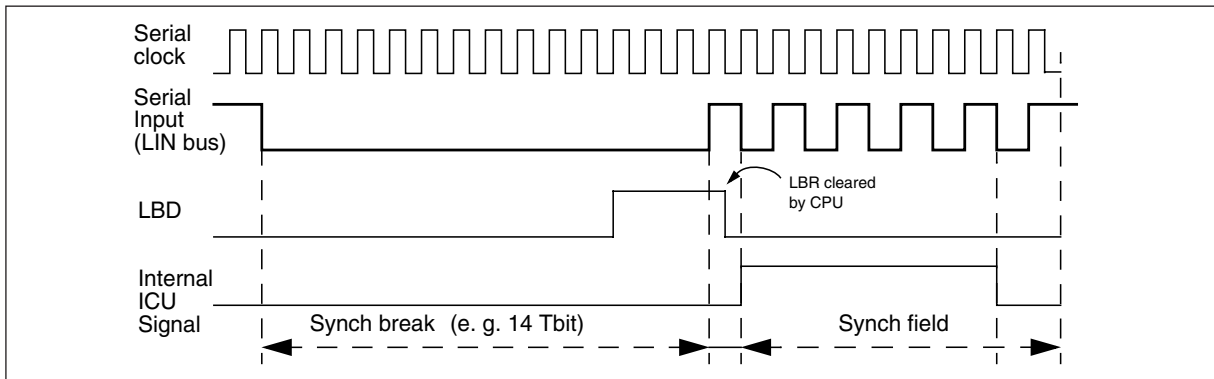


The figure above demonstrates the LIN synch break detection and flag set timing.

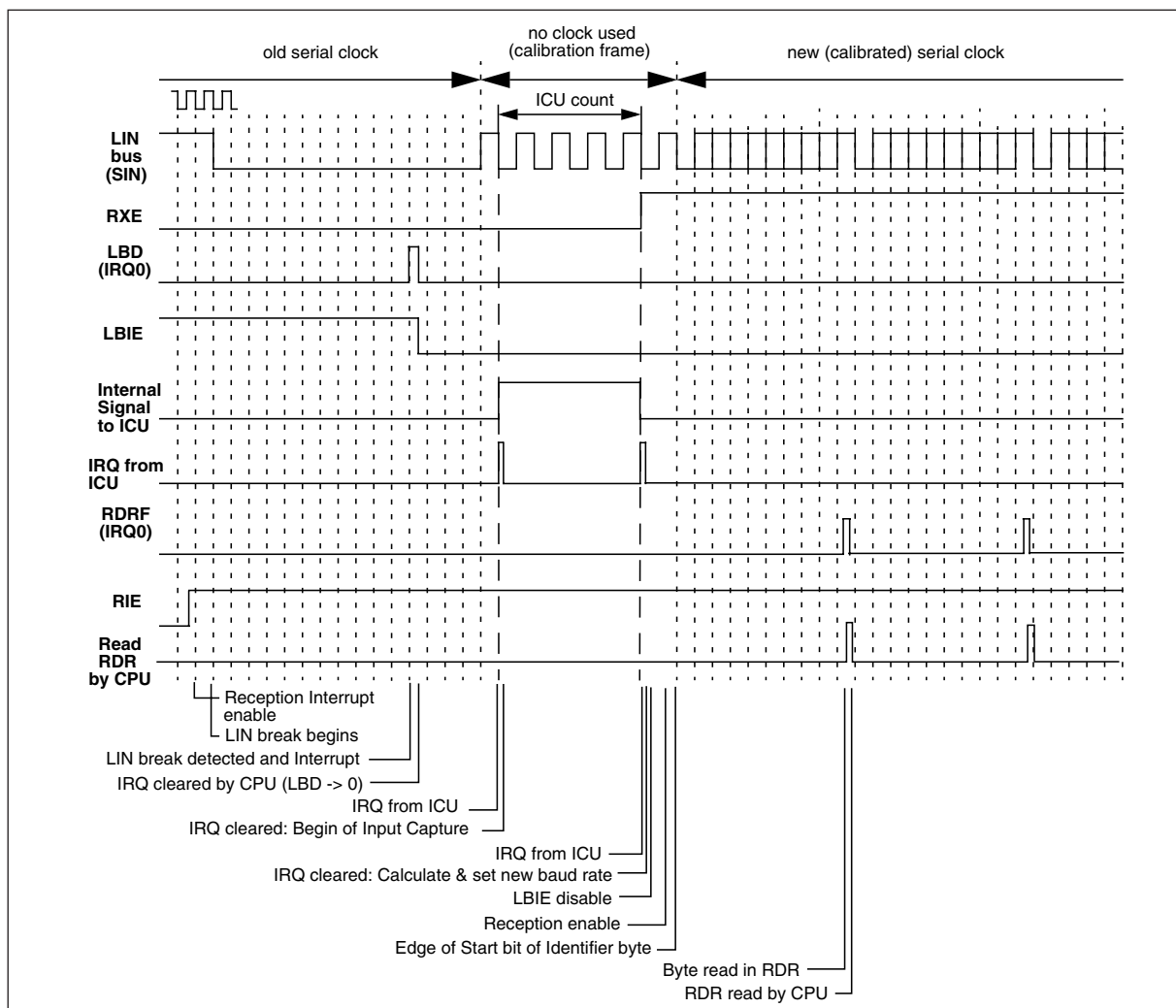
Note, that if reception is enabled ($RXE = 1$) and reception interrupt is enabled ($RIE = 1$) the Reception Data Framing Error (FRE) flag bit of the SSR will cause a reception interrupt 2 bit times (“8N1”) earlier than the LIN break interrupt, so it is recommended to turn off RXE, if a LIN break is expected.

LBD is only supported in operation mode 3.

The Figure 20.7-7 “UART behavior as slave in LIN mode” shows a typical start of a LIN message frame and the behavior of the UART.

Figure 20.7-7 UART behavior as slave in LIN mode

● LIN bus timing

Figure 20.7-8 LIN bus timing and UART signals

20.7.4 Direct Access to Serial Pins

UART allows the user to directly access to the transmission pin (SOTn) or the reception pin (SINn).

■ UART direct pin access

The UART provides the ability for the software to access directly to serial input or output pin. The software can always monitor the incoming serial data by reading the SIOP bit of the ESCR. If setting the Serial Output Pin direct access Enable (SOPE) bit of the ESCR the software can force the SOTn pin to a desired value. Note, that this access is only possible, if the transmission shift register is empty (i. e. no transmission activity).

In LIN mode this function can be used for reading back the own transmission and is used for error handling if something is physically wrong with the single-wire LIN-bus.

Notes:

- Write the desired value to the SIOP pin before enabling the output pin access to prevent undesired output level because SIOP holds the last written value.
- During a Read-Modify-Write operation the SIOP bit returns the actual value of the SOTn pin in the read cycle instead of the value of SINn during a normal read instruction.

20.7.5 Bidirectional Communication Function (Normal Mode)

In operation mode 0 or 2, normal serial bidirectional communication is available. Select operation mode 0 for asynchronous communication and operation mode 2 for synchronous communication.

■ Bidirectional communication function

The settings shown in Figure 20.7-9 “Settings for UART operation mode 0 and 2” are required to operate UART in normal mode (operation mode 0 or 2).

Figure 20.7-9 Settings for UART operation mode 0 and 2

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SCRn, SMRn	PEN	P	SBL	CL	AD	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 0	⊙	⊙	⊙	⊙	x	0	⊙	⊙	0	0	0	0	0	0	1	⊙
Mode 2	□	□	□	+	x	0	⊙	⊙	1	0	⊙	⊙	0	0	⊙	⊙
SSRn, TDRn /RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)							
Mode 0	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙								
Mode 2	□	⊙	□	⊙	⊙	⊙	⊙	⊙								
ESCRn, ECCRn	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES	/	LBR	MS	SCDE	SSM	/	RBI	TBI
Mode 0	x	x	x	x	⊙	⊙	0	0		0	x	x	x	0	⊙	⊙
Mode 2	x	x	x	x	⊙	⊙	□	⊙		x	⊙	⊙	⊙	0	□	□

⊙ : Bit used
 x : Bit not used
 1 : Set 1
 0 : Set 0
 □ : Bit used if SSM = 1 (Synchronous start-/stop-bit mode)
 + : Bit automatically set to correct value

n = 0, 1, 2, 3, 4

● Inter-CPU connection

As shown in Figure 20.7-10 “Connection example of UART mode 2 bidirectional communication”, interconnect two CPUs in UART mode 2

Figure 20.7-10 Connection example of UART mode 2 bidirectional communication

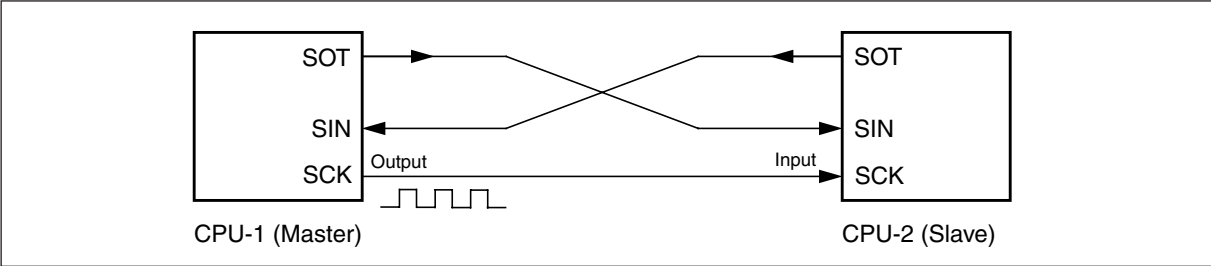
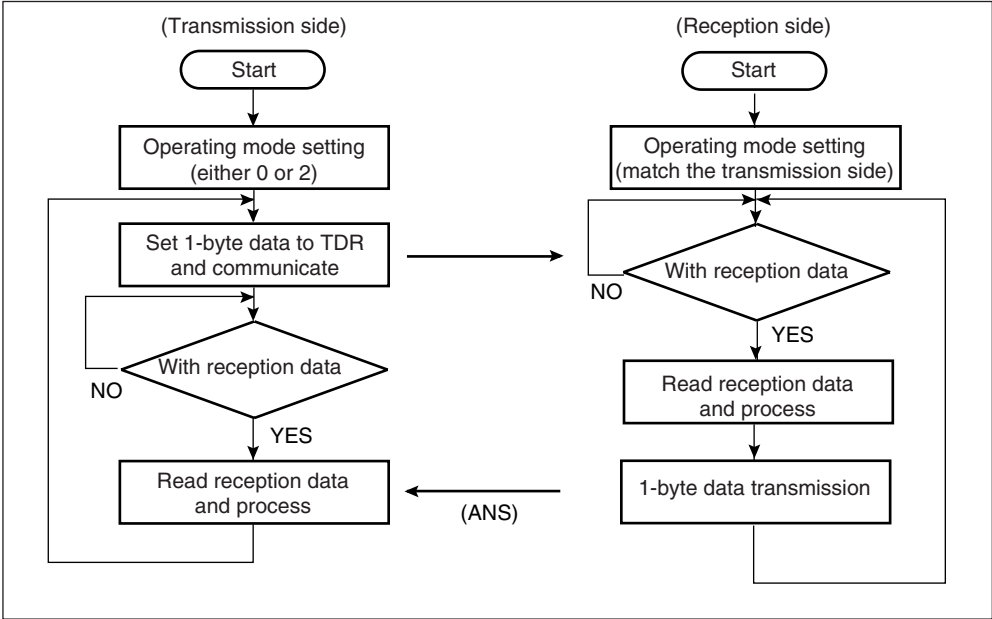


Figure 20.7-11 Example of master-slave communication flowchart



20.7.6 Master-Slave Communication Function (Multiprocessor Mode)

UART communication with multiple CPUs connected in master-slave mode is available for both master or slave systems.

■ Master-slave communication function

The settings shown in Figure 20.7-12 “Settings for UART operation mode 1” are required to operate UART in multiprocessor mode (operation mode 1).

Figure 20.7-12 Settings for UART operation mode 1

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SCRn, SMRn	PEN	P	SBL	CL	AD	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 1	+	x	⊙	⊙	⊙	0	⊙	⊙	0	1	0	0	0	0	1	⊙
SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)							
Mode 1	x	⊙	⊙	⊙	⊙	⊙	⊙	⊙								
ESCRn, ECCRn	LBIE	LBD	LBL1	LBLO	SOPE	SIOP	CCO	SCES	/	LBR	MS	SCDE	SSM	/	RBI	TBI
Mode 1	x	x	x	x	⊙	⊙	0	0		x	x	x	x	0	⊙	⊙

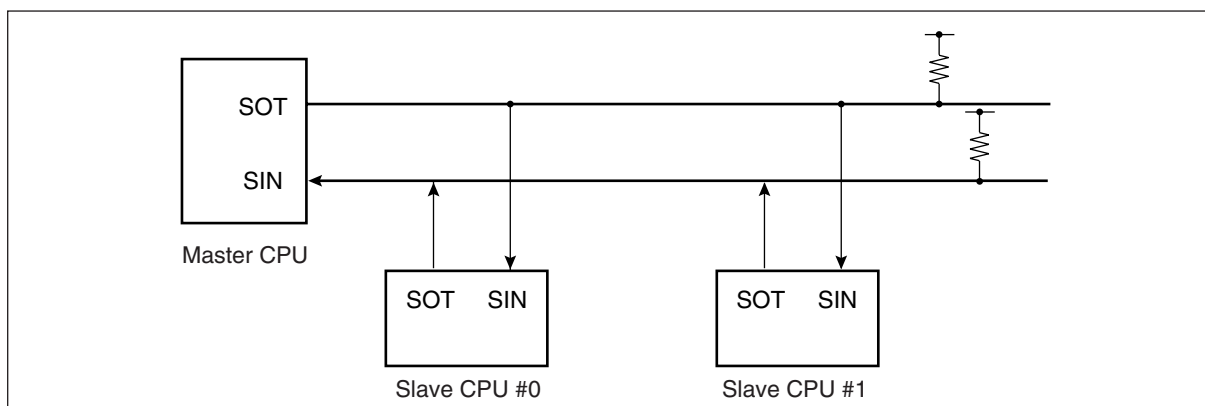
⊙ : Bit used
 x : Bit not used
 1 : Set 1
 0 : Set 0
 + : Bit automatically set to correct value

n = 0, 1, 2, 3, 4

● Inter-CPU connection

As shown in Figure 20.7-13 “Connection example of UART master-slave communication”, a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. UART can be used for the master or slave CPU.

Figure 20.7-13 Connection example of UART master-slave communication



● Function selection

Select the operation mode and data transfer mode for master-slave communication as shown in Table 20.7-3 “Selection of the master-slave communication function”.

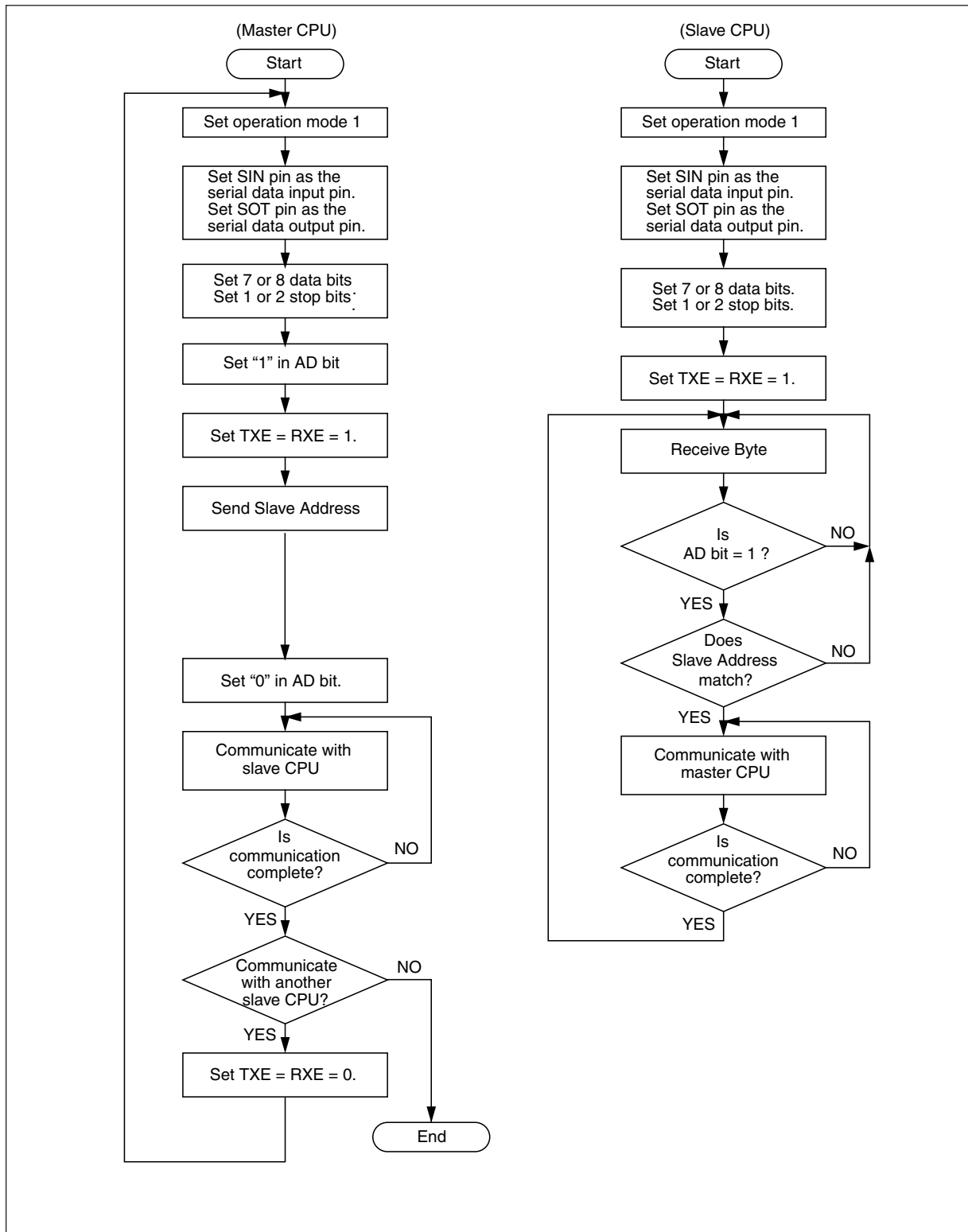
Table 20.7-3 Selection of the master-slave communication function

	Operation mode		Data	Parity	Synchronization method	Stop bit	Bit direction
	Master CPU	Slave CPU					
Address transmission and reception	Mode 1 (transmit/receive AD-bit)	Mode 1 (transmit/receive AD-bit)	AD="1" + 7- or 8-bit address	None	Asynchronous	1 or 2 bits	LSB or MSB first
Data transmission and reception			AD="0" + 7- or 8-bit data				

Communication procedure

When the master CPU transmits address data, communication starts. The A/D bit in the address data is set to 1, and the communication destination slave CPU is selected. Each slave CPU checks the address data using a program. When the address data indicates the address assigned to a slave CPU, the slave CPU communicates with the master CPU.

Figure 20.7-14 “Master-slave communication flowchart” shows a flowchart of master-slave communication (multiprocessor mode)

Figure 20.7-14 Master-slave communication flowchart

20.7.7 LIN Communication Function

UART communication with LIN devices is available for both LIN master or LIN slave systems.

■ LIN-master-slave communication function

The settings shown in the figure below are required to operate UART in LIN communication mode (operation mode 3).

Figure 20.7-15 Settings for UART in operation mode 3 (LIN)

SCRn, SMRn	PEN	P	SBL	CL	AD	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 3	+	x	+	+	x	0	⊙	⊙	1	1	0	0	0	0	1	⊙
SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)							
Mode 3	x	⊙	⊙	⊙	⊙	+	⊙	⊙								
ESCRx, ECCRx	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES	/	LBR	MS	SCDE	SSM	/	RBI	TBI
Mode 3	⊙	⊙	⊙	⊙	⊙	⊙	0	0		⊙	x	x	x	0	⊙	⊙

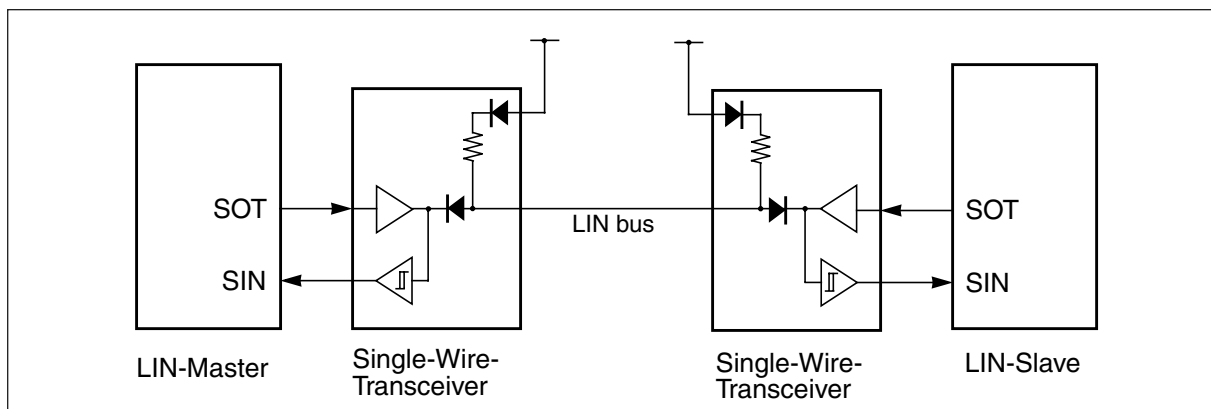
⊙ : Bit used
 x : Bit not used
 1 : Set 1
 0 : Set 0
 + : Bit automatically set to correct value

n = 0, 1, 2, 3, 4

● LIN device connection

As shown in the Figure below, a communication system of one LIN-Master device and a LIN-Slave device. UART can operate both as LIN-Master or LIN-Slave.

Figure 20.7-16 Connection example of a small LIN-Bus system

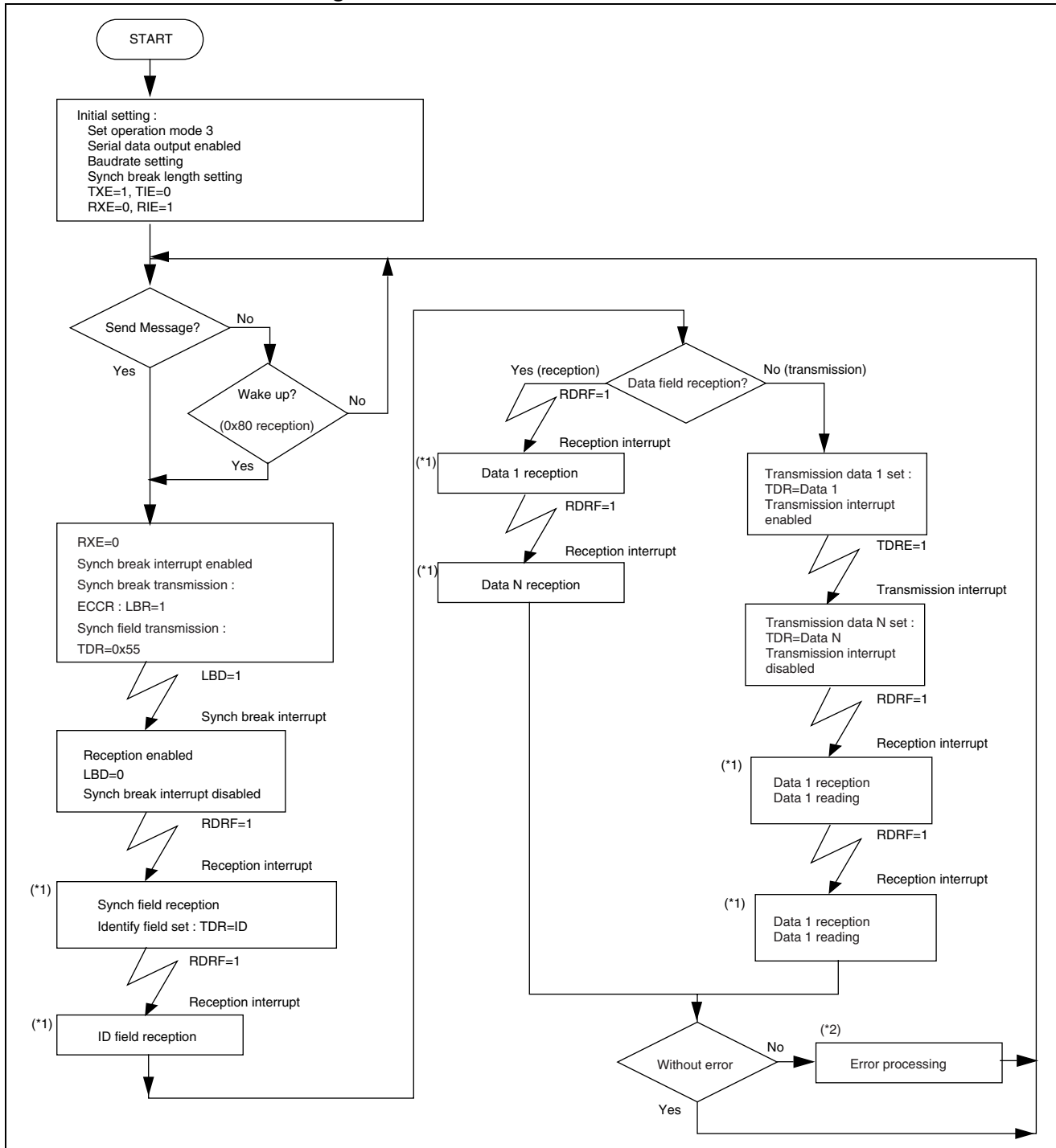


20.7.8 Sample Flowcharts for UART in LIN communication (Operation Mode 3)

This section contains sample flowcharts for UART in LIN communication.

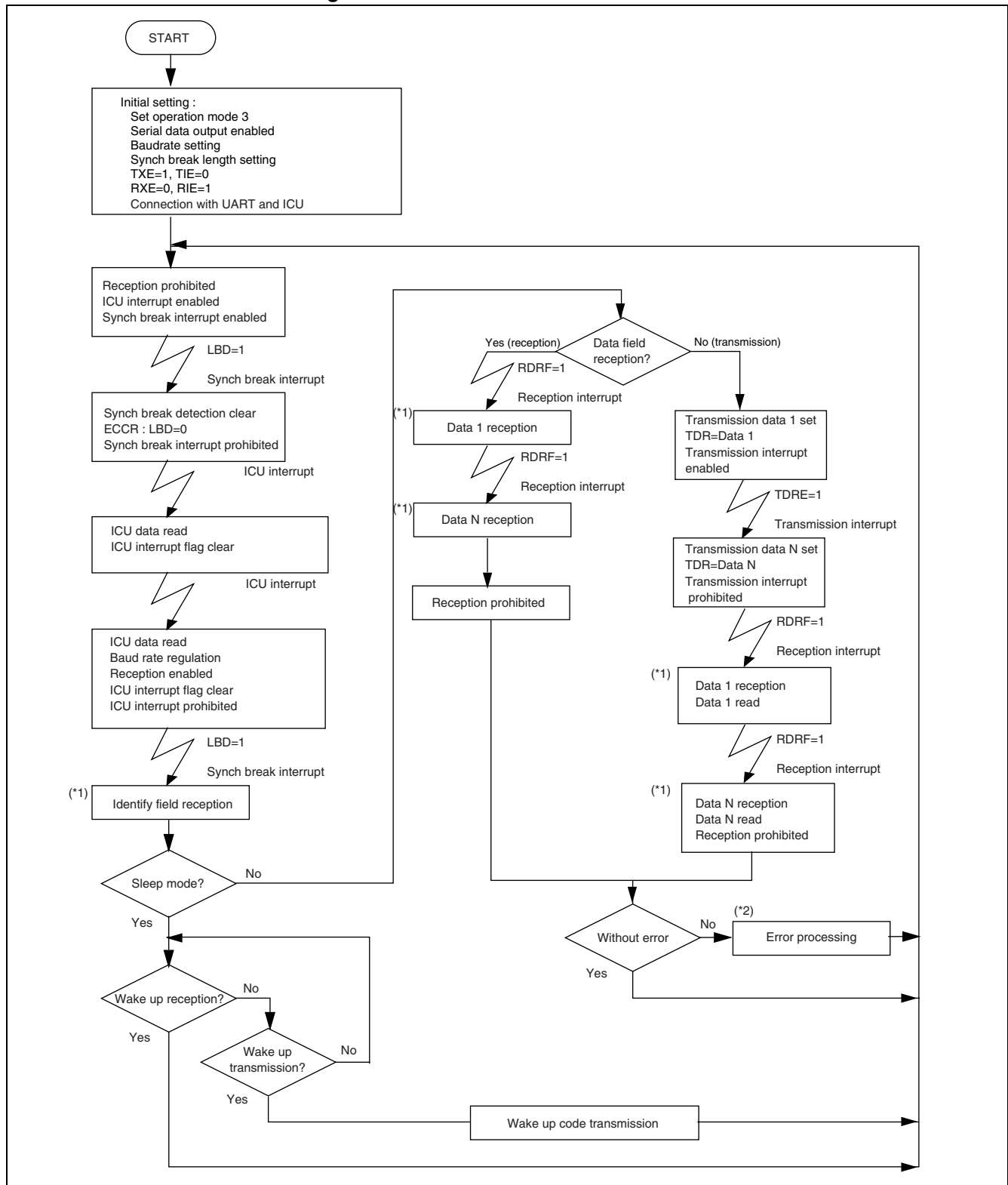
■ UART as master device

Figure 20.7-17 UART LIN master flow chart



■ UART as slave device

Figure 20.7-18 UART LIN slave flow chart



20.8 Notes on Using UART

Notes on using UART are given below.

■ Notes on using UART

● Enabling operations

In UART, the control register (SCR) has TXE (transmission) and RXE (reception) operation enable bits. Both, transmission and reception operations, must be enabled before the communication starts because they have been disabled as the default value (initial value). The operation can also be canceled by disabling these bits.

● Communication mode setting

Set the communication mode while the system is not operating. If the mode is changed during transmission or reception, the transmission or reception is stopped and possible data will be lost.

● Transmission interrupt enabling timing

The default (initial value) of the transmission data empty flag bit (SSR: TDRE) is “1” (no transmission data and transmission data write enable state). A transmission interrupt request is generated as soon as the transmission interrupt request is enabled (SSR: TIE=1). Be sure to set the TIE flag to “1” after setting the transmission data to avoid an immediate interrupt.

● Using LIN operation mode 3

The LIN features are available in mode 3 (transmitting, receiving synch break), but using mode 3 sets the UART data format automatically to LIN format (8N1, LSB first). Note, that the length of the synch break for transmission is variable but for reception it is fixed 11-bit time.

Note:

During LIN operation, please set SCES bit of ESCR register to 0.

● Changing operation settings

It is strongly recommended to reset UART after changing operation settings. Particularly if (for example) start-/stop-bits added to or removed from the data format.

Note:

If settings in the Serial Mode Register (SMR) are desired, it is not useful to set the UPCL bit at the same time to reset UART. The correct operation settings are not guaranteed in this case. Thus it is recommended to set the bits of the SMR and then to set them again plus the UPCL bit.

● LIN slave settings

Set the baud rate before receiving the first LIN synch break for the slave operation. Otherwise, duration of the synch break can not be correctly checked against the minimum requirement of the LIN specification (13 master bit time and 11 slave bit time).

● Software compatibility

Although this UART is similar to other UART in other microcontrollers it is not software compatible to them. The programming models may be the same, but the structure of the registers differ. Furthermore the setting of the baud rate is now determined by a reload value instead of selecting a predefined value.

● Bus idle function

The Bus Idle Function cannot be used in synchronous mode 2.

● AD bit (serial control register (SCR): address/data type select bit)

Special care has to be taken when using the AD bit (Address-Data-Bit for multiprocessor mode 1) of the Serial Control Register. This bit is both a control and a flag bit, because writing to it sets the AD bit for transmission, whereas reading from it returns the last received AD bit. Internally, the received and the transmitted value are stored in different registers, but in Read-Modify-Write instructions, the received value is read, modified and then written back for transmission. This can lead to a wrong value in the AD bit, when one of the other bits in the same register is accessed by an instruction of this kind.

Therefore, this bit should be written by the last register access before transmission. Alternatively, using byte wise access and writing the correct values for all bits at once avoids this problem.

Furthermore, the AD bit is not buffered like the transmission data register.

Changing the bit when transmission will alter the AD bit of the currently transmitted data.

● Software reset of UART

Perform the software reset (SMR: UPCL=1), when the TXE bit of the SCR register is "1".

● LIN Synch field wait state

In mode 3 (LIN operation), the LBD bit in the ESCR register is set to "1" if the input signal is kept at "0" for more than equal to 11-bit time. Then the UART waits for the following synch field to be received. If the UART is set into this state for other reasons than the synch break, it should be initialized by the software reset (SMR: UPCL=1).

CHAPTER 21

400 kHz I²C INTERFACE

This section describes the functions and operation of the fast I²C interface.

21.1 “I²C Interface Overview”

21.2 “I²C Interface Registers”

21.3 “I²C Interface Operation”

21.4 “Programming Flow Charts”

21.1 I²C Interface Overview

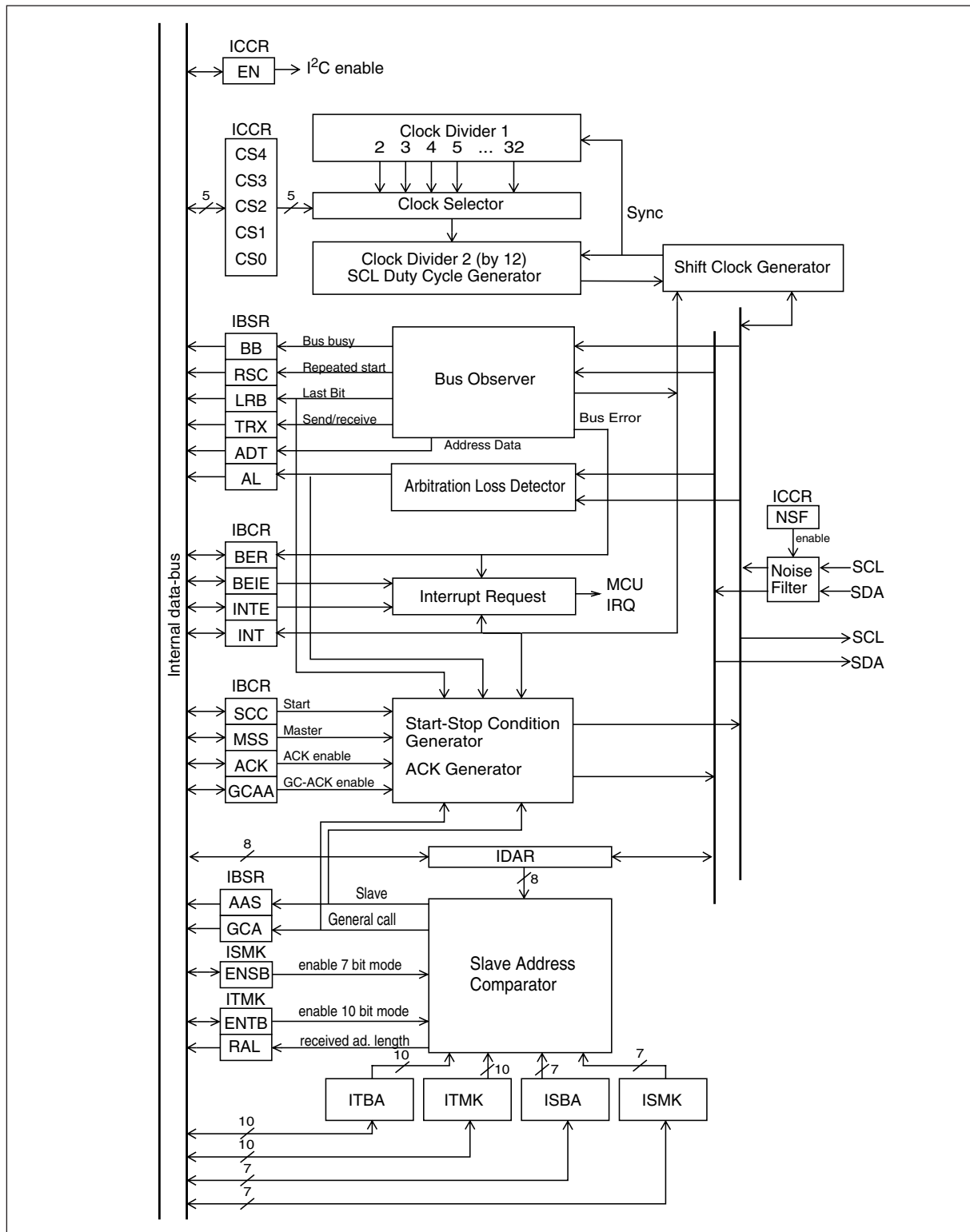
The I²C interface is a serial I/O port supporting the Inter IC bus, operating as a master/slave device on the I²C bus.

The MB90340 Series contain two I²C Interface modules (only devices with suffix-"c".)

■ Features

- Master/slave transmitting and receiving functions
- Arbitration function
- Clock synchronization function
- General call addressing support
- Transfer direction detection function
- Repeated start condition generation and detection function
- Bus error detection function
- 7 bit addressing as master and slave
- 10 bit addressing as master and slave
- Possibility to give the interface a seven and a ten bit slave address
- Acknowledging upon slave address reception can be disabled (Master-only operation)
- Address masking to give interface several slave addresses (in 7 and 10 bit mode)
- Up to 400 KBit transfer rate
- Possibility to use built-in noise filters for SDA and SCL
- Can receive data at 400 KBit if machine clock is higher than 6 MHz regardless of prescaler setting
- Can generate MCU interrupts on transmission and bus error events
- Supports being slowed down by a slave on bit and byte level

The I²C interface does not support SCL clock stretching on bit level since it can receive the full 400 kBit data rate if the machine clock is higher than 6 MHz regardless of the prescaler setting. However, clock stretching on byte level is performed since SCL is pulled low during an interrupt (INT="1" in IBCR register).

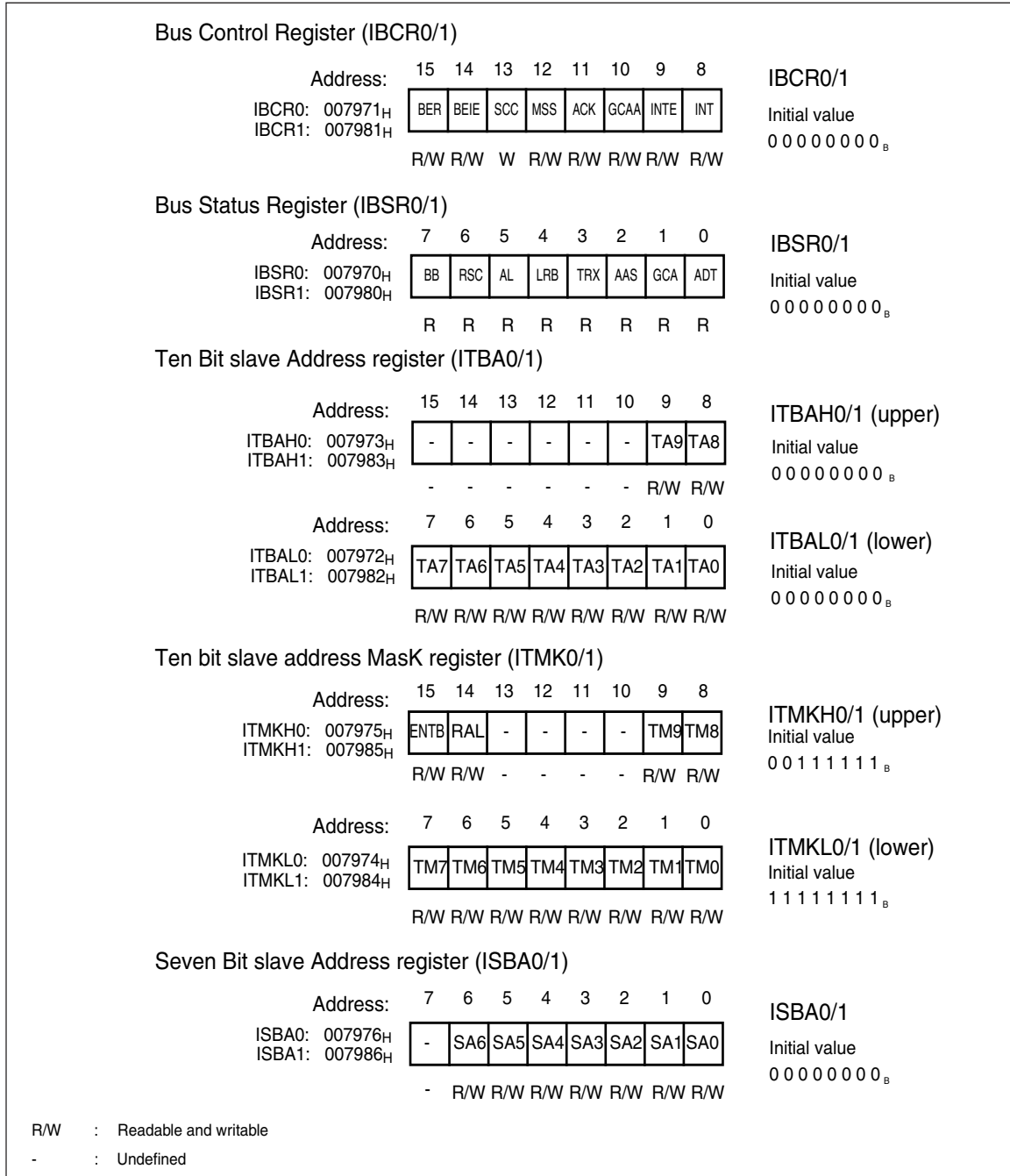
Figure 21.1-1 Block diagram

21.2 I²C Interface Registers

This section describes the function of the I²C interface registers in detail.

■ I²C Interface registers

Figure 21.2-1 I²C Interface registers



Seven bit slave address MasK register (ISMK0/1)

Address:	15	14	13	12	11	10	9	8	
ISMK0: 007977 _H	ENSB	SM6	SM5	SM4	SM3	SM2	SM1	SM0	ISMK0/1
ISMK1: 007987 _H									Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	0 1 1 1 1 1 1 1 _B

Data Register (IDAR0/1)

Address:	7	6	5	4	3	2	1	0	
IDAR0: 007978 _H	D7	D6	D5	D4	D3	D2	D1	D0	IDAR0/1
IDAR1: 007988 _H									Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	0 0 0 0 0 0 0 0 _B

Clock control register (ICCR0/1)

Address:	15	14	13	12	11	10	9	8	
ICCR0: 00797B _H	-	NSF	EN	CS4	CS3	CS2	CS1	CS0	ICCR0/1
ICCR1: 00798B _H									Initial value
	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	0 0 0 1 1 1 1 1 _B

R/W : Readable and writable
 - : Undefined

21.2.1 Bus Status Register (IBSR)

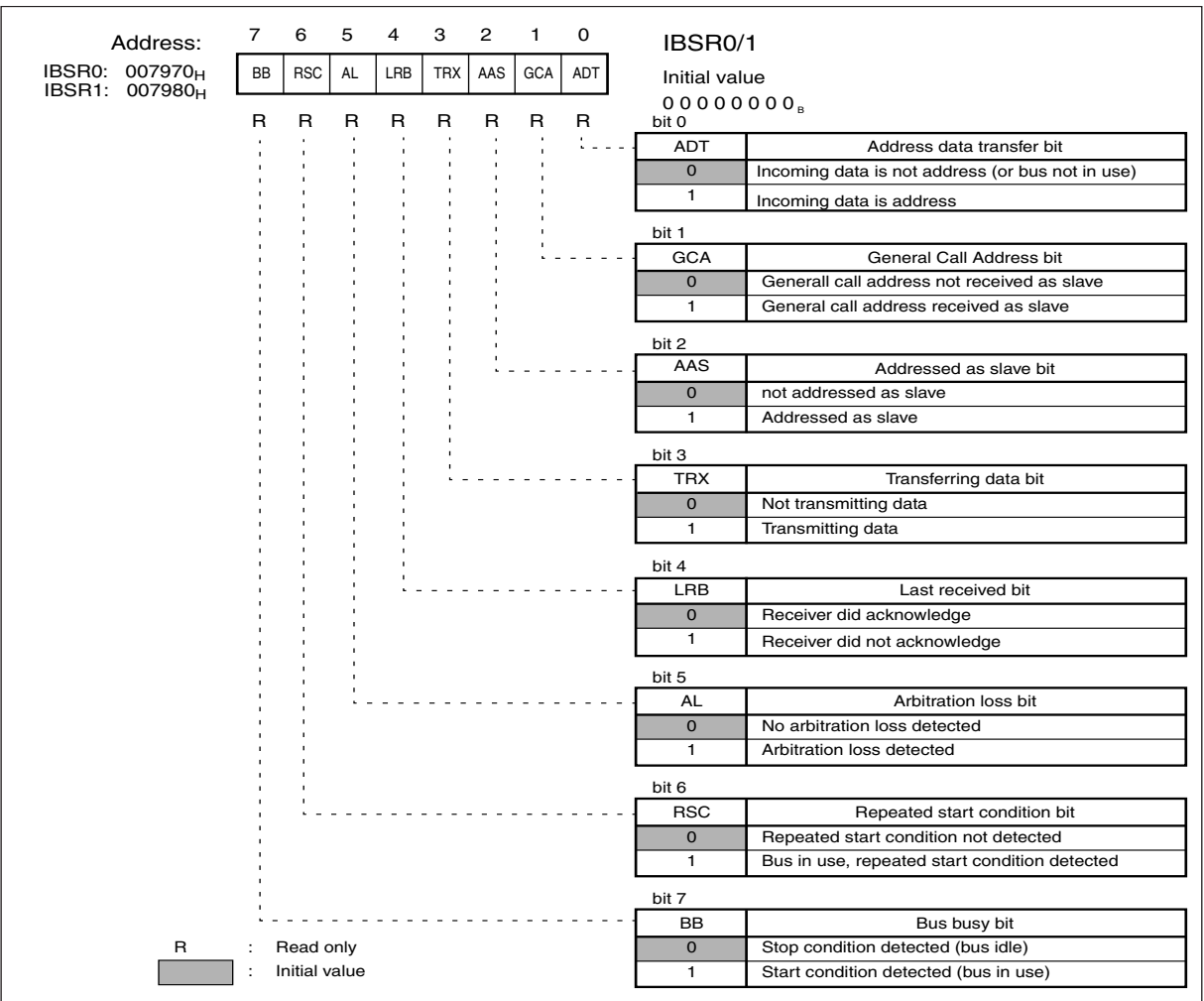
The bus status register (IBSR) has the following functions:

- Bus busy detection
- Repeated start condition detection
- Arbitration loss detection
- Acknowledge detection
- Data transfer direction indication
- Addressing detection as slave
- General call address detection
- Address/data detection

■ Bus status register (IBSR)

This register is read-only, all bits are controlled by the hardware. All bits are cleared if the interface is not enabled (EN = "0" in ICCR).

Figure 21.2-2 Configuration of the bus status register (IBSR0/1)



■ Bus status register (IBSR) contents

Table 21.2-1 Function of each bit of the bus status register (IBSR0/1) (1/2)

Bit name		Function
bit 7	BB: Bus busy bit	This bit indicates the status of the I ² C bus. "0": Stop condition detected (bus idle) "1": Start condition detected (bus in use) This bit is set to "1" if a start condition is detected. It is reset upon a stop condition.
bit 6	RSC: Repeated start condition bit	This bit indicates detection of a repeated start condition. "0": Repeated start condition not detected. "1": Repeated start condition detected (bus in use). This bit is cleared at the end of an address data transfer (ADT="0") or detection of a stop condition.
bit 5	AL: Arbitration loss bit	This bit indicates an arbitration loss. "0": No arbitration loss detected. "1": Arbitration loss occurred during master sending. This bit is cleared by writing "0" to the INT bit or by writing "1" to the MSS bit in the IBCR register. An arbitration loss occurs if: - the data sent does not match the data read on the SDA line at the rising SCL edge - a repeated start condition is generated by another master in the first bit of a data byte - The interface could not generate a start or stop condition because signal transition caused from "1" to "0" by a certain external condition observed at the SCL line.
bit 4	LRB: Last received bit	This bit is used to indicate the acknowledge from the receiving device. "0": Receiver acknowledged. "1": Receiver did not acknowledge. It is changed by the hardware upon reception of bit 9 (acknowledge bit) and is also cleared by a start or stop condition.
bit 3	TRX: Transferring data bit	This bit indicates data transmission operation. "0": Not transmitting data. "1": Transmitting data. It is set to "1": - if a start condition was generated in master mode - if addressed as slave in read access. It is set to "0" if: - the bus is idle (BB="0") - an arbitration loss occurred - "1" is written to the SCC bit during master interrupt (MSS="1" and INT="1") - the MSS bit being cleared during master interrupt (MSS="1" and INT="1") - the interface is in slave mode and the last transferred byte was not acknowledged - the interface is in slave mode and it is receiving data - the interface is in master mode and is reading data from a slave
bit 2	AAS: Addressed as slave bit	This bit indicates detection of a slave addressing. "0": Not addressed as slave. "1": Addressed as slave. This bit is cleared by a (repeated-) start or stop condition. It is set if the interface detects its seven and/or ten bit slave address.

Table 21.2-1 Function of each bit of the bus status register (IBSR0/1) (2/2)

Bit name		Function
bit 1	GCA: General call address bit	This bit indicates detection of a general call address (0x00). "0": General call address not received as slave. "1": General call address received as slave. This bit is cleared by a (repeated-) start or stop condition.
bit 0	ADT: Address data transfer bit	This bit indicates the detection of an address data transfer. "0": Incoming data is not address data (or bus is not in use). "1": Incoming data is address data. This bit is set to "1" by a start condition. It is cleared after the second byte if a ten bit slave address header with write access is detected, else it is cleared after the first byte. This bit is also cleared when: - "0" is written to the MSS bit during a master interrupt (MSS="1" and INT="1" in IBCR) - "1" is written to the SCC bit during a master interrupt (MSS="1" and INT="1" in IBCR) - the INT bit is being cleared - the beginning of every byte transfer if the interface is not involved in the current transfer as master or slave

21.2.2 Bus Control Register (IBCR)

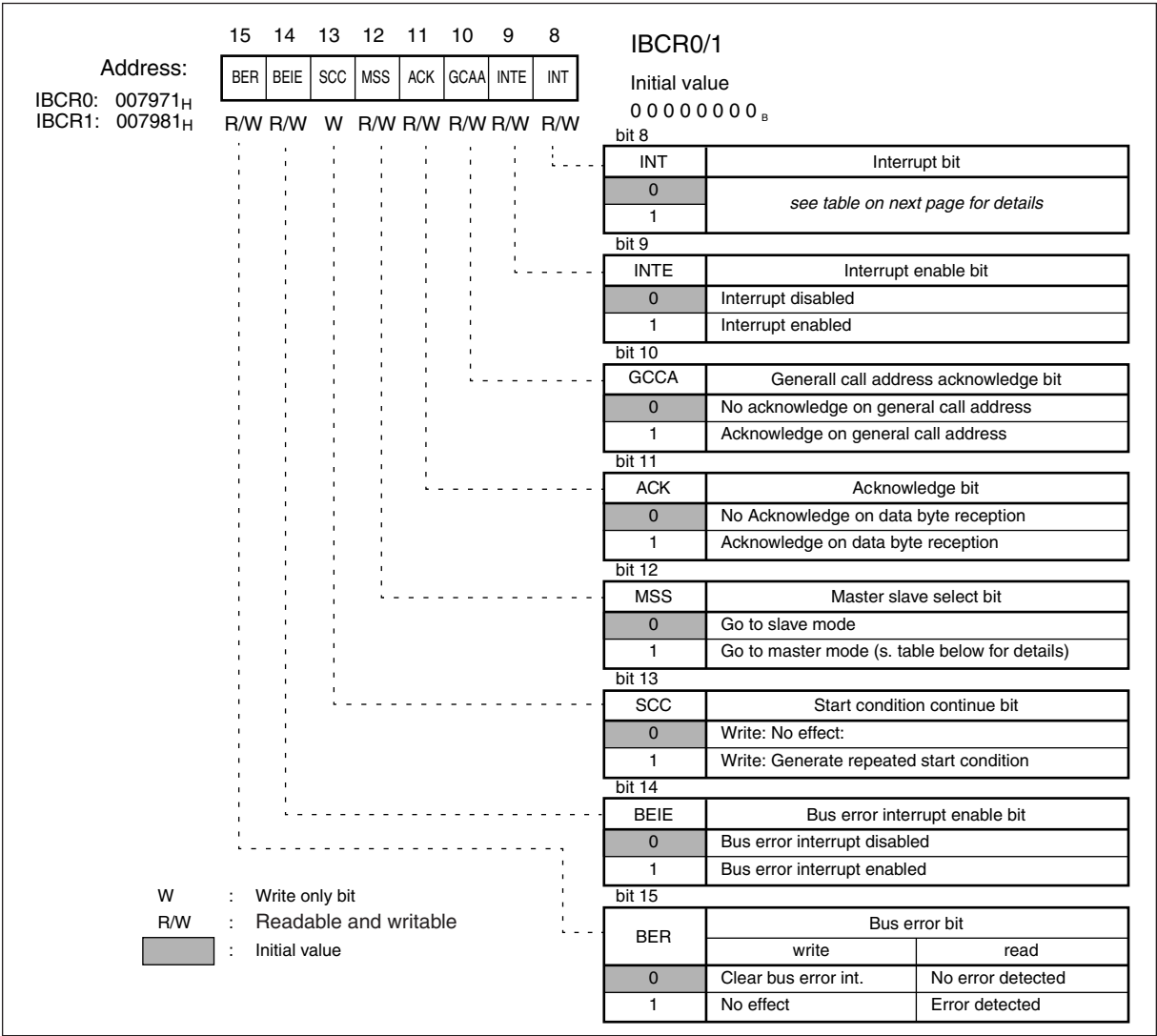
The bus control register (IBCR) has the following functions:

- Interrupt enabling flags
 - Interrupt generation flag
 - Bus error detection flag
 - Repeated start condition generation
 - Master / slave mode selection
 - General call acknowledge generation enabling
 - Data byte acknowledge generation enabling
-

■ Bus control register (IBCR)

Write access to this register should only occur while the INT="1" or if a transfer is to be started. The user should not write to this register during an ongoing transfer since changes to the ACK or GCAA bits could result in bus errors. All bits in this register except the BER and the BEIE bit are cleared if the interface is not enabled (EN="0" in ICCR).

Figure 21.2-3 Configuration of the bus control register (IBCR0/1)



■ Bus control register (IBCR) contents

Table 21.2-2 Function of each bit of the bus control register (IBCR0/1) (1/3)

Bit name		Function
bit 15	BER: Bus error bit	This bit is the bus error interrupt flag. It is set by the hardware and cleared by the user. It always reads "1" in a Read-Modify-Write access. Write access: "0": Clear bus error interrupt flag "1": No effect Read access: "0": No bus error detected "1": One of the error conditions described below detected When this bit is set, the EN bit in the ICCR register is cleared, the I²C interface goes to pause status, data transfer is interrupted and all bits in the IBSR and the IBCR registers except BER and BEIE are cleared. The BER bit must be cleared before the interface may be reenabled. This bit is set to "1" if: - start or stop conditions are detected at wrong places: during an address data transfer or during the transfer of the bits two to nine (acknowledge bit) - a ten bit address header with read access is received before a ten bit write access
bit 14	BEIE: Bus error interrupt enable bit	This bit enables the bus error interrupt. It only can be changed by the user. "0": Bus error interrupt disabled "1": Bus error interrupt enabled Setting this bit to "1" enables MCU interrupt generation when the BER bit is set to "1".
bit 13	SCC Start condition continue bit	This bit is used to generate a repeated start condition. It is write only - it always reads "0". "0": No effect "1": Generate repeated start condition during master transfer A repeated start condition is generated if a "1" is written to this bit while an interrupt in master mode (MSS="1" and INT="1") and the INT bit is cleared automatically.

Table 21.2-2 Function of each bit of the bus control register (IBCR0/1) (2/3)

Bit name		Function
bit 12	MSS: Master slave select bit	This is the master/slave mode selection bit. It can only be set by the user, but it can be cleared by the user and the hardware. "0": Go to slave mode "1": Go to master mode, generate start condition and send address data byte in IDAR register. It is cleared if an arbitration loss event occurs during master sending. If a "0" is written to it during a master interrupt (MSS="1" and INT="1"), the INT bit is cleared automatically, a stop condition will be generated and the data transfer ends. Note that the MSS bit is reset immediately, the generation of the stop condition can be checked by polling the BB bit in the IBSR register. If a "1" is written to it while the bus is idle (MSS="0" and BB="0"), a start condition is generated and the contents of the IDAR register (which should be address data) is sent. If a "1" is written to the MSS bit while the bus is in use (BB="1" and TRX="0" in IBSR; MSS="0" in IBCR), the interface waits until the bus is free and then starts sending. If the interface is addressed as slave with write access (data reception) in the meantime, it will start sending after the transfer ended and the bus is free again. If the interface is sending data as slave in the meantime (AAS="1" and TRX="1" in IBSR), it will not start sending data if the bus is free again. It is important to check whether the interface was addressed as slave (AAS="1" in IBSR), sent the data byte successfully (MSS="1" in IBCR) or failed to send the data byte (AL="1" in IBSR) at the next interrupt!
bit 11	ACK: Acknowledge bit	This bit enables the acknowledge generation on data byte reception. It only can be changed by the user. "0": The interface will not acknowledge on data byte reception "1": The interface will acknowledge on data byte reception This bit is not valid when receiving address bytes in slave mode - if the interface detects its 7 or 10 bit slave address, it will acknowledge if the corresponding enable bit (ENTB in ITMK or ENSB in ISMK) is set. Write access to this bit should occur during an interrupt (INT="1") or if the bus is idle (BB="0" in the IBSR register) only.
bit 10	GCAA: General call address acknowledge bit	This bit enables acknowledge generation when a general call address is received. It only can be changed by the user. "0": The interface will not acknowledge on general call address byte reception. "1": The interface will acknowledge on general call address byte reception. Write access to this bit should occur during an interrupt (INT="1") or if the bus is idle (BB="0" in IBSR register) or the interface is disabled (EN="0" in ICCR register) only.
bit 9	INTE: Interrupt enable bit	This bit enables the MCU interrupt generation. It only can be changed by the user. "0": Interrupt disabled "1": Interrupt enabled Setting this bit to "1" enables MCU interrupt generation when the INT bit is set to "1" (by the hardware).

Table 21.2-2 Function of each bit of the bus control register (IBCR0/1) (3/3)

Bit name		Function
bit 8	INT: Interrupt flag bit	This bit is the transfer end interrupt request flag. It is changed by the hardware and can be cleared by the user. It always reads "1" in a Read-Modify-Write access. Write access: "0": Clear transfer end interrupt request flag "1": No effect Read access: "0": Transfer not ended or not involved in current transfer or bus is idle "1": Set at the end of a 1-byte data transfer or reception including the acknowledge bit under the following conditions: Device is bus master. Device is addressed as slave. General call address received. Arbitration loss occurred. Set at the end of an address data reception (after first byte if seven bit address received, after second byte if ten bit address received) including the acknowledge bit if the device is addressed as slave. While this bit is "1" the SCL line will hold an "L" level signal. Writing "0" to this bit clears the setting, releases the SCL line, and executes transfer of the next byte or a repeated start or stop condition is generated. Additionally, this bit is cleared if a "1" is written to the SCC bit or the MSS bit is being cleared.

■ SCC, MSS and INT bit competition

Simultaneously writing to the SCC, MSS and INT bits causes a competition to transfer the next byte, to generate a repeated start condition or to generate a stop condition. In these cases the order of priority is as follows:

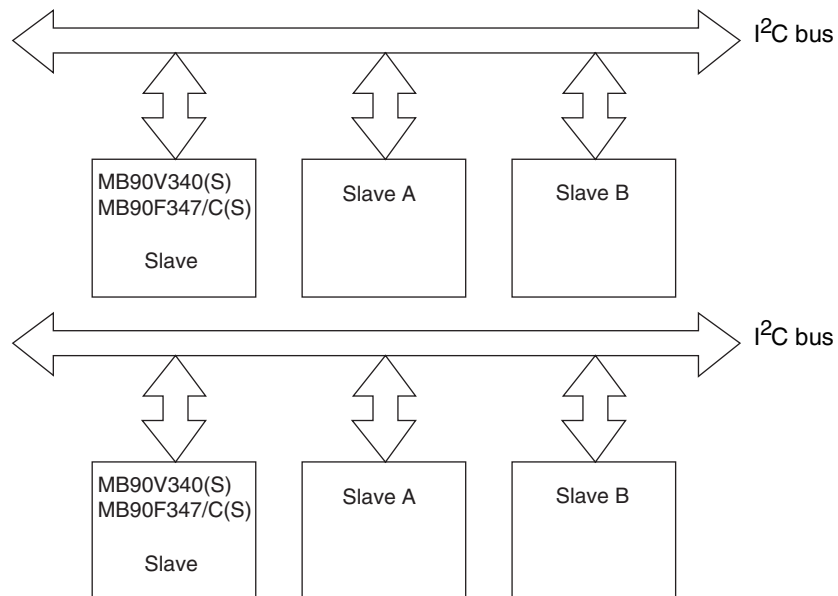
- Next byte transfer and stop condition generation.
When "0" is written to the INT bit and "0" is written to the MSS bit, the MSS bit takes priority and a stop condition is generated.
- Next byte transfer and start condition generation.
When "0" is written to the INT bit and "1" is written to the SCC bit, the SCC bit takes priority. A repeated start condition is generated and the contents of the IDAR register is sent.
- Repeated start condition generation and stop condition generation.
When a "1" is written to the SCC bit and "0" to the MSS bit, the MSS bit clearing takes priority. A stop condition is generated and the interface enters slave mode.

Note :

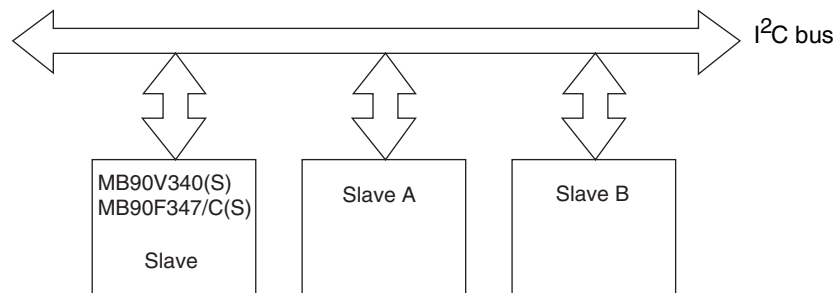
Note on using MB90V340(S), MB90F347/C(S) (part number without A suffix)

If there are some other master mode LSIs on the bus, the device can not be used as the master mode.

- Example of usable configuration



- Example of unusable configuration

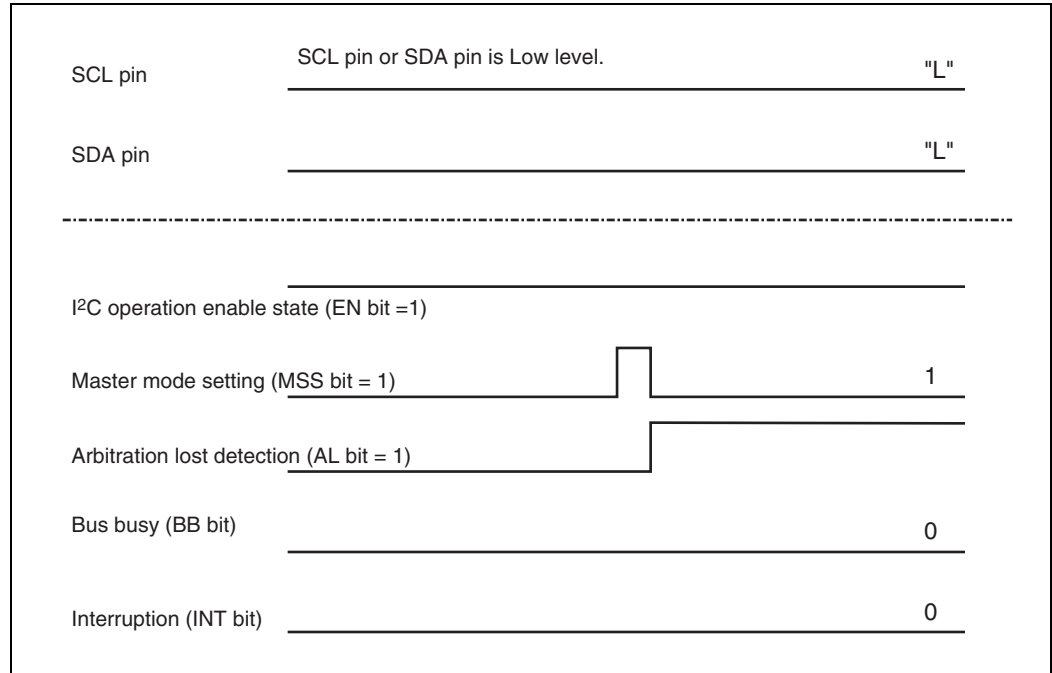


If specific conditions are met, the AL (arbitration lost) bit does not set the INT (interrupt) bit. Those conditions are presented in figure 21.2-4 and 21.2-5.

● **Case 1: When SCL and SDA signals are kept at "L";**

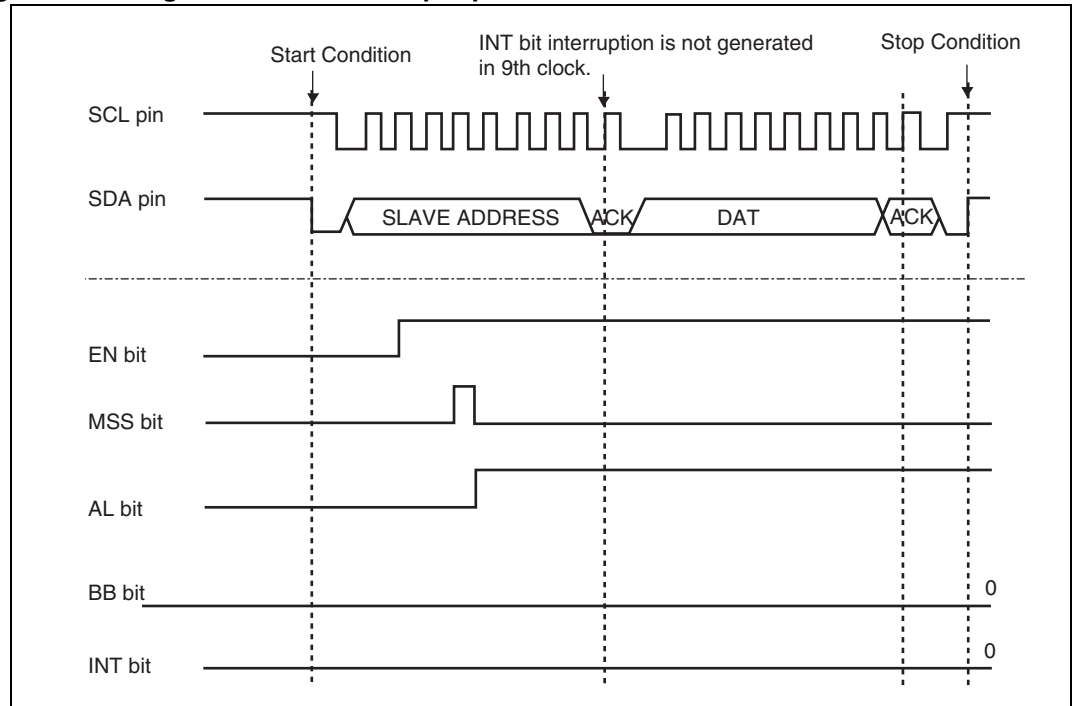
The AL bit is set immediately after the MSS bit set to "1" while the BB bit is indicating "0" (no start condition is detected). However the AL bit will not set the INT bit under this circumstances.

Figure 21.2-4 Diagram of timing at which an interrupt upon detection of " AL bit = 1 " does not occur



● **Case 2: When I²C interface is enabled while there is ongoing communication with another bus master;**

The interface participates in the I²C bus while the bus is occupied with ongoing communication if the EN bit is set from "0" to "1". In this case the BB bit stays "0" (no start condition is detected) and setting the MSS bit to "1" results in the AL bit indicating "1". However the AL bit will not set the INT bit under this circumstances.

Figure 21.2-5 Diagram of timing at which an interrupt upon detection of "AL bit = 1" does not occur

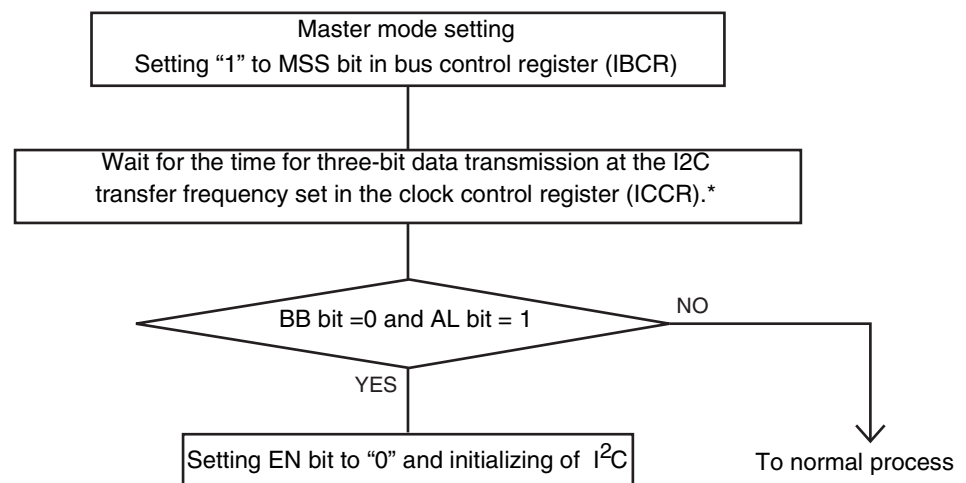
If a symptom as described above can occur, follow the procedure below for software processing.

1. Execute the instruction that generates a start condition (set the MSS bit to 1).
2. Use, for example, the timer function to wait for the time for three - bit data transmission at the I²C transfer frequency set in the ICCR register.*

Example: Time for three-bit data transmission at an I²C transfer frequency of 100 kHz = $(1/100 \times 10^3) \times 3 = 30$

3. Check the AL and BB bits in the IBSR register and, if the AL and BB bits are 1 and 0, respectively, set the EN bit in the ICCR register to 0 to initialize I²C. When the AL and BB bits are not so, perform normal processing.

A sample flow is given below.

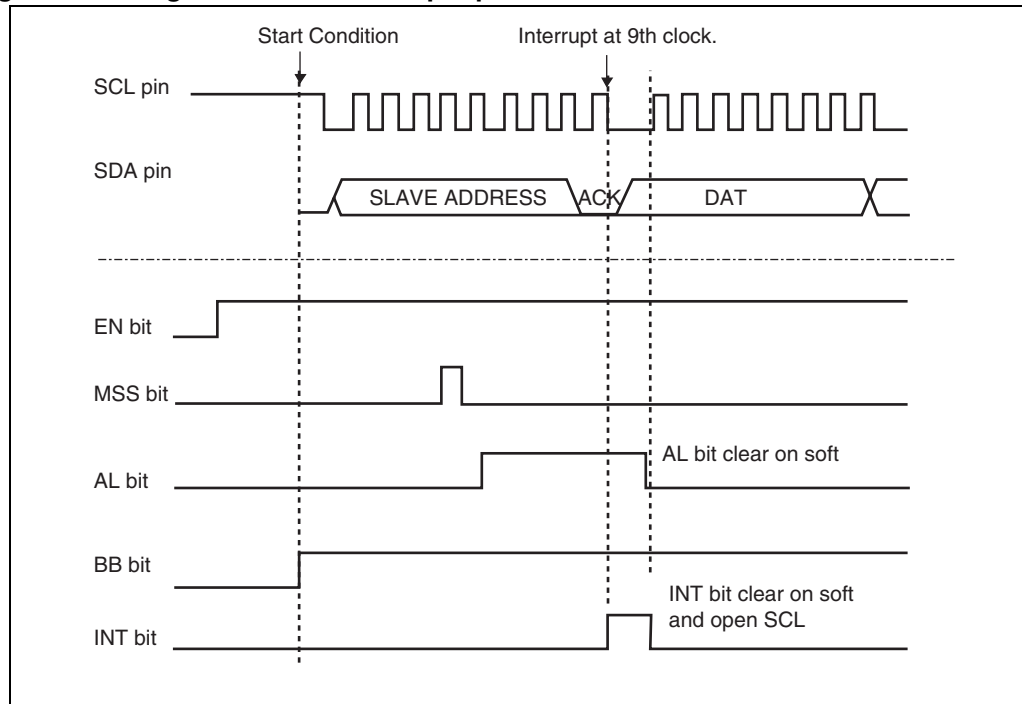


*: Arbitration lost is detected within 3-bit time after setting the MSS bit to "1".

● Example of occurrence of an interrupt (INT bit = 1) upon detection of "AL bit = 1"

When an instruction which generates a start condition is executed (setting the MSS bit to 1) with "bus busy" detected (BB bit = 1) and arbitration is lost, the INT bit interrupt occurs upon detection of "AL bit = 1".

Figure 21.2-6 Diagram of timing at which an interrupt upon detection of "AL bit = 1" occurs



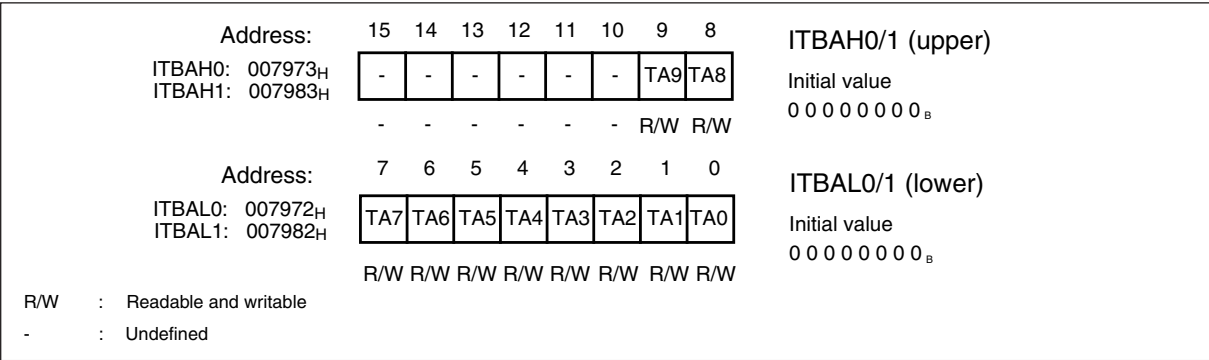
21.2.3 Ten Bit Slave Address Register (ITBA)

This register (ITBAH / ITBAL) designates the ten bit slave address.

■ Ten bit slave address register (ITBA)

Write access to this register is only possible if the interface is disabled (EN="0" in ICCR).

Figure 21.2-7 Configuration of the ten bit slave address register (IBTA0/1)



■ Ten bit slave address register (ITBA) contents

Table 21.2-3 Function of each bit of the ten bit slave address register (ITBA0/1)

Bit name		Function
bits 15 to 10	Undefined	These bits always return "0".
bits 9 to 0	TBA9 to 0: Ten bit slave address	When address data is received in slave mode, it is compared to the ITBA register if the ten bit address is enabled (ENTB="1" in the ITMK register). An acknowledge is sent to the master after reception of a ten bit address header with write access1. Then, the second incoming byte is compared to the TBAL register. If a match is detected, an acknowledge signal is sent to the master device and the AAS bit is set. Additionally, the interface acknowledges upon the reception of a ten bit header with read access2 after a repeated start condition. All bits of the slave address may be masked using the ITMK register. The received ten bit slave address is written back to the ITBA register, it is only valid while the AAS bit in the IBSR register is "1".

Note:

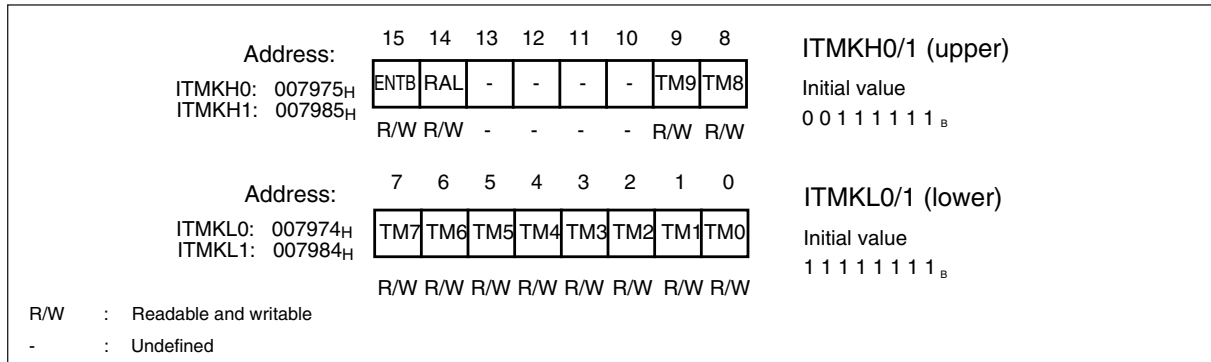
- A ten bit header (write access) consists of the following bit sequence: 11110, TA9, TA8, 0.
- A ten bit header (read access) consists of the following bit sequence: 11110, TA9, TA8, 1.

21.2.4 Ten Bit Address Mask Register (ITMK)

This register contains the ten bit slave address mask and the ten bit slave address enable bit.

■ Ten bit address mask register (ITMK)

Figure 21.2-8 Configuration of the ten bit address mask register (ITMK0/1)



■ Ten bit address mask register (ITMK) contents

Table 21.2-4 Function of each bit of the ten bit address mask register (ITMK0/1)

Bit name		Function
bit 15	ENTB: Enable ten bit slave address bit	This bit enables the ten bit slave address (and the acknowledging upon its reception). Write access to this bit is only possible if the interface is disabled (EN="0" in ICCR). "0": Ten bit address disabled "1": Ten bit address enabled
bit 14	RAL: Received slave address length bit	This bit indicates whether the interface was addressed as a seven or ten bit slave. It is read-only. "0": Addressed as seven bit slave "1": Addressed as ten bit slave This bit can be used to determine whether the interface was addressed as a seven or ten bit slave if both slave addresses are enabled (ENTB="1" and ENSB="1"). Its contents is only valid if the AAS bit in the IBSR register is "1". This bit is also reset if the interface is disabled (EN="0" in ICCR).
bit 13 to 10	Undefined	These bits always return "1" during reading.
bit 9 to 0	TMK: Ten bit slave address mask bits	This register is used to mask the ten bit slave address of the interface. Write access to these bits is only possible if the interface is disabled (EN="0" in ICCR). "0": Bit is not used in slave address comparison "1": Bit is used in slave address comparison This can be used to make the interface acknowledge on multiple ten bit slave addresses. Only the bits set to "1" in this register are used in the ten bit slave address comparison. The received slave address is written back to the ITBA register and thus may be determined by reading the ITBA register if the AAS bit in the IBSR register is "1". Note: If the address mask is changed after the interface had been enabled, the slave address should also be set again since it could have been overwritten by a previously received slave address.

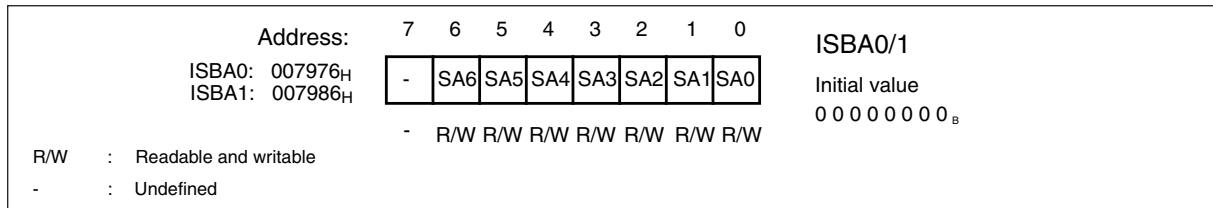
21.2.5 Seven Bit Slave Address Register (ISBA)

This register designates the seven bit slave address.

■ Seven bit slave address register

Write access to this register is only possible if the interface is disabled (EN="0" in ICCR).

Figure 21.2-9 Configuration of the seven bit slave address register (ISBA0/1)



■ Seven bit slave address register contents

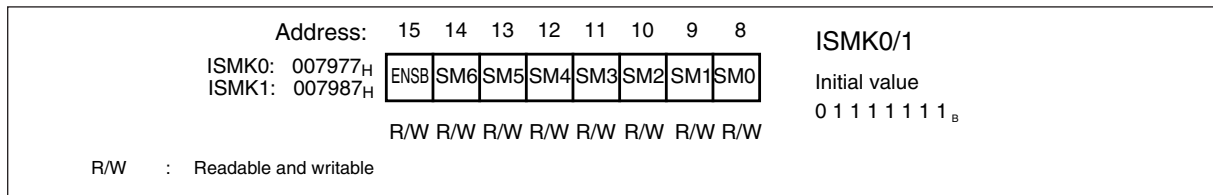
Table 21.2-5 Function of each bit of the seven bit slave address register

Bit name		Function
bit 7	Undefined	This bit always returns "0" during reading.
bit 6 to 0	SA6 to SA0: Seven bit slave address bits	When address data is received in slave mode, it is compared to the ISBA register if the seven bit address is enabled (ENSB="1" in the ISMK register). If a match is detected, an acknowledge signal is sent to the master device and the AAS bit is set. All bits of the slave address may be masked using the ISMK register. The received seven bit slave address is written back to the ISBA register, it is only valid while the AAS bit in the IBSR register is "1". The interface does not compare the contents of this register to the incoming data if a ten bit header or a general call is received.

■ Seven bit slave address mask register (ISMK)

This register contains the seven bit slave address mask and the seven bit mode enable bit. Write access to this register is only possible if the interface is disabled (EN="0" in ICCR).

Figure 21.2-10 Configuration of the seven bit address mask register (ISMK0/1)



■ Seven bit slave address mask register contents

Table 21.2-6 Function of each bit of the seven bit slave address mask register (ISMK0/1)

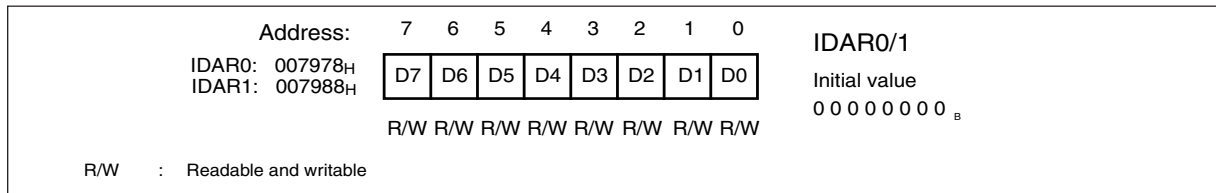
Bit name		Function
bit 15	ENSB: Enable seven bit slave address bit	This bit enables the seven bit slave address (and the acknowledging upon its reception). "0": Seven bit slave address disabled "1": Seven bit slave address enabled
bit 14 to 8	SMK: Seven bit slave address mask bits	This register is used to mask the seven bit slave address of the interface. "0": Bit is not used in slave address comparison. "1": Bit is used in slave address comparison. This can be used to make the interface acknowledge on multiple seven bit slave addresses. Only the bits set to "1" in this register are used in the seven bit slave address comparison. The received slave address is written back to the ISBA register and may thus may be determined by reading the ISBA register if the AAS bit in the IBSR register is "1". Note: If the address mask is changed after the interface had been enabled, the slave address should also be set again since it could have been overwritten by a previously received slave address.

21.2.6 Data Register (IDAR)

Data Register for the 400 kHz I²C Interface.

■ Data register (IDAR)

Figure 21.2-11 Configuration of the data register (IDAR0/1)



■ Data register contents

Table 21.2-7 Function of each bit of the data register

Bit name		Function
bits 7 to 0	D7 to D0: Data bits	The data register is used in serial data transfer, and transfers data MSB-first. This register is double buffered on the write side, so that when the bus is in use (BB="1"), write data can be loaded to the register for serial transfer. The data byte is loaded into the internal transfer register if the INT bit in the IBCR register is being cleared or the bus is idle (BB="0" in IBSR). In a read access, the internal register is read directly, therefore received data values in this register are only valid if INT="1" in the IBCR register.

21.2.7 Clock Control Register (ICCR)

The clock control register (ICCR) has the following functions:

- Enable test mode
- Enable IO pad noise filters
- Enable I²C interface operation
- Setting the serial clock frequency

■ Clock control register (ICCR)

Figure 21.2-12 Configuration of the clock control register (ICCR0/1)

Address:	15	14	13	12	11	10	9	8	ICCR0/1
ICCR0: 00797B _H	-	NSF	EN	CS4	CS3	CS2	CS1	CS0	Initial value
ICCR1: 00798B _H	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	0 0 0 1 1 1 1 1 _B
R/W	:	Readable and writable							
-	:	Undefined							

■ Clock control register (ICCR) contents

Table 21.2-8 Function of each bit of the clock control register (ICCR0/1)

Bit name		Function																																																	
bit 15	Undefined	This bit always returns "0" during reading.																																																	
bit 14	NSF: IO pad noise filter enable bit	This bit enables the noise filters built into the SDA and SCL IO pads. The noise filter will suppress single spikes with a pulse width of 0 ns (minimum) and between 1 and 1.5 cycles of the machine clock. The maximum depends on the phase relationship between I ² C signals (SDA, ACL) and machine clock. It should be set to “1” if the interface is transmitting or receiving at data rates above 100 KBit.																																																	
bit 13	EN: Enable bit	This bit enables the I²C interface operation. It can only be set by the user but may be cleared by the user and the hardware. "0": Interface disabled "1": Interface enabled When this bit is set to “0” all bits in the IBSR register and IBCR register (except the BER and BEIE bits) are cleared and the module is disabled and the I²C lines are left open. It is cleared by the hardware if a bus error occurs (BER="1" in IBCR). Note: The interface immediately stops transmitting or receiving if it is being disabled. This might leave the I²C bus in an undesired state.																																																	
bits 12 to 8	CS4 to CS0: Clock prescaler bits	These bits select the serial bit rate. They can only be changed if the interface is disabled (EN="0") or the EN bit is being cleared in the same write access. <table><tr><td>n</td><td>CS4</td><td>CS3</td><td>CS2</td><td>CS1</td><td>CS0</td><td></td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>Bitrate: Φ / 28(+1)</td></tr><tr><td>2</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Bitrate: Φ / 40(+1)</td></tr><tr><td>3</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>Bitrate: Φ / 52(+1)</td></tr><tr><td>4</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>Bitrate: Φ / 64(+1)</td></tr><tr><td colspan="6">...</td><td></td></tr><tr><td>31</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>Bitrate: Φ / 400(+1)</td></tr></table> (+1) means: Add 1 to divisor, if noise filter is enabled	n	CS4	CS3	CS2	CS1	CS0		1	0	0	0	0	1	Bitrate: Φ / 28(+1)	2	0	0	0	1	0	Bitrate: Φ / 40(+1)	3	0	0	0	1	1	Bitrate: Φ / 52(+1)	4	0	0	1	0	0	Bitrate: Φ / 64(+1)	...							31	1	1	1	1	1	Bitrate: Φ / 400(+1)
n	CS4	CS3	CS2	CS1	CS0																																														
1	0	0	0	0	1	Bitrate: Φ / 28(+1)																																													
2	0	0	0	1	0	Bitrate: Φ / 40(+1)																																													
3	0	0	0	1	1	Bitrate: Φ / 52(+1)																																													
4	0	0	1	0	0	Bitrate: Φ / 64(+1)																																													
...																																																			
31	1	1	1	1	1	Bitrate: Φ / 400(+1)																																													

■ Clock prescaler settings

The calculation formula for CS0 to CS4 is determined as follows:

$$\text{Bitrate} = \frac{\phi}{n \cdot 12 + 16} \quad n > 0 \quad \phi: \text{machine clock, Noise filter disabled}$$

$$\text{Bitrate} = \frac{\phi}{n \cdot 12 + 17} \quad n > 0 \quad \phi: \text{machine clock, Noise filter enabled}$$

Table 21.2-9 Prescaler settings:

n	CS4	CS3	CS2	CS1	CS0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
...					
31	1	1	1	1	1

Note:

Do not use n=0 prescaler setting, it violates SDA/SCL timings.

■ Common machine clock frequencies

The most common machine clock frequencies with their prescaler settings and the resulting sending bit rate:

Table 21.2-10 Common machine clock frequencies

Machine Clock [MHz]	100 KBit (Noise filter disabled)		400 KBit (Noise filter enabled)	
	n	Bit rate [K bit]	n	Bit rate [K bit]
24	19	98	4	369
20	16	96	3	377
16	12	100	2	390
40/3 = 13.3	10	98	2	325
12	9	96	2	292
64/6 = 10.6	8	94	1	367
10	7	100	1	344
8	6	90	1	275

21.3 I²C Interface Operation

The I²C bus executes communication using two bi-directional bus lines, the serial data line (SDA) and serial clock line (SCL). The I²C interface has two open-drain I/O pins (SDA/SCL) corresponding to these lines, enabling wired logic applications.

■ Start conditions

When the bus is free (BB="0" in IBSR, MSS="0" in IBCR), writing "1" to the MSS bit places the I²C interface in master mode and generates a start condition.

If a "1" is written to it while the bus is idle (MSS="0" and BB="0"), a start condition is generated and the contents of the IDAR register (which should be address data) is sent.

Repeated start conditions can be generated by writing "1" to the SCC bit when in bus master mode and interrupt status (MSS="1" and INT="1" in IBCR).

If a "1" is written to the MSS bit while the bus is in use (BB="1" and TRX="0" in IBSR; MSS="0" and INT="0" in IBCR), the interface waits until the bus is free and then starts sending.

If the interface is addressed as slave with write access (data reception) in the meantime, it will start sending after the transfer ended and the bus is free again. If the interface is sending data as slave in the meantime, it will not start sending data if the bus is free again. It is important to check whether the interface was addressed as slave (MSS="0" in IBCR and AAS="1" in IBSR), sent the data byte successfully (MSS="1" in IBCR) or failed to send the data byte (AL="1" in IBSR) at the next interrupt.

Writing "1" to the MSS bit or SCC bit in any other situation has no significance.

■ Stop conditions

Writing "0" to the MSS bit in master mode (MSS="1" and INT="1" in IBCR) generates a stop condition and places the device in slave mode. Writing "0" to the MSS bit in any other situation has no significance.

After clearing the MSS bit, the interface tries to generate a stop condition which might fail if a certain external condition causes signal transition caused from "1" to "0" at the SCL line before the generation of this stop condition. In this case, the AL bit is set to "1" and interrupt is signaled at the end of the next byte.

■ Slave address detection

In slave mode, after a start condition is generated the BB is set to "1" and data sent from the master device is received into the IDAR register.

After the reception of eight bits, the contents of the IDAR register is compared to the ISBA register using the bit mask stored in ISMK if the ENSB bit in the ISMK register is "1". If a match results, the AAS bit is set to "1" and an acknowledge signal is sent to the master. Then bit 0 of the received data (bit 0 of the IDAR register) is inverted and stored in the TRX bit.

If the ENTB bit in the ITMK register is "1" and a ten bit address header (11110, TA1, TA0, write access) is detected, the interface sends an acknowledge signal to the master and stores the inverted last data bit in the TRX register. No interrupt is generated. Then, the next transferred byte is compared (using the bit mask stored in ITMK) to the lower byte of the ITBA register. If a match is found, an acknowledge signal is sent to the master, the AAS bit is set and an interrupt is generated.

If the interface was addressed as slave and detects a repeated start condition, the AAS bit is set after reception of the ten bit address header (11110, TA1, TA0, read access) and an interrupt is generated.

Since there are separate registers for the ten and seven bit address and their bit masks, it is possible to make the interface acknowledge on both addresses by setting the ENSB (in ISMK) and ENTB (in ITMK) bits. The received slave address length (seven or ten bit) may be determined by reading the RAL bit in the ITMK register (this bit is valid if the AAS bit is set only).

It is also possible to give the interface no slave address by setting both bits to "0" if it is only used as a master.

All slave address bits may be masked with their corresponding mask register (ITMK or ISMK).

■ Slave address masking

Only the bits set to "1" in the mask registers (ITMK / ISMK) are used for address comparison, all other bits are ignored. The received slave address can be read from the ITBA (if ten bit address received, RAL="1") or ISBA (if seven bit address received, RAL="0") register if the AAS bit in the IBSR register is "1".

If the bit masks are cleared, the interface can be used as a bus monitor since it will always be addressed as slave. Note that this is not a real bus monitor because it acknowledges upon any slave address reception, even if there is no other slave listening.

■ Addressing slaves

In master mode, after a start condition is generated the BB and TRX bits are set to "1" and the contents of the IDAR register is sent in MSB first order. After address data is sent and an acknowledge signal was received from the slave device, bit 0 of the sent data (bit 0 of the IDAR register after sending) is inverted and stored in the TRX bit. Acknowledgement by the slave may be checked using the LRB bit in the IBSR register. This procedure also applies to a repeated start condition.

In order to address a ten bit slave for write access, two bytes have to be sent. The first one is the ten bit address header which consists of the bit sequence "1 1 1 1 0 A9 A8 0", it is followed by the second byte containing the lower eight bits of the ten bit slave address (A7 to A0).

A ten bit slave is accessed for reading by sending the above byte sequence and generating a repeated start condition (SCC bit in IBCR) followed by a ten bit address header with read access (1 1 1 1 0 A9 A8 1).

Summary of the address data bytes:

7 bit slave, write access: Start condition - A6 A5 A4 A3 A2 A1 A0 0.

7 bit slave, read access: Start condition - A6 A5 A4 A3 A2 A1 A0 1.

10 bit slave, write access: Start condition - 1 1 1 1 0 A9 A8 0 - A7 A6 A5 A4 A3 A2 A1 A0.

10 bit slave, read access: Start condition - 1 1 1 1 0 A9 A8 1 - A7 A6 A5 A4 A3 A2 A1 A0 - repeated start - 1 1 1 1 0 A9 A8 1.

■ Arbitration

During sending in master mode, if another master device is sending data at the same time, arbitration is performed. If a device is sending the data value "1" and the data on the SDA line has an "L" level value, the device is considered to have lost arbitration, and the AL bit is set to "1." Also, the AL bit is set to "1" if a start condition is detected at the first bit of a data byte but the interface did not want to generate one or the generation of a start or stop condition failed by some reason.

Arbitration loss detection clears both the MSS and TRX bit and immediately places the device in slave mode so it is able to acknowledge if its own slave address is being sent.

■ Acknowledgement

Acknowledge bits are sent from the receiver to the transmitter. The ACK bit in the IBCR register can be used to select whether to send an acknowledgment when data bytes are received.

When data is send in slave mode (read access from another master), if no acknowledgement is received from the master, the TRX bit is set to "0" and the device goes to receiving mode. This enables the master to generate a stop condition as soon as the slave has released the SCL line.

In master mode, acknowledgement by the slave can be checked by reading the LRB bit in the IBSR register.

21.4 Programming Flow Charts

Each programming flow charts for the 400 kHz I²C interface is shown below.

■ Programming flow charts

Figure 21.4-1 Example of slave addressing and sending data

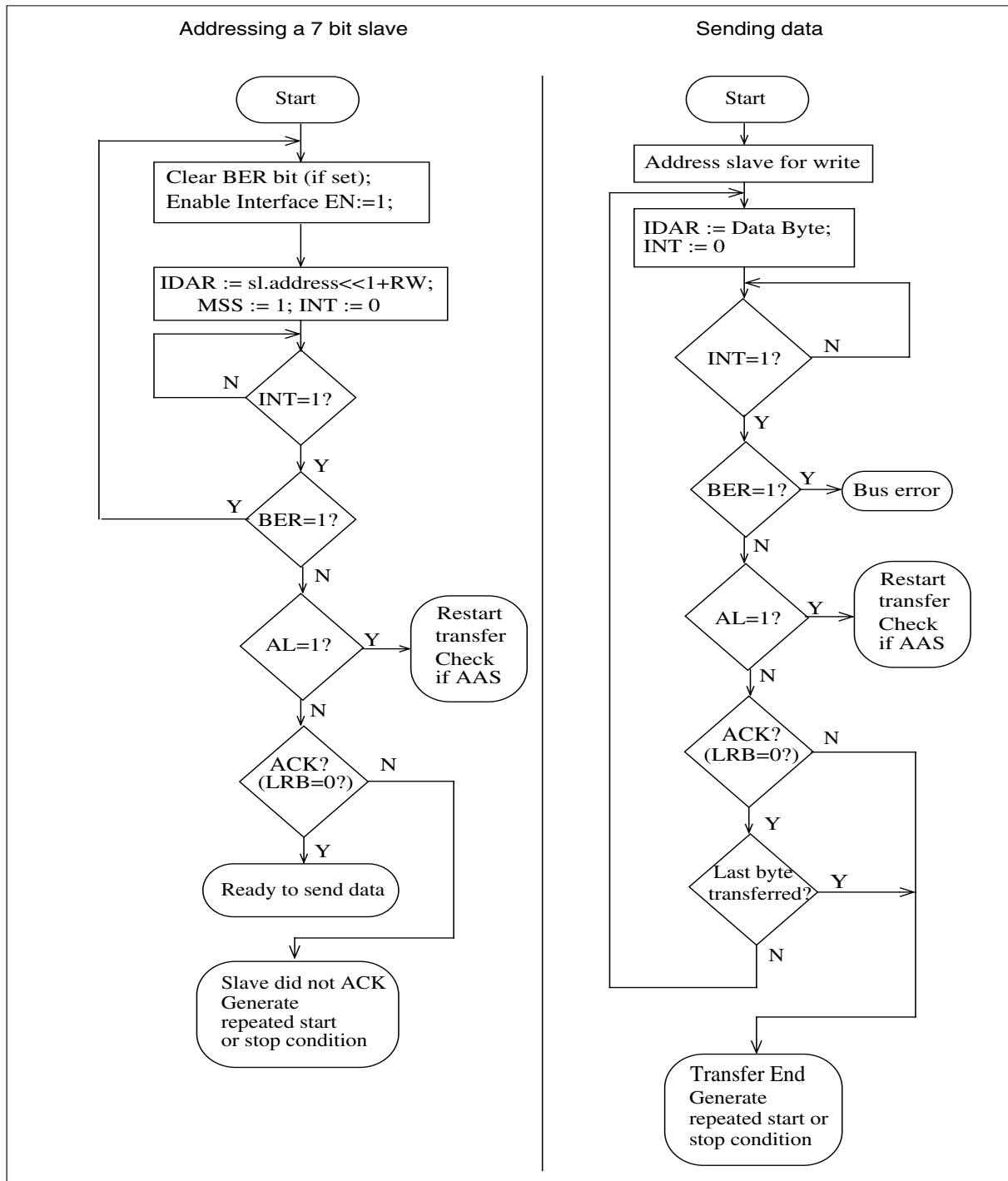
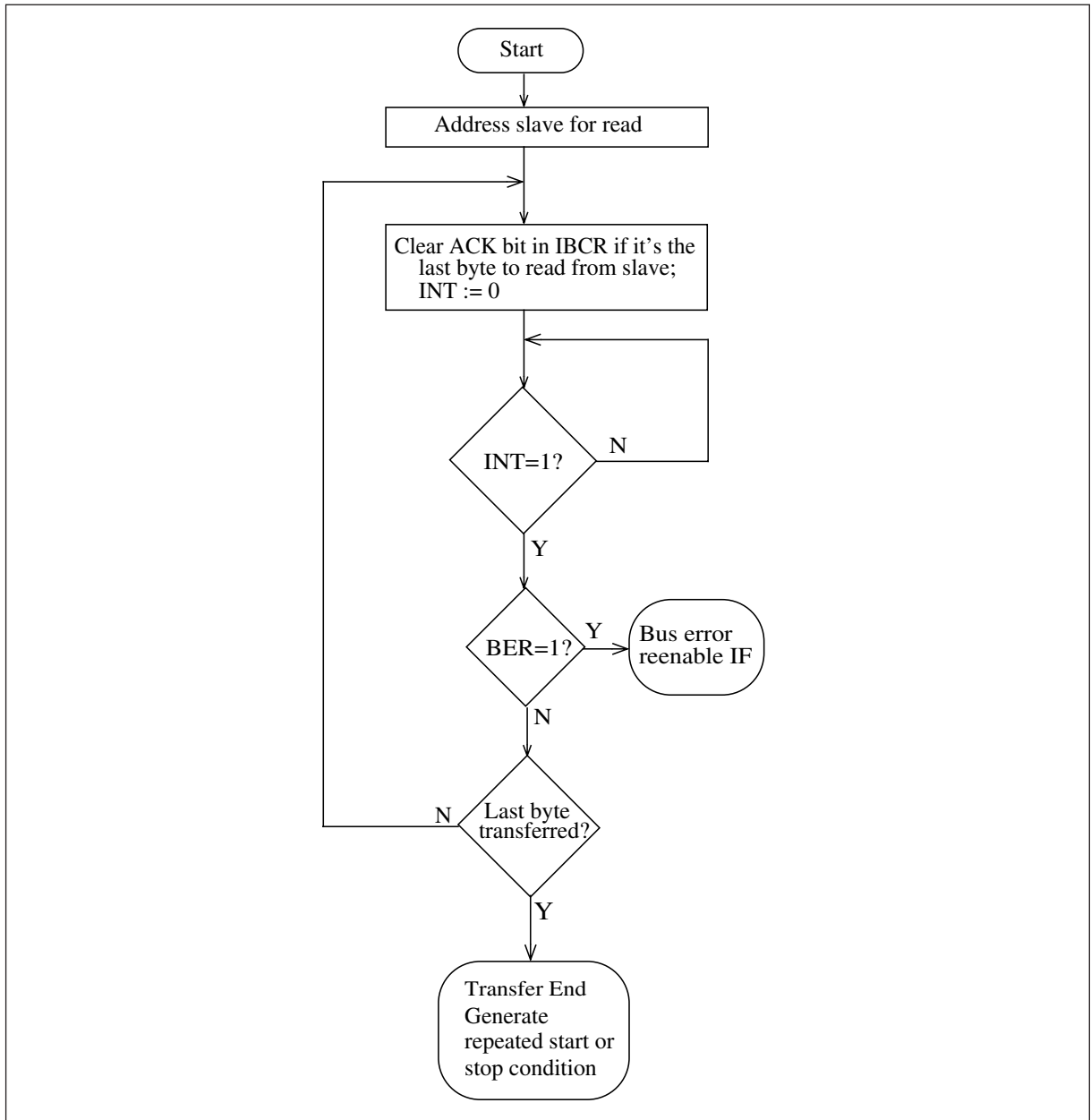


Figure 21.4-2 Example of receiving data

CHAPTER 22

CAN CONTROLLER

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

Note: CAN controller 2 is not specified in the MB90340 series. For this reason Fujitsu recommends not to use these features.

- 22.1 “Features of CAN Controller”
- 22.2 “Block Diagram of CAN Controller”
- 22.3 “List of Overall Control Registers”
- 22.4 “List of Message Buffers (ID Registers)”
- 22.5 “List of Message Buffers (DLC Registers and Data Registers)”
- 22.6 “Classifying the CAN Controller Registers”
- 22.7 “Transmission of CAN Controller”
- 22.8 “Reception of CAN Controller”
- 22.9 “Reception Flowchart of CAN Controller”
- 22.10 “How to Use the CAN Controller”
- 22.11 “Procedure for Transmission by Message Buffer (x)”
- 22.12 “Procedure for Reception by Message Buffer (x)”
- 22.13 “Setting Configuration of Multi-level Message Buffer”
- 22.14 “Setting the redirection of CAN1 and 3 RX/TX pin”
- 22.15 “Setting the CAN Direct Mode Register”

22.1 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:

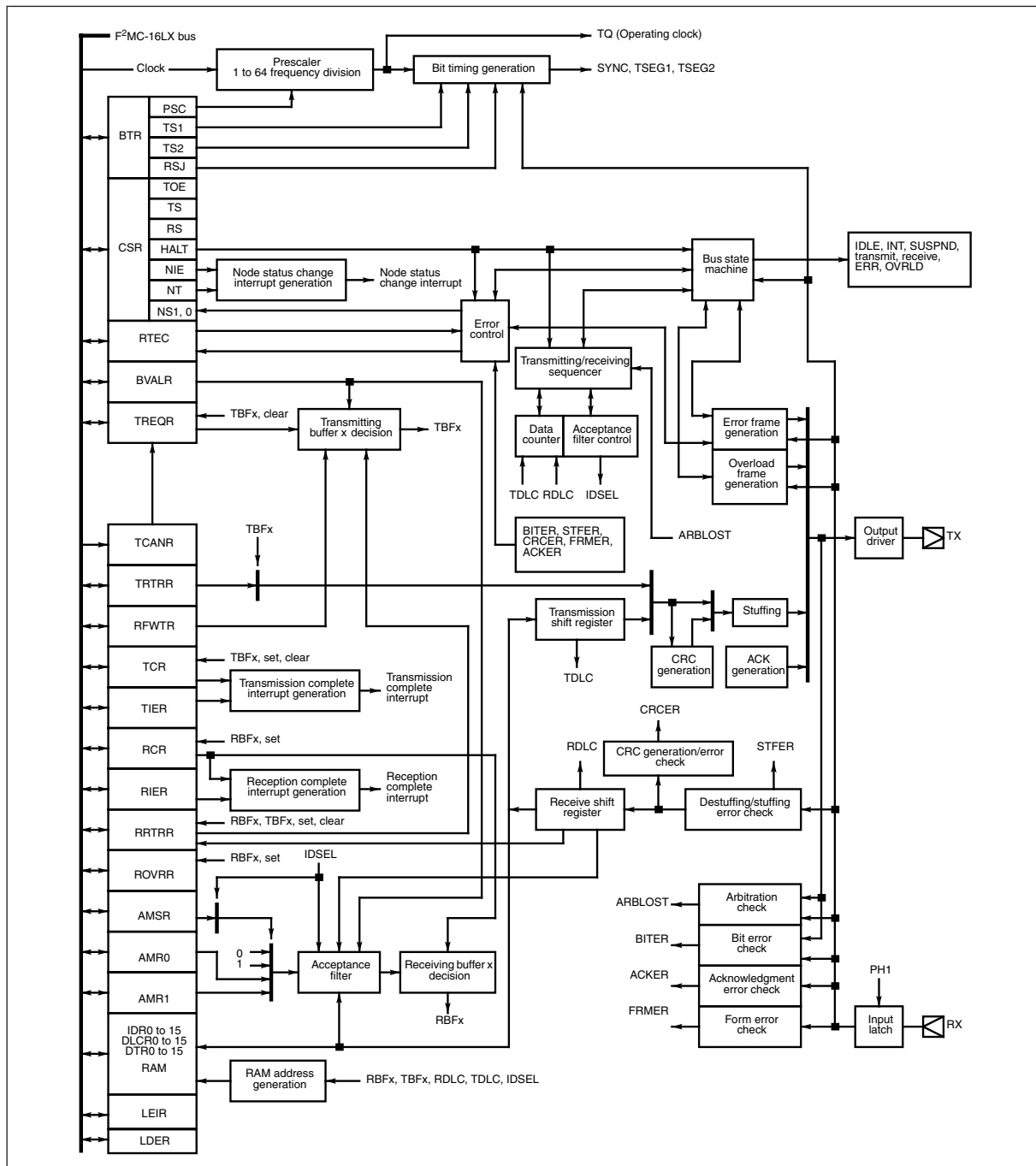
- Conforms to CAN Specification Version 2.0 Part A and B
 - Supports transmission/reception in standard frame and extended frame formats
- Supports transmitting of data frames by receiving remote frames
- 16 transmitting/receiving message buffers
 - 29-bit ID and 8-byte data
 - Multi-level message buffer 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)

22.2 Block Diagram of CAN Controller

Figure 22.2-1 “Block diagram of CAN controller” shows a block diagram of the CAN controller.

■ Block diagram of CAN controller

Figure 22.2-1 Block diagram of CAN controller



22.3 List of Overall Control Registers

Table 22.3-1 “List of overall control registers” lists overall control registers.

■ List of overall control registers

Table 22.3-1 List of overall registers (1/2)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
000070 _H	000080 _H	0000E0 _H	Message buffer valid register	BVALR	R/W	00000000 00000000
000071 _H	000081 _H	0000E1 _H				
000072 _H	000082 _H	0000E2 _H	Transmit request register	TREQR	R/W	00000000 00000000
000073 _H	000083 _H	0000E3 _H				
000074 _H	000084 _H	0000E4 _H	Transmit cancel register	TCANR	W	00000000 00000000
000075 _H	000085 _H	0000E5 _H				
000076 _H	000086 _H	0000E6 _H	Transmit complete register	TCR	R/W	00000000 00000000
000077 _H	000087 _H	0000E7 _H				
000078 _H	000088 _H	0000E8 _H	Receive complete register	RCR	R/W	00000000 00000000
000079 _H	000089 _H	0000E9 _H				
00007A _H	00008A _H	0000EA _H	Remote request receiving register	RRTRR	R/W	00000000 00000000
00007B _H	00008B _H	0000EB _H				
00007C _H	00008C _H	0000EC _H	Receive overrun register	ROVRR	R/W	00000000 00000000
00007D _H	00008D _H	0000ED _H				
00007E _H	00008E _H	0000EE _H	Receive interrupt enable register	RIER	R/W	00000000 00000000
00007F _H	00008F _H	0000EF _H				
007B00 _H	007D00 _H	007F00 _H	Control status register	CSR	R/W, R	00XXX000 0XXXX0X1
007B01 _H	007D01 _H	007F01 _H				
007B02 _H	007D02 _H	007F02 _H	Last event indicator register	LEIR	R/W	XXXXXXXX 000X0000
007B03 _H	007D03 _H	007F03 _H				
007B04 _H	007D04 _H	007F04 _H	Receive/transmit error counter	RTEC	R	00000000 00000000
007B05 _H	007D05 _H	007F05 _H				
007B06 _H	007D06 _H	007F06 _H	Bit timing register	BTR	R/W	X1111111 11111111
007B07 _H	007D07 _H	007F07 _H				
007B08 _H	007D08 _H	007F08 _H	IDE register	IDER	R/W	XXXXXXXX XXXXXXXX
007B09 _H	007D09 _H	007F09 _H				
007B0A _H	007D0A _H	007F0A _H	Transmit RTR register	TRTRR	R/W	00000000 00000000
007B0B _H	007D0B _H	007F0B _H				

Table 22.3-1 List of overall registers (2/2)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007B0C _H	007D0C _H	007F0C _H	Remote frame receive waiting register	R/WTR	R/W	XXXXXXXX XXXXXXXX
007B0D _H	007D0D _H	007F0D _H				
007B0E _H	007D0E _H	007F0E _H	Transmit interrupt enable register	TIER	R/W	00000000 00000000
007B0F _H	007D0F _H	007F0F _H				
007B10 _H	007D10 _H	007F10 _H	Acceptance mask select register	AMSR	R/W	XXXXXXXX XXXXXXXX
007B11 _H	007D11 _H	007F11 _H				
007B12 _H	007D12 _H	007F12 _H				XXXXXXXX XXXXXXXX
007B13 _H	007D13 _H	007F13 _H				
007B14 _H	007D14 _H	007F14 _H	Acceptance mask register 0	AMR0	R/W	XXXXXXXX XXXXXXXX
007B15 _H	007D15 _H	007F15 _H				
007B16 _H	007D16 _H	007F16 _H				XXXXXXXX XXXXXXXX
007B17 _H	007D17 _H	007F17 _H				
007B18 _H	007D18 _H	007F18 _H	Acceptance mask register 1	AMR1	R/W	XXXXXXXX XXXXXXXX
007B19 _H	007D19 _H	007F19 _H				
007B1A _H	007D1A _H	007F1A _H				XXXXXXXX XXXXXXXX
007B1B _H	007D1B _H	007F1B _H				

22.4 List of Message Buffers (ID Registers)

Table 22.4-1 “List of message buffers (ID registers)” lists message buffers (ID registers).

■ List of message buffers (ID registers)

Table 22.4-1 List of message buffers (ID registers) (1/3)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007A00 _H to 007A1F _H	007C00 _H to 007C1F _H	007E00 _H to 007E1F _H	General- purpose RAM	--	R/W	XXXXXXXX to XXXXXXXX
007A20 _H	007C20 _H	007E20 _H	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX
007A21 _H	007C21 _H	007E21 _H				
007A22 _H	007C22 _H	007E22 _H				XXXXXXXX XXXXXXXX
007A23 _H	007C23 _H	007E23 _H				
007A24 _H	007C24 _H	007E24 _H	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX
007A25 _H	007C25 _H	007E25 _H				
007A26 _H	007C26 _H	007E26 _H				XXXXXXXX XXXXXXXX
007A27 _H	007C27 _H	007E27 _H				
007A28 _H	007C28 _H	007E28 _H	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX
007A29 _H	007C29 _H	007E29 _H				
007A2A _H	007C2A _H	007E2A _H				XXXXXXXX XXXXXXXX
007A2B _H	007C2B _H	007E2B _H				
007A2C _H	007C2C _H	007E2C _H	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX
007A2D _H	007C2D _H	007E2D _H				
007A2E _H	007C2E _H	007E2E _H				XXXXXXXX XXXXXXXX
007A2F _H	007C2F _H	007E2F _H				
007A30 _H	007C30 _H	007E30 _H	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX
007A31 _H	007C31 _H	007E31 _H				
007A32 _H	007C32 _H	007E32 _H				XXXXXXXX XXXXXXXX
007A33 _H	007C33 _H	007E33 _H				
007A34 _H	007C34 _H	007E34 _H	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX
007A35 _H	007C35 _H	007E35 _H				
007A36 _H	007C36 _H	007E36 _H				XXXXXXXX XXXXXXXX
007A37 _H	007C37 _H	007E37 _H				

Table 22.4-1 List of message buffers (ID registers) (2/3)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007A38 _H	007C38 _H	007E38 _H	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX
007A39 _H	007C39 _H	007E39 _H				
007A3A _H	007C3A _H	007E3A _H				XXXXXXXX XXXXXXXX
007A3B _H	007C3B _H	007E3B _H				
007A3C _H	007C3C _H	007E3C _H	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX
007A3D _H	007C3D _H	007E3D _H				
007A3E _H	007C3E _H	007E3E _H				XXXXXXXX XXXXXXXX
007A3F _H	007C3F _H	007E3F _H				
007A40 _H	007C40 _H	007E40 _H	ID register 8	IDR8	R/W	XXXXXXXX XXXXXXXX
007A41 _H	007C41 _H	007E41 _H				
007A42 _H	007C42 _H	007E42 _H				XXXXXXXX XXXXXXXX
007A43 _H	007C43 _H	007E43 _H				
007A44 _H	007C44 _H	007E44 _H	ID register 9	IDR9	R/W	XXXXXXXX XXXXXXXX
007A45 _H	007C45 _H	007E45 _H				
007A46 _H	007C46 _H	007E46 _H				XXXXXXXX XXXXXXXX
007A47 _H	007C47 _H	007E47 _H				
007A48 _H	007C48 _H	007E48 _H	ID register 10	IDR10	R/W	XXXXXXXX XXXXXXXX
007A49 _H	007C49 _H	007E49 _H				
007A4A _H	007C4A _H	007E4A _H				XXXXXXXX XXXXXXXX
007A4B _H	007C4B _H	007E4B _H				
007A4C _H	007C4C _H	007E4C _H	ID register 11	IDR11	R/W	XXXXXXXX XXXXXXXX
007A4D _H	007C4D _H	007E4D _H				
007A4E _H	007C4E _H	007E4E _H				XXXXXXXX XXXXXXXX
007A4F _H	007C4F _H	007E4F _H				
007A50 _H	007C50 _H	007E50 _H	ID register 12	IDR12	R/W	XXXXXXXX XXXXXXXX
007A51 _H	007C51 _H	007E51 _H				
007A52 _H	007C52 _H	007E52 _H				XXXXXXXX XXXXXXXX
007A53 _H	007C53 _H	007E53 _H				
007A54 _H	007C54 _H	007E54 _H	ID register 13	IDR13	R/W	XXXXXXXX XXXXXXXX
007A55 _H	007C55 _H	007E55 _H				
007A56 _H	007C56 _H	007E56 _H				XXXXXXXX XXXXXXXX
007A57 _H	007C57 _H	007E57 _H				

Table 22.4-1 List of message buffers (ID registers) (3/3)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007A58 _H	007C58 _H	007E58 _H	ID register 14	IDR14	R/W	XXXXXXXX XXXXXXXX
007A59 _H	007C59 _H	007E59 _H				
007A5A _H	007C5A _H	007E5A _H				XXXXXXXX XXXXXXXX
007A5B _H	007C5B _H	007E5B _H				
007A5C _H	007C5C _H	007E5C _H	ID register 15	IDR15	R/W	XXXXXXXX XXXXXXXX
007A5D _H	007C5D _H	007E5D _H				
007A5E _H	007C5E _H	007E5E _H				XXXXXXXX XXXXXXXX
007A5F _H	007C5F _H	007E5F _H				

22.5 List of Message Buffers (DLC Registers and Data Registers)

Table 22.5-1 “List of message buffers (DLC registers and data registers)” lists message buffers (DLC registers) and message buffers (data registers).

■ List of message buffers (DLC registers and data registers)

Table 22.5-1 List of message buffers (DLC registers and data register) (1/3)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007A60 _H	007C60 _H	007E60 _H	DLC register 0	DLCR0	R/W	XXXXXXXX
007A61 _H	007C61 _H	007E61 _H				
007A62 _H	007C62 _H	007E62 _H	DLC register 1	DLCR1	R/W	XXXXXXXX
007A63 _H	007C63 _H	007E63 _H				
007A64 _H	007C64 _H	007E64 _H	DLC register 2	DLCR2	R/W	XXXXXXXX
007A65 _H	007C65 _H	007E65 _H				
007A66 _H	007C66 _H	007E66 _H	DLC register 3	DLCR3	R/W	XXXXXXXX
007A67 _H	007C67 _H	007E67 _H				
007A68 _H	007C68 _H	007E68 _H	DLC register 4	DLCR4	R/W	XXXXXXXX
007A69 _H	007C69 _H	007E69 _H				
007A6A _H	007C6A _H	007E6A _H	DLC register 5	DLCR5	R/W	XXXXXXXX
007A6B _H	007C6B _H	007E6B _H				
007A6C _H	007C6C _H	007E6C _H	DLC register 6	DLCR6	R/W	XXXXXXXX
007A6D _H	007C6D _H	007E6D _H				
007A6E _H	007C6E _H	007E6E _H	DLC register 7	DLCR7	R/W	XXXXXXXX
007A6F _H	007C6F _H	007E6F _H				
007A70 _H	007C70 _H	007E70 _H	DLC register 8	DLCR8	R/W	XXXXXXXX
007A71 _H	007C71 _H	007E71 _H				
007A72 _H	007C72 _H	007E72 _H	DLC register 9	DLCR9	R/W	XXXXXXXX
007A73 _H	007C73 _H	007E73 _H				
007A74 _H	007C74 _H	007E74 _H	DLC register 10	DLCR10	R/W	XXXXXXXX
007A75 _H	007C75 _H	007E75 _H				
007A76 _H	007C76 _H	007E76 _H	DLC register 11	DLCR11	R/W	XXXXXXXX
007A77 _H	007C77 _H	007E77 _H				

Table 22.5-1 List of message buffers (DLC registers and data register) (2/3)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007A78 _H	007C78 _H	007E78 _H	DLC register 12	DLCR12	R/W	XXXXXXXX
007A79 _H	007C79 _H	007E79 _H				
007A7A _H	007C7A _H	007E7A _H	DLC register 13	DLCR13	R/W	XXXXXXXX
007A7B _H	007C7B _H	007E7B _H				
007A7C _H	007C7C _H	007E7C _H	DLC register 14	DLCR14	R/W	XXXXXXXX
007A7D _H	007C7D _H	007E7D _H				
007A7E _H	007C7E _H	007E7E _H	DLC register 15	DLCR15	R/W	XXXXXXXX
007A7F _H	007C7F _H	007E7F _H				
007A80 _H to 007A87 _H	007C80 _H to 007C87 _H	007E80 _H to 007E87 _H	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX to XXXXXXXX
007A88 _H to 007A8F _H	007C88 _H to 007C8F _H	007E88 _H to 007E8F _H	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX to XXXXXXXX
007A90 _H to 007A97 _H	007C90 _H to 007C97 _H	007E90 _H to 007E97 _H	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX to XXXXXXXX
007A98 _H to 007A9F _H	007C98 _H to 007C9F _H	007E98 _H to 007E9F _H	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX to XXXXXXXX
007AA0 _H to 007AA7 _H	007CA0 _H to 007CA7 _H	007EA0 _H to 007EA7 _H	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX to XXXXXXXX
007AA8 _H to 007AAF _H	007CA8 _H to 007CAF _H	007EA8 _H to 007EAF _H	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX to XXXXXXXX
007AB0 _H to 007AB7 _H	007CB0 _H to 007CB7 _H	007EB0 _H to 007EB7 _H	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX to XXXXXXXX
007AB8 _H to 007ABF _H	007CB8 _H to 007CBF _H	007EB8 _H to 007EBF _H	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX to XXXXXXXX
007AC0 _H to 007AC7 _H	007CC0 _H to 007CC7 _H	007EC0 _H to 007EC7 _H	Data register 8 (8 bytes)	DTR8	R/W	XXXXXXXX to XXXXXXXX
007AC8 _H to 007ACF _H	007CC8 _H to 007CCF _H	007EC8 _H to 007ECF _H	Data register 9 (8 bytes)	DTR9	R/W	XXXXXXXX to XXXXXXXX
007AD0 _H to 007AD7 _H	007CD0 _H to 007CD7 _H	007ED0 _H to 007ED7 _H	Data register 10 (8 bytes)	DTR10	R/W	XXXXXXXX to XXXXXXXX
007AD8 _H to 007ADF _H	007CD8 _H to 007CDF _H	007ED8 _H to 007EDF _H	Data register 11 (8 bytes)	DTR11	R/W	XXXXXXXX to XXXXXXXX

Table 22.5-1 List of message buffers (DLC registers and data register) (3/3)

Address			Register	Abbreviation	Access	Initial Value
CAN0	CAN1	CAN2				
007AE0 _H to 007AE7 _H	007CE0 _H to 007CE7 _H	007EE0 _H to 007EE7 _H	Data register 12 (8 bytes)	DTR12	R/W	XXXXXXXX to XXXXXXXX
007AE8 _H to 007AEF _H	007CE8 _H to 007CEF _H	007EE8 _H to 007EEF _H	Data register 13 (8 bytes)	DTR13	R/W	XXXXXXXX to XXXXXXXX
007AF0 _H to 007AF7 _H	007CF0 _H to 007CF7 _H	007EF0 _H to 007EF7 _H	Data register 14 (8 bytes)	DTR14	R/W	XXXXXXXX to XXXXXXXX
007AF8 _H to 007AFF _H	007CF8 _H to 007CFF _H	007EF8 _H to 007EFF _H	Data register 15 (8 bytes)	DTR15	R/W	XXXXXXXX to XXXXXXXX

22.6 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:

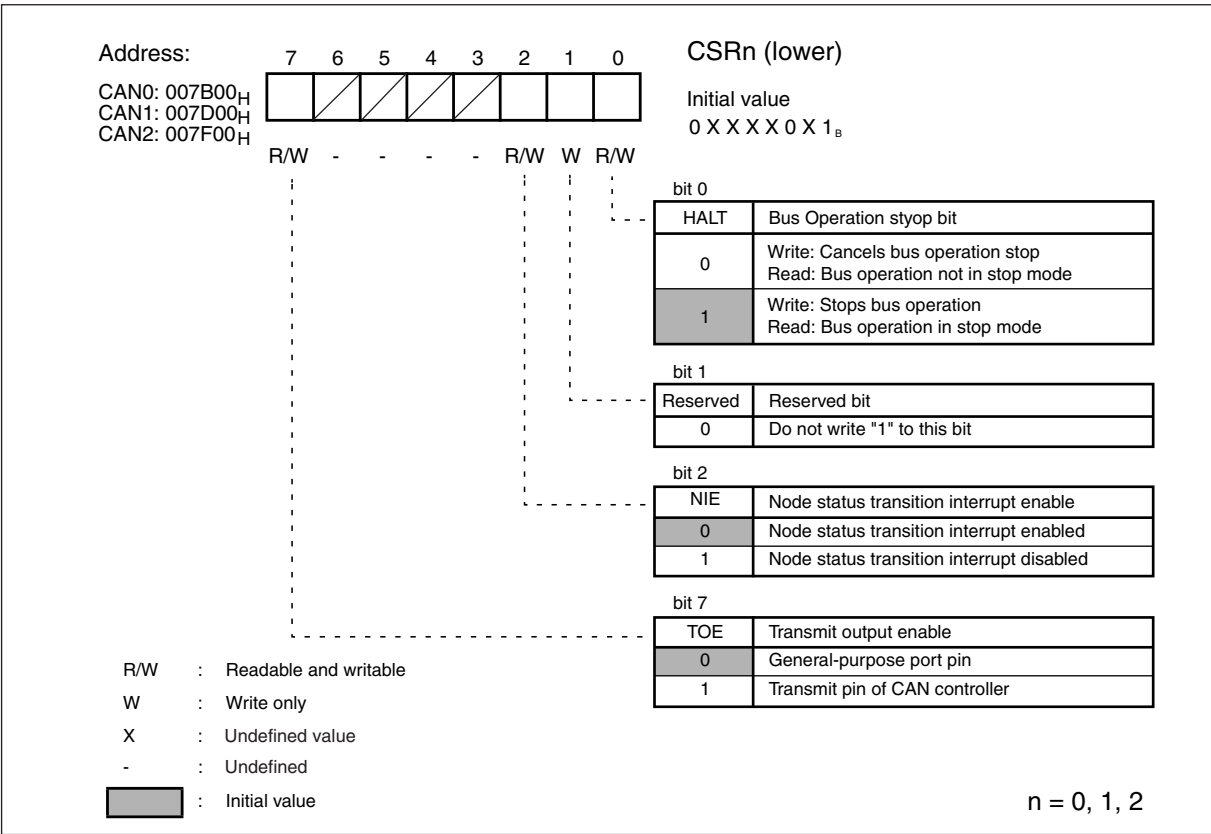
- ID register x ($x = 0$ to 15) (IDRx)
- DLC register x ($x = 0$ to 15) (DLCRx)
- Data register x ($x = 0$ to 15) (DTRx)

22.6.1 Control Status Register (CSR)

The lower 8 bits with the control status register (CSR) is prohibited from executing any bit manipulation instructions (Read-Modify-Write instructions). Only in the case of HALT bits unchanged, use any bit manipulation instructions without problems (initialization of the macro instructions etc.).

■ Control status register (CSR) (lower)

Figure 22.6-1 Configuration of the control status register (lower byte)



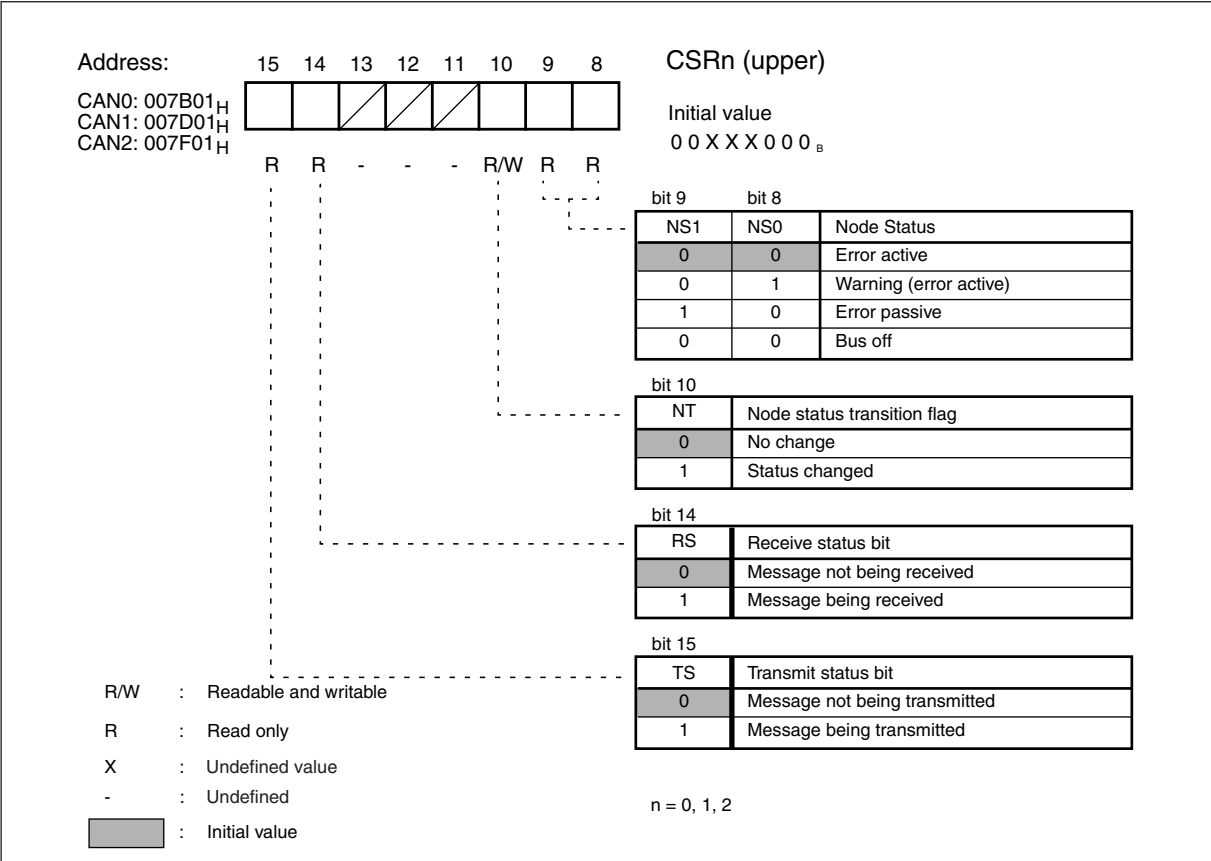
■ Control status register (CSR-lower) contents

Table 22.6-1 Function of each bit of the control status register (lower)

Bit Name		Function
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 6 to 3	Undefined	
bit 2	NIE: Node status transition flag 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
bit1	Reserved bit	This is a reserved bit. Do not write "1" to this bit.
bit 0	HALT: Bus operation stop bit	This bit controls the bus halt. The halt state of the bus can be checked by reading this bit. Writing to this bit 0: Cancels bus halt 1: Halt bus Reading this bit 0: Bus operation not in stop state 1: Bus operation in stop state Note: When write 0 to this bit during the node status is Bus Off, ensure that 1 is written to this bit. Example program: <pre>switch (IO_CANCT0.CSR.bit.NS) { case 0 : /* error active */ break; case 1 : /* warning */ break; case 2 : /* error passive */ break; default : /* bus off */ for (i=0; (i <= 500) && (IO_CANCT0.CSR.bit.HALT == 0); i++); IO_CANCT0.CSR.word = 0x0084; /* HALT = 0 */ break; }</pre> <u>* The variable "i" is used for fail-safe.</u>

■ Control status register (CSR) (upper)

Figure 22.6-2 Configuration of the control status register (upper byte)



■ Control status register (CSR-upper) contents

Table 22.6-2 Function of each bit of the control status register (upper)

Bit Name		Function
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 a error/overload frame is on the bus.
bit 13 to 11	Undefined	
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 a Read Modify Write instruction is performed.
bit 9, 8	NS1, NS0: Node status bit 1 and 0	These bits indicate the current node status. See Table 22.6-3 “Correspondence between NS1 and NS0 and node status” below for details.

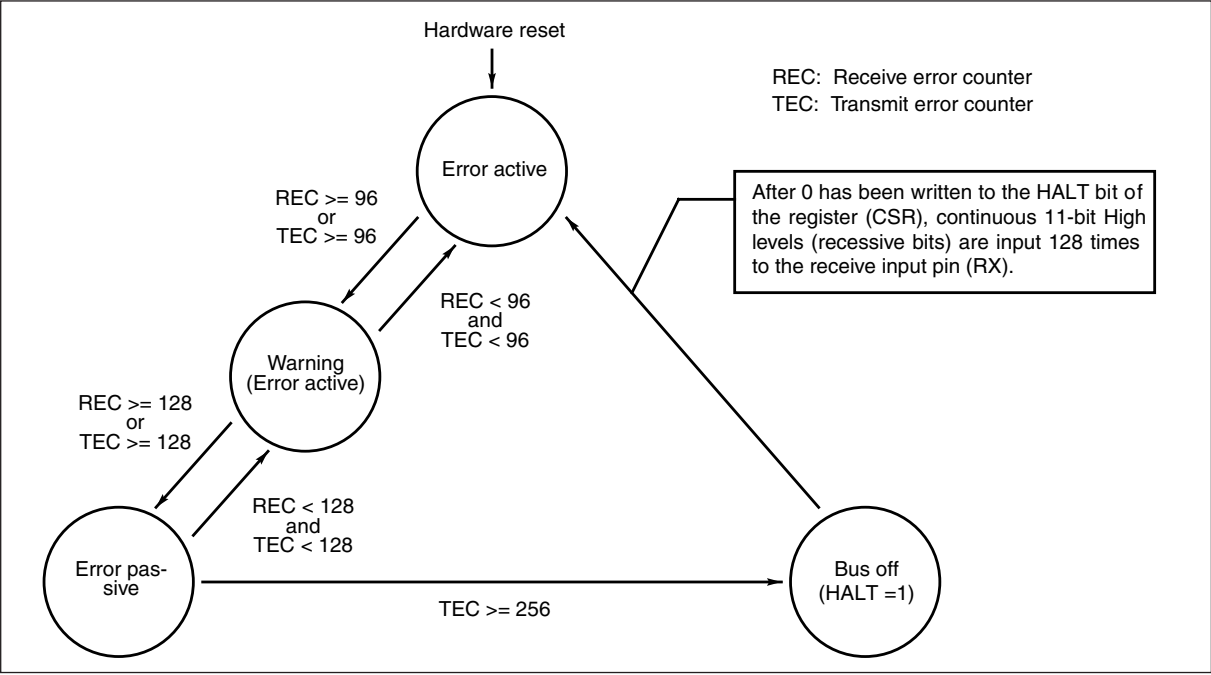
Table 22.6-3 Correspondence between NS1 and NS0 and node status

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 22.6-3 “Node status transition diagram”.

Figure 22.6-3 Node status transition diagram



22.6.2 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 and timebase timer 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.
 - When write 0 to HALT during the node status is Bus Off, ensure that 1 is written to this bit.
-

■ 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:

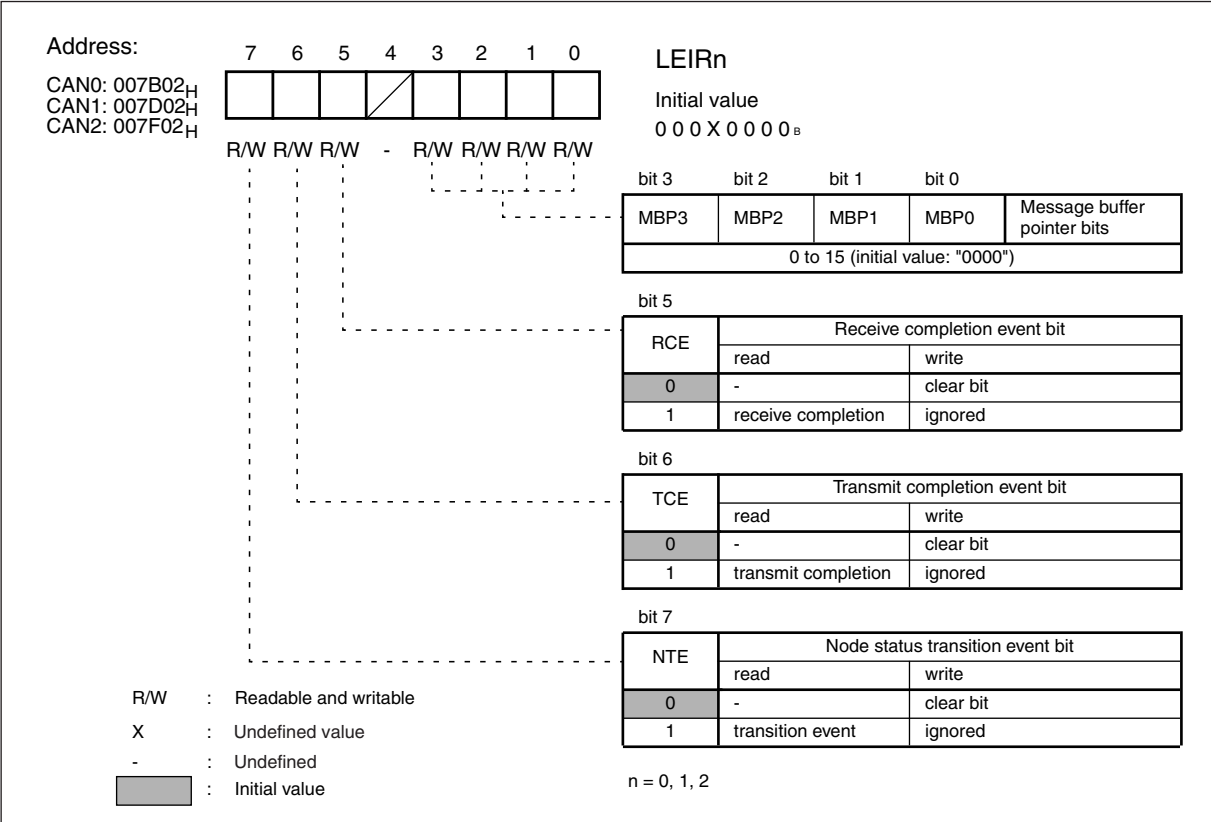
The bit timing register (BTR) should be set during bus operation stop (HALT = 1).

22.6.3 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)

Figure 22.6-4 Configuration of the last event indicator register (LEIR)



■ Last event indicator register (LEIR) contents

Table 22.6-4 Function of each bit of the last event indicator register

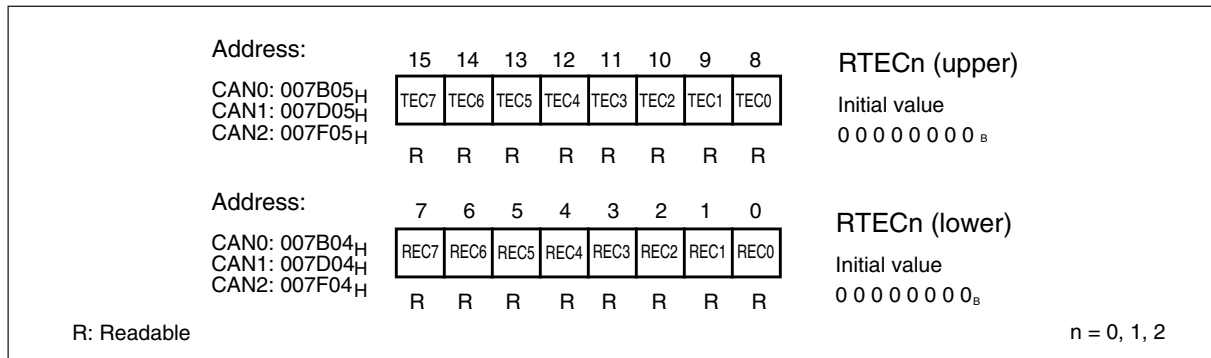
Bit Name		Function
bit 3 to 0	MBP3, MBP2, MBP1, MBP0	When the TCE or RCE bit is set to 1, these bits indicate the corresponding numbers of the message buffers (0 to 15). If the NTE bit is set to 1, these bits have no meaning. Writing 0 sets these bits to 0s. Writing 1 to these bits is ignored. 1s are read when a 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.
bit 4	Undefined	
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 a 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.
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 a 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 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 a Read Modify Write instruction is executed.

22.6.4 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)

Figure 22.6-5 Configuration of the receive and transmit error counters



■ Receive and transmit error counters (RTEC) contents

Table 22.6-5 Function of each bit of the receive and transmit error counters (RTEC)

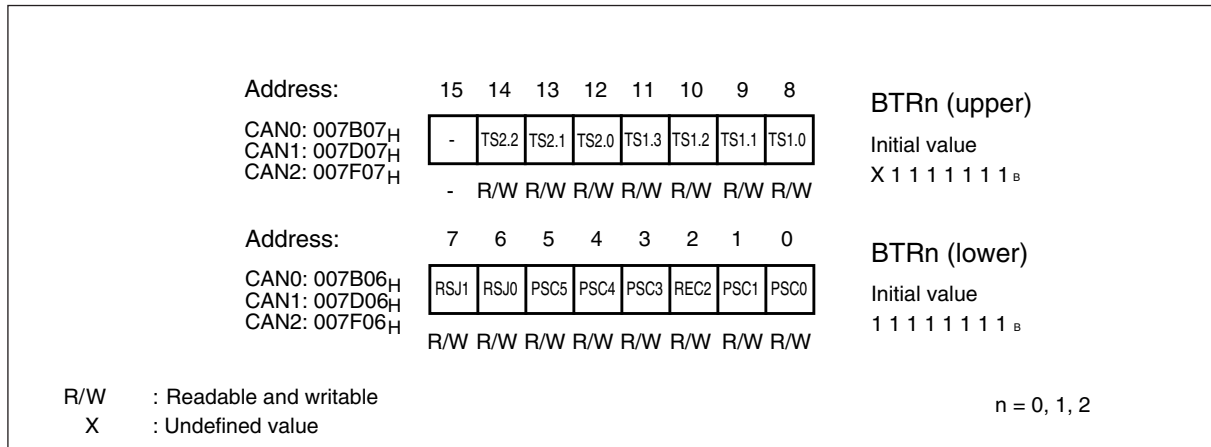
Bit Name		Function
bit 15 to 8	TEC7 to 0: Transmit error counter bits	These are transmit error counters. TEC7 to TEC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Bus Off is indicated for the node status (NS1 and NS0 of control status register CSR = 11).
bit 7 to 0	REC7 to 0: Receive error counter bits	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).

22.6.5 Bit Timing Register (BTR)

Bit timing register (BTR) stores the prescaler and bit timing setting.

■ Bit timing register (BTR)

Figure 22.6-6 Configuration of the bit timing register (BTR)



Note:

This register should be set during bus operation stop (HALT = 1).

■ Bit timing register (BTR) contents

Table 22.6-6 Function of each bit of the bit timing register (BTR)

Bit Name		Function
bit 15	Undefined	
bit 14 to 12	TS2.2 to 2.0: Time segment2 setting bits	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.
bit 11 to 8	TS1.3 to 1.0: Time segment1 setting bits	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.
bit 7, 6	RSJ1,0: Resynchronization jump width setting bits	These bits define the number of the time quanta (TQ's) for the resynchronization jump width.
bit 5 to 0	PSC5 to 0: Prescaler setting bits	These bits define the time quanta (TQ) of the CAN controller. (see below for details.)

■ Prescaler settings

The bit time segments defined in the CAN specification, and the CAN controller are shown in Figure 22.6-7 “Bit time segment in CAN specification” and Figure 22.6-8 “Bit time segment in CAN controller” respectively.

Figure 22.6-7 Bit time segment in CAN specification

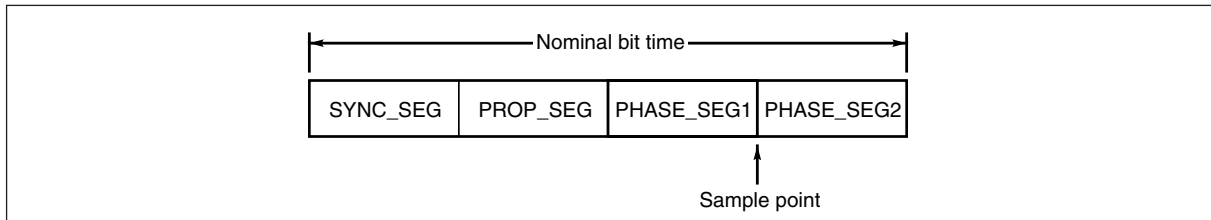
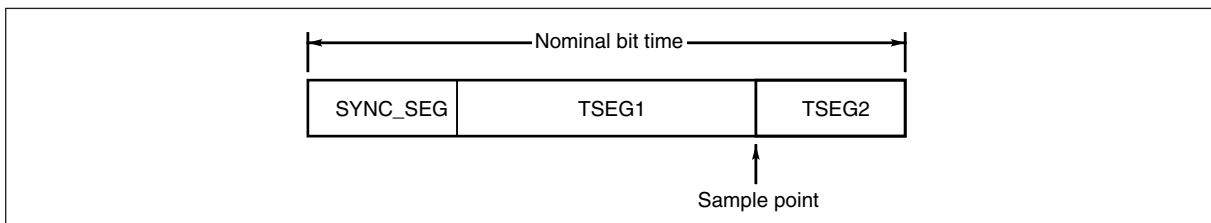


Figure 22.6-8 Bit time segment in CAN controller



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 $\text{PSC} = 0$:
 - $\text{TSEG1} \geq 5\text{TQ}$
 - $\text{TSEG2} \geq \text{RSJW} + 2\text{TQ}$

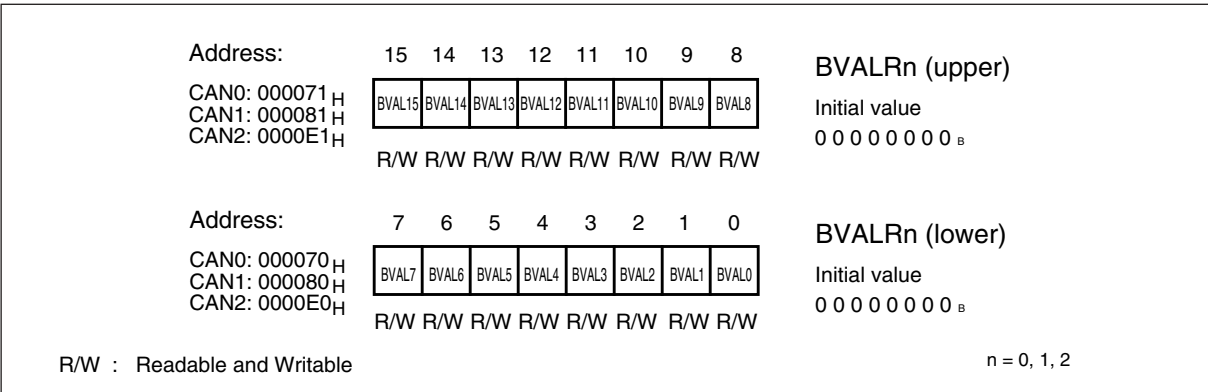
In order to meet the bit timing requirements defined in the CAN specification, additions have to be met, e.g. the propagation delay has to be considered.

22.6.6 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)

Figure 22.6-9 Configuration of the message buffer valid register (BVALR)



- 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.

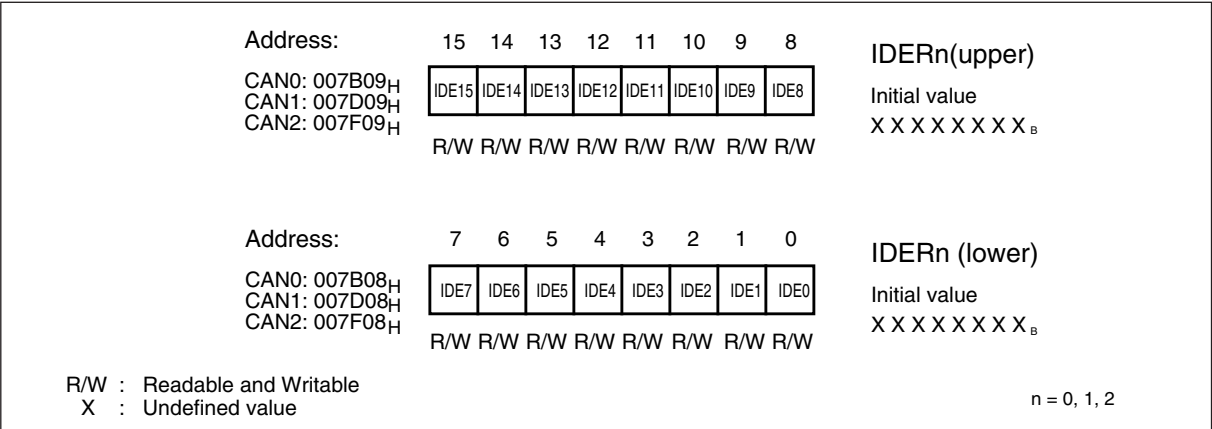
To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in Section 22.16 “Precautions when Using CAN Controller”.

22.6.7 IDE register (IDER)

This register stores the frame format used by the message buffers (x) during transmission/reception.

■ IDE register (IDER)

Figure 22.6-10 Configuration of the IDE register (IDER)



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.

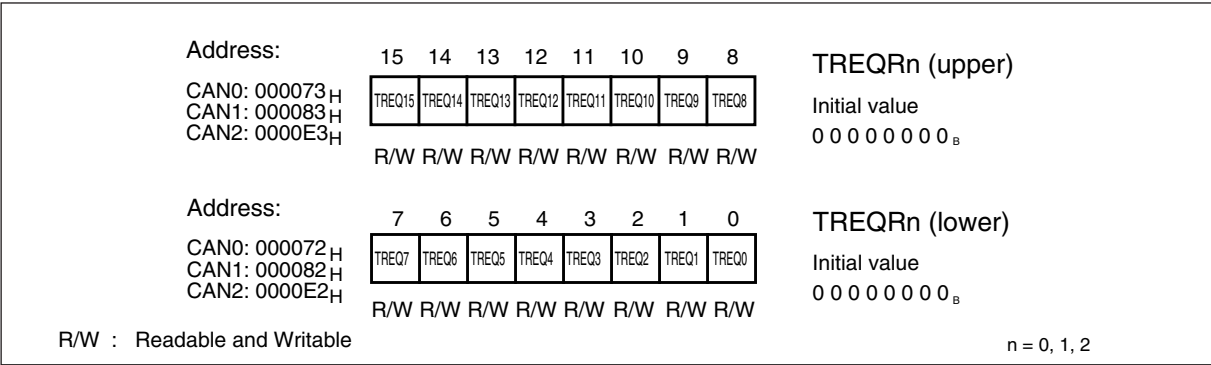
To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in Section 22.16 “Precautions when Using CAN Controller”.

22.6.8 Transmission Request Register (TREQR)

Transmission request register (TREQR) stores transmission requests to the message buffers (x) or displays their state.

■ Transmission request register (TREQR)

Figure 22.6-11 Configuration of the transmission request register (TREQR)



When 1 is written to TREQx, transmission to the message buffer (x) starts. If RFWTx of the remote frame receiving wait register (RFWTR) ^{*1} is 0, transmission starts immediately. However, if RFWTx = 1, transmission starts after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR)*1 becomes 1). Transmission starts ^{*2} immediately even when RFWTx = 1, if RRTRx is already 1 when 1 is written to TREQx.

*1: For RFWTR and TRTRR, see Sections 22.6.9 "Transmission RTR Register (TRTRR)" and 22.6.10 "Remote Frame Receiving Wait Register (RFWTR)".

*2: For cancellation of transmission, see Sections 22.6.11 "Transmission Cancel Register (TCANR)" and 22.6.12 "Transmission Complete Register (TCR)".

Writing 0 to TREQx is ignored.

0 is read when a 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).

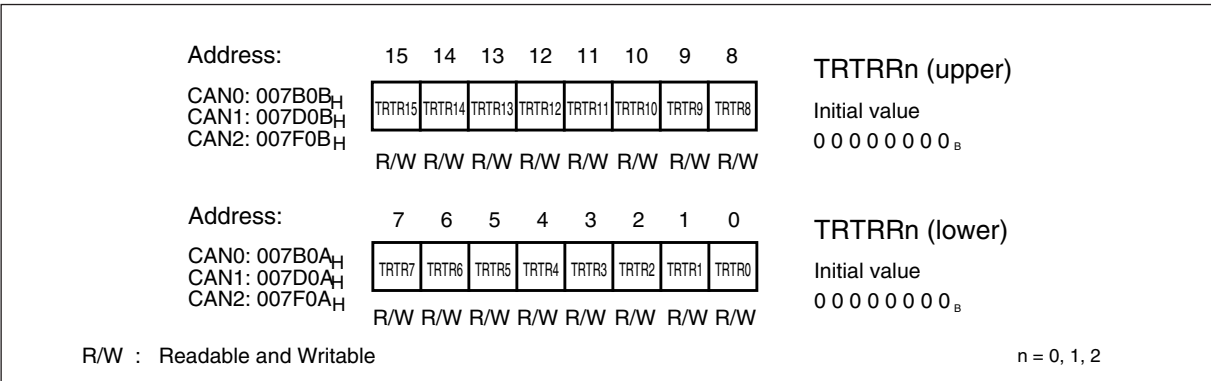
TREQx is 1 while transmission is pending, and becomes 0 when transmission is completed or canceled.

22.6.9 Transmission RTR Register (TRTRR)

This register stores the RTR (Remote Transmission Request) bits for the message buffers (x).

■ Transmission RTR register (TRTRR)

Figure 22.6-12 Configuration of the transmission RTR register (TRTRR)



- 0: Data frame.
- 1: Remote frame.

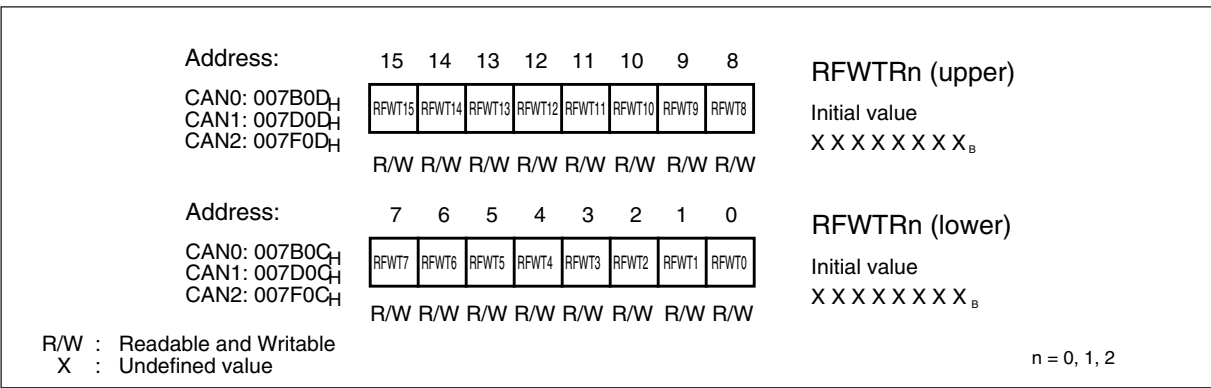
22.6.10 Remote Frame Receiving Wait Register (RFWTR)

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)

Figure 22.6-13 Configuration of the remote frame receiving wait register (RFWTR)



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.

22.6.11 Transmission Cancel Register (TCANR)

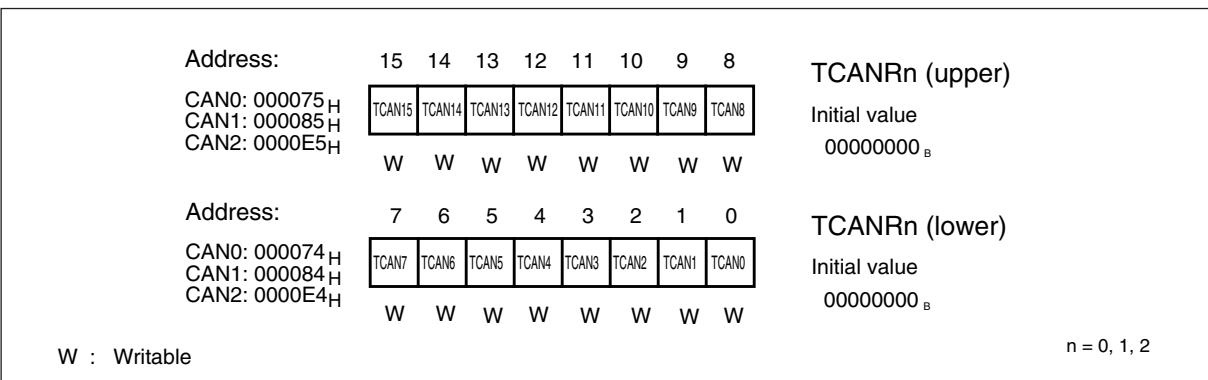
When 1 is written to TCANx, this register cancels a pending request for transmission to the message buffer (x).

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)

Figure 22.6-14 Configuration of the transmission cancel register (TCANR)



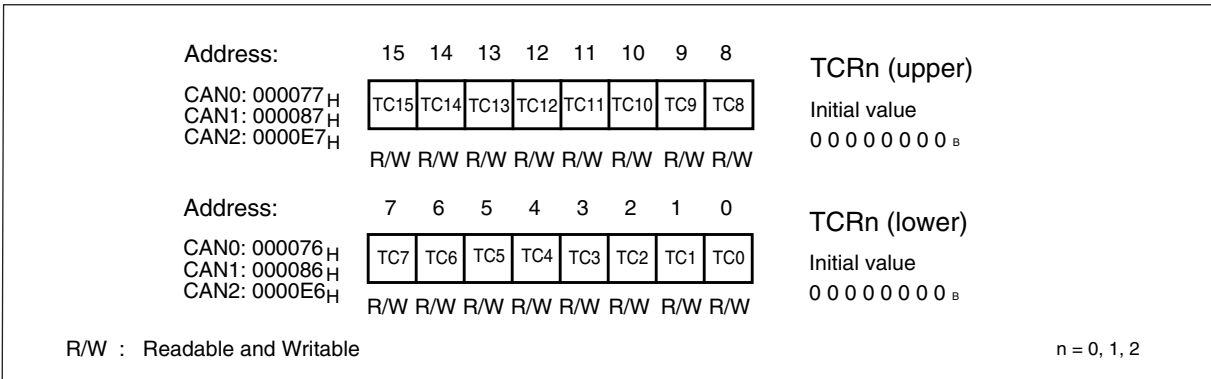
22.6.12 Transmission Complete Register (TCR)

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)

Figure 22.6-15 Configuration of the transmission complete register (TCR)



● 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 a Read Modify Write instruction is performed.

Note:

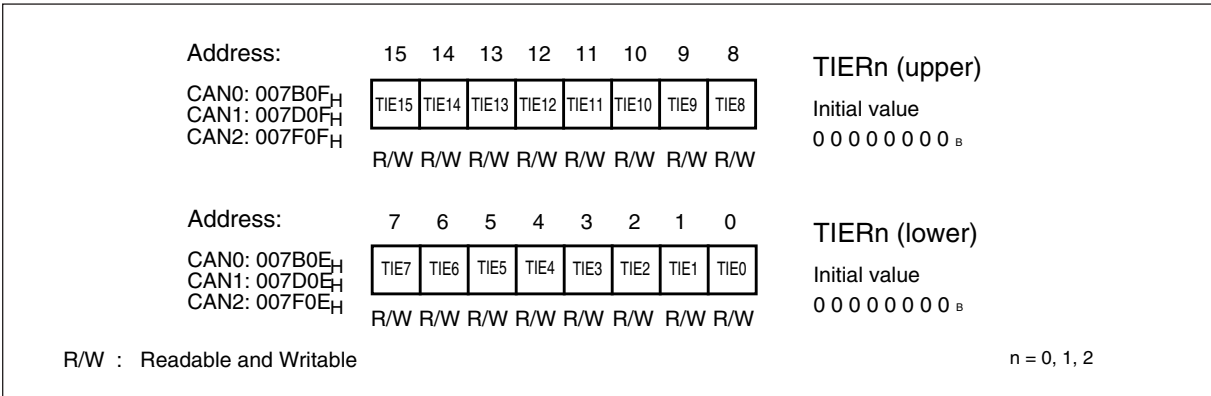
If setting to 1 by completion of the transmit operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

22.6.13 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)

Figure 22.6-16 Configuration of the transmission interrupt enable register (TIER)



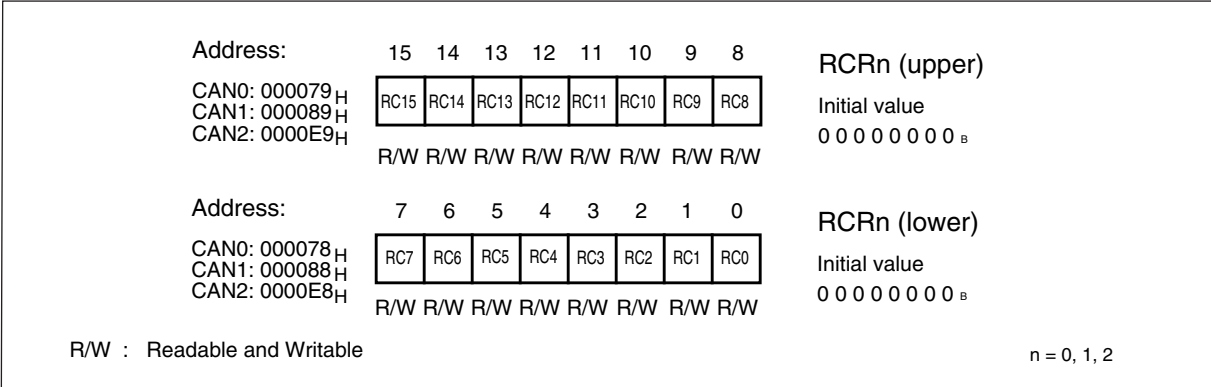
- 0: Transmission interrupt disabled.
- 1: Transmission interrupt enabled.

22.6.14 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)

Figure 22.6-17 Configuration of the reception complete register (RCR)



● 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 a Read Modify Write instruction is performed.

Note:

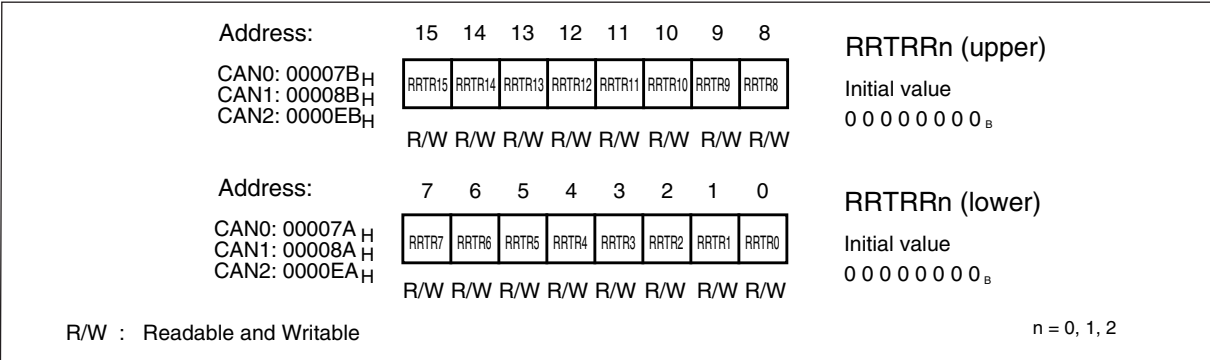
If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

22.6.15 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)

Figure 22.6-18 Configuration of the remote request receiving register (RRTRR)



● 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 a Read Modify Write instruction is performed.

Note:

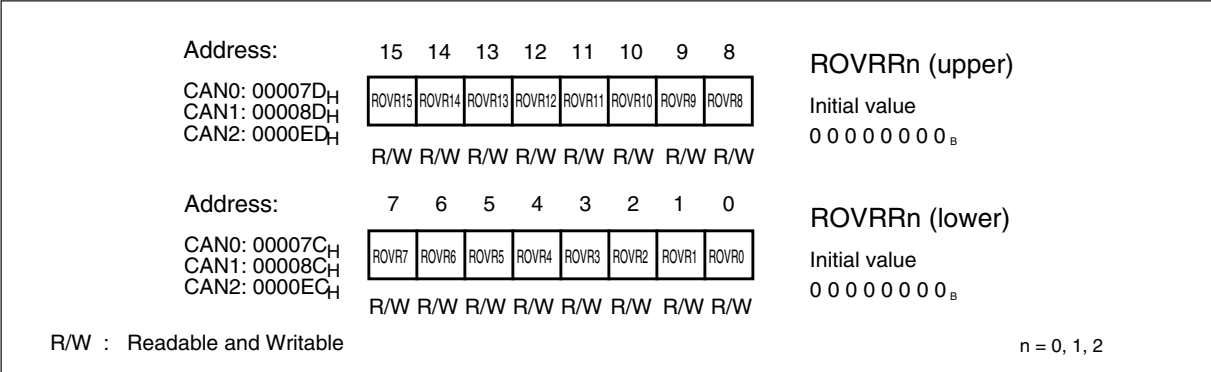
If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

22.6.16 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)

Figure 22.6-19 Configuration of the receive overrun register (ROVRR)



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 a 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.

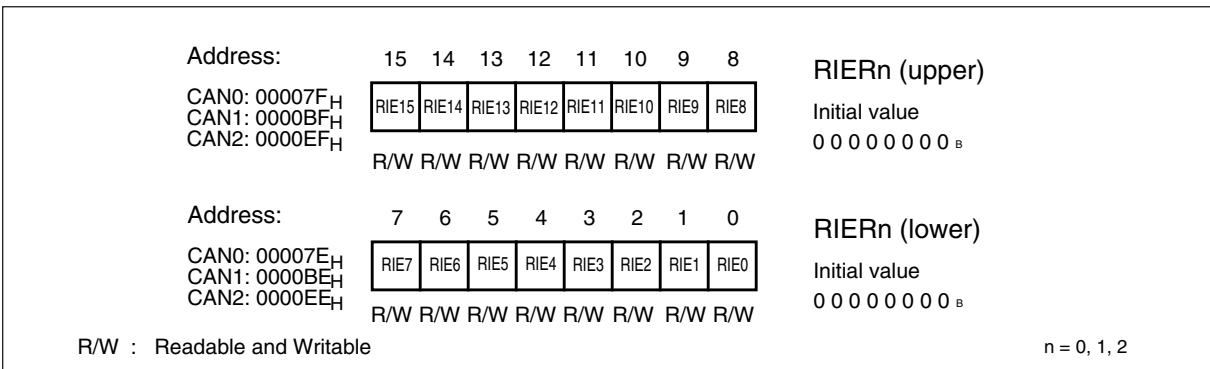
22.6.17 Reception Interrupt Enable Register (RIER)

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)

Figure 22.6-20 Configuration of the reception interrupt enable register (RIER)



- 0: Reception interrupt disabled.
- 1: Reception interrupt enabled.

22.6.18 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)

Figure 22.6-21 Configuration of the acceptance mask select register (AMSR)

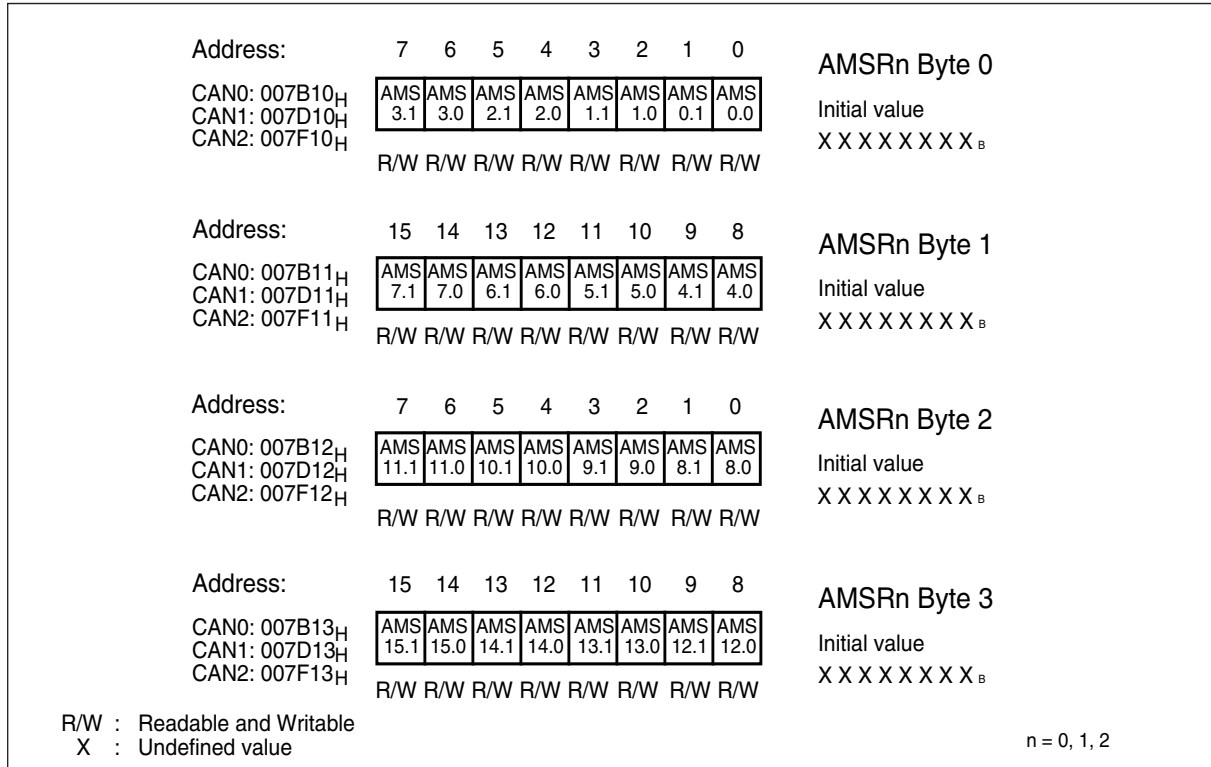


Table 22.6-7 Selection of acceptance mask

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 the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in Section 22.16 "Precautions when Using CAN Controller".

22.6.19 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)

Figure 22.6-22 Configuration of the acceptance mask register 0 (AMR0)

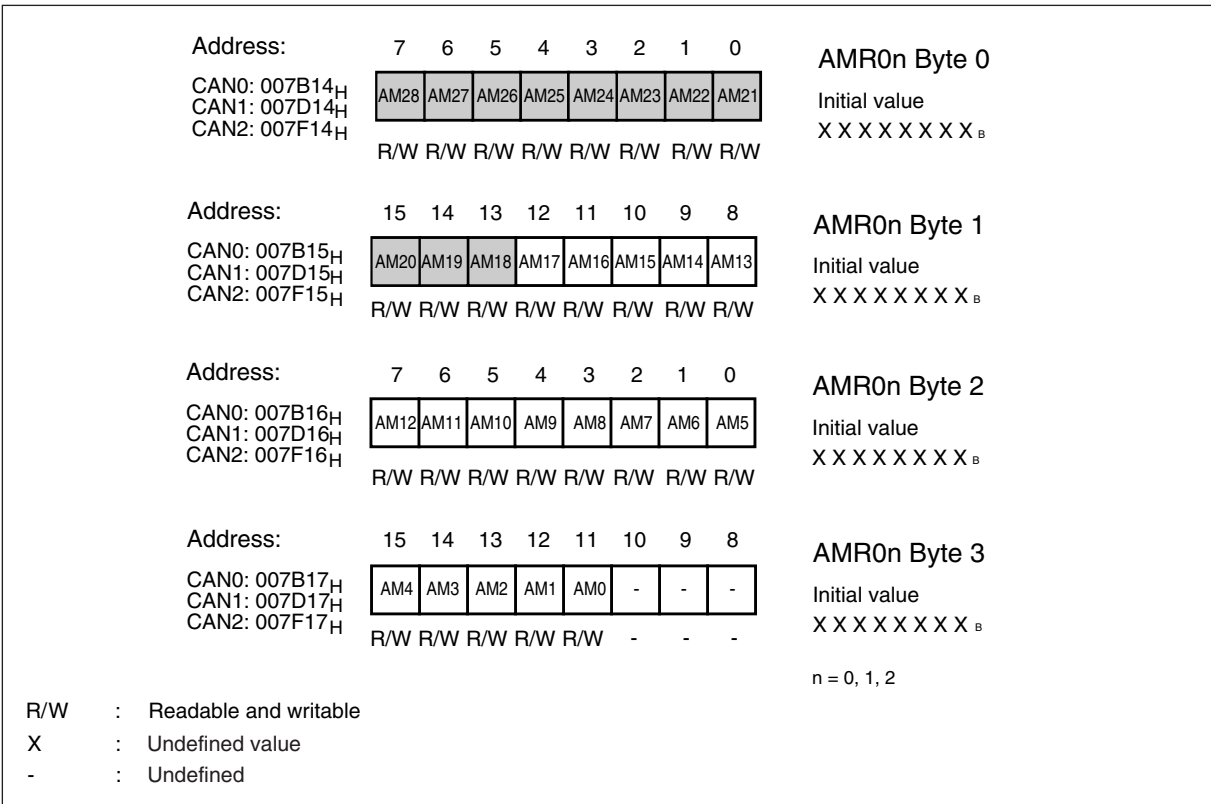
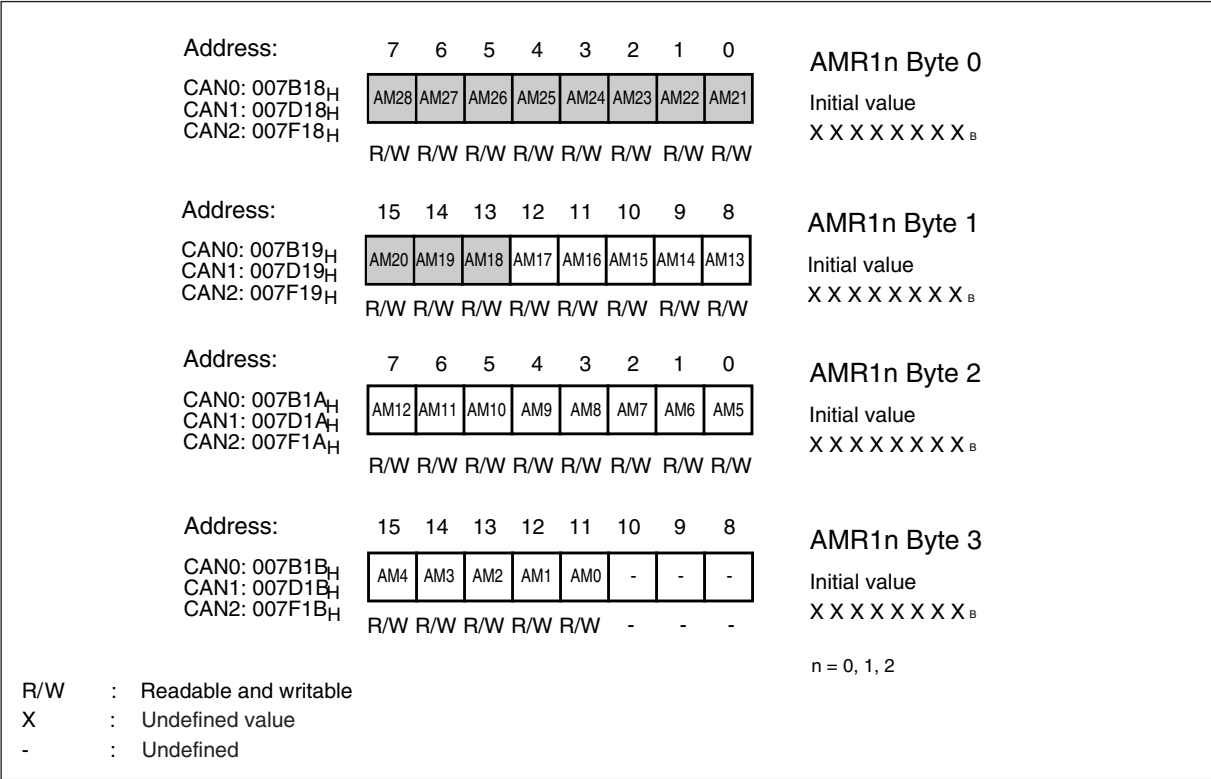


Figure 22.6-23 Configuration of the acceptance mask register 1 (AMR1)



● 0: Compare

Compare the bit of the acceptance code (ID register IDRx 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 (IDRx) 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 (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffers are 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 the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in Section 22.16 "Precautions when Using CAN Controller".

22.6.20 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 Section 22.7 "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 Section 22.8 "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 Section 22.12 "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.

22.6.21 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)

Figure 22.6-24 Configuration of the ID registers (IDRx)

Address:	7 6 5 4 3 2 1 0	IDRxn Byte 0
CAN0: 007A20 _H + 4×x	ID28 ID27 ID26 ID25 ID24 ID23 ID22 ID21	Initial value
CAN1: 007C20 _H + 4×x		X X X X X X X _B
CAN2: 007E20 _H + 4×x	R/W R/W R/W R/W R/W R/W R/W R/W	
Address:	15 14 13 12 11 10 9 8	IDRxn Byte 1
CAN0: 007A21 _H + 4×x	ID20 ID19 ID18 ID17 ID16 ID15 ID14 ID13	Initial value
CAN1: 007C21 _H + 4×x		X X X X X X X _B
CAN2: 007E21 _H + 4×x	R/W R/W R/W R/W R/W R/W R/W R/W	
Address:	7 6 5 4 3 2 1 0	IDRxn Byte 2
CAN0: 007A22 _H + 4×x	ID12 ID11 ID10 ID9 ID8 ID7 ID6 ID5	Initial value
CAN1: 007C22 _H + 4×x		X X X X X X X _B
CAN2: 007E22 _H + 4×x	R/W R/W R/W R/W R/W R/W R/W R/W	
Address:	15 14 13 12 11 10 9 8	IDRxn Byte 3
CAN0: 007A23 _H + 4×x	ID4 ID3 ID2 ID1 ID0 - - -	Initial value
CAN1: 007C23 _H + 4×x		X X X X X X X _B
CAN2: 007E23 _H + 4×x	R/W R/W R/W R/W R/W - - -	

R/W : Readable and Writable
 X : Undefined value
 - : Undefined

x = 0, ..., 15
 n = 0, 1, 2

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.

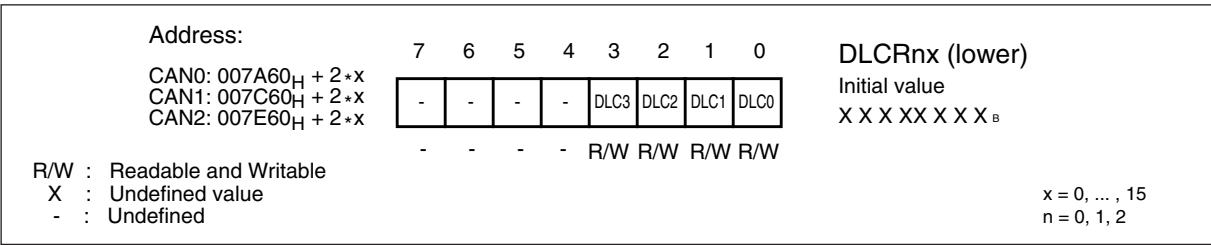
To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in Section 22.16 "Precautions when Using CAN Controller".

22.6.22 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)

Figure 22.6-25 Configuration of the DLC registers (DLCRx)



● 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.

22.6.23 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)

Figure 22.6-26 Configuration of the data registers (DTRx)

Address: CAN0: 007A80 _H + 8*x CAN1: 007C80 _H + 8*x CAN2: 007E80 _H + 8*x	7 6 5 4 3 2 1 0 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 0 Initial value X X X X X X X _B
Address: CAN0: 007A81 _H + 8*x CAN1: 007C81 _H + 8*x CAN2: 007E81 _H + 8*x	15 14 13 12 11 10 9 8 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 1 Initial value X X X X X X X _B
Address: CAN0: 007A82 _H + 8*x CAN1: 007C82 _H + 8*x CAN2: 007E82 _H + 8*x	7 6 5 4 3 2 1 0 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 2 Initial value X X X X X X X _B
Address: CAN0: 007A83 _H + 8*x CAN1: 007C83 _H + 8*x CAN2: 007E83 _H + 8*x	15 14 13 12 11 10 9 8 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 3 Initial value X X X X X X X _B
Address: CAN0: 007A84 _H + 8*x CAN1: 007C84 _H + 8*x CAN2: 007E84 _H + 8*x	7 6 5 4 3 2 1 0 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 4 Initial value X X X X X X X _B
Address: CAN0: 007A85 _H + 8*x CAN1: 007C85 _H + 8*x CAN2: 007E85 _H + 8*x	15 14 13 12 11 10 9 8 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 5 Initial value X X X X X X X _B
Address: CAN0: 007A86 _H + 8*x CAN1: 007C86 _H + 8*x CAN2: 007E86 _H + 8*x	7 6 5 4 3 2 1 0 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 6 Initial value X X X X X X X _B
Address: CAN0: 007A87 _H + 8*x CAN1: 007C87 _H + 8*x CAN2: 007E87 _H + 8*x	15 14 13 12 11 10 9 8 D7 D6 D5 D4 D3 D2 D1 D0 R/W R/W R/W R/W R/W R/W R/W R/W	DTRxn Byte 7 Initial value X X X X X X X _B

R/W : Readable and Writable
X : Undefined value

x = 0, 1, ..., 15
n = 0, 1, 2

- 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.

22.7 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.

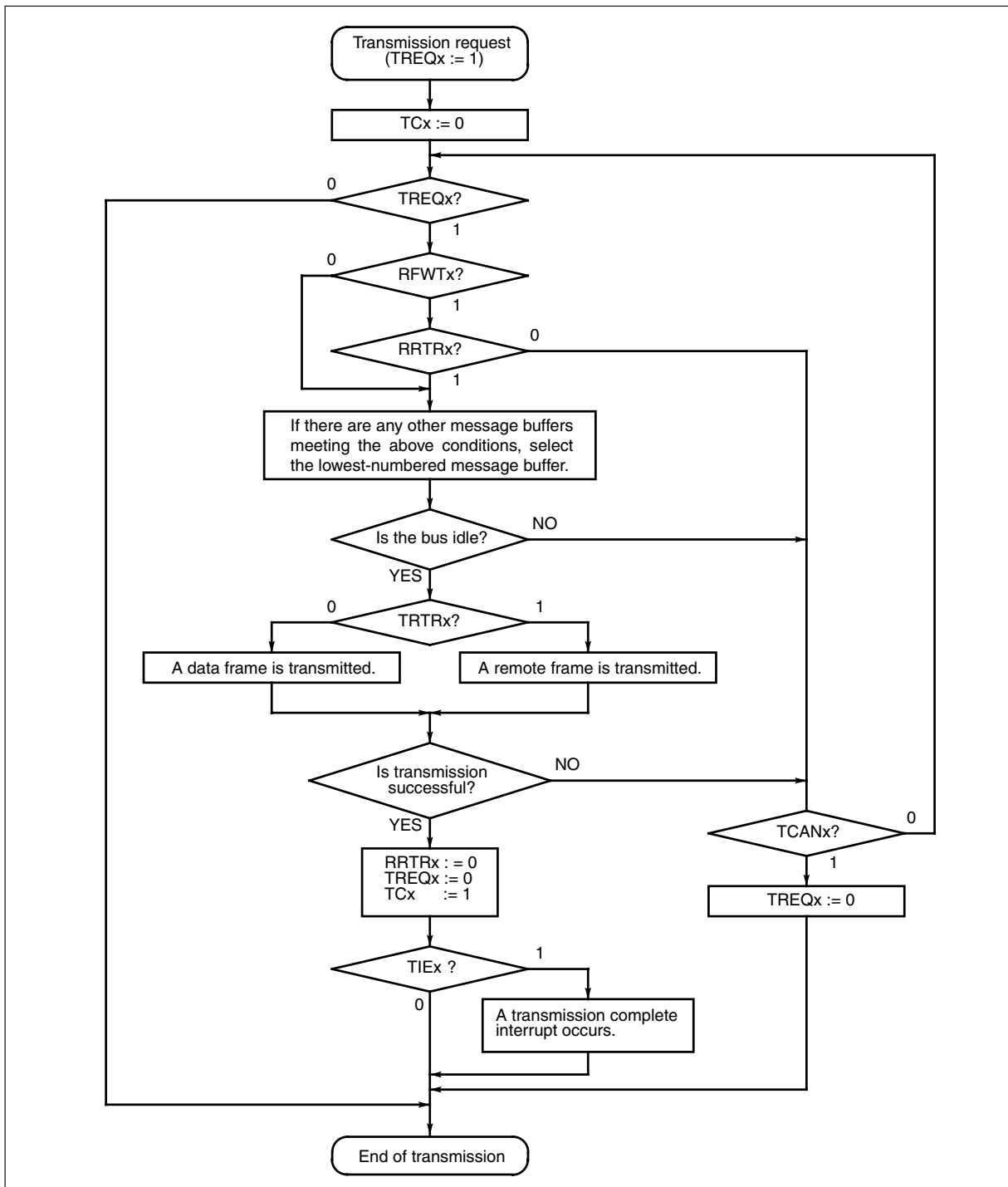
■ Completing transmission of the CAN controller

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 22.7-1 Transmission flowchart of the CAN controller



22.8 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.

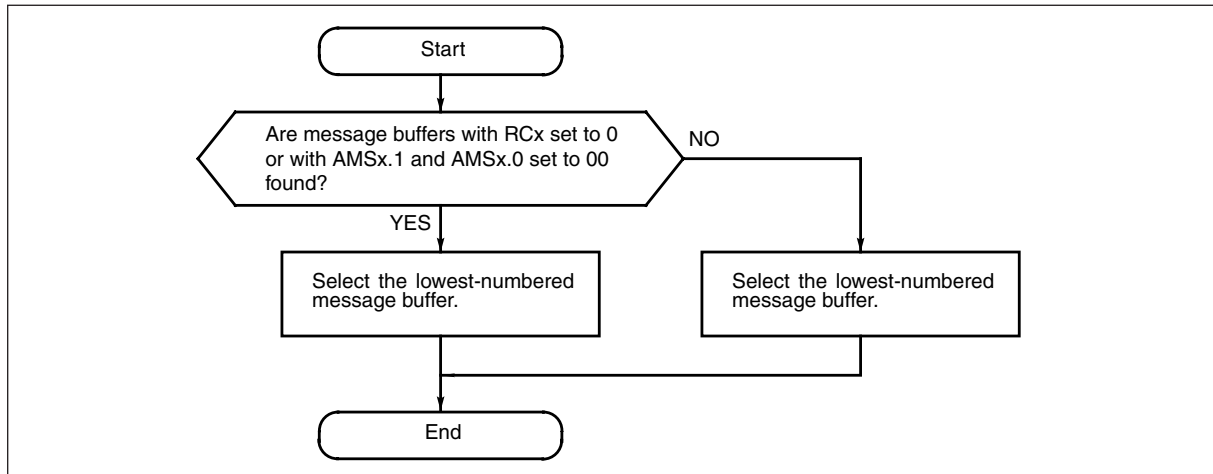
When receiving remote frames, received messages are stored only in the IDRx and DLCRx, and the DTRx remains unchanged.

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 22.8-1 "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 22.8-1 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 22.7-1 "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:

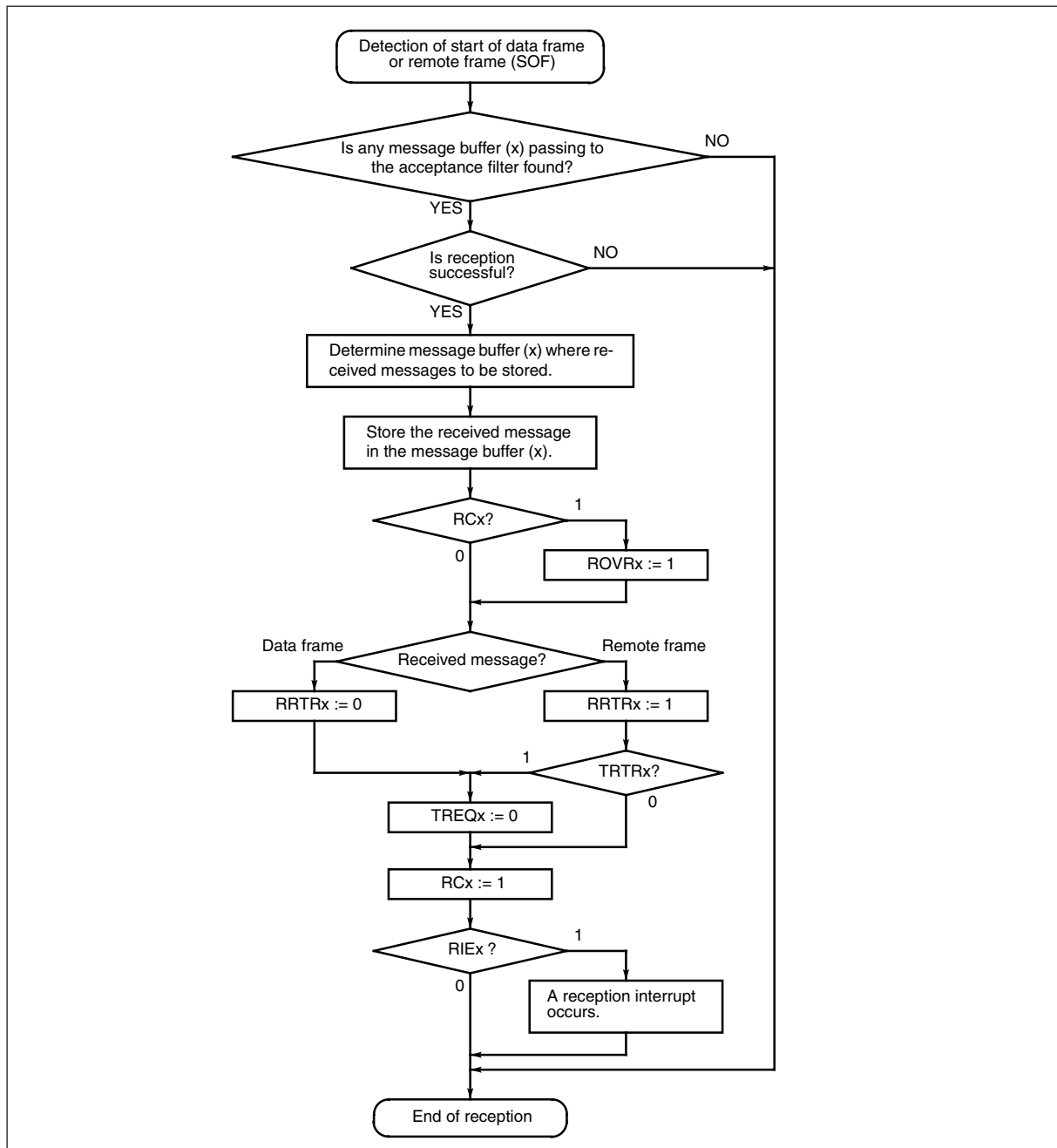
This CAN controller will not receive any messages transmitted by itself.

22.9 Reception Flowchart of CAN Controller

Figure 22.9-1 "Reception flowchart of the CAN controller" shows a reception flowchart of the CAN controller.

■ Reception flowchart of the CAN controller

Figure 22.9-1 Reception flowchart of the CAN controller



22.10 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 Sections 22.6.18 "Acceptance Mask Select Register (AMSR)" and 22.6.19 "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 and Timebase timer), 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).

22.11 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 TIEx of the transmission complete interrupt enable register (TIER) to 1.

When not generating a transmission complete interrupt, set TIEx to 0.

● Setting transmission request

For a transmission request, set TREQx 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 TCANx of the transmission cancel register (TCANR).

Check TREQx. For TREQx = 0, transmission cancellation is terminated or transmission is completed. Check TCx of the transmission complete register (TCR). For TCx = 0, transmission cancellation is terminated. For TCx = 1, transmission is completed.

● Processing for completion of transmission

If transmission is successful, 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.

After checking the transmission completion, write 0 to TCx 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.

22.12 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 RIEx of the reception interrupt enable register (RIER) to 1.

To disable reception interrupt, set RIEx to 0.

● Starting reception

When starting reception after setting, set BVALx 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 RCx of the reception complete register (RCR) becomes 1. For data frame reception, RRTRx of the remote request receiving register (RRTRR) becomes 0. For remote frame reception, RRTRx becomes 1.

If a reception interrupt is enabled (RIEx of the reception interrupt enable register (RIER) is 1), an interrupt occurs.

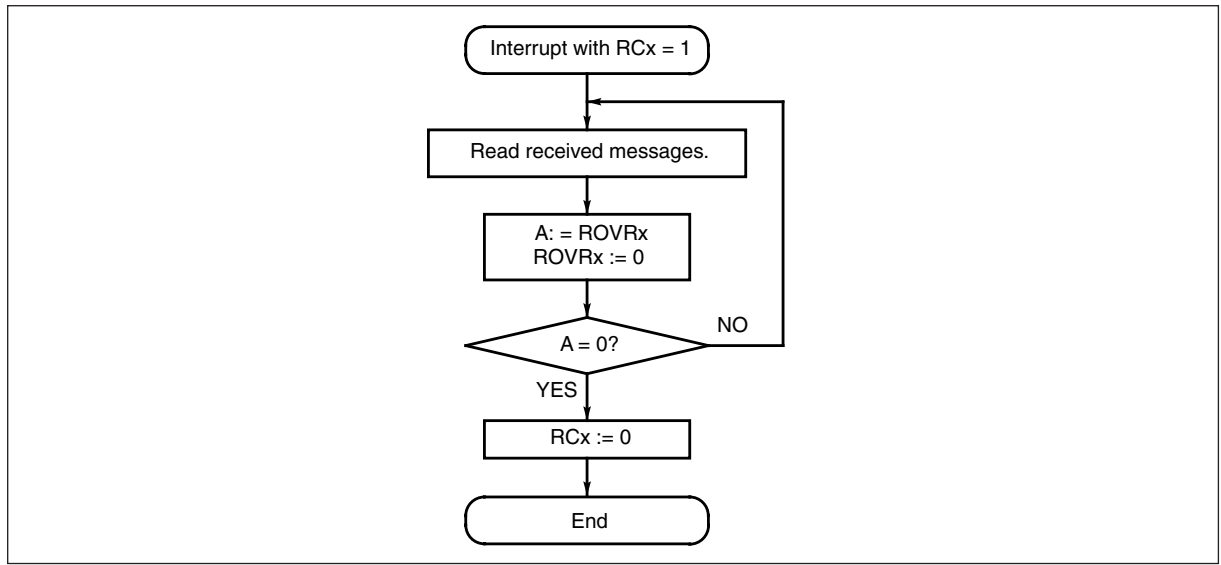
After checking the reception completion (RCx = 1), process the received message.

After completion of processing the received message, check ROVRx of the reception overrun register (ROVRR).

If ROVRx = 0, the processed received message is valid. Write 0 to RCRx to set it to 0 (the reception complete interrupt is also canceled) to terminate reception.

If ROVRx = 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 ROVRx bit to 0 by writing 0 to it.

Figure 22.12-1 "Example of receive interrupt handling" shows an example of receive interrupt handling.

Figure 22.12-1 Example of receive interrupt handling

22.13 Setting Configuration of Multi-level Message Buffer

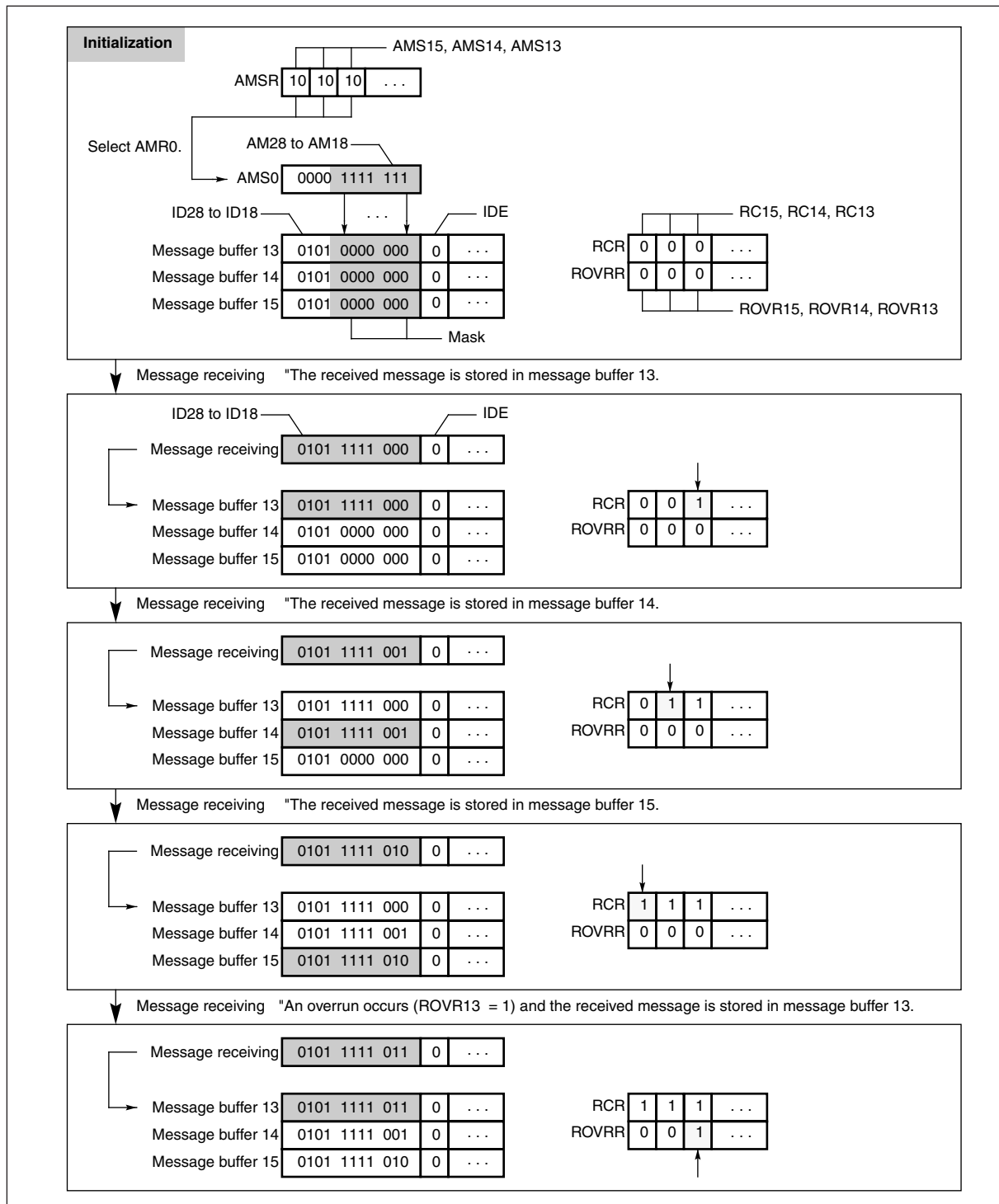
If the receptions are performed frequently, or if several different ID's of messages are received, in other words, if there is insufficient time for handling messages, more than one message buffer can be combined into a multi-level message buffer to provide allowance for processing time of the received 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 ((AMSRx.1, AMSR.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 RCx bit of the receive completion register (RCR), so received messages are always stored in lower-numbered (higher-priority) message buffers even if All Bits Compare and identical acceptance code (ID register (IDRx)) 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 22.13-1 "Examples of operation of multi-level message buffer" shows operational examples of multi-level message buffers.

Figure 22.13-1 Examples of operation of multi-level message buffer**Note:**

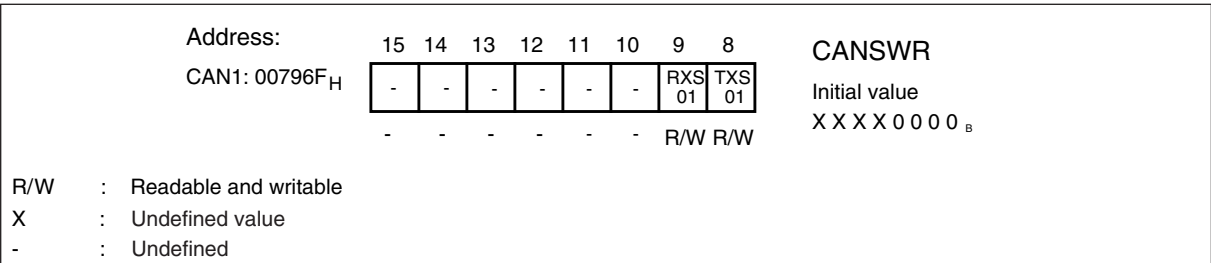
Four messages are received with the same acceptance filter set in message buffers 13, 14 and 15.

22.14 Setting the redirection of CAN1 RX/TX pin

CAN1 can be changed the redirection CAN1 RX/TX pin (RX1/TX1) to RX0/TX0 pin by the CANSWR register.

■ CAN1 RX/TX pin switching register (CANSWR)

Figure 22.14-1 CAN1 RX/TX pin switching register (CANSWR)

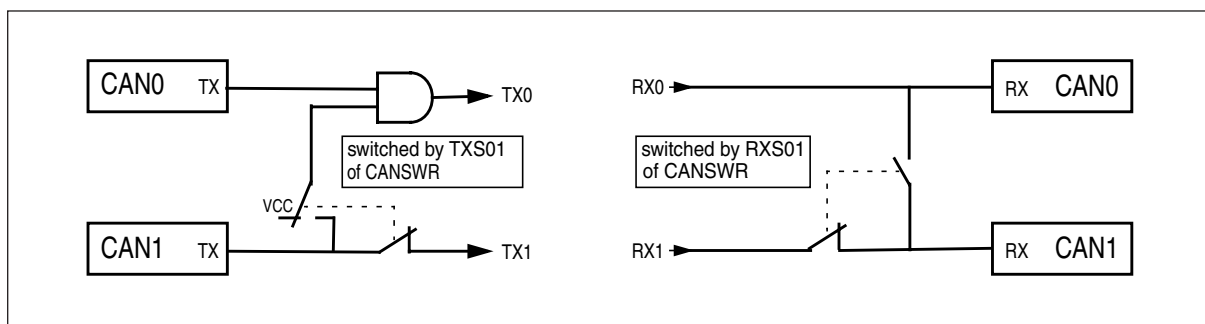


■ CAN switch register (CANSWR) contents

Table 22.14-1 Function of each bit of the CAN switch register

Bit Name		Function
bit 15 to 10	Undefined	
bit 9	RXS01: Reception switch 0/1	If "0" is written to this bit, input of CAN1 is inputted from RX1 pin. If "1" is written to this bit, input of CAN1 is inputted from RX0 pin.
bit 8	TXS01: Transmission switch 0/1	If "0" is written to this bit, output of CAN1 is outputted from TX1 pin. If "1" is written to this bit, output of CAN1 is outputted from TX0 pin.

CAN1 can be changed the redirection CAN1 RX/TX pin, and shared with CAN0, as shown in Figure 22.14-2 "Redirection of CAN1 RX/TX". Therefore this feature allows the saving of one external CAN transceiver. This function is selected by CANSWR register.

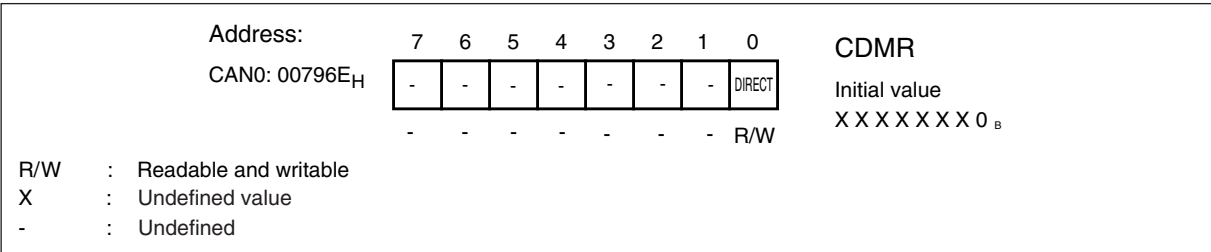
Figure 22.14-2 Redirection of CAN1 RX/TX

22.15 Setting the CAN Direct Mode Register

The clock modulation can be set on the system clock in MB90340 series.
To operate CAN normally, this register must be set correctly.

■ CAN direct mode register (CDMR)

Figure 22.15-1 Configuration of the CAN direct mode register (CDMR)



■ CAN direct mode register contents

Table 22.15-1 Function of the DIRECT bit of the CAN direct mode register

Bit Name		Function
bit 7 to 1	Undefined	-
bit 0	DIRECT	If the clock modulation is set (initial state), the bit should be set "0". If the clock modulation is not set, the bit should be set "1".

Note:

The clock modulation function can't be used, because the function is tested now. So, at using CAN controller, the DIRECT bit of the register must be set "1".

22.16 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 (the read value of the CSR: HALT bit is 0 and CAN Controller is ready to transmit messages). This section shows the work around of this malfunction.

● Condition

When following two conditions occur at the same time, the CAN Controller will not perform to transmit messages normally.

- CAN Controller is participating in the CAN communication. (i.e. The read value of the CSR: HALT bit is 0 and CAN Controller is ready to transmit messages)
- Message buffers are read when BVAL bits disable the message buffers.

● Work around

Operation for suppressing transmission request

Do not 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).

In case a buffer needs to be disabled, ensure that no transmission request is pending (if it was requested before)! Therefore, do not reset BVALx-Bit before testing, if a transmission is ongoing:

- a) Cancel the transmission request (TCANx=1;), if necessary
- b) and wait for the transmission completion (while (TREQx==1);) by polling or interrupt.

Only after that the transmission buffer can be disabled (BVALx=0;).

Note for case a), if transmission of that buffer has already started, canceling the request is ignored and disabling the buffer is delayed until the end of the transmission.

■ Setting of the CAN direct mode

The DIRECT bit of the CAN direct mode register CDMR should be set following, depends on the setting of clock modulator. If the DIRECT bit is not set correctly, the CAN controller does not operate normally.

● Not using the clock modulator

- The clock modulator control register CMCR:PMOD bit ... 0 (initial value)
- The CAN direct mode register CDMR: DIRECT bit ... 1

● Using the clock modulator

- The clock modulator control register CMCR:PMOD bit ... 1
- The CAN direct mode register CDMR: DIRECT bit ... 0 (initial value)

Note:

The clock modulation can't be used, because the function is tested now. So, at using CAN controller, the "Not using the clock modulator" setting must keep on be set.

CHAPTER 23

ADDRESS MATCH DETECTION FUNCTION

This chapter explains the address match detection function and operation.

- 23.1 “Outline of the Address Match Detection Function”
- 23.2 “Registers of the Address Match Detection Function”
- 23.3 “Operation of the Address Match Detection Function”
- 23.4 “Example of the Address Match Detection Function”

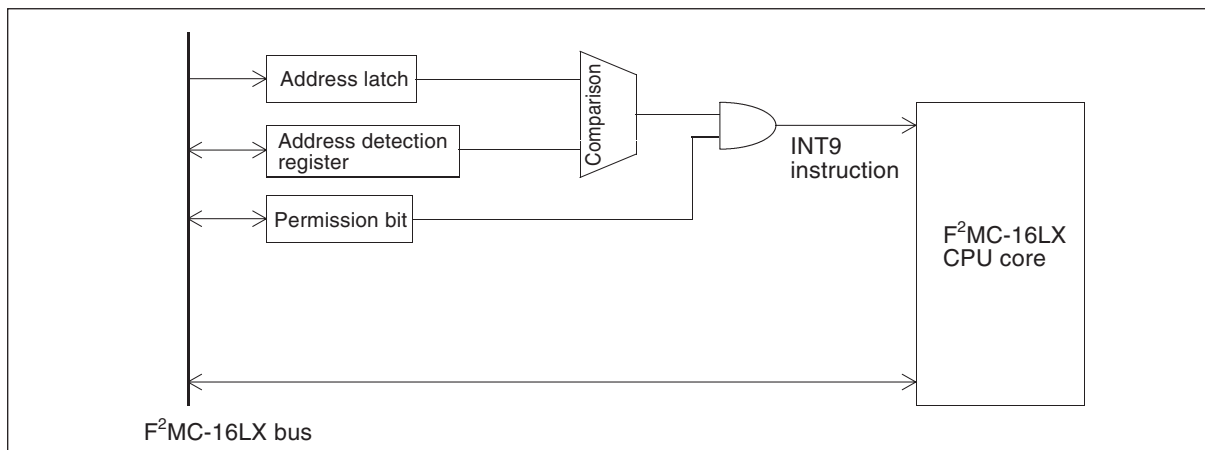
23.1 Outline of the Address Match Detection Function

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 (01_H). 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 6 address detection registers, each with an interrupt permission bit. When an address matches the value set in the address detection register and the interrupt permission bit is 1, the instruction code to be read by the CPU is replaced with the INT9 instruction code.

■ Block diagram of the address match detection function

Figure 23.1-1 Block diagram of the address match detection function



23.2 Registers of the Address Match Detection Function

The two types of registers for the address match detection function are as follows:

- Program address detection registers (PADR0 to PADR5)
- Program address detection control status register (PACSR0 and PACSR1)

■ Program address detection registers (PADR0 to PADR5)

The program address detection registers 0 to 5 (PADR0 to PADR5) compare the address with the value written in each register. If they match when the interrupt permission bit corresponding to ADCSR is 1, the CPU is requested to issue the INT9 instruction.

When the corresponding interrupt bit is 0, nothing occurs.

Figure 23.2-1 Program address detection registers (PADR0 to PADR5)

Program address detection registers	byte	byte	byte	Access	Initial value
PADR0 79E2 _H /79E1 _H /79E0 _H				R/W	Not defined
PADR1 79E5 _H /79E4 _H /79E3 _H				R/W	Not defined
PADR2 79E8 _H /79E7 _H /79E6 _H				R/W	Not defined
PADR3 79F2 _H /79F1 _H /79F0 _H				R/W	Not defined
PADR4 79F5 _H /79F4 _H /79F3 _H				R/W	Not defined
PADR5 79F8 _H /79F7 _H /79F6 _H				R/W	Not defined

Table 23.2-1 “Correspondence between PADR0 to PADR5 registers and PACSR0 and PACSR1 registers” lists the correspondence between the program address detection registers (PADR0 to PADR5) and PACSR.

Table 23.2-1 Correspondence between PADR0 to PADR5 registers and PACSR0 and PACSR1 registers

Address detection register	Interrupt permission bit
PADR0	AD0E (bit 1)
PADR1	AD1E (bit 3)
PADR2	AD2E (bit 5)
PADR3	AD3E (bit 1)
PADR4	AD4E (bit 3)
PADR5	AD5E (bit 5)

■ Program address detection control status register (PACSR)

The program address detection control status register (PACSR) controls the operation of the address detection function.

Figure 23.2-2 Program address detection control status registers (PACSR0/1)

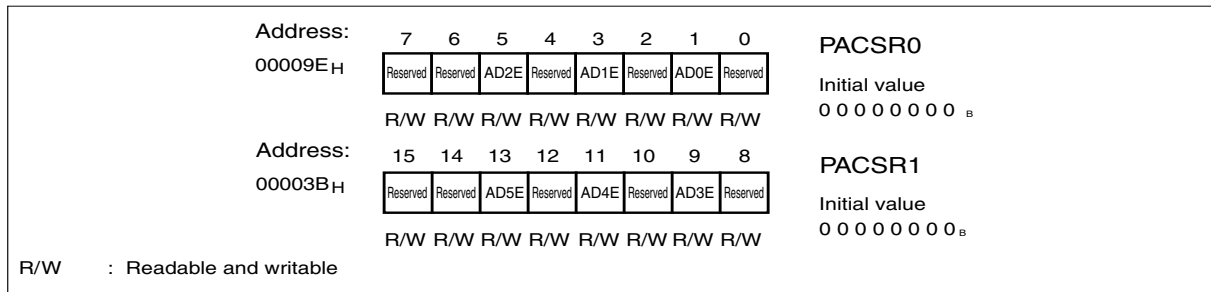


Table 23.2-2 Function of each bit of PACSR1 and PACSR0

Name		Function
bit 15 to 14	Reserved bits	Bits 15 to 14 are reserved. Set these bits to 0 before setting PACSR1.
bit 13	AD5E: Address detect register 1 enable	The AD5E bit is the operation permission bit of ASIE ADR5. When this bit is 1, the address is compared with the PADR5 register. If they match, the INT9 instruction is issued.
bit 12	Reserved bit	Bit 12 is reserved. Set this bit to 0 before setting PACSR1.
bit 11	AD4E: Address detect register 1 enable	The AD4E bit is the operation permission bit of ASIE ADR4. When this bit is 1, the address is compared with the PADR4 register. If they match, the INT9 instruction is issued.
bit 10	Reserved bit	Bit 10 is reserved. Set this bit to 0 before setting PACSR1.
bit 9	AD3E: Address detect register 1 enable	The AD3E bit is the operation permission bit of ASIE ADR3. When this bit is 1, the address is compared with the PADR3 register. If they match, the INT9 instruction is issued.
bit 8	Reserved bit	Bit 8 is reserved. Set this bit to 0 before setting PACSR1.
bit 7 to 6	Reserved bits	Bits 7 to 6 are reserved. Set these bits to 0 before setting PACSR0.
bit 5	AD2E: Address detect register 1 enable	The AD2E bit is the operation permission bit of ASIE ADR2. When this bit is 1, the address is compared with the PADR2 register. If they match, the INT9 instruction is issued.
bit 4	Reserved bits	Bits 4 is reserved. Set these bits to 0 before setting PACSR0.
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.
bit 2	Reserved bit	Bit 2 is reserved. Set this bit to 0 before setting PACSR0.
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 PACSR0.

23.3 Operation of the Address Match Detection Function

If the program counter specifies the same address as the address match detection register, the INT9 instruction is executed. The address match detection function can be achieved by processing the INT9 instruction routine.

■ Operation of the address match detection function

There are 6 address detection registers with a compare enable bit. When the value set in the address detection register and the value of the program counter match and the compare enable bit is set to 1, the CPU executes the INT9 instruction.

Note:

If the value of the address detection register and the value of the program counter match, the contents of internal data bus is changed to 01_H. Consequently, the INT9 instruction is executed. Before changing the contents of the address detection register, always set the compare enable bit to 0. While the compare enable bit is set to 1, changing the contents of the address detection register may result in a malfunction.

23.4 Example of the Address Match Detection Function

Figure 23.4-1 “System configuration example of the address match detection function” shows a system configuration example of the address match detection function. Table 23.4-1 “EEPROM memory map” lists the EEPROM memory map.

■ System configuration example of the address match detection function

Figure 23.4-1 System configuration example of the address match detection function

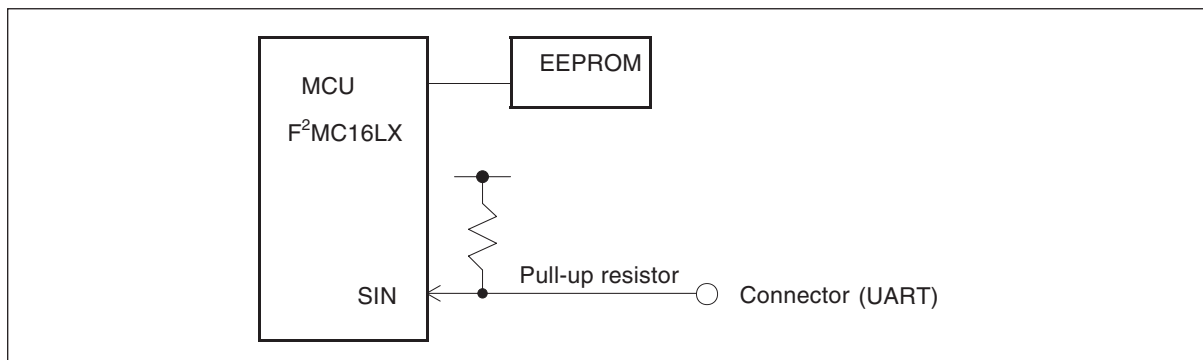


Table 23.4-1 EEPROM memory map

Address	Description
0000 _H	Number of bytes of patch program No.0 (If 0, no program error exists.)
0001 _H	Program address No.0 bits 7 to 0
0002 _H	Program address No.0 bits 15 to 8
0003 _H	Program address No.0 bits 24 to 16
0004 _H	Number of bytes of patch program No.1 (If 0, no program error exists.)
0005 _H	Program address No.1 bits 7 to 0
0006 _H	Program address No.1 bits 15 to 8
0007 _H	Program address No.1 bits 24 to 16
0010 _H or higher	Main body of patch program No. 0

● Initial status

EEPROM is set to all 0s.

● When a program error occurs:

The main body of the patch program and program address are transferred to the MCU through the connector (UART). The MCU writes the information to EEPROM.

● Reset sequence

The MCU reads the value of EEPROM after reset. If the number of bytes of the patch program is not 0, the main body of the patch program is read from EEPROM and written to RAM. The MCU then uses either PADR0 or PADR1 to set the patch address and sets the compare enable bit. If the relocatable patch program is required, the first address of the patched program can be written to the RAM area. In this case, the INT9 routine accesses this user-defined RAM area and jumps to the patched program.

● INT9 interrupt

The interrupt routine can know the address where the interrupt occurs by checking the value of the stack program counter. The information that has been placed on the stack during the interrupt is discarded.

■ Example of program patch processing

Figure 23.4-2 Example of program patch processing

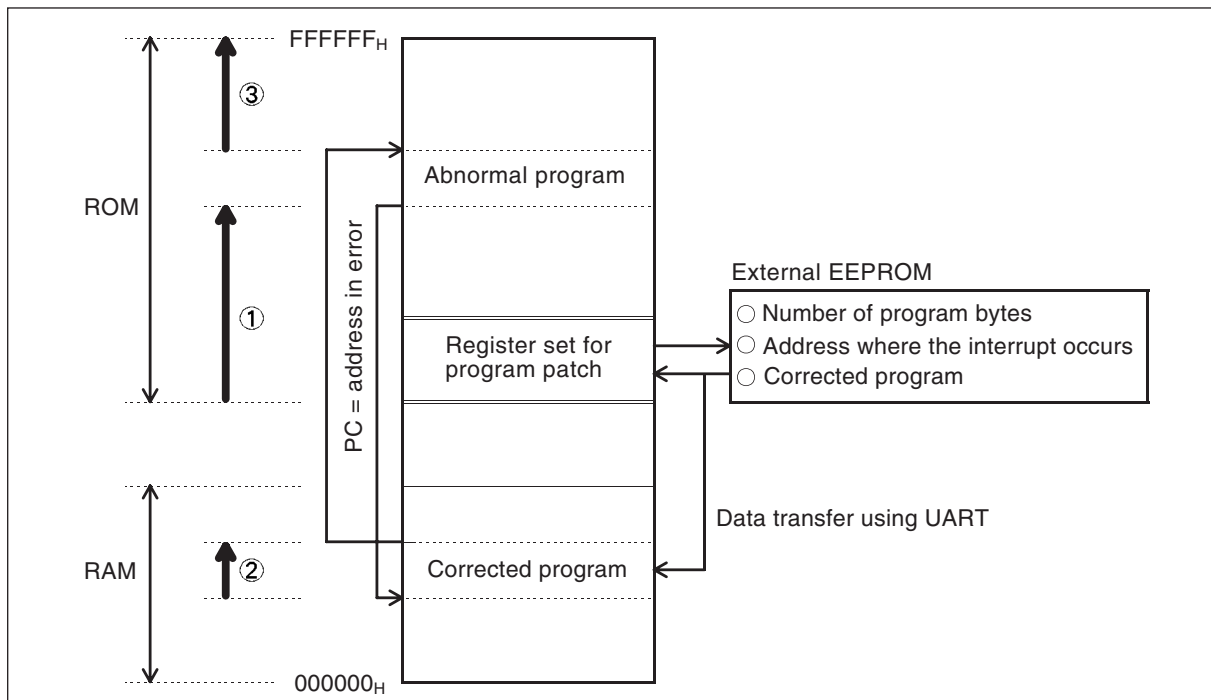
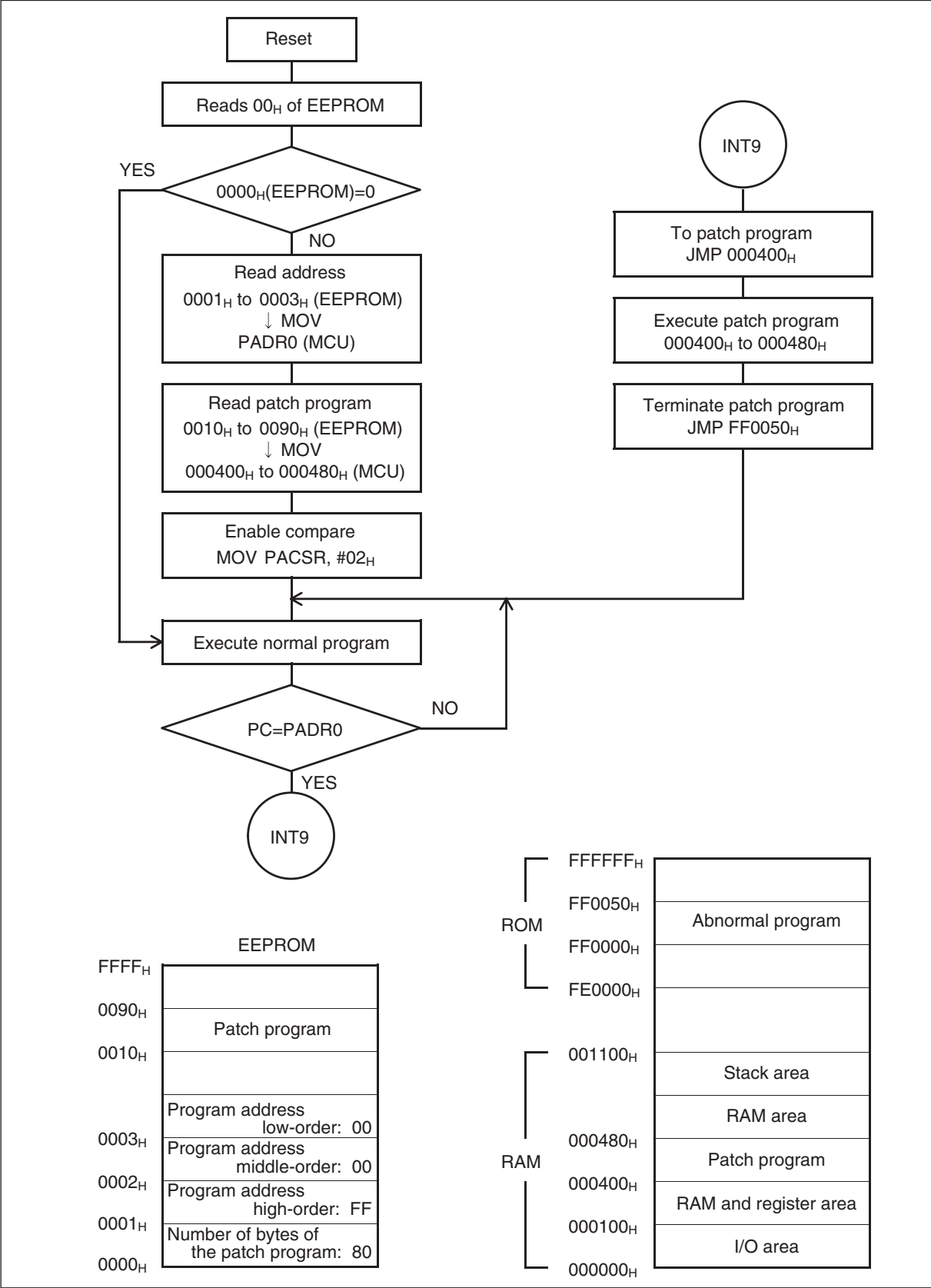


Figure 23.4-3 Flow of program patch processing



CHAPTER 24

ROM MIRRORING MODULE

This chapter explains the ROM mirroring module.

24.1 “Outline of ROM Mirroring Module”

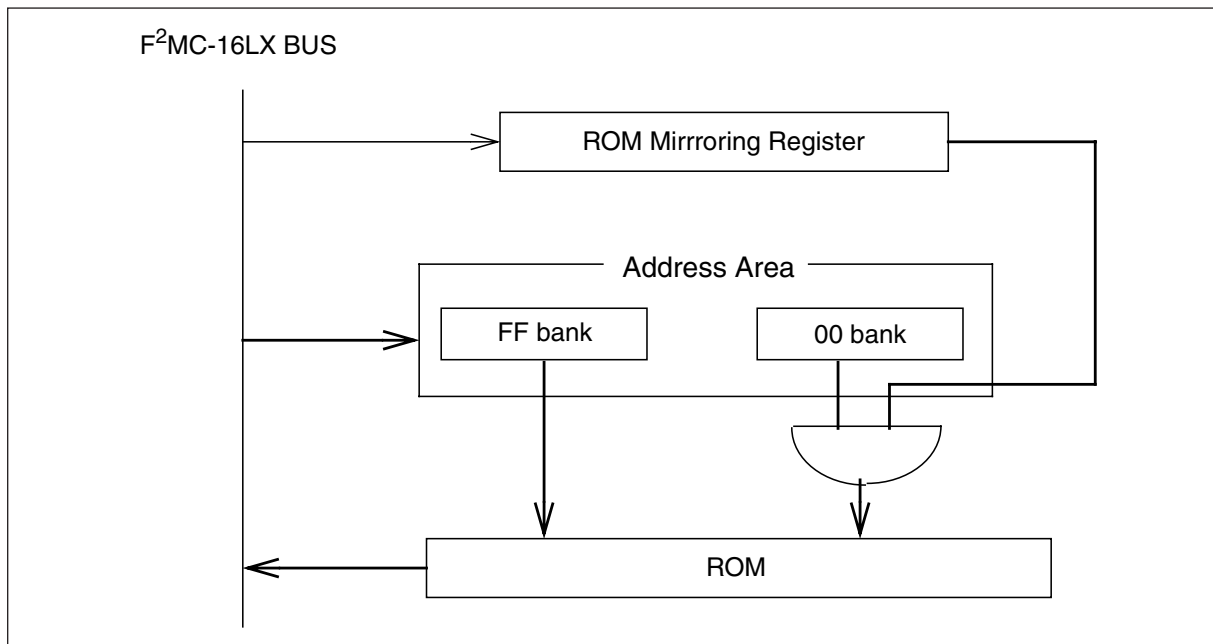
24.2 “ROM Mirroring Register (ROMM)”

24.1 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 24.1-1 Block diagram of ROM mirroring module



24.2 ROM Mirroring Register (ROMM)

Do not access the ROM mirroring register (ROMM) when addresses 008000_H to 00FFFF_H are being accessed.

■ ROM mirroring register (ROMM)

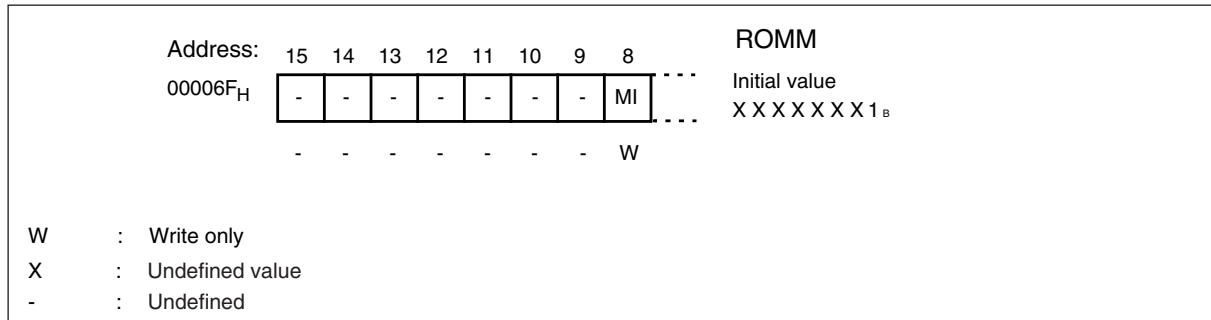


Table 24.2-1 Function of each bit of ROM mirroring register

Bit Name		Function
bit 15 to 9	Undefined	
bit 8	MI: Mirror bit	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 FF8000_H to FFFFFFF_H is mirrored to 008000_H to 00FFFF_H when the ROM mirroring function is activated. Therefore, addresses FF0000_H to FF7FFF_H will not be mirrored to 00 bank.

CHAPTER 25

0.5M/1M/2M/3M/4M-BIT FLASH MEMORY

This chapter explains the functions and operation of the 0.5M/1M/2M/3M/4M-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”.

- 25.1 “Outline of 0.5M/1M/2M/3M/4M-bit Flash Memory”
- 25.2 “Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory”
- 25.3 “Write/Erase Modes”
- 25.4 “Flash Memory Control Status Register (FMCS)”
- 25.5 “Starting the Flash Memory Automatic Algorithm”
- 25.6 “Confirming the Automatic Algorithm Execution State”
- 25.7 “Detailed Explanation of Writing to and Erasing Flash Memory”
- 25.8 “Notes on Using 0.5M/1M/2M/3M/4M-bit Flash Memory”
- 25.9 “Flash Security Feature”
- 25.10 “Example of Programming 1M-bit Flash Memory”

25.1 Overview of 0.5M/1M/2M/3M/4M-bit Flash Memory

The 0.5M/1M/2M/3M/4M-bit flash memory is mapped to the FF/FE/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.

■ 0.5M/1M/2M/3M/4M-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 reading cycle time: Minimum of 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 control status register (FMCS)

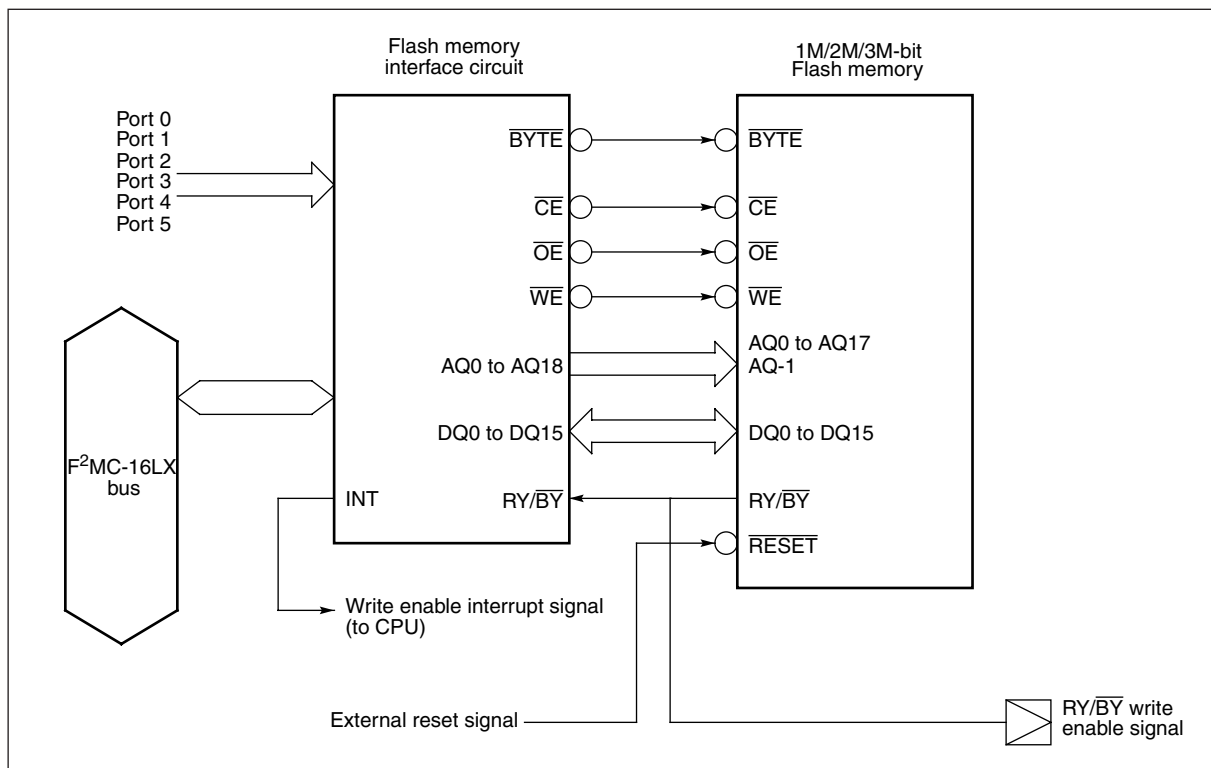
FMCS	7	6	5	4	3	2	1	0	Initial value
Address: 0000AE _H	INTE	RDYINT	WE	RDY	Reserved	Reserved	Reserved	Reserved	000X0000 _B
	(R/W)	(R/W)	(R/W)	(R)	-	-	-	-	

25.2 Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory

Figure 25.2 “Block diagram of the entire flash memory” shows a block diagram of the entire flash memory with the flash memory interface circuit included. Figure 25.2 “Sector configuration of the 0.5M/1M/2M/3M/4M-bit flash memory” shows the sector configuration of the flash memory.

■ Block diagram of the entire flash memory

Figure 25.2-1 Block diagram of the entire flash memory



■ Sector configuration of the 0.5M/1M/2M/3M/4M-bit flash memory

Figure 25.2-2 “Sector configuration of the 0.5M/1M/2M/3M/4M-bit flash memory” shows the sector configuration of the 0.5M/1M/2M/3M/4M-bit flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

Figure 25.2-2 Sector configuration of the 3M/2M-bit flash memory

MB90F343/C/S/CS	Programmer address*	CPU address	MB90F342/C/S/CS MB90F349/C/S/CS	Programmer address*	CPU address
SA8 (16 KBytes)	7FFFF _H	FFFFFF _H	SA6 (16 KBytes)	7FFFF _H	FFFFFF _H
SA7 (8 KBytes)	7BFFF _H	FFBFFF _H	SA5 (8 KBytes)	7BFFF _H	FFBFFF _H
SA6 (8 KBytes)	79FFF _H	FF9FFF _H	SA4 (8 KBytes)	79FFF _H	FF9FFF _H
SA5 (32 KBytes)	77FFF _H	FF7FFF _H	SA3 (32 KBytes)	77FFF _H	FF7FFF _H
SA4 (64 KBytes)	6FFFF _H	FEFFFF _H	SA2 (64 KBytes)	6FFFF _H	FEFFFF _H
SA3 (64 KBytes)	5FFFF _H	FDFFFF _H	SA1 (64 KBytes)	5FFFF _H	FDFFFF _H
Unused	4FFFF _H	FCFFFF _H	SA0 (64 KBytes)	4FFFF _H	FCFFFF _H
SA2 (64 KBytes)	3FFFF _H	FBFFFF _H		40000 _H	FC0000 _H
SA1 (64 KBytes)	2FFFF _H	FAFFFF _H			
SA0 (64 KBytes)	1FFFF _H	F9FFFF _H			
Unused	0FFFF _H	F8FFFF _H			
	00000 _H	F80000 _H			

* The programmer address is equivalent to the CPU address when data is written to the flash memory using a parallel programmer. When a general programmer is used for writing/erasing, this address is used for writing/erasing.

Figure 25.2-3 Sector configuration of the 1M-bit flash memory

MB90F347A/CA/AS/CAS	Programmer address*	CPU address
SA4 (16 KBytes)	7FFFF _H	FFFFFF _H
SA3 (8 KBytes)	7BFFF _H	FFBFFF _H
SA2 (8 KBytes)	79FFF _H	FF9FFF _H
SA1 (32 KBytes)	77FFF _H	FF7FFF _H
SA0 (64 KBytes)	6FFFF _H	FEFFFF _H
	60000 _H	FE0000 _H

* The programmer address is equivalent to the CPU address when data is written to the flash memory using a parallel programmer. When a general programmer is used for writing/erasing, this address is used for writing/erasing.

Figure 25.2-4 Sector configuration of the 4M-bit flash memory

MB90F345/C/S/CS	Writer address*	CPU address
SA13 (16 K byte)	7FFFFH	FFFFFFH
SA12 (8 K byte)	7BFFFH	FFBFFFH
SA11 (8 K byte)	79FFFH	FF9FFFH
SA10 (32 K byte)	77FFFH	FF7FFFH
SA9 (64 K byte)	6FFFFH	FEFFFFH
SA8 (64 K byte)	5FFFFH	FDFFFFH
SA7 (64 K byte)	4FFFFH	FCFFFFH
SA6 (16 K byte)	3FFFFH	FBFFFFH
SA5 (8 K byte)	3BFFFH	FBBFFFH
SA4 (8 K byte)	39FFFH	FB9FFFH
SA3 (32 K byte)	37FFFH	FB7FFFH
SA2 (64 K byte)	2FFFFH	FAFFFFH
SA1 (64 K byte)	1FFFFH	F9FFFFH
SA0 (64 K byte)	0FFFFH	F8FFFFH
	00000H	F80000H

*: The programmer address is equivalent to the CPU address when data is written to the flash memory using a parallel programmer. When a general programmer is used for writing/erasing, this address is used for writing/erasing.

Figure 25.2-5 Sector configuration of the 0.5M-bit flash memory

MB90F346A/CA/AS/CAS	Writer address*	CPU address
SA3 (16 K byte)	7FFFFH	FFFFFFH
SA2 (8 K byte)	7BFFFH	FFBFFFH
SA1 (8 K byte)	79FFFH	FF9FFFH
SA0 (32 K byte)	77FFFH	FF7FFFH
	70000H	FF0000H

*: The programmer address is equivalent to the CPU address when data is written to the flash memory using a parallel programmer. When a general programmer is used for writing/erasing, this address is used for writing/erasing.

25.3 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, 1, 2, 3, 4 and 5, 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 25.3-1 “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 25.3-1 Flash memory control signals

MB90F34X				MBM29LV200
Pin number		Normal function	Flash memory mode	
LQFP	QFP			
3	5	P30	AQ16	A15
4	6	P31	$\overline{CE}$	$\overline{CE}$
5	7	P32	$\overline{OE}$	$\overline{OE}$
6	8	P33	$\overline{WE}$	$\overline{WE}$
7(8)	9 (10)	P34 (P35)	AQ17 (AQ18)	A16
9	11	P36	$\overline{BYTE}$	$\overline{BYTE}$
10	12	P37	RY/ $\overline{BY}$	RY/ $\overline{BY}$
16 to 19	18 to 21	P42 to P45	AQ8 to AQ11	A7 to A10
20 to 21	22 to 23	P46 to P47	AQ12 to AQ13	A11 to A12
22 to 23	24 to 25	P50 to P51	AQ14 to AQ15	A13 to A14
51	53	MD0	MDO	A9 (V _{ID})
50	52	MD1	MD1	$\overline{RESET}$ (V _{ID})
49	51	MD2	MD2	OE (V _{ID})
75 to 82	77 to 84	P00 to P07	DQ0 to DQ7	DQ0 to DQ7
83 to 86	85 to 88	P10 to P13	DQ8 to DQ11	DQ8 to DQ11
52	54	$\overline{RST}$	$\overline{RESET}$	$\overline{RESET}$
87, 92 to 94	89, 94 to 96	P14 to P17	DQ12 to DQ15	DQ12 to DQ15
99 to 2, 95 to 98	1 to 4, 97 to 100	P20 to P27	AQ0 to AQ7	A-1, A0 to A6

25.4 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)

Figure 25.4-1 Flash memory control status register (FMCS)

FMCS	7	6	5	4	3	2	1	0	Initial value
Address: 0000AE _H	INTE	RDYINT	WE	RDY	Reserved	Reserved	Reserved	Reserved	000X0000 _B
	(R/W)	(R/W)	(R/W)	(R)	-	-	-	-	

● 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 25.5 “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 25.5 “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).

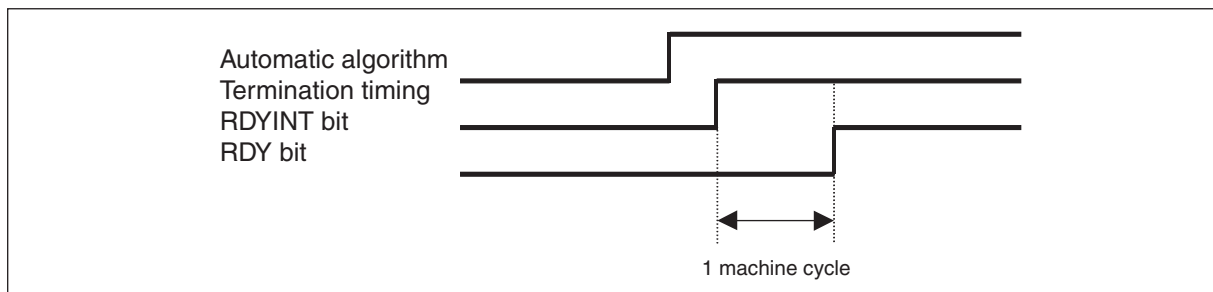
[Bits 3 to 0] Reserved bits

These bits are reserved for testing. During regular use, they should always be set to 0.

Note:

- This register can be accessed only in byte-access mode.
- 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.

Figure 25.4-2 Transitions of the RDYINT and RDY bits



25.5 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 25.5-1 “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 25.5-1 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, FC, FB, FA and F9 for 0.5M/1M/2M/3M/4M-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 25.2 “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 25.5-1 “Command sequence table” is used to know the state of sector protection. When using the Auto-select command, set the address as follows.

Table 25.5-2 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".

25.6 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 25.5-1 “Command sequence table” in Section 25.5 “Starting the Flash Memory Automatic Algorithm”. Table 25.6-1 “Bit assignments of hardware sequence flags” lists the bit assignments of the hardware sequence flags.

Table 25.6-1 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 25.6-2 “Hardware sequence flag functions” lists the functions of the hardware sequence flags.

Table 25.6-2 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	*

*: If the DQ5 outputs "1" (exceed the timing limit), successive reads from a writing or erasing sector cause DQ2 to toggle. DQ2 does not toggle when the successive reads are executed from other sectors.

25.6.1 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 25.6-3 "Data polling flag state transitions (state change for normal operation)" and Table 25.6-4 "Data polling flag state transitions (state change for abnormal operation)" list the state transitions of the data polling flag.

Table 25.6-3 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 25.6-4 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.

25.6.2 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 25.6-5 "Toggle bit flag state transitions (state change for normal operation)" and Table 25.6-6 "Toggle bit flag state transitions (state change for abnormal operation)" list the state transitions of the toggle bit flag.

Table 25.6-5 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 25.6-6 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.

25.6.3 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 25.6-7 "Timing limit exceeded flag state transitions (state change for normal operation)" and Table 25.6-8 "Timing limit exceeded bit flag state transitions (state change for abnormal operation)" list the state transitions of the timing limit exceeded flag.

Table 25.6-7 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 25.6-8 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.

25.6.4 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 25.6-9 "Sector erase timer flag state transitions (state change for normal operation)" and Table 25.6-10 "Sector erase timer flag state transitions (state change for abnormal operation)" list the state transitions of the sector erase timer flag.

Table 25.6-9 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 25.6-10 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.

- Read access during 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. If this address does not belong to the sector being erased, the flash memory outputs bit 3 (DATA:3) of the corresponding memory value.

25.6.5 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 25.6-11 "Toggle bit-2 flag state transitions (state change for normal operation)" and Table 25.6-12 "Toggle bit-2 flag state transitions (state change for abnormal operation)" list the state transitions of the toggle bit flag.

Table 25.6-11 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 25.6-12 Toggle bit-2 flag state transitions (state change for abnormal operation)

Operating state	Write	Chip/sector erase
DQ2	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 100μs, and then returns to the read/reset mode without writing the data.

25.7 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 25.5-1 "Command sequence table" in Section 25.5 "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

25.7.1 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 25.5-1 "Command sequence table" in Section 25.5 "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.

25.7.2 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 25.5-1 "Command sequence table" in Section 25.5 "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.

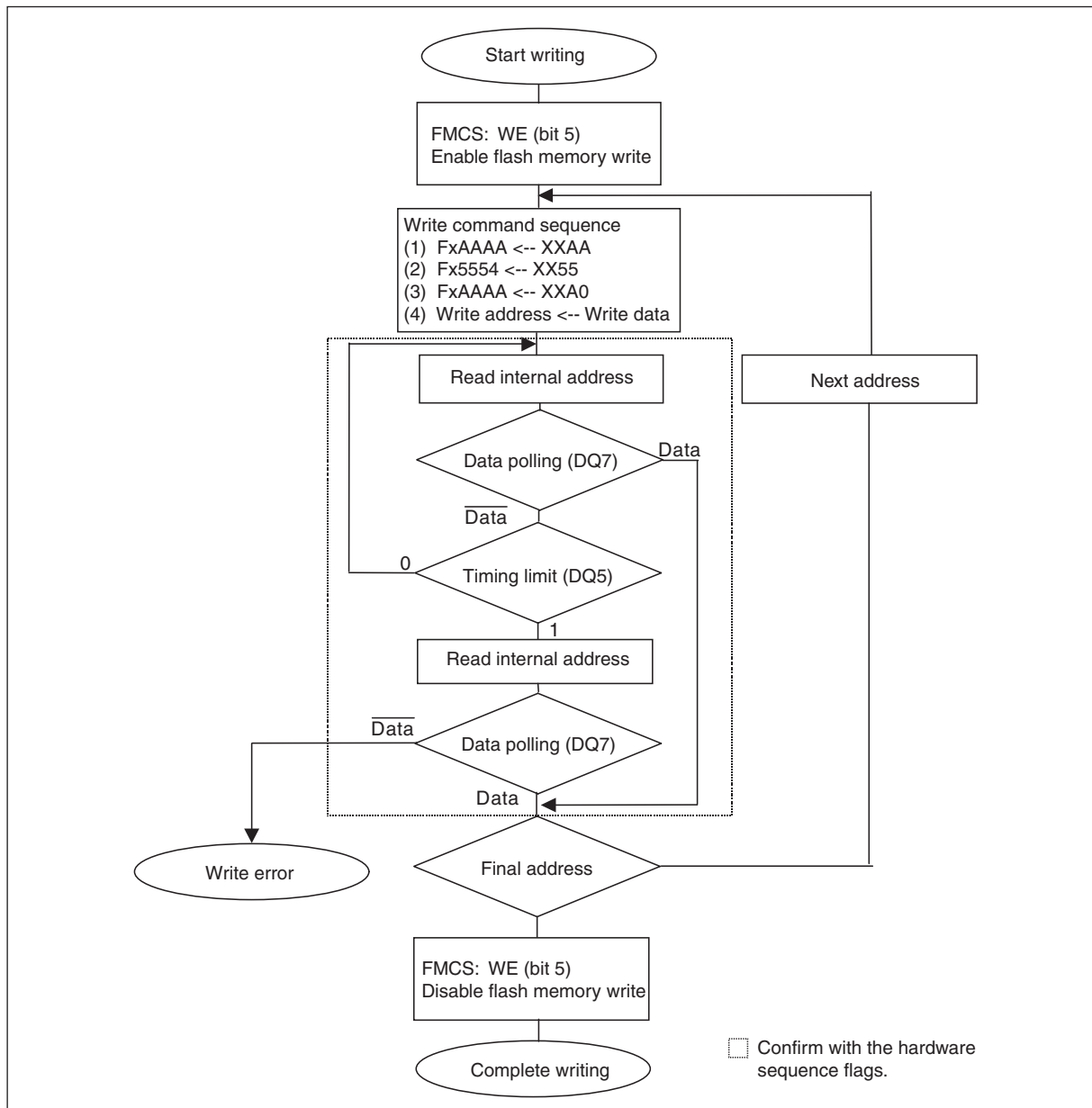
■ Writing to the flash memory

Figure 25.7-1 "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 25.6 "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 25.7-1 Example of the flash memory write procedure

25.7.3 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 25.5-1 "Command sequence table" in Section 25.5 "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.

25.7.4 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 25.5-1 "Command sequence table" in Section 25.5 "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.

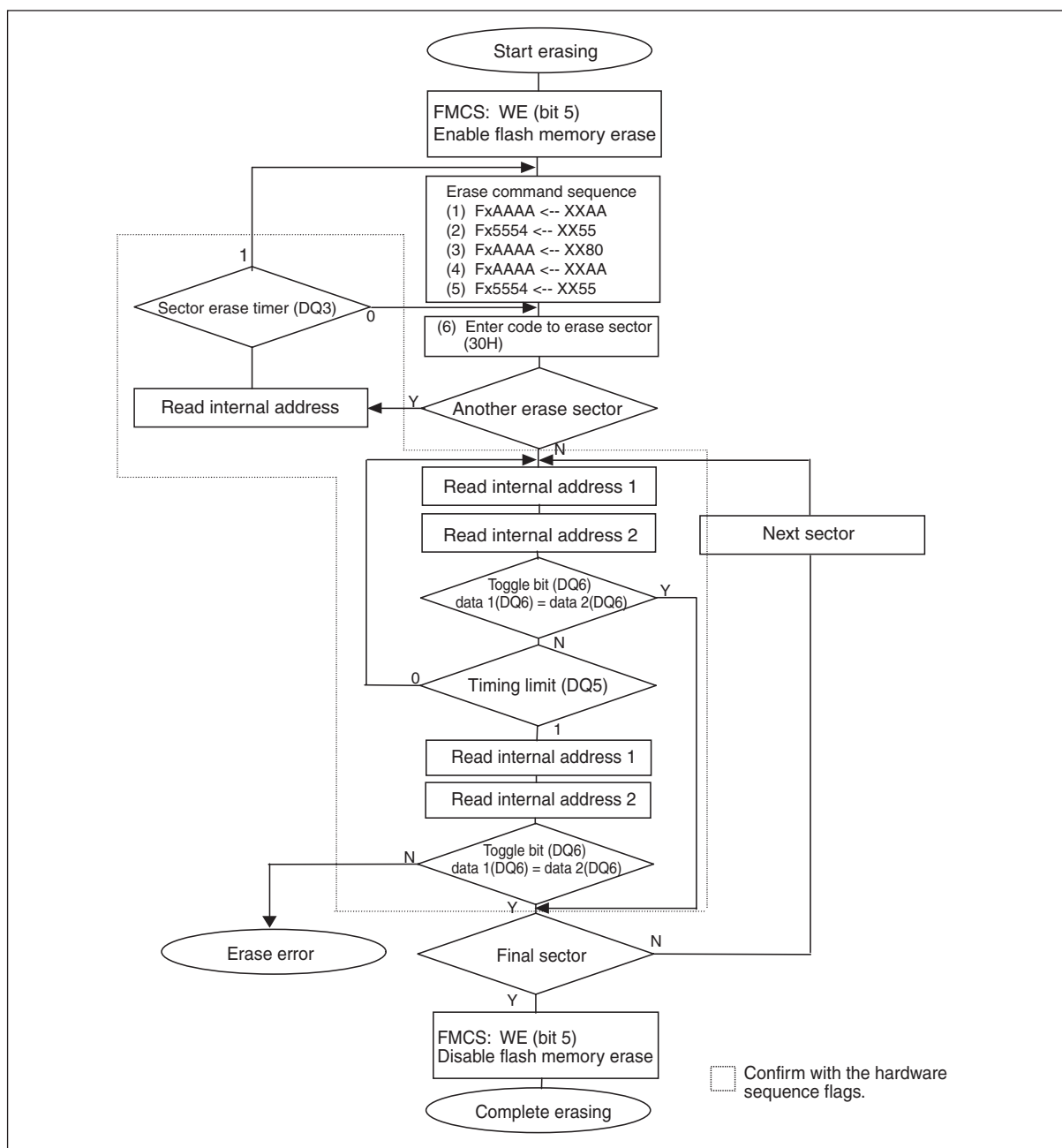
■ Erasing sectors in the flash memory

The hardware sequence flags (see Section 25.6 "Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 25.7-2 "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 25.7-2 Example of the flash memory sector erase procedure

25.7.5 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 25.5-1 "Command sequence table" in Section 25.5 "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.

25.7.6 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 25.5-1 "Command sequence table" in Section 25.5 "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.

25.8 Notes on using 0.5M/1M/2M/3M/4M-bit Flash Memory

This section contains notes on using 0.5M/1M/2M/3M/4M-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 and watchdog timer reset

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 (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.

25.9 Flash Security Feature

Flash Security Feature provides possibilities to protect the content of the flash memory from being read from external pins. This function is available for all flash devices except for MB90F346A(S), MB90F346CA(S).

■ Abstract

By writing the protection code of "01H" to one predefined address of the flash memory(1M-bit flash memory:FE0001H, 2M-bit flash memory:FC0001H, 3M-bit flash memory:F90001H), access to the flash memory is restricted. Once the flash memory is protected, performing the chip erase operation only can unlock the function. Otherwise read/write access to the flash memory from the external pins is not possible.

This function is suitable for applications requiring security of self-containing and data stored in the flash memory.

■ How to enable the Flash Security Feature

After writing the code "01H" to the predefined address of each flash memory(1M-bit flash memory:FE0001_H, 2M-bit flash memory:FC0001_H, 3M-bit flash memory:F90001_H), the following external reset or power-on enables the Flash Security Feature.

■ How to disable the Flash Security Feature

Performing the chip erase operation.

■ Behavior under the Flash Security Feature

Read operation: invalid data read

Write operation: ignored

■ 5. others

(1) About configuration of the standard parallel programmer, please follow to the specification of parallel programmer.

(2) Writing the protection code at the last of flash memory programming is recommended, in order to prevent the device from enabling the Flash Security Feature accidentally.

25.10 Example of Programming 1M-bit Flash Memory

This section presents a programming example of 1M-bit flash memory.

■ Programming example of 1M-bit flash memory

Flash Memory Sample Program

```

NAME    FLASHWE
TITLE   FLASHWE
;-----
;1M-bit-FLASH test program
;
;1: Transmits the program (address: FFA000H, sector: SA3) from FLASH to RAM
;   (address: 001500H).
;2: Executes the program on RAM.
;3: Writes the PDR1 value to FLASH (address: FF0000H, sector: SA1).
;4: Reads the written value (address: FF0000H, 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    DDRO,#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 (FFA000H)" to RAM (1500H address)
;    ;////////////////////////////////////
    MOVW   A,#1500H      ;Transfer destination RAM area
    MOVW   A,#0A000H      ;Transfer source address (program position)
    MOVW   RW0,#100H      ;Number of bytes to be transferred
    MOVS   ADB,PCB        ;Transfer of 100H from FFA000H to 001500H
    CALLP  001500H        ;Jump to the address containing the transferred
;                           program
;    ;////////////////////////////////////
;    Data output
;    ;////////////////////////////////////
OUT    MOV    A,#0FFH
        MOV    ADB,A
        MOVW   RW2,#0000H
        MOVW   A,@RW2+00
        MOV    PDR2,A
END     JMP    *
CODE    ENDS
;////////////////////////////////////
;FLASH write erase program (SA3)
;////////////////////////////////////
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 FF:0000
        MOV    A,#00H        ;DTB modification
        MOV    DTB,A         ;Bank specification for @RW0
        MOV    A,#0FFH       ;ADB modification 1
        MOV    ADB,A         ;Bank specification for write mode specification
;                                   address
;    MOV    PDR3,#00H        ;Switch initialization
        MOV    DDR3,#00H
;
WAIT1   BBC     PDR3:0,WAIT1    ;PDR3: 0(write start at high level)
;

```

```

;////////////////////////////////////////
;Write (SA1)
;////////////////////////////////////////
        MOV     A,PDR1
        MOVW    @RW0+00,A           ;PDR1 data allocation to RAM
        MOV     FMCS,#20H           ;Write mode setting
        MOVW    ADB:COMADR1,#00AAH  ;Flash write command 1
        MOVW    ADB:COMADR2,#0055H  ;Flash write command 2
        MOVW    ADB:COMADR1,#00A0H  ;Flash write command 3
;
        MOVW    A,@RW0+00           ;Input data (RW0) write to flash memory (RW2)
        MOVW    @RW2+00,A
WRITE    ;Wait time check
;
;////////////////////////////////////////
;
; ERROR when the time limit excess check flag is set and toggle operation is
; in progress
;
;////////////////////////////////////////
        MOV     A,@RW2+00
        AND     A,#20H               ;DQ5 time limit check
        BZ      NTOW                 ;Time limit over
        MOVW    A,@RW2+00            ;AH
        MOVW    A,@RW2+00            ;AL
        XORW    A                    ;XOR of AH and AL (1 when the values differ)
        AND     A,#40H               ;Is the DQ6 toggle bit different?
        BNZ     ERROR                ;To ERROR when the DQ6 toggle bit is different
;
; Write termination check (FMCS-RDY)
;
;////////////////////////////////////////
;
; NTOW
        MOV     A,FMCS
        AND     A,#10H               ;Extraction of FMCS RDY bit (bit 4)
        BZ      WRITE                ;End of write?
        MOV     FMCS,#00H            ;Write mode release
;
; Write data output
;
;////////////////////////////////////////
        MOVW    RW2,#0000H           ;Write data output
        MOVW    A,@RW2+00
        MOV     PDR2,A
;
WAIT2    BBC     PDR3:1,WAIT2         ;PDR3: 1(sector erase start at high level)
;
;////////////////////////////////////////
;Sector erase (SA1)
;////////////////////////////////////////
        MOV     @RW2+00,#0000H       ;Address initialization
        MOV     FMCS,#20H            ;Erase mode setting
        MOVW    ADB:COMADR1,#00AAH  ;Flash erase command 1
        MOVW    ADB:COMADR2,#0055H  ;Flash erase command 2
        MOVW    ADB:COMADR1,#0080H  ;Flash erase command 3
        MOVW    ADB:COMADR1,#00AAH  ;Flash erase command 4
        MOVW    ADB:COMADR2,#0055H  ;Flash erase command 5
        MOV     @RW2+00,#0030H       ;Issuance of erase command 6 to the sector
                                         to be erased
ELS      ;Wait time check
;
;////////////////////////////////////////
;
; ERROR when the time limit excess check flag is set and toggle operation is
; in progress
;
;////////////////////////////////////////
        MOVW    A,@RW2+00
        AND     A,#20H               ;DQ5 time limit check
        BZ      NTOE                 ;Time limit over
        MOVW    A,@RW2+00            ;AH High and Low are alternately output from
        MOVW    A,@RW2+00            ;AL DQ6 per read during write operation.
        XORW    A                    ;XOR of AH and AL (If the DQ6 value differs,
;                                         write operation is in progress (1)).
        AND     A,#40H               ;Is the DQ6 toggle bit High?
        BNZ     ERROR                ;ERROR when the DQ6 toggle bit is High

```

```

;          //////////////////////////////////////////
;          Erase termination check (FMCS-RDY)
;          //////////////////////////////////////////
NTOE      MOV      A,FMCS          ;
          AND      A,#10H          ;Extraction of FMCS RDY bit (bi
          BZ       ELS            ;End of sector erase?
          MOV      FMCS,#00H       ;FLASH erase mode release
          RETP      ;Return to the main program
;          //////////////////////////////////////////
;Error
;          //////////////////////////////////////////
ERROR     MOV      FMCS,#00H       ;FLASH mode release
          MOV      PDR0,#0FFH      ;Error handling check
          MOV      ADB:COMADR1,#0F0H ;Reset command (read is enable
          RETP      ;Return to the main program
RAMPRG    ENDS
;          //////////////////////////////////////////
VECT      CSEG      ABS=0FFH
          ORG      0FFDCH
          DSL      START
          DB       00H
VECT      ENDS
;

```


CHAPTER 26

EXAMPLES OF MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) SERIAL PROGRAMMING CONNECTION

This chapter provides examples of F²MC-16LX MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) serial programming connection.

- 26.1 “Basic Configuration of F²MC-16LX MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) Serial Programming Connection”
- 26.2 “Example of Serial Programming Connection (User Power Supply Used)”
- 26.3 “Example of Serial Programming Connection (Power Supplied from the Programmer)”
- 26.4 “Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)”
- 26.5 “Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)”

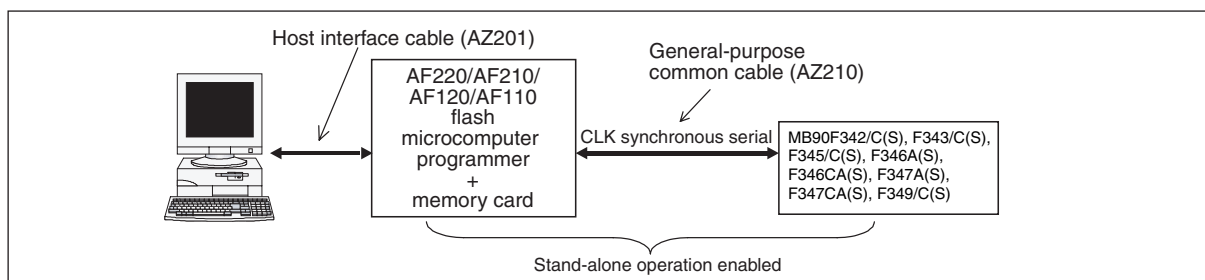
26.1 Basic Configuration of MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) Serial Programming Connection

The MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) supports flash ROM serial onboard programming (Fujitsu standard). This section describes the specifications.

■ Basic configuration of MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) serial programming connection

The AF220/AF210/AF120/AF110 flash microcomputer programmer from Yokogawa Digital Computer Corporation is used for Fujitsu standard serial onboard programming.

Figure 26.1-1 Fujitsu standard serial onboard programming of MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S)



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 26.1-1 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, because the oscillation clock frequency becomes the internal operation clock signal, the oscillator used for serial reprogramming is 3 MHz to 16 MHz.															
P00, P01	programming activation pins	<table border="1"> <thead> <tr> <th>P01</th><th>P00</th><th>function</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>machine clock = 16 MHz Baud rate = 19200bps asynchronous</td></tr> <tr> <td>0</td><td>1</td><td>machine clock = 16 MHz Baud rate = 19200 bps asynchronous</td></tr> <tr> <td>1</td><td>0</td><td>clock synchronous mode</td></tr> <tr> <td>1</td><td>1</td><td>reserved</td></tr> </tbody> </table>	P01	P00	function	0	0	machine clock = 16 MHz Baud rate = 19200bps asynchronous	0	1	machine clock = 16 MHz Baud rate = 19200 bps asynchronous	1	0	clock synchronous mode	1	1	reserved
P01	P00	function															
0	0	machine clock = 16 MHz Baud rate = 19200bps asynchronous															
0	1	machine clock = 16 MHz Baud rate = 19200 bps asynchronous															
1	0	clock synchronous mode															
1	1	reserved															
$\overline{\text{RST}}$	Reset pin	-															
SIN0	Serial data input pin	Serial input-output is used.															
SOT0	Serial data output pin																
SCK0	Serial clock signal input pin																
C	C pin	This external capacitor pin is used to stabilize the power supply. Connect a ceramic capacitor of approximately 0.1 or more μF to the outside.															
V_{CC}	Power voltage supply pin	If the programming voltage ($5\text{ V} \pm 10\%$) 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.															

Even if the P00, P01, SIN0, SOT0, and SCK0 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.

Sections 26.2 “Example of Serial Programming Connection (User Power Supply Used)” to 26.5 “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)

Figure 26.1-2 Connecting user circuitry for serial programming

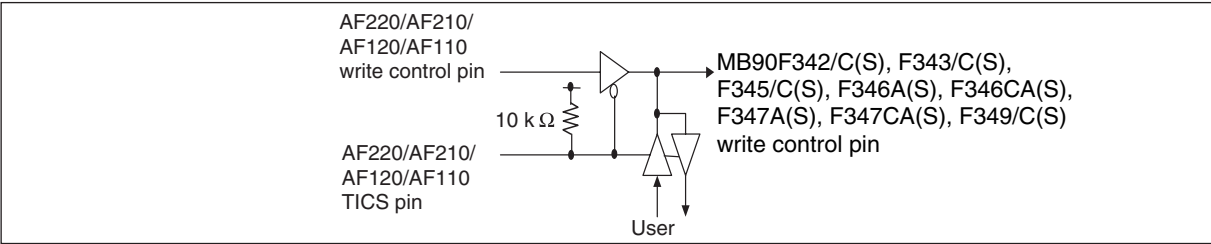


Table 26.1-2 System configuration of flash microcomputer programmers (manufactured by Yokogawa Digital Computer Corporation)

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)-42-333-6222

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 26.2 “Example of Serial Programming Connection (User Power Supply Used)” and 26.3 “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 MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S). Set an appropriate serial clock input frequency in the flash microcomputer programmer according to the oscillating clock frequency in use.

$$f_{SC} = 0.125 * f_{OSC},$$

where f_{SC} is the serial clock frequency and f_{OSC} is the oscillating clock frequency.

Table 26.1-3 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

*: External clock only.

26.2 Example of Serial Programming Connection (User Power Supply Used)

Figure 26.2-1 “Example of serial programming connection for MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) internal vector modes (user power supply used)”

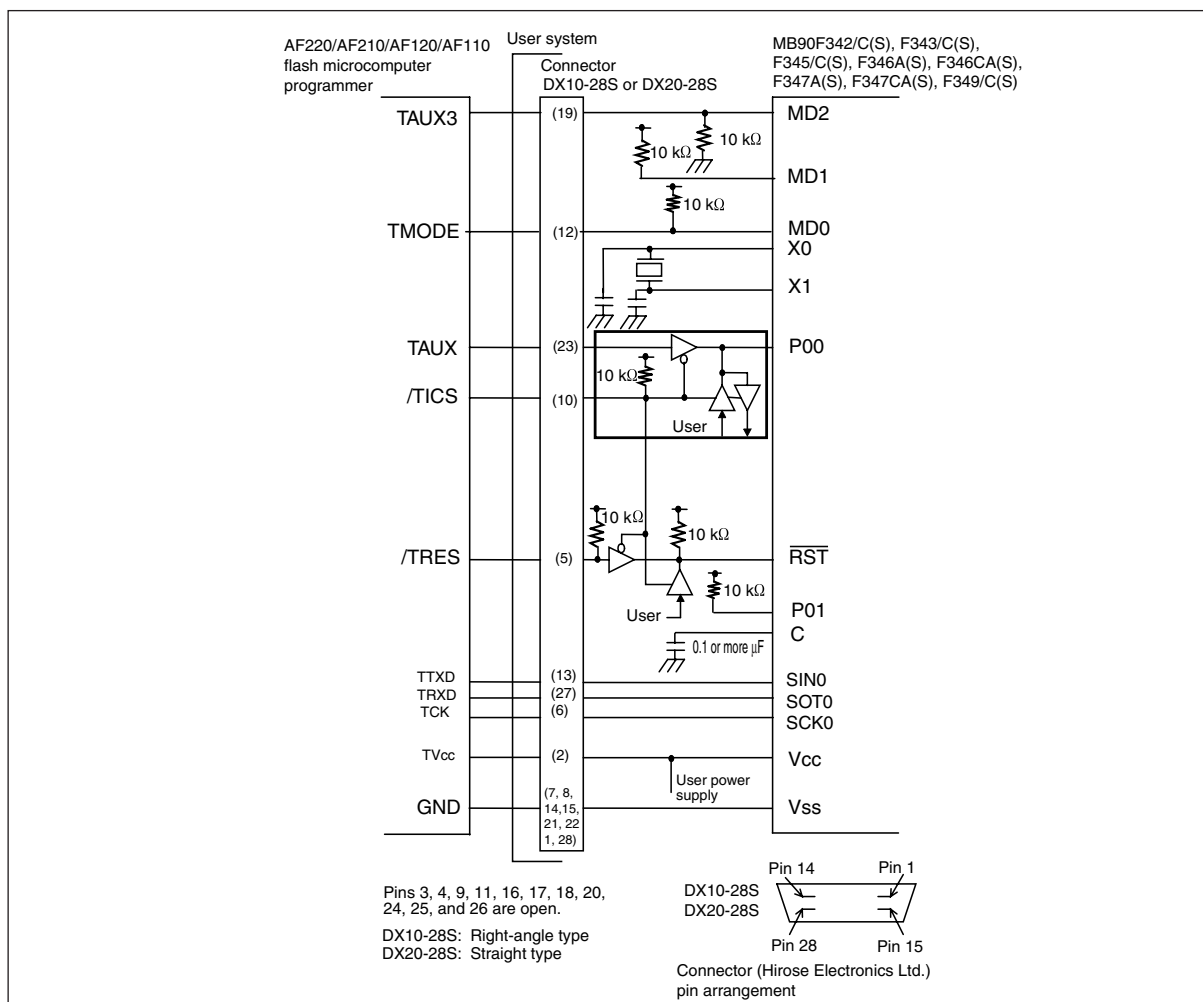
is an example of a serial programming connection for internal vector modes (single-chip mode) when the user power supply is used.

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_B.

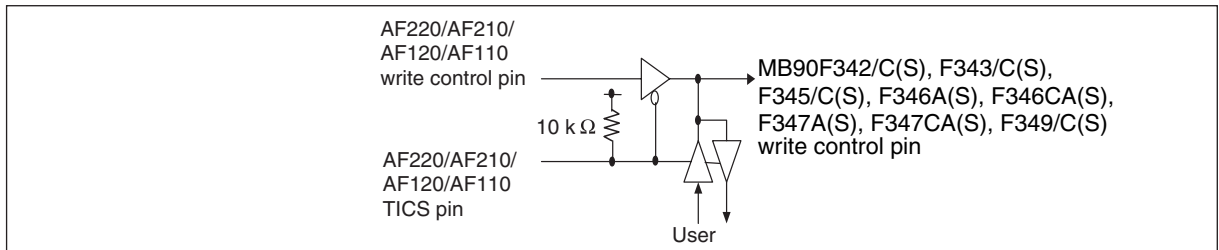
■ Example of serial programming connection (user power supply used)

Figure 26.2-1 Example of serial programming connection for MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) internal vector modes (user power supply used)



- Even if the SIN0, SOT0, and SCK0 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.

Figure 26.2-2 Connecting user circuitry (detail)



26.3 Example of Serial Programming Connection (Power Supplied from the Programmer)

Figure 26.3-1 “Example of serial programming connection for MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) internal vector modes (power supplied from the programmer)”

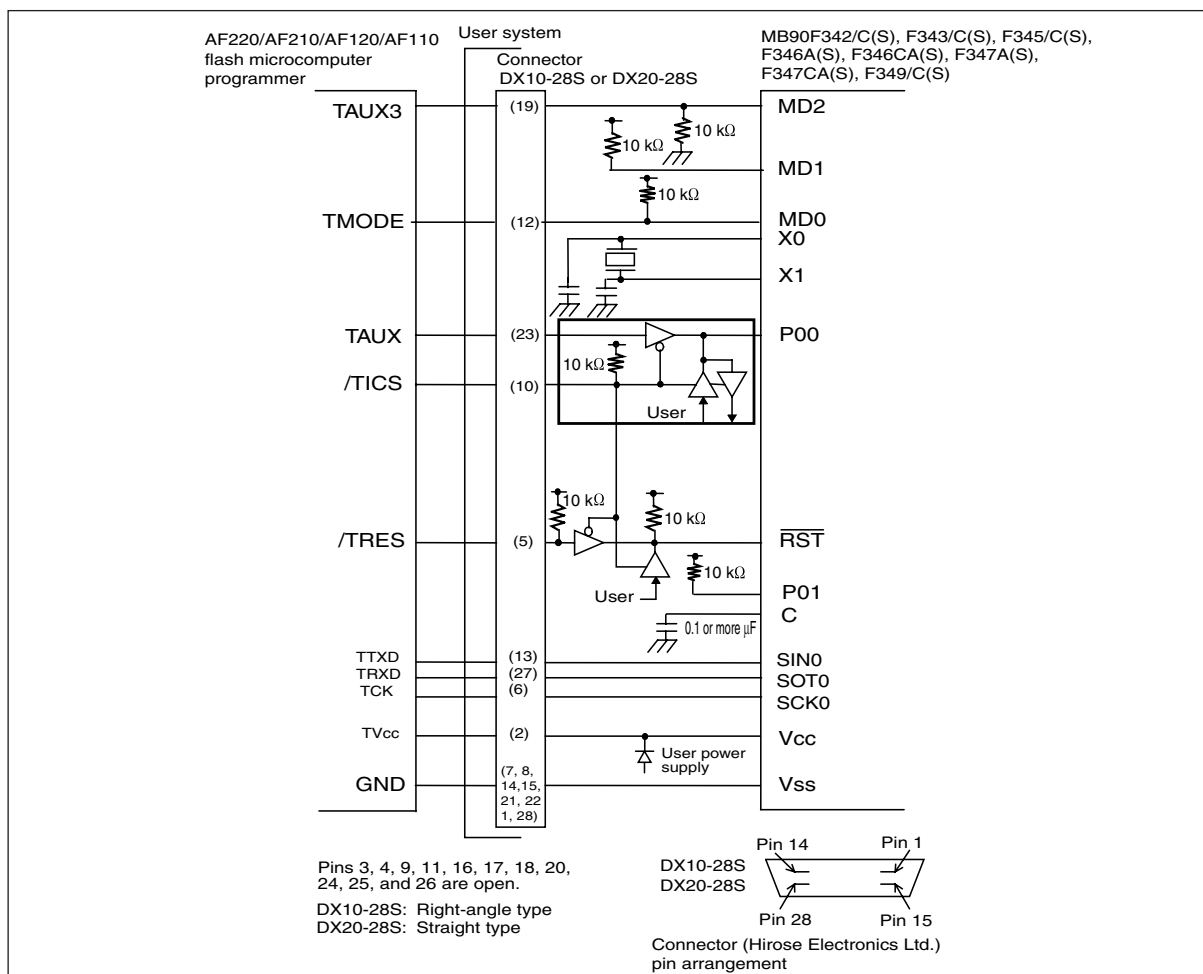
is an example of a serial programming connection for internal vector modes (single-chip mode) when power is supplied from the programmer.

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_B.

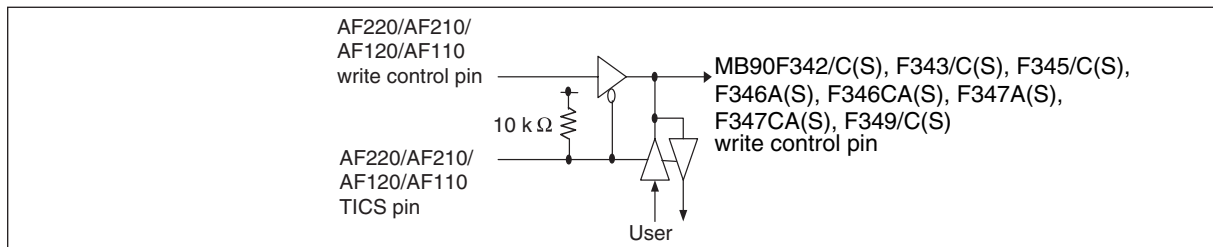
■ Example of serial programming connection (power supplied from the programmer)

Figure 26.3-1 Example of serial programming connection for MB90F342/C(S), F343/C(S), F345/C(S), F346A(S), F346CA(S), F347A(S), F347CA(S), F349/C(S) internal vector modes (power supplied from the programmer)



- Even if the SIN0, SOT0, and SCK0 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.

Figure 26.3-2 Connecting user circuitry (detail)



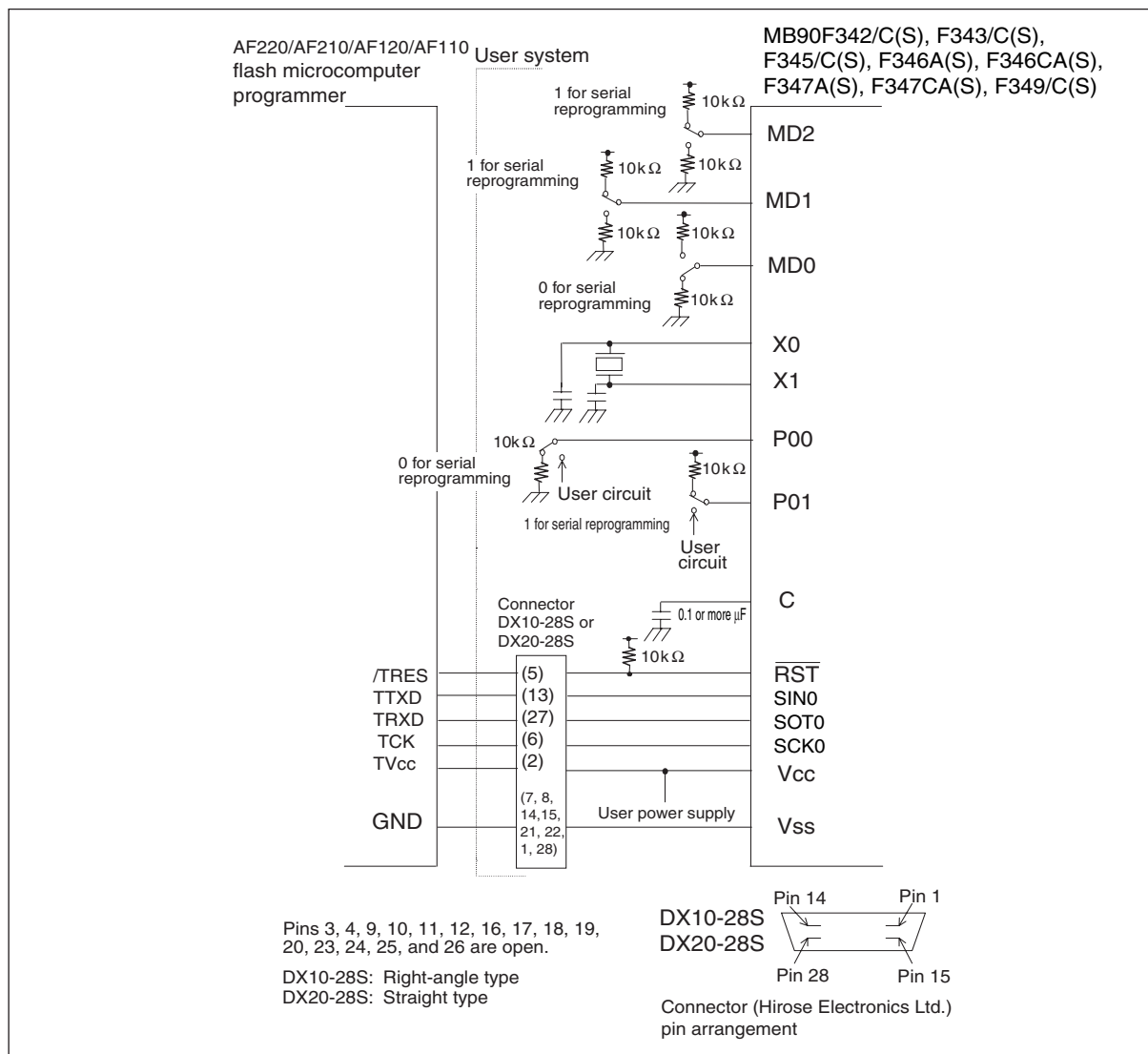
26.4 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)

Figure 26.4-1 “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_B.

■ Example of minimum connection to the flash microcomputer programmer (user power supply used)

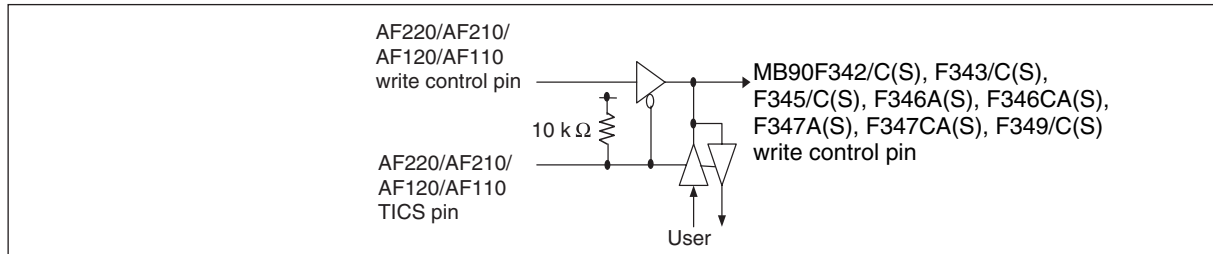
For a flash memory write, the MD2, MD1, MD0, and P00 pins and flash microcomputer programmer need not be connected if the pins are set as described below.

Figure 26.4-1 Example of minimum connection to the flash microcomputer programmer (user power supply used)



- Even if the SIN0, SOT0, and SCK0 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.
- Connect the AF220/AF210/AF120/AF110 while the user power is off.

Figure 26.4-2 Connecting user circuitry (detail)



26.5 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Programmer)

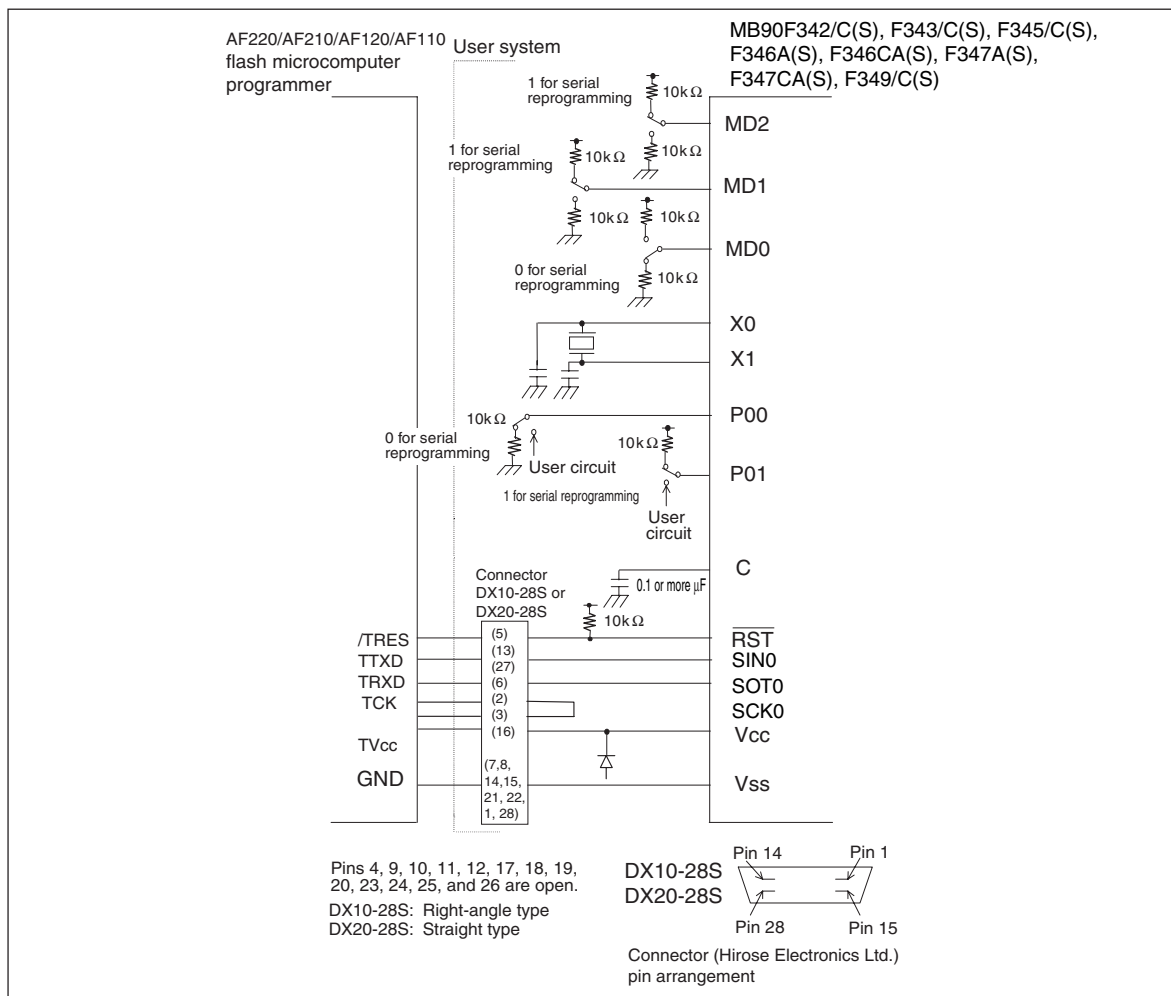
Figure 26.5-1 “Example of minimum connection to the flash microcomputer programmer (power supplied from the programmer)” is an example of the minimum connection to the flash microcomputer programmer when power is supplied from the Programmer.

Serial reprogramming mode: MD2, MD1, MD0 = 110_B.

■ **Example of minimum connection to the flash microcomputer programmer (power supplied from the programmer)**

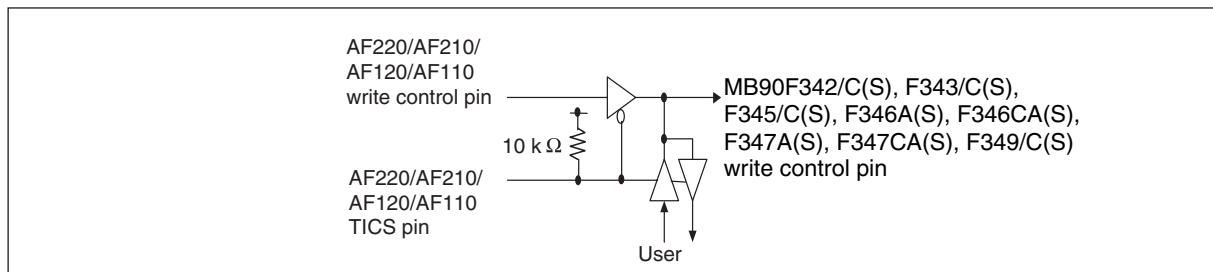
For a flash memory write, the MD2, MD1, MD0, and P00 pins and flash microcomputer programmer need not be connected if the pins are set as described below.

Figure 26.5-1 Example of minimum connection to the flash microcomputer programmer (power supplied from the programmer)



- Even if the SIN0, SOT0, and SCK0 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial programming.
- 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.

Figure 26.5-2 Connecting user circuitry (detail)



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 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 (00XX addresses)

Table A-1 I/O map (1/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
00 _H	Port 0 data register	PDR0	R/W	Port 0	XXXXXXXX
01 _H	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX
02 _H	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX
03 _H	Port 3 data register	PDR3	R/W	Port 3	XXXXXXXX
04 _H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX
05 _H	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX
06 _H	Port 6 data register	PDR6	R/W	Port 6	XXXXXXXX
07 _H	Port 7 data register	PDR7	R/W	Port 7	XXXXXXXX
08 _H	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX
09 _H	Port 9 data register	PDR9	R/W	Port 9	XXXXXXXX
0A _H	Port A data register	PDRA	R/W	Port A	XXXXXXXX
0B _H	Analog Input Enable Port 5	ADER5	R/W	Port 5, A/D	11111111
0C _H	Analog Input Enable Port 6	ADER6	R/W	Port 6, A/D	11111111
0D _H	Analog Input Enable Port 7	ADER7	R/W	Port 7, A/D	11111111
0E _H	Input level select register0	ILSR0	R/W	Ports	XXXXXXXX
0F _H	Input level select register1	ILSR1	R/W	Ports	XXXXXXXX
10 _H	Port 0 direction register	DDR0	R/W	Port 0	00000000
11 _H	Port 1 direction register	DDR1	R/W	Port 1	00000000
12 _H	Port 2 direction register	DDR2	R/W	Port 2	00000000
13 _H	Port 3 direction register	DDR3	R/W	Port 3	00000000
14 _H	Port 4 direction register	DDR4	R/W	Port 4	00000000
15 _H	Port 5 direction register	DDR5	R/W	Port 5	00000000

Table A-1 I/O map (2/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
16 _H	Port 6 direction register	DDR6	R/W	Port 6	00000000
17 _H	Port 7 direction register	DDR7	R/W	Port 7	00000000
18 _H	Port 8 direction register	DDR8	R/W	Port 8	00000000
19 _H	Port 9 direction register	DDR9	R/W	Port 9	00000000
1A _H	Port A direction register	DDRA	R/W	Port A	00000100
16 _H	Port 6 direction register	DDR6	R/W	Port 6	00000000
1B _H	Reserved				
1C _H	Port 0 Pull-up control register	PUCR0	R/W	Port 0	00000000
1D _H	Port 1 Pull-up control register	PUCR1	R/W	Port 1	00000000
1E _H	Port 2 Pull-up control register	PUCR2	R/W	Port 2	00000000
1F _H	Port 3 Pull-up control register	PUCR3	W, R/W	Port 3	00000000
20 _H	Serial Mode Register	SMR0	W,R/W	UART0	00000000
21 _H	Serial Control Register	SCR0	W,R/W		00000000
22 _H	Reception/Transmission Data Register	RDR0/TDR0	R/W		00000000
23 _H	Serial Status Register	SSR0	R,R/W		00001000
24 _H	Extended Communication Control Reg.	ECCR0	R,W,R/W		000000XX
25 _H	Extended Status/Control Register	ESCR0	R/W		00000100
26 _H	Baud Rate Register 0	BGR00	R/W		00000000
27 _H	Baud Rate Register 1	BGR01	R/W		00000000
28 _H	Serial Mode Register	SMR1	W,R/W	UART1	00000000
29 _H	Serial Control Register	SCR1	W,R/W		00000000
2A _H	Reception/Transmission Data Register	RDR1/TDR1	R/W		00000000
2B _H	Serial Status Register	SSR1	R,R/W		00001000
2C _H	Extended Communication Control Reg.	ECCR1	R,W,R/W		000000XX
2D _H	Extended Status/Control Register	ESCR1	R/W		00000100
2E _H	Baud Rate Register 0	BGR10	R/W		00000000

Table A-1 I/O map (3/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
2F _H	Baud Rate Register 1	BGR11	R/W	UART1	00000000
30 _H	PPG0 operation mode control register	PPGC0	W,R/W	16-bit Programable Pulse Generator 0/1	0X000XX1
31 _H	PPG1 operation mode control register	PPGC1	W,R/W		0X000001
32 _H	PPG0 and PPG1 clock select register	PPG01	R/W		000000X0
33 _H	Reserved				
34 _H	PPG2 operation mode control register	PPGC2	W,R/W	16-bit Programable Pulse Generator 2/3	0X000XX1
35 _H	PPG3 operation mode control register	PPGC3	W,R/W		0X000001
36 _H	PPG2 and PPG3 clock select register	PPG23	R/W		000000X0
37 _H	Reserved				
38 _H	PPG4 operation mode control register	PPGC4	W,R/W	16-bit Programable Pulse Generator 4/5	0X000XX1
39 _H	PPG5 operation mode control register	PPGC5	W,R/W		0X000001
3A _H	PPG4 and PPG5 clock select register	PPG45	R/W		000000X0
3B _H	ROM Correction Control Status 1	PACSR1	R/W	ROM Correction 1	00000000
3C _H	PPG6 operation mode control register	PPGC6	W,R/W	16-bit Programable Pulse Generator 6/7	0X000XX1
3D _H	PPG7 operation mode control register	PPGC7	W,R/W		0X000001
3E _H	PPG6 and PPG7 clock select register	PPG67	R/W		000000X0
3F _H	Reserved				
40 _H	PPG8 operation mode control register	PPGC8	W,R/W	16-bit Programable Pulse Generator 8/9	0X000XX1
41 _H	PPG9 operation mode control register	PPGC9	W,R/W		0X000001
42 _H	PPG8 and PPG9 clock select register	PPG89	R/W		000000X0
43 _H	Reserved				

Table A-1 I/O map (4/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
43 _H	Reserved				
44 _H	PPGA operation mode control register	PPGCA	W,R/W	16-bit Programable Pulse Generator A/B	0X000XX1
45 _H	PPGB operation mode control register	PPGCB	W,R/W		0X000001
46 _H	PPGA and PPGB clock select register	PPGAB	R/W		000000X0
47 _H	Reserved				
48 _H	PPGC operation mode control register	PPGCC	W,R/W	16-bit Programable Pulse Generator C/D	0X000XX1
49 _H	PPGD operation mode control register	PPGCD	W,R/W		0X000001
4A _H	PPGC and PPGD clock select register	PPGCD	R/W		000000X0
4B _H	Reserved				
4C _H	PPGE operation mode control register	PPGCE	W,R/W	16-bit Programable Pulse Generator E/F	0X000XX1
4D _H	PPGF operation mode control register	PPGCF	W,R/W		0X000001
4E _H	PPGE and PPGF clock select register	PPGEF	R/W		000000X0
4F _H	Reserved				
50 _H	Input Capture Control Status 0/1	ICS01	R/W	Input Capture 0/1	00000000
51 _H	Input Capture Edge 0/1	ICE01	R/W		XXX0X0XX
52 _H	Input Capture Control Status 2/3	ICS23	R/W	Input Capture 2/3	00000000
53 _H	Input Capture Edge 2/3	ICE23	R/W		XXXXXXXXXX
54 _H	Input Capture Control Status 4/5	ICS45	R/W	Input Capture 4/5	00000000
55 _H	Input Capture Edge 4/5	ICE45	R/W		XXXXXXXXXX
56 _H	Input Capture Control Status 6/7	ICS67	R/W	Input Capture 6/7	00000000
57 _H	Input Capture Edge 6/7	ICE67	R/W		XXX000XX
58 _H	Output Compare Control Status 0	OCS0	R/W	Output Compare 0/1	0000XX00
59 _H	Output Compare Control Status 1	OCS1	R/W		0XX00000

Table A-1 I/O map (5/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
5A _H	Output Compare Control Status 2	OCS2	R/W	Output Compare 2/3	0000XX00
5B _H	Output Compare Control Status 3	OCS3	R/W		0XX00000
5C _H	Output Compare Control Status 4	OCS4	R/W	Output Compare 4/5	0000XX00
5D _H	Output Compare Control Status 5	OCS5	R/W		0XX00000
5E _H	Output Compare Control Status 6	OCS6	R/W	Output Compare 6/7	0000XX00
5F _H	Output Compare Control Status 7	OCS7	R/W		0XX00000
60 _H	Timer Control Status 0	TMCSR0	R/W	16-bit Reload Timer 0	00000000
61 _H	Timer Control Status 0	TMCSR0	R/W		XXXX0000
62 _H	Timer Control Status 1	TMCSR1	R/W	16-bit Reload Timer 1	00000000
63 _H	Timer Control Status 1	TMCSR1	R/W		XXXX0000
64 _H	Timer Control Status 2	TMCSR2	R/W	16-bit Reload Timer 2	00000000
65 _H	Timer Control Status 2	TMCSR2	R/W		XXXX0000
66 _H	Timer Control Status 3	TMCSR3	R/W	16-bit Reload Timer 3	00000000
67 _H	Timer Control Status 3	TMCSR3	R/W		XXXX0000
68 _H	A/D Control Status 0	ADCS0	R/W	A/D Converter	000XXXX0
69 _H	A/D Control Status 1	ADCS1	R/W		0000000X
6A _H	A/D Data 0	ADCR0	R		00000000
6B _H	A/D Data 1	ADCR1	R		XXXXXX00
6C _H	ADC Setting 0	ADSR0	R/W		00000000
6D	ADC Setting 1	ADSR1	R/W		00000000
6E _H	Reserved				
6F _H	ROM Mirror	ROMM	W	ROM Mirror	XXXXXXXX1
70 to 8F _H	Reserved for CAN Interface 0/1. Refer to section about CAN controller				
90 to 9A _H	Reserved				
9B _H	DMA Descriptor Channel Select	DCSR	R/W	DMA	00000000
9C _H	DMA Status L	DSRL	R/W		00000000
9D _H	DMA Status H	DSRH	R/W		00000000
9E _H	ROM Correction Control Status 0	PACSR0	R/W	ROM Correction 0	00000000

Table A-1 I/O map (6/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
9F _H	Delayed Interrupt/release	DIRR	R/W	Delayed Interrupt	XXXXXXXX0
A0 _H	Low-power Mode Control	LPMCR	W,R/W	Low Power Controller	00011000
A1 _H	Clock Selection	CKSCR	R,R/W	Low Power Controller	11111100
A2 to A3 _H	Reserved				
A4 _H	DMA Stop Status	DSSR	R/W	DMA	00000000
A5 _H	Automatic ready function select reg.	ARSR	W	External Memory Access	0011XX00
A6 _H	External address output control reg.	HACR	W		00000000
A7 _H	Bus control signal selection register	ECSR	W		0000000X
A8 _H	Watchdog Control	WDTC	R,W	Watchdog Timer	XXXXX111
A9 _H	Time Base Timer Control	TBTC	W,R/W	Time Base Timer	1XX00100
AA _H	Watch Timer Control register	WTC	R,R/W	Watch Timer	1X001000
AB _H	Reserved				
AC _H	DMA Enable L	DERL	R/W	DMA	00000000
AD _H	DMA Enable H	DERH	R/W		00000000
AE _H	Flash Control Status (Flash device only. Otherwise reserved)	FMCS	R,R/W	Flash Memory	000X0000
AF _H	Reserved				
B0 _H	Interrupt control register 00	ICR00	W,R/W	Interrupt controller	00000111
B1 _H	Interrupt control register 01	ICR01	W,R/W		00000111
B2 _H	Interrupt control register 02	ICR02	W,R/W		00000111
B3 _H	Interrupt control register 03	ICR03	W,R/W		00000111
B4 _H	Interrupt control register 04	ICR04	W,R/W		00000111
B5 _H	Interrupt control register 05	ICR05	W,R/W		00000111
B6 _H	Interrupt control register 06	ICR06	W,R/W		00000111
B7 _H	Interrupt control register 07	ICR07	W,R/W		00000111
B8 _H	Interrupt control register 08	ICR08	W,R/W		00000111

Table A-1 I/O map (7/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
B9 _H	Interrupt control register 09	ICR09	W,R/W	Interrupt controller	00000111
BA _H	Interrupt control register 10	ICR10	W,R/W		00000111
BB _H	Interrupt control register 11	ICR11	W,R/W		00000111
BC _H	Interrupt control register 12	ICR12	W,R/W		00000111
BD _H	Interrupt control register 13	ICR13	W,R/W		00000111
BE _H	Interrupt control register 14	ICR14	W,R/W		00000111
BF _H	Interrupt control register 15	ICR15	W,R/W		00000111
C0 _H	D/A Converter data 0	DAT0	R/W	D/A Converter	XXXXXXXXX
C1 _H	D/A Converter data 1	DAT1	R/W		XXXXXXXXX
C2 _H	D/A Control 0	DACR0	R/W		XXXXXXXX0
C3 _H	D/A Control 1	DACR1	R/W		XXXXXXXX0
C4 to C5 _H	Reserved				
C6 _H	External Interrupt Enable 0	ENIR0	R/W	External Interrupt 0	00000000
C7 _H	External Interrupt Request 0	EIRR0	R/W		XXXXXXXXX
C8 _H	External Interrupt Level 0	ELVR0	R/W		00000000
C9 _H	External Interrupt Level 0	ELVR0	R/W		00000000
CA _H	External Interrupt Enable 1	ENIR1	R/W	External Interrupt 1	00000000
CB _H	External Interrupt Request 1	EIRR1	R/W		XXXXXXXXX
CC _H	External Interrupt Level 1	ELVR1	R/W		00000000
CD _H	External Interrupt Level 1	ELVR1	R/W		00000000
CE _H	External Interrupt 1 Source Select	EISSR	R/W		00000000
CF _H	PLL/Subclock Control register	PSCCR	W	PLL	XXXX0000
D0 _H	DMA Buffer Addrss Pointer L	BAPL	R/W	DMA	XXXXXXXXX
D1 _H	DMA Buffer Addrss Pointer M	BAPM	R/W		XXXXXXXXX
D2 _H	DMA Buffer Addrss Pointer H	BAPH	R/W		XXXXXXXXX
D3 _H	DMA Control	DMACS	R/W		XXXXXXXXX
D4 _H	I/O Register Address Pointer L	IOAL	R/W		XXXXXXXXX
D5 _H	I/O Register Address Pointer H	IOAH	R/W		XXXXXXXXX

Table A-1 I/O map (8/8)

Address	Register	Abbreviation	Access	Peripheral	Initial value
D6 _H	Data Counter L	DCTL	R/W	DMA	XXXXXXXX
D7 _H	Data Counter H	DCTH	R/W		XXXXXXXX
D8 _H	Serial Mode Register	SMR2	W,R/W	UART2	00000000
D9 _H	Serial Control Register	SCR2	W,R/W		00000000
DA _H	Reception/Transmission Data Register	RDR2/TDR2	R/W		00000000
DB _H	Serial Status Register	SSR2	R,R/W		00001000
DC _H	Extended Communication Control Reg.	ECCR2	R,W,R/W		000000XX
DD _H	Extended Status/Control Register	ESCR2	R/W		00000100
DE _H	Baud Rate Register 0	BGR20	R/W		00000000
DF _H	Baud Rate Register 1	BGR21	R/W		00000000
E0 to EF _H	Reserved for CAN Interface 2. Refer to section about CAN controller				
F0 to FF _H	External				

■ I/O map (79XX - 7FXX addresses)

Table A-2 I/O map (79XX - 7FXX addresses) (1/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7900 _H	Reload L	PRLLO	R/W	16-bit Programable Pulse Generator 0/1	XXXXXXXX
7901 _H	Reload H	PRLH0	R/W		XXXXXXXX
7902 _H	Reload L	PRLLO1	R/W		XXXXXXXX
7903 _H	Reload H	PRLH1	R/W		XXXXXXXX
7904 _H	Reload L	PRLLO2	R/W	16-bit Programable Pulse Generator 2/3	XXXXXXXX
7905 _H	Reload H	PRLH2	R/W		XXXXXXXX
7906 _H	Reload L	PRLLO3	R/W		XXXXXXXX
7907 _H	Reload H	PRLH3	R/W		XXXXXXXX
7908 _H	Reload L	PRLLO4	R/W	16-bit Programable Pulse Generator 4/5	XXXXXXXX
7909 _H	Reload H	PRLH4	R/W		XXXXXXXX
790A _H	Reload L	PRLLO5	R/W		XXXXXXXX
790B _H	Reload H	PRLH5	R/W		XXXXXXXX
790C _H	Reload L	PRLLO6	R/W	16-bit Programable Pulse Generator 6/7	XXXXXXXX
790D _H	Reload H	PRLH6	R/W		XXXXXXXX
790E _H	Reload L	PRLLO7	R/W		XXXXXXXX
790F _H	Reload H	PRLH7	R/W		XXXXXXXX
7910 _H	Reload L	PRLLO8	R/W	16-bit Programable Pulse Generator 8/9	XXXXXXXX
7911 _H	Reload H	PRLH8	R/W		XXXXXXXX
7912 _H	Reload L	PRLLO9	R/W		XXXXXXXX
7913 _H	Reload H	PRLH9	R/W		XXXXXXXX
7914 _H	Reload L	PRLLOA	R/W	16-bit Programable Pulse Generator A/B	XXXXXXXX
7915 _H	Reload H	PRLHA	R/W		XXXXXXXX
7916 _H	Reload L	PRLLOB	R/W		XXXXXXXX
7917 _H	Reload H	PRLHB	R/W		XXXXXXXX
7918 _H	Reload L	PRLLOC	R/W	16-bit Programable Pulse Generator C/D	XXXXXXXX
7919 _H	Reload H	PRLHC	R/W		XXXXXXXX

Table A-2 I/O map (79XX - 7FXX addresses) (2/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
791A _H	Reload L	PRLLD	R/W	16-bit Programable Pulse Generator C/D	XXXXXXXX
791B _H	Reload H	PRLHD	R/W		XXXXXXXX
791C _H	Reload L	PRLLE	R/W	16-bit Programable Pulse Generator E/F	XXXXXXXX
791D _H	Reload H	PRLHE	R/W		XXXXXXXX
791E _H	Reload L	PRLLF	R/W		XXXXXXXX
791F _H	Reload H	PRLHF	R/W		XXXXXXXX
7920 _H	Input Capture 0	IPCP0	R	Input Capture 0/1	XXXXXXXX
7921 _H	Input Capture 0	IPCP0	R		XXXXXXXX
7922 _H	Input Capture 1	IPCP1	R		XXXXXXXX
7923 _H	Input Capture 1	IPCP1	R		XXXXXXXX
7924 _H	Input Capture 2	IPCP2	R	Input Capture 2/3	XXXXXXXX
7925 _H	Input Capture 2	IPCP2	R		XXXXXXXX
7926 _H	Input Capture 3	IPCP3	R		XXXXXXXX
7927 _H	Input Capture 3	IPCP3	R		XXXXXXXX
7928 _H	Input Capture 4	IPCP4	R	Input Capture 4/5	XXXXXXXX
7929 _H	Input Capture 4	IPCP4	R		XXXXXXXX
792A _H	Input Capture 5	IPCP5	R		XXXXXXXX
792B _H	Input Capture 5	IPCP5	R		XXXXXXXX
792C _H	Input Capture 6	IPCP6	R	Input Capture 6/7	XXXXXXXX
792D _H	Input Capture 6	IPCP6	R		XXXXXXXX
792E _H	Input Capture 7	IPCP7	R		XXXXXXXX
792F _H	Input Capture 7	IPCP7	R		XXXXXXXX
7930 _H	Output Compare 0	OCCP0	R/W	Output Compare 0/1	XXXXXXXX
7931 _H	Output Compare 0	OCCP0	R/W		XXXXXXXX
7932 _H	Output Compare 1	OCCP1	R/W		XXXXXXXX
7933 _H	Output Compare 1	OCCP1	R/W		XXXXXXXX

Table A-2 I/O map (79XX - 7FXX addresses) (3/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7934 _H	Output Compare 2	OCCP2	R/W	Output Compare 2/3	XXXXXXXX
7935 _H	Output Compare 2	OCCP2	R/W		XXXXXXXX
7936 _H	Output Compare 3	OCCP3	R/W		XXXXXXXX
7937 _H	Output Compare 3	OCCP3	R/W		XXXXXXXX
7938 _H	Output Compare 4	OCCP4	R/W	Output Compare 4/5	XXXXXXXX
7939 _H	Output Compare 4	OCCP4	R/W		XXXXXXXX
793A _H	Output Compare 5	OCCP5	R/W		XXXXXXXX
793B _H	Output Compare 5	OCCP5	R/W		XXXXXXXX
793C _H	Output Compare 6	OCCP6	R/W	Output Compare 6/7	XXXXXXXX
793D _H	Output Compare 6	OCCP6	R/W		XXXXXXXX
793E _H	Output Compare 7	OCCP7	R/W		XXXXXXXX
793F _H	Output Compare 7	OCCP7	R/W		XXXXXXXX
7940 _H	Timer Data 0	TCDT0	R/W	I/O Timer 0	00000000
7941 _H	Timer Data 0	TCDT0	R/W		00000000
7942 _H	Timer Control 0	TCCSL0	R/W		00000000
7943 _H	Timer Control 0	TCCSH0	R/W		0XXXXXXXX
7944 _H	Timer Data 1	TCDT1	R/W	I/O Timer 1	00000000
7945 _H	Timer Data 1	TCDT1	R/W		00000000
7946 _H	Timer Control 1	TCCSL1	R/W		00000000
7947 _H	Timer Control 1	TCCSH1	R/W		0XXXXXXXX
7948 _H	Timer 0/Reload 0	TMR0/ TMRLR0	R/W	16-bit Reload Timer 0	XXXXXXXX
7949 _H			R/W		XXXXXXXX
794A _H	Timer 1/Reload 1	TMR1/ TMRLR1	R/W	16-bit Reload Timer 1	XXXXXXXX
794B _H			R/W		XXXXXXXX
794C _H	Timer 2/Reload 2	TMR2/ TMRLR2	R/W	16-bit Reload Timer 2	XXXXXXXX
794D _H			R/W		XXXXXXXX
794E _H	Timer 3/Reload 3	TMR3/ TMRLR3	R/W	16-bit Reload Timer 3	XXXXXXXX
794F _H			R/W		XXXXXXXX

Table A-2 I/O map (79XX - 7FXX addresses) (4/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7950 _H	Serial Mode Register	SMR3	W,R/W	UART3	00000000
7951 _H	Serial Control Register	SCR3	W,R/W		00000000
7952 _H	Reception/Transmission Data Register	RDR3/TDR3	R/W		00000000
7953 _H	Serial Status Register	SSR3	R,R/W		00001000
7954 _H	Extended Communication Control Reg.	ECCR3	R,W,R/W		000000XX
7955 _H	Extended Status/Control Register	ESCR3	R/W		00000100
7956 _H	Baud Rate Register 0	BGR30	R/W		00000000
7957 _H	Baud Rate Register 1	BGR31	R/W		00000000
7958 _H	Serial Mode Register	SMR4	W,R/W	UART4	00000000
7959 _H	Serial Control Register	SCR4	W,R/W		00000000
795A _H	Reception/Transmission Data Register	RDR4/TDR4	R/W		00000000
795B _H	Serial Status Register	SSR4	R,R/W		00001000
795C _H	Extended Communication Control Reg.	ECCR4	R,W,R/W		000000XX
795D _H	Extended Status/Control Register	ESCR4	R/W		00000100
795E _H	Baud Rate Register 0	BGR40	R/W		00000000
795F _H	Baud Rate Register 1	BGR41	R/W		00000000
7960 to 796B _H	Reserved				
796C _H	Clock output enable register	CLKR	R/W	Clock Monitor	XXXX0000
796D _H	Reserved				
796E _H	CAN Direct Mode Register	CDMR	R/W	CAN clock sync	XXXXXXX0
796F _H	CAN RX/TX redirect register	CANSWR	R/W	CAN 0/1	XXXXXXX0

Table A-2 I/O map (79XX - 7FXX addresses) (5/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7970 _H	I ² C bus status register	IBSR0	R	I ² C Interface 0	00000000
7971 _H	I ² C bus control register	IBCR0	W,R/W		00000000
7972 _H	I ² C ten bit slave address register	ITBAL0	R/W		00000000
7973 _H		ITBAH0	R/W		00000000
7974 _H	I ² C ten bit address mask register	ITMKL0	R/W		11111111
7975 _H		ITMKH0	R/W		00111111
7976 _H	I ² C seven bit slave address register	ISBA0	R/W		00000000
7977 _H	I ² C seven bit address mask register	ISMK0	R/W		01111111
7978 _H	I ² C data register	IDAR0	R/W		00000000
7979 _H to 797A _H	Reserved				
797B _H	I ² C clock control register	ICCR0	R/W	I ² C Interface 0	00011111
797C to 797F _H	Reserved				
7980 _H	I ² C bus status register	IBSR1	R	I ² C Interface 1	00000000
7981 _H	I ² C bus control register	IBCR1	W,R/W		00000000
7982 _H	I ² C ten bit slave address register	ITBAL1	R/W		00000000
7983 _H		ITBAH1	R/W		00000000
7984 _H	I ² C ten bit address mask register	ITMKL1	R/W		11111111
7985 _H		ITMKH1	R/W		00111111
7986 _H	I ² C seven bit slave address register	ISBA1	R/W		00000000
7987 _H	I ² C seven bit address mask register	ISMK1	R/W		01111111
7988 _H	I ² C data register	IDAR1	R/W		00000000
7989 _H to 798A _H	Reserved				
798B _H	I ² C clock control register	ICCR1	R/W	I ² C Interface 1	00011111

Table A-2 I/O map (79XX - 7FFX addresses) (6/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
798C to 79C1 _H	Reserved				
79C2 _H	Clock Modulator Control Register	CMCR	R,R/W	Clock Modulator	0001X000
79C3 to 79DF _H	Reserved				
79E0 _H	ROM Correction Address 0	PADR0	R/W	ROM Correction 0	XXXXXXXX
79E1 _H	ROM Correction Address 0	PADR0	R/W		XXXXXXXX
79E2 _H	ROM Correction Address 0	PADR0	R/W		XXXXXXXX
79E3 _H	ROM Correction Address 1	PADR1	R/W		XXXXXXXX
79E4 _H	ROM Correction Address 1	PADR1	R/W		XXXXXXXX
79E5 _H	ROM Correction Address 1	PADR1	R/W		XXXXXXXX
79E6 _H	ROM Correction Address 2	PADR2	R/W		XXXXXXXX
79E7 _H	ROM Correction Address 2	PADR2	R/W		XXXXXXXX
79E8 _H	ROM Correction Address 2	PADR2	R/W		XXXXXXXX
79E9 to 79EF _H	Reserved				
79F0 _H	ROM Correction Address 3	PADR3	R/W	ROM Correction 1	XXXXXXXX
79F1 _H	ROM Correction Address 3	PADR3	R/W		XXXXXXXX
79F2 _H	ROM Correction Address 3	PADR3	R/W		XXXXXXXX
79F3 _H	ROM Correction Address 4	PADR4	R/W		XXXXXXXX
79F4 _H	ROM Correction Address 4	PADR4	R/W		XXXXXXXX
79F5 _H	ROM Correction Address 4	PADR4	R/W		XXXXXXXX
79F6 _H	ROM Correction Address 5	PADR5	R/W		XXXXXXXX
79F7 _H	ROM Correction Address 5	PADR5	R/W		XXXXXXXX
79F8 _H	ROM Correction Address 5	PADR5	R/W		XXXXXXXX
79F9 to 79FF _H	Reserved				
7A00 to 7AFF _H	Reserved for CAN Interface 0. Refer to section about CAN controller				

Table A-2 I/O map (79XX - 7FXX addresses) (7/7)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7B00 to 7BFF _H	Reserved for CAN Interface 0. Refer to section about CAN controller				
7C00 to 7CFF _H	Reserved for CAN Interface 1. Refer to section about CAN controller				
7D00 to 7DFF _H	Reserved for CAN Interface 1. Refer to section about CAN controller				
7E00 to 7EFF _H	Reserved for CAN Interface 2. Refer to section about CAN controller				
7F00 to 7FFF _H	Reserved for CAN Interface 2. Refer to section about CAN controller				

Note: Any write access to reserved addresses in I/O map should not be performed. A read access to reserved address results in reading "X".

Note:

Any write access to reserved addresses in I/O map should not be performed.
A read access to reserved address results in reading "X".

● 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.

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

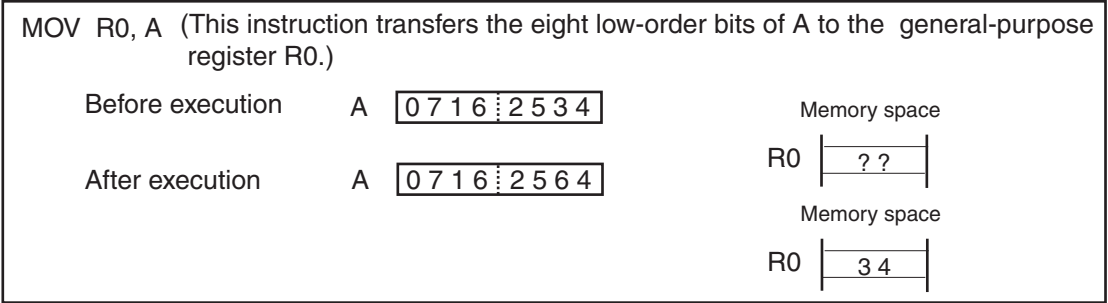
Specify a register explicitly as an operand. Table B.3-1 "Direct addressing registers" lists the registers that can be specified. Figure B.3-2 "Example of register direct addressing" shows an example of register direct addressing.

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, R5W, 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.

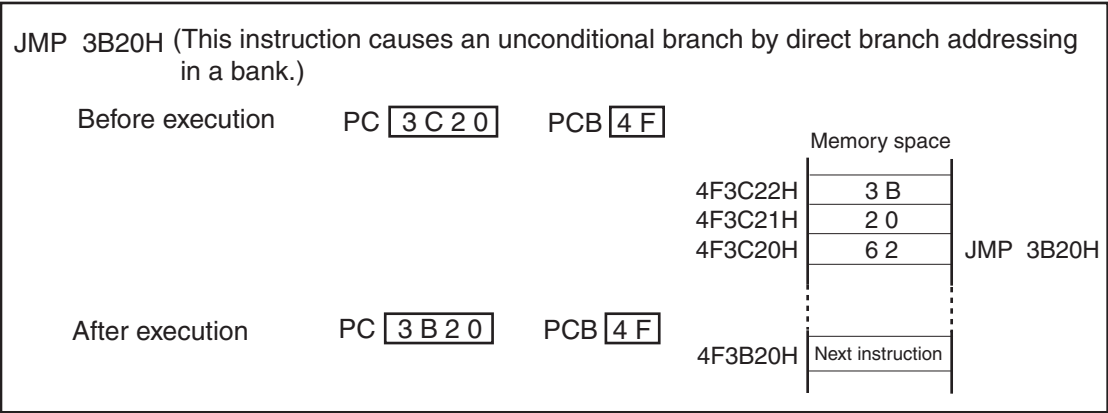
Figure B.3-2 Example of register direct addressing



● 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).

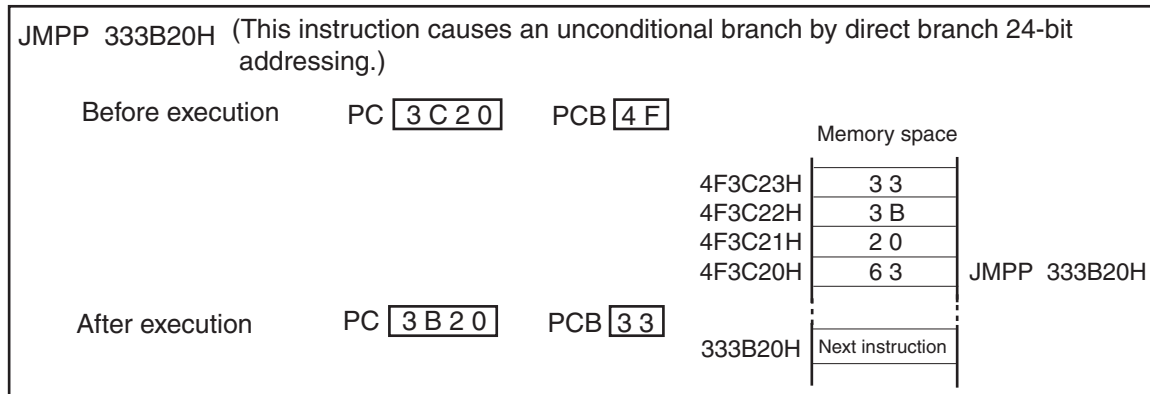
Figure B.3-3 Example of direct branch addressing (addr16)



● 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.

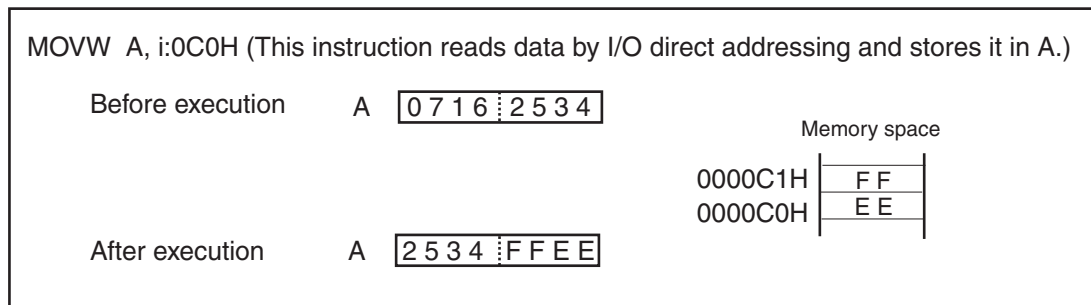
Figure B.3-4 Example of direct branch addressing (addr24)



● 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.

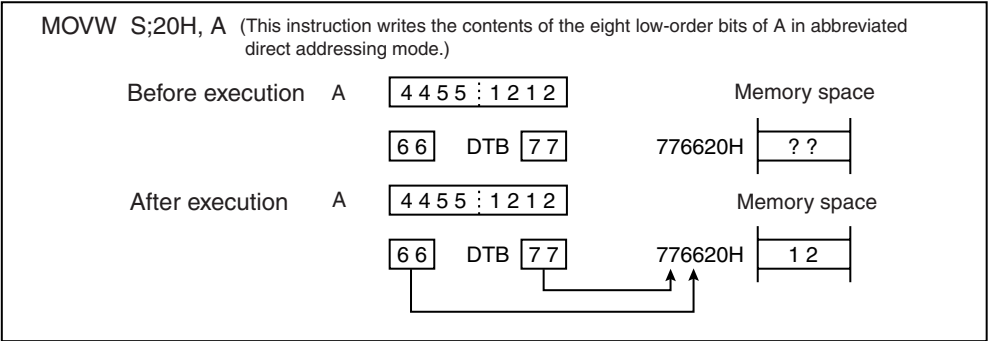
Figure B.3-5 Example of I/O direct addressing (io)



● 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).

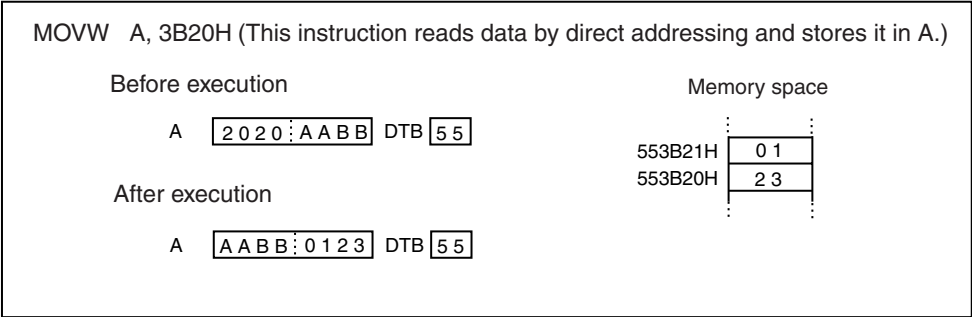
Figure B.3-6 Example of abbreviated direct addressing (dir)



● 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.

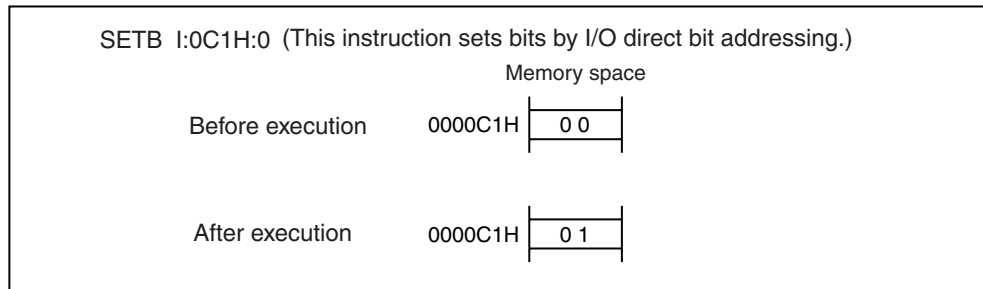
Figure B.3-7 Example of direct addressing (addr16)



● 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).

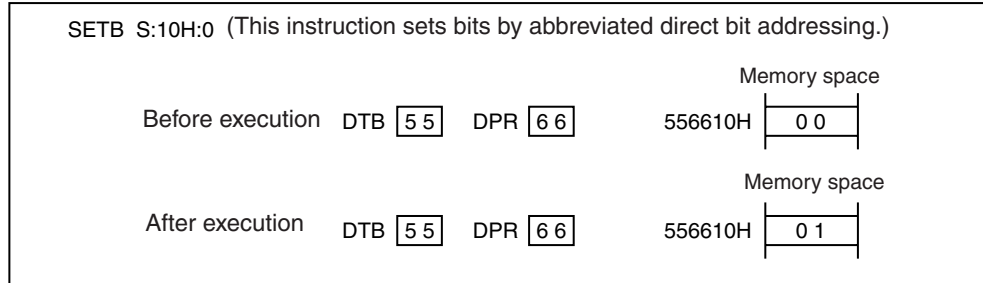
Figure B.3-8 Example of I/O direct bit addressing (io:bp)



● 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).

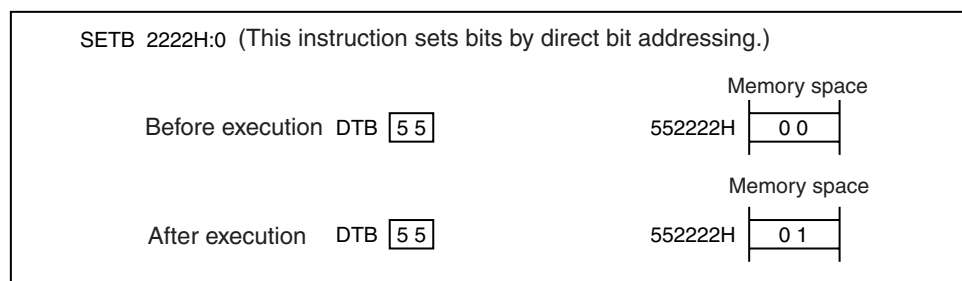
Figure B.3-9 Example of abbreviated direct bit addressing (dir:bp)



● 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)

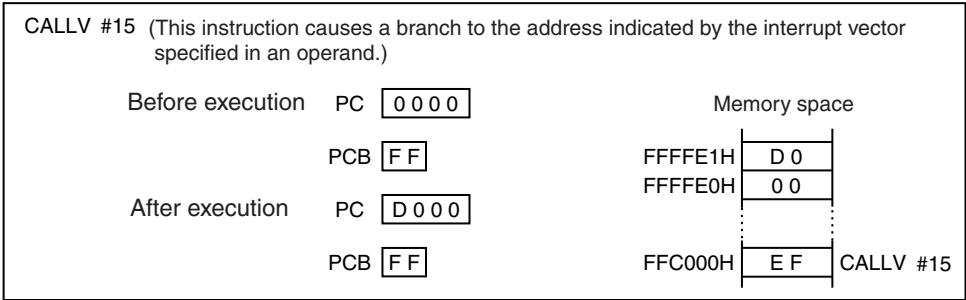


Table B.3-2 CALLV vector list

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 INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2 "CALLV vector list").

B.4 Indirect Addressing

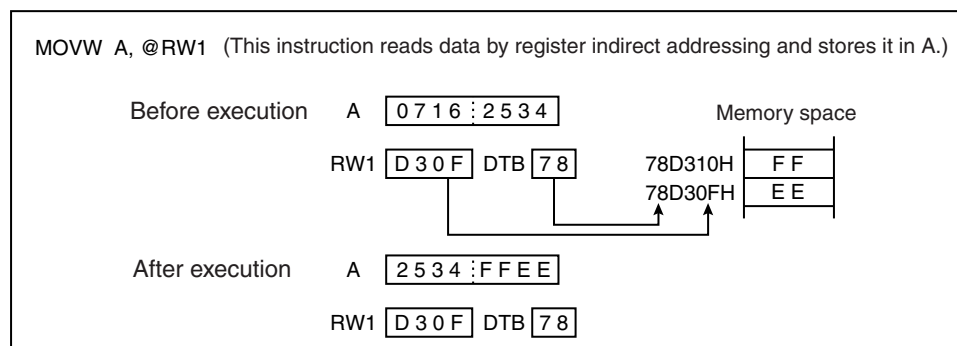
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ 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.

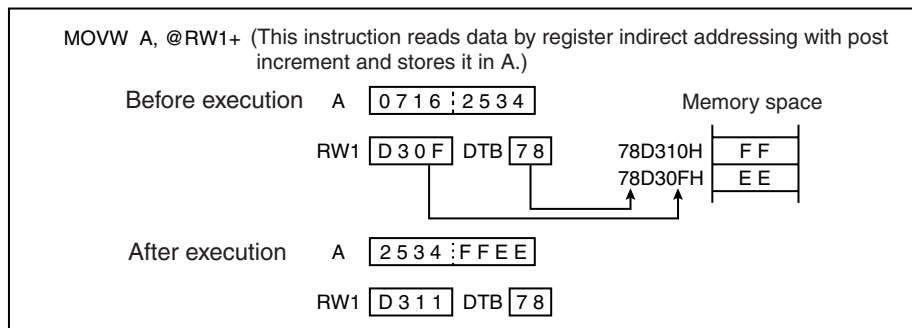
Figure B.4-1 Example of register indirect addressing (@RWj j = 0 to 3)



● Register indirect addressing with post increment (@RWj+ j = 0 to 3)

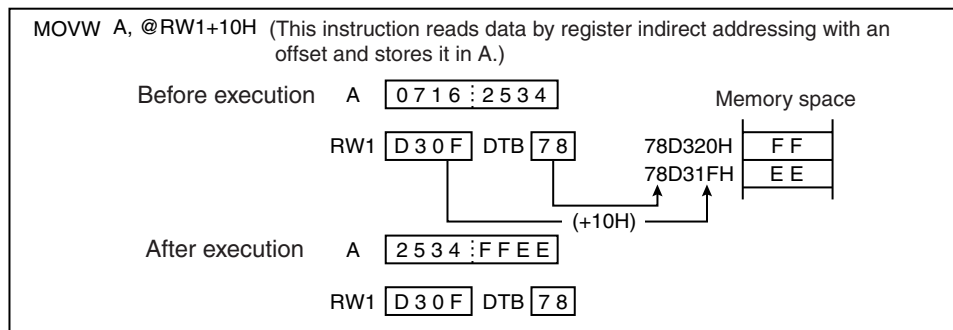
Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system 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)

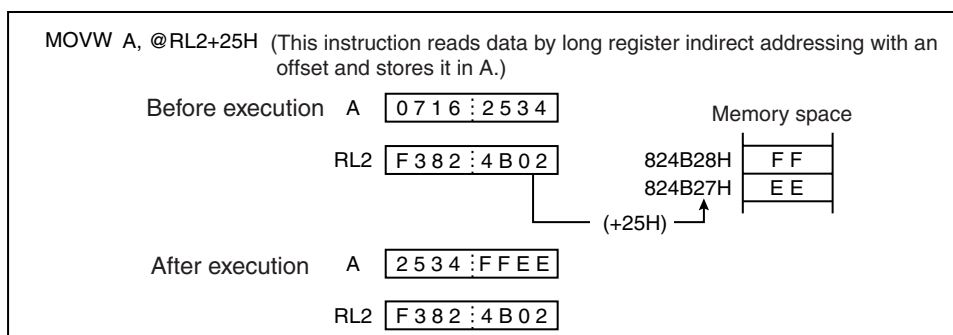
● Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)

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.

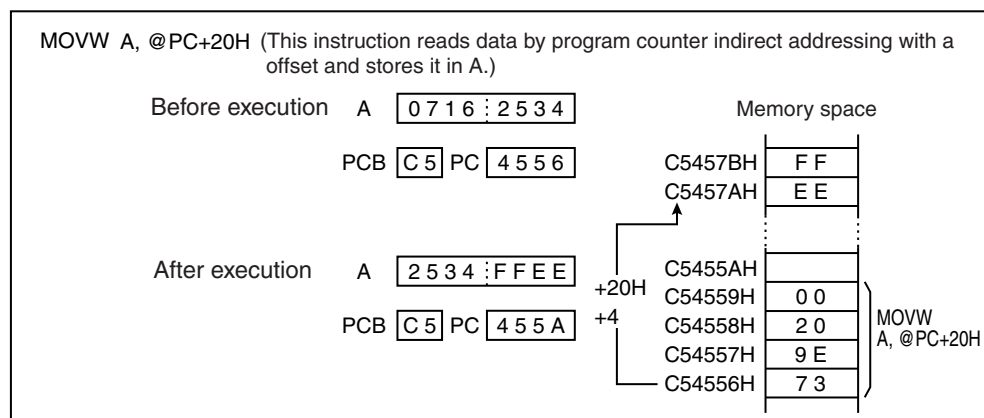
Figure B.4-4 Example of long register indirect addressing with offset (@RLi + disp8 i = 0 to 3)

● 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

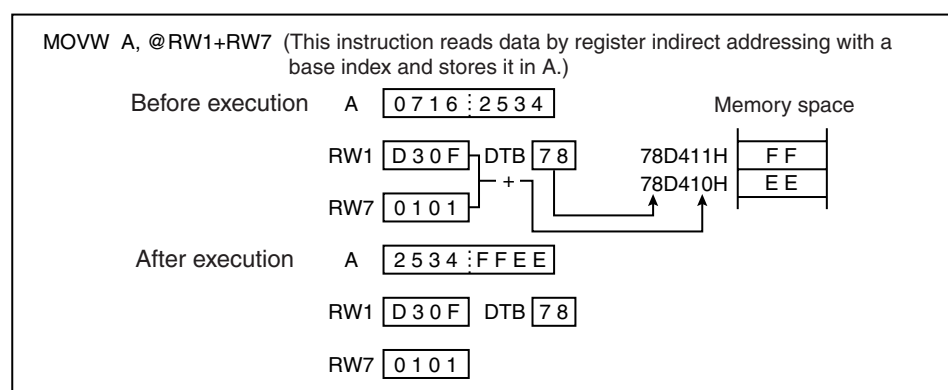
Figure B.4-5 Example of program counter indirect addressing with offset (@PC + disp16)



● 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).

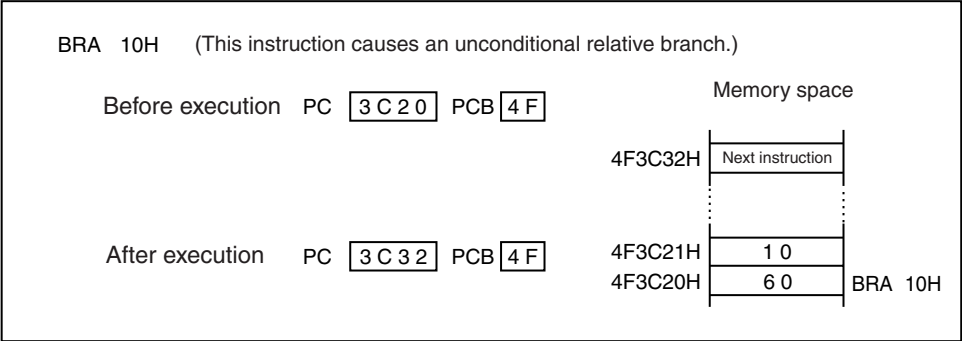
Figure B.4-6 Example of register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)



● 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

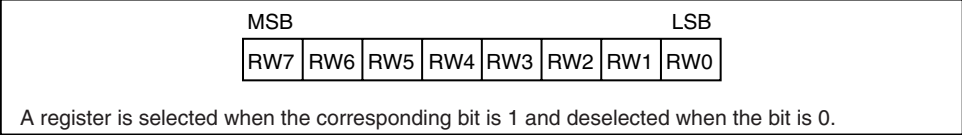
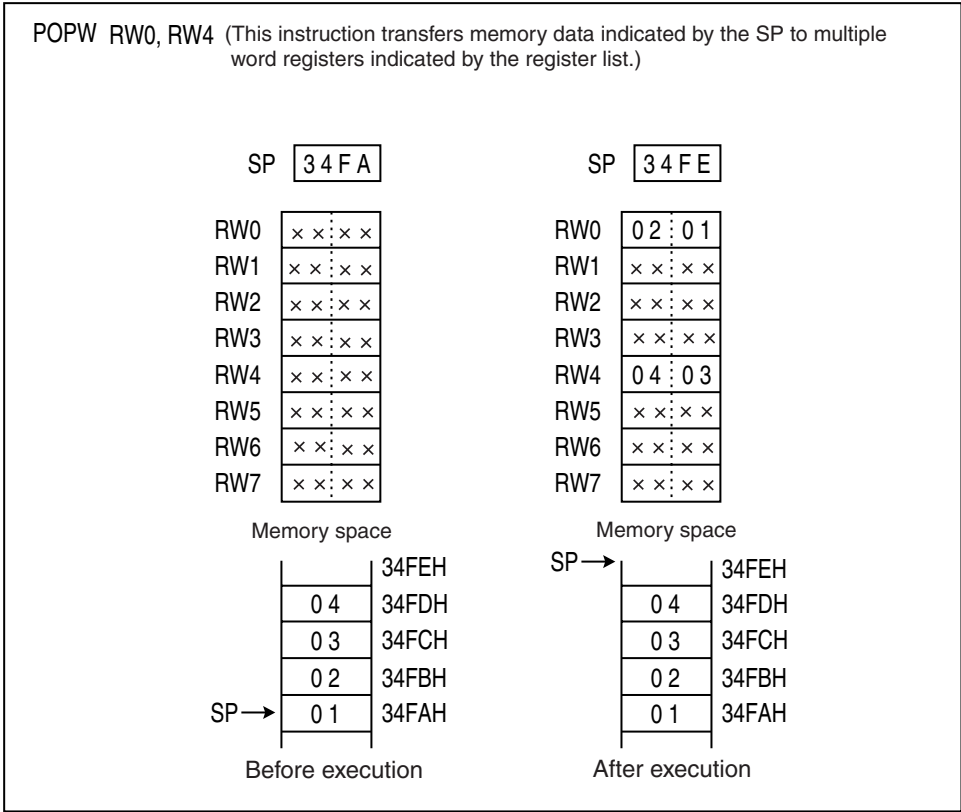


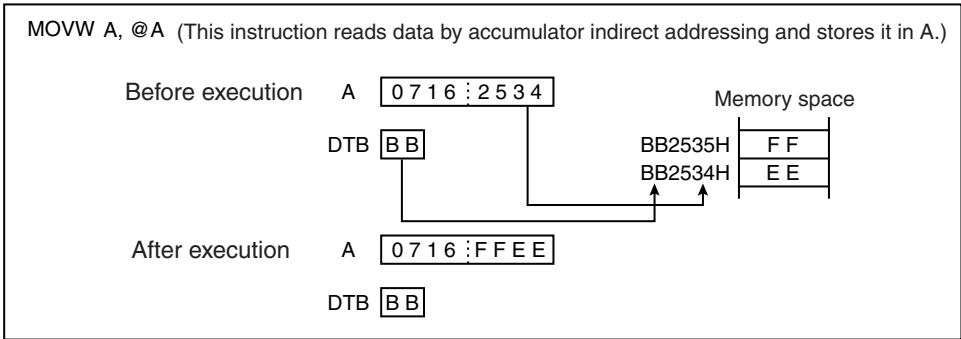
Figure B.4-9 Example of register list (rlist)



● 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).

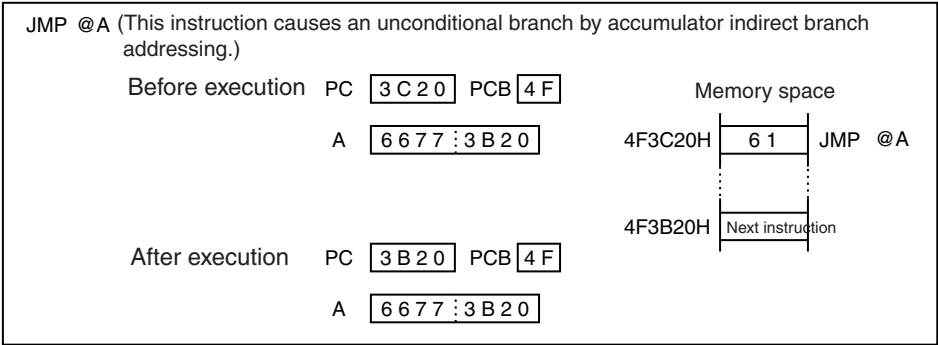
Figure B.4-10 Example of accumulator indirect addressing (@A)



● 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.

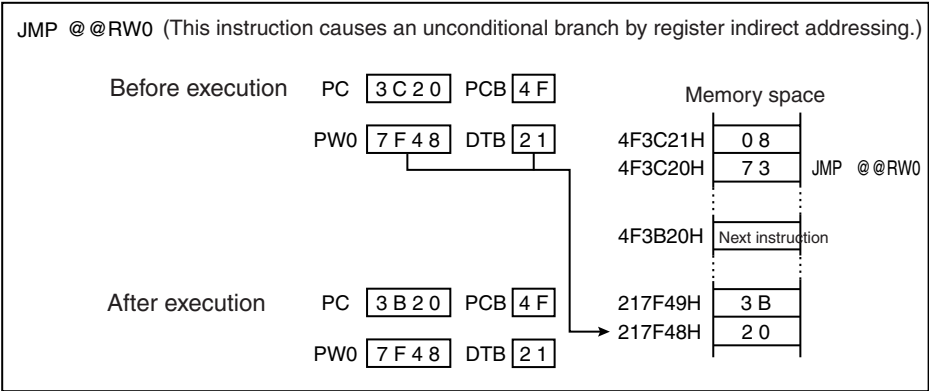
Figure B.4-11 Example of accumulator indirect branch addressing (@A)



● Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

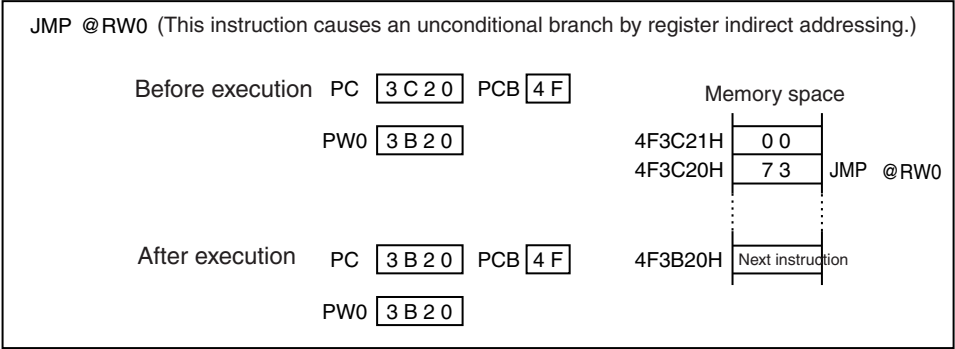
Figure B.4-12 Example of indirect specification branch addressing (@ear)



● 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 $\sim$ (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 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-						
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3					Register indirect	0				
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+							Register indirect with post increment	0		
10 11 12 13 14 15 16 17	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8 @RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8									Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16										
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16					Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	0 0 2 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	2+	4 + (a)	1	(c)	word (RWi) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVW ear,RWi	2	4	2	0	word (ear) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW eam,RWi	2+	5 + (a)	1	(c)	word (eam) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,#imm16	3	2	1	0	word (RWi) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW io,#imm16	4	5	0	(c)	word (io) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW ear,#imm16	4	2	1	0	word (ear) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW eam,#imm16	4+	4 + (a)	0	(c)	word (eam) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW @AL,AH / 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.

Table B.8-3 42 Addition/subtraction instructions (byte, word, long word)

Mnemonic	#	~	RG	B	Operation	L H	A H	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	-	-	-	-	*	*	*	*	-
ADDC 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.

Table B.8-6 11 Unsigned multiplication/division instructions (word, long word)

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
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)	-	-	-	-	-	-	-	*	*	-
MULL A	1	*8	0	0	byte (AH) * byte (AL) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULL A,ear	2	*9	1	0	byte (A) * byte (ear) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULL A,eam	2+	*10	0	(b)	byte (A) * byte (eam) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULEY A	1	*11	0	0	word (AH) * word (AL) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULEY A,ear	2	*12	1	0	word (A) * word (ear) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULEY 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	L H	A H	I	S	T	N	Z	V	C	R M W
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.
-

Table B.8-8 39 Logic 1 instructions (byte, word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
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	L H	A H	I	S	T	N	Z	V	C	R M W
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	L H	A H	I	S	T	N	Z	V	C	R M W
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	L H	A H	I	S	T	N	Z	V	C	R M W
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)

Table B.8-12 18 Shift instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
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) <-- addr24 0-15, (PCB) <-- addr24 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	L H	A H	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.

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

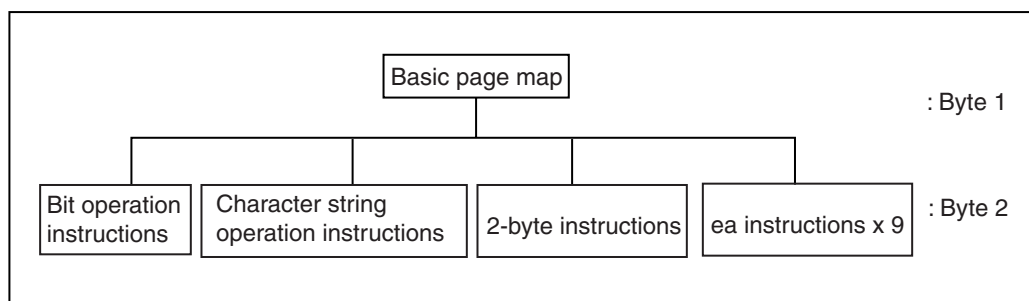
See Table B.5-1 "Execution cycle counts in each addressing mode" and Table B.5-2 "Cycle count correction values for counting execution cycles" for information on (a) to (d) in the table.

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 = 7F_H)" summarize the F²MC-16LX instruction map.

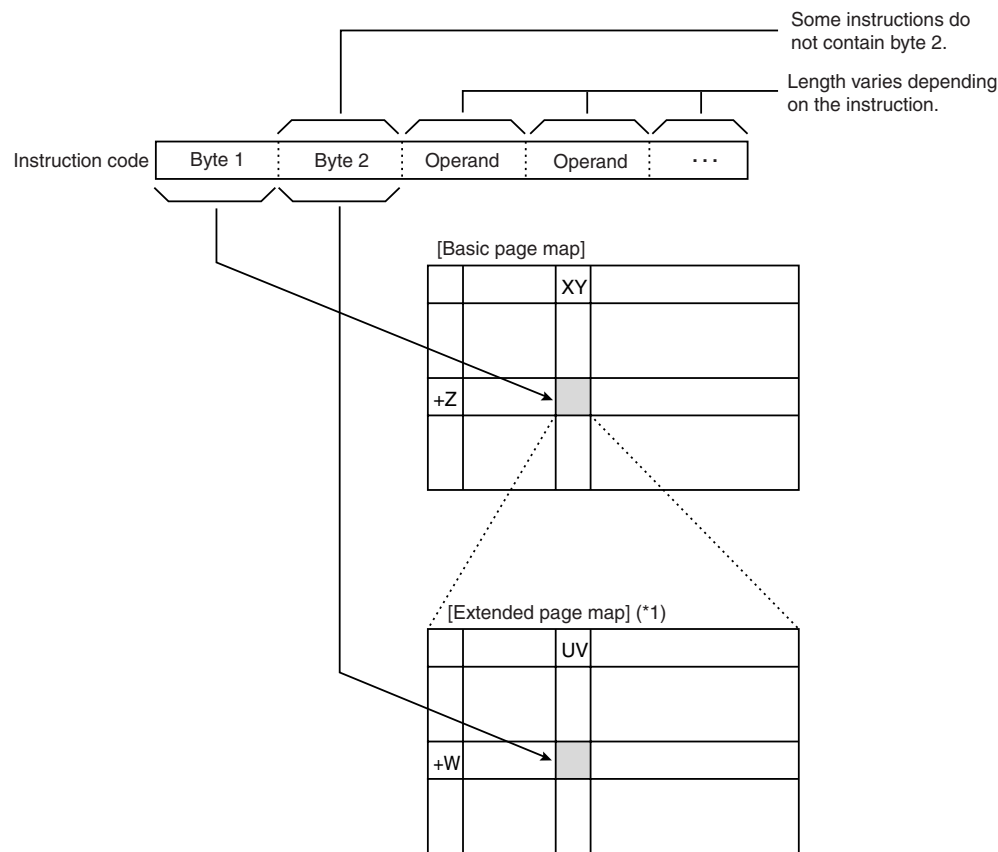
■ Structure of instruction map

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 instruc- tion 1	MOV A,Ri	MOV Ri,A	MOV Ri,#8	MOVX A,Ri	MOVX A, @RWi+d8	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 instruc- tion 2								BNZ/BNE rel
+ 2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A,#8	MOV A,addr16	JMP addr16	ea instruc- tion 3								BC/BLO rel
+ 3	NEG A	JCTX @A	CMP A	CMP A,#8	MOVX A,#8	MOV addr16,A	JMP addr24	ea instruc- tion 4								BNC/BHS rel
+ 4	PCB	EXT	AND CCR,#8	AND A,#8	MOV dir,#8	MOV io,#8	CALL addr16	ea instruc- tion 5								BN rel
+ 5	DTB	ZEXT	OR CCR,#8	OR A,#8	MOVX A,dir	MOVX A,io	CALLP addr24	ea instruc- tion 6								BP rel
+ 6	ADB	SWAP	DIVU A	XOR A,#8	MOVW A,SP	MOVW io,#16	RETP	ea instruc- tion 7								BV rel
+ 7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP,A	MOVX A,addr16	RET	ea instruc- tion 8								BNV rel
+ 8	LINK #imm8	ADDL A,#32	ADDW A	ADDW A,#16	MOVW A,dir	MOVW io,A	INT #vct8	ea instruc- tion 9	MOVW A,RWi	MOVW RWi,A	MOVW RWi,#16	MOVW A, @RWi+d8	MOVW @R Wi+d8,A			BT rel
+ 9	UNLINK	SUBL A,#32	SUBW A	SUBW A,#16	MOVW dir,A	MOVW io,A	INTP	MOVEA RWi,ea								BNT rel
+ A	MOV RP,#8	MOV ILM,#8	CBNE A, #8,rel	CBNE A, #16,rel	MOVW A,#16	MOVW A,addr16	MOV addr24	MOV Ri,ea								BLT rel
+ B	NEGW	CMPL A,#32	CMPL A	CMPL A,#16	MOVL A,#32	MOVW addr16,A	RETI	MOVW RWi,ea								BGE rel
+ C	LSLW A	EXTW	ANDW A	ANDW A,#16	PUSHW A	POPW A	Bit operation instruction	MOV ea,Ri								BLE rel
+ D		ZEXTW	ORW A	ORW A,#16	PUSHW AH	POPW AH		MOVW ea,RWi								BGT rel
+ E	ASRW A	SWAPW	XORW A	XORW A,#16	PUSHW PS	POPW PS	Character string opera- tion instruction	XCH Ri,ea								BLS rel
+ F	LSRW A	ADDSP #16	MULW A	NOTW A	PUSHW r1st	POPW r1st	2-byte instruction	XCHW RWi,ea								BHI rel

Table B.9-3 Bit operation instruction map (first byte = 6C_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV A,io:bp		MOV io:bp,A		CLRB io:bp		SETB io:bp		BBC io:bp,rel		BBS io:bp,rel		WBTS io:bp		WBTC io:bp	
+ 1																
+ 2																
+ 3																
+ 4																
+ 5																
+ 6																
+ 7																
+ 8	MOV A,dir:bp	MOV A,a ddr16:bp	MOV dir:bp,A	MOV addr16:bp,A	CLRB dir:bp	CLRB a ddr16:bp	SETB dir:bp	SETB a ddr16:bp	BBC dir:bp,rel	BBC a ddr16:bp	BBS dir:bp,rel	BBS a ddr16:bp				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 PCB,PCB	MOVSWI PCB,PCB	MOVSWD PCB,PCB					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB		FILSWI PCB	
+1									DTB	DTB	DTB	DTB	DTB		DTB	
+2									ADB	ADB	ADB	ADB	ADB		ADB	
+3									SPB	SPB	SPB	SPB	SPB		SPB	
+4																
+5																
+6																
+7																
+8																
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table B.9-5 2-byte instruction map (first byte = 6F_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV A,DTB	MOV DTB,A	MOVX A, @RL0+d8	MOV @RL0 +d8,A	MOV A, @RL0+d8											
+ 1	MOV A,ADB	MOV ADB,A														
+ 2	MOV A,SSB	MOV SSB,A	MOVX A, @RL1+d8	MOV @RL1 +d8,A	MOV A, @RL1+d8											
+ 3	MOV A,USB	MOV USB,A														
+ 4	MOV A,DPR	MOV DPR,A	MOVX A, @RL2+d8	MOV @RL2 +d8,A	MOV A, @RL2+d8											
+ 5	MOV A,@A	MOV @AL,AH														
+ 6	MOV A,PCB	MOVX A,@A	MOVX A, @RL3+d8	MOV @RL3 +d8,A	MOV A, @RL3+d8											
+ 7	ROL A	ROLC A														
+ 8				MOVW @RL 0+d8,A	MOVW A, @RL0+d8		MUL A									
+ 9							MULW A									
+ A				MOVW @RL 1+d8,A	MOVW A, @RL1+d8		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
					CMPLNE, I	CMPLNE, I									CMPLNE, I	
+ 0	ADDL A, @RW0-d8	SUBL A, @RW0-d8	SUBL A, @RW0-d8	SUBL A, @RW0-d8	RW0, #16,rel	@RW0-d8, #16,rel	CMPL A, @RW0-d8	CMPL A, @RW0-d8	ANDL A, @RW0-d8	ANDL A, @RW0-d8	ORL A, @RW0-d8	ORL A, @RW0-d8	XORL A, @RW0-d8	XORL A, @RW0-d8	R0, #8,rel	@RW0-d8, #8,rel
+ 1	ADDL A, @RW1-d8	SUBL A, @RW1-d8	SUBL A, @RW1-d8	SUBL A, @RW1-d8	RW1, #16,rel	@RW1-d8, #16,rel	CMPL A, @RW1-d8	CMPL A, @RW1-d8	ANDL A, @RW1-d8	ANDL A, @RW1-d8	ORL A, @RW1-d8	ORL A, @RW1-d8	XORL A, @RW1-d8	XORL A, @RW1-d8	R1, #8,rel	@RW1-d8, #8,rel
+ 2	ADDL A, @RW2-d8	SUBL A, @RW2-d8	SUBL A, @RW2-d8	SUBL A, @RW2-d8	RW2, #16,rel	@RW2-d8, #16,rel	CMPL A, @RW2-d8	CMPL A, @RW2-d8	ANDL A, @RW2-d8	ANDL A, @RW2-d8	ORL A, @RW2-d8	ORL A, @RW2-d8	XORL A, @RW2-d8	XORL A, @RW2-d8	R2, #8,rel	@RW2-d8, #8,rel
+ 3	ADDL A, @RW3-d8	SUBL A, @RW3-d8	SUBL A, @RW3-d8	SUBL A, @RW3-d8	RW3, #16,rel	@RW3-d8, #16,rel	CMPL A, @RW3-d8	CMPL A, @RW3-d8	ANDL A, @RW3-d8	ANDL A, @RW3-d8	ORL A, @RW3-d8	ORL A, @RW3-d8	XORL A, @RW3-d8	XORL A, @RW3-d8	R3, #8,rel	@RW3-d8, #8,rel
+ 4	ADDL A, @RW4-d8	SUBL A, @RW4-d8	SUBL A, @RW4-d8	SUBL A, @RW4-d8	RW4, #16,rel	@RW4-d8, #16,rel	CMPL A, @RW4-d8	CMPL A, @RW4-d8	ANDL A, @RW4-d8	ANDL A, @RW4-d8	ORL A, @RW4-d8	ORL A, @RW4-d8	XORL A, @RW4-d8	XORL A, @RW4-d8	R4, #8,rel	@RW4-d8, #8,rel
+ 5	ADDL A, @RW5-d8	SUBL A, @RW5-d8	SUBL A, @RW5-d8	SUBL A, @RW5-d8	RW5, #16,rel	@RW5-d8, #16,rel	CMPL A, @RW5-d8	CMPL A, @RW5-d8	ANDL A, @RW5-d8	ANDL A, @RW5-d8	ORL A, @RW5-d8	ORL A, @RW5-d8	XORL A, @RW5-d8	XORL A, @RW5-d8	R5, #8,rel	@RW5-d8, #8,rel
+ 6	ADDL A, @RW6-d8	SUBL A, @RW6-d8	SUBL A, @RW6-d8	SUBL A, @RW6-d8	RW6, #16,rel	@RW6-d8, #16,rel	CMPL A, @RW6-d8	CMPL A, @RW6-d8	ANDL A, @RW6-d8	ANDL A, @RW6-d8	ORL A, @RW6-d8	ORL A, @RW6-d8	XORL A, @RW6-d8	XORL A, @RW6-d8	R6, #8,rel	@RW6-d8, #8,rel
+ 7	ADDL A, @RW7-d8	SUBL A, @RW7-d8	SUBL A, @RW7-d8	SUBL A, @RW7-d8	RW7, #16,rel	@RW7-d8, #16,rel	CMPL A, @RW7-d8	CMPL A, @RW7-d8	ANDL A, @RW7-d8	ANDL A, @RW7-d8	ORL A, @RW7-d8	ORL A, @RW7-d8	XORL A, @RW7-d8	XORL A, @RW7-d8	R7, #8,rel	@RW7-d8, #8,rel
+ 8	ADDL A, @RW0-d16	SUBL A, @RW0-d16	SUBL A, @RW0-d16	SUBL A, @RW0-d16	@RW0, #16,rel	@RW0-d16, #16,rel	CMPL A, @RW0-d16	CMPL A, @RW0-d16	ANDL A, @RW0-d16	ANDL A, @RW0-d16	ORL A, @RW0-d16	ORL A, @RW0-d16	XORL A, @RW0-d16	XORL A, @RW0-d16	@RW0, #8,rel	@RW0-d16, #8,rel
+ 9	ADDL A, @RW1-d16	SUBL A, @RW1-d16	SUBL A, @RW1-d16	SUBL A, @RW1-d16	@RW1, #16,rel	@RW1-d16, #16,rel	CMPL A, @RW1-d16	CMPL A, @RW1-d16	ANDL A, @RW1-d16	ANDL A, @RW1-d16	ORL A, @RW1-d16	ORL A, @RW1-d16	XORL A, @RW1-d16	XORL A, @RW1-d16	@RW1, #8,rel	@RW1-d16, #8,rel
+ A	ADDL A, @RW2-d16	SUBL A, @RW2-d16	SUBL A, @RW2-d16	SUBL A, @RW2-d16	@RW2, #16,rel	@RW2-d16, #16,rel	CMPL A, @RW2-d16	CMPL A, @RW2-d16	ANDL A, @RW2-d16	ANDL A, @RW2-d16	ORL A, @RW2-d16	ORL A, @RW2-d16	XORL A, @RW2-d16	XORL A, @RW2-d16	@RW2, #8,rel	@RW2-d16, #8,rel
+ B	ADDL A, @RW3-d16	SUBL A, @RW3-d16	SUBL A, @RW3-d16	SUBL A, @RW3-d16	@RW3, #16,rel	@RW3-d16, #16,rel	CMPL A, @RW3-d16	CMPL A, @RW3-d16	ANDL A, @RW3-d16	ANDL A, @RW3-d16	ORL A, @RW3-d16	ORL A, @RW3-d16	XORL A, @RW3-d16	XORL A, @RW3-d16	@RW3, #8,rel	@RW3-d16, #8,rel
+ C	ADDL A, @RW0-RW7	SUBL A, @RW0-RW7	SUBL A, @RW0-RW7	SUBL A, @RW0-RW7	Use prohibited	@RW0-RW7, #16,rel	CMPL A, @RW0-RW7	CMPL A, @RW0-RW7	ANDL A, @RW0-RW7	ANDL A, @RW0-RW7	ORL A, @RW0-RW7	ORL A, @RW0-RW7	XORL A, @RW0-RW7	XORL A, @RW0-RW7	Use prohibited	@RW0-RW7, #8,rel
+ D	ADDL A, @RW1-RW7	SUBL A, @RW1-RW7	SUBL A, @RW1-RW7	SUBL A, @RW1-RW7	Use prohibited	@RW1-RW7, #16,rel	CMPL A, @RW1-RW7	CMPL A, @RW1-RW7	ANDL A, @RW1-RW7	ANDL A, @RW1-RW7	ORL A, @RW1-RW7	ORL A, @RW1-RW7	XORL A, @RW1-RW7	XORL A, @RW1-RW7	Use prohibited	@RW1-RW7, #8,rel
+ E	ADDL A, @PC-d16	SUBL A, @PC-d16	SUBL A, @PC-d16	SUBL A, @PC-d16	Use prohibited	@PC-d16, #16,rel	CMPL A, @PC-d16	CMPL A, @PC-d16	ANDL A, @PC-d16	ANDL A, @PC-d16	ORL A, @PC-d16	ORL A, @PC-d16	XORL A, @PC-d16	XORL A, @PC-d16	Use prohibited	@PC-d16, #8,rel
+ F	ADDL A, @RW3+	SUBL A, @RW3+	SUBL A, @RW3+	SUBL A, @RW3+	Use prohibited	@RW3+, #16,rel	CMPL A, @RW3+	CMPL A, @RW3+	ANDL A, @RW3+	ANDL A, @RW3+	ORL A, @RW3+	ORL A, @RW3+	XORL A, @RW3+	XORL A, @RW3+	Use prohibited	@RW3+, #8,rel

Table B.9-7 ea instruction 2 (first byte = 71_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMPP @RL0	JMPP @RL0	CALLP @RL0	CALLP @RL0	INCL @RL0	INCL @RL0	DECL @RL0	DECL @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0
+ 1	JMPP @RL0	JMPP @RL0	CALLP @RL0	CALLP @RL0	INCL @RL0	INCL @RL0	DECL @RL0	DECL @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0	MOV @RL0
+ 2	JMPP @RL1	JMPP @RL1	CALLP @RL1	CALLP @RL1	INCL @RL1	INCL @RL1	DECL @RL1	DECL @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1
+ 3	JMPP @RL1	JMPP @RL1	CALLP @RL1	CALLP @RL1	INCL @RL1	INCL @RL1	DECL @RL1	DECL @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1	MOV @RL1
+ 4	JMPP @RL2	JMPP @RL2	CALLP @RL2	CALLP @RL2	INCL @RL2	INCL @RL2	DECL @RL2	DECL @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2
+ 5	JMPP @RL2	JMPP @RL2	CALLP @RL2	CALLP @RL2	INCL @RL2	INCL @RL2	DECL @RL2	DECL @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2	MOV @RL2
+ 6	JMPP @RL3	JMPP @RL3	CALLP @RL3	CALLP @RL3	INCL @RL3	INCL @RL3	DECL @RL3	DECL @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3
+ 7	JMPP @RL3	JMPP @RL3	CALLP @RL3	CALLP @RL3	INCL @RL3	INCL @RL3	DECL @RL3	DECL @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3	MOV @RL3
+ 8	JMPP @RW0	JMPP @RW0	CALLP @RW0	CALLP @RW0	INCL @RW0	INCL @RW0	DECL @RW0	DECL @RW0	MOV @RW0	MOV @RW0	MOV @RW0	MOV @RW0	MOV @RW0	MOV @RW0	MOV @RW0	MOV @RW0
+ 9	JMPP @RW1	JMPP @RW1	CALLP @RW1	CALLP @RW1	INCL @RW1	INCL @RW1	DECL @RW1	DECL @RW1	MOV @RW1	MOV @RW1	MOV @RW1	MOV @RW1	MOV @RW1	MOV @RW1	MOV @RW1	MOV @RW1
+ A	JMPP @RW2	JMPP @RW2	CALLP @RW2	CALLP @RW2	INCL @RW2	INCL @RW2	DECL @RW2	DECL @RW2	MOV @RW2	MOV @RW2	MOV @RW2	MOV @RW2	MOV @RW2	MOV @RW2	MOV @RW2	MOV @RW2
+ B	JMPP @RW3	JMPP @RW3	CALLP @RW3	CALLP @RW3	INCL @RW3	INCL @RW3	DECL @RW3	DECL @RW3	MOV @RW3	MOV @RW3	MOV @RW3	MOV @RW3	MOV @RW3	MOV @RW3	MOV @RW3	MOV @RW3
+ C	JMPP @RW0+	JMPP @RW0+	CALLP @RW0+	CALLP @RW0+	INCL @RW0+	INCL @RW0+	DECL @RW0+	DECL @RW0+	MOV @RW0+	MOV @RW0+	MOV @RW0+	MOV @RW0+	MOV @RW0+	MOV @RW0+	MOV @RW0+	MOV @RW0+
+ D	JMPP @RW1+	JMPP @RW1+	CALLP @RW1+	CALLP @RW1+	INCL @RW1+	INCL @RW1+	DECL @RW1+	DECL @RW1+	MOV @RW1+	MOV @RW1+	MOV @RW1+	MOV @RW1+	MOV @RW1+	MOV @RW1+	MOV @RW1+	MOV @RW1+
+ E	JMPP @PC-d16	JMPP @PC-d16	CALLP @PC-d16	CALLP @PC-d16	INCL @PC-d16	INCL @PC-d16	DECL @PC-d16	DECL @PC-d16	MOV @PC-d16	MOV @PC-d16	MOV @PC-d16	MOV @PC-d16	MOV @PC-d16	MOV @PC-d16	MOV @PC-d16	MOV @PC-d16
+ F	JMPP @RW3+	JMPP @RW3+	CALLP @RW3+	CALLP @RW3+	INCL @RW3+	INCL @RW3+	DECL @RW3+	DECL @RW3+	MOV @RW3+	MOV @RW3+	MOV @RW3+	MOV @RW3+	MOV @RW3+	MOV @RW3+	MOV @RW3+	MOV @RW3+

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 R0	ROL @RW0-d8	RORC R0	RORC @RW0-d8	INC R0	INC @RW0-d8	DEC R0	DEC @RW0-d8	MOV A,R0	MOV A,@RW0-d8	MOV R0,A	MOV @R,W0-d8A	MOVX A,R0	MOVX A,@RW0-d8	XCH A,R0	XCH A,@RW0-d8
+ 1	ROL R1	ROL @RW1-d8	RORC R1	RORC @RW1-d8	INC R1	INC @RW1-d8	DEC R1	DEC @RW1-d8	MOV A,R1	MOV A,@RW1-d8	MOV R1,A	MOV @R,W1-d8A	MOVX A,R1	MOVX A,@RW1-d8	XCH A,R1	XCH A,@RW1-d8
+ 2	ROL R2	ROL @RW2-d8	RORC R2	RORC @RW2-d8	INC R2	INC @RW2-d8	DEC R2	DEC @RW2-d8	MOV A,R2	MOV A,@RW2-d8	MOV R2,A	MOV @R,W2-d8A	MOVX A,R2	MOVX A,@RW2-d8	XCH A,R2	XCH A,@RW2-d8
+ 3	ROL R3	ROL @RW3-d8	RORC R3	RORC @RW3-d8	INC R3	INC @RW3-d8	DEC R3	DEC @RW3-d8	MOV A,R3	MOV A,@RW3-d8	MOV R3,A	MOV @R,W3-d8A	MOVX A,R3	MOVX A,@RW3-d8	XCH A,R3	XCH A,@RW3-d8
+ 4	ROL R4	ROL @RW4-d8	RORC R4	RORC @RW4-d8	INC R4	INC @RW4-d8	DEC R4	DEC @RW4-d8	MOV A,R4	MOV A,@RW4-d8	MOV R4,A	MOV @R,W4-d8A	MOVX A,R4	MOVX A,@RW4-d8	XCH A,R4	XCH A,@RW4-d8
+ 5	ROL R5	ROL @RW5-d8	RORC R5	RORC @RW5-d8	INC R5	INC @RW5-d8	DEC R5	DEC @RW5-d8	MOV A,R5	MOV A,@RW5-d8	MOV R5,A	MOV @R,W5-d8A	MOVX A,R5	MOVX A,@RW5-d8	XCH A,R5	XCH A,@RW5-d8
+ 6	ROL R6	ROL @RW6-d8	RORC R6	RORC @RW6-d8	INC R6	INC @RW6-d8	DEC R6	DEC @RW6-d8	MOV A,R6	MOV A,@RW6-d8	MOV R6,A	MOV @R,W6-d8A	MOVX A,R6	MOVX A,@RW6-d8	XCH A,R6	XCH A,@RW6-d8
+ 7	ROL R7	ROL @RW7-d8	RORC R7	RORC @RW7-d8	INC R7	INC @RW7-d8	DEC R7	DEC @RW7-d8	MOV A,R7	MOV A,@RW7-d8	MOV R7,A	MOV @R,W7-d8A	MOVX A,R7	MOVX A,@RW7-d8	XCH A,R7	XCH A,@RW7-d8
+ 8	ROL @RW0	ROL @RW0-d16	RORC @RW0	RORC @RW0-d16	INC @RW0	INC @RW0-d16	DEC @RW0	DEC @RW0-d16	MOV A,@RW0	MOV A,@RW0-d16	MOV @RW0,A	MOV @R,W0-d16A	MOVX A,@RW0	MOVX A,@RW0-d16	XCH A,@RW0	XCH A,@RW0-d16
+ 9	ROL @RW1	ROL @RW1-d16	RORC @RW1	RORC @RW1-d16	INC @RW1	INC @RW1-d16	DEC @RW1	DEC @RW1-d16	MOV A,@RW1	MOV A,@RW1-d16	MOV @RW1,A	MOV @R,W1-d16A	MOVX A,@RW1	MOVX A,@RW1-d16	XCH A,@RW1	XCH A,@RW1-d16
+ A	ROL @RW2	ROL @RW2-d16	RORC @RW2	RORC @RW2-d16	INC @RW2	INC @RW2-d16	DEC @RW2	DEC @RW2-d16	MOV A,@RW2	MOV A,@RW2-d16	MOV @RW2,A	MOV @R,W2-d16A	MOVX A,@RW2	MOVX A,@RW2-d16	XCH A,@RW2	XCH A,@RW2-d16
+ B	ROL @RW3	ROL @RW3-d16	RORC @RW3	RORC @RW3-d16	INC @RW3	INC @RW3-d16	DEC @RW3	DEC @RW3-d16	MOV A,@RW3	MOV A,@RW3-d16	MOV @RW3,A	MOV @R,W3-d16A	MOVX A,@RW3	MOVX A,@RW3-d16	XCH A,@RW3	XCH A,@RW3-d16
+ C	ROL @RW0+	ROL @RW0-RW7	RORC @RW0+	RORC @RW0-RW7	INC @RW0+	INC @RW0-RW7	DEC @RW0+	DEC @RW0-RW7	MOV A,@RW0+	MOV A,@RW0-RW7	MOV @RW0+,A	MOV @R,W0-RW7A	MOVX A,@RW0+	MOVX A,@RW0-RW7	XCH A,@RW0+	XCH A,@RW0-RW7
+ D	ROL @RW1+	ROL @RW1-RW7	RORC @RW1+	RORC @RW1-RW7	INC @RW1+	INC @RW1-RW7	DEC @RW1+	DEC @RW1-RW7	MOV A,@RW1+	MOV A,@RW1-RW7	MOV @RW1+,A	MOV @R,W1-RW7A	MOVX A,@RW1+	MOVX A,@RW1-RW7	XCH A,@RW1+	XCH A,@RW1-RW7
+ E	ROL @RW2+	ROL @PC-d16	RORC @RW2+	RORC @PC-d16	INC @RW2+	INC @PC-d16	DEC @RW2+	DEC @PC-d16	MOV A,@RW2+	MOV A,@PC-d16	MOV @RW2+,A	MOV C-d16A	MOVX A,@RW2+	MOVX A,@PC-d16	XCH A,@RW2+	XCH A,@PC-d16
+ F	ROL @RW3+	ROL addr16	RORC @RW3+	RORC addr16	INC @RW3+	INC addr16	DEC @RW3+	DEC addr16	MOV A,@RW3+	MOV addr16	MOV @RW3+,A	MOV addr16A	MOVX A,@RW3+	MOVX addr16	XCH A,@RW3+	XCH addr16

Table B.9-9 ea instruction 4 (first byte = 73_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMP @RW0	JMP @RW0-d8	CALL @RW0	CALL @RW0-d8	INCW RW0	INCW @RW0-d8	DECW RW0	DECW @RW0-d8	MOVW A, RW0	MOVW A, @RW0-d8	MOVW RW0, A	MOVW @RW0-d8, A	MOVW RW0, A	MOVW @RW0-d8, A	XCHW A, RW0	XCHW A, @RW0-d8
+ 1	JMP @RW1	JMP @RW1-d8	CALL @RW1	CALL @RW1-d8	INCW RW1	INCW @RW1-d8	DECW RW1	DECW @RW1-d8	MOVW A, RW1	MOVW A, @RW1-d8	MOVW RW1, A	MOVW @RW1-d8, A	MOVW RW1, A	MOVW @RW1-d8, A	XCHW A, RW1	XCHW A, @RW1-d8
+ 2	JMP @RW2	JMP @RW2-d8	CALL @RW2	CALL @RW2-d8	INCW RW2	INCW @RW2-d8	DECW RW2	DECW @RW2-d8	MOVW A, RW2	MOVW A, @RW2-d8	MOVW RW2, A	MOVW @RW2-d8, A	MOVW RW2, A	MOVW @RW2-d8, A	XCHW A, RW2	XCHW A, @RW2-d8
+ 3	JMP @RW3	JMP @RW3-d8	CALL @RW3	CALL @RW3-d8	INCW RW3	INCW @RW3-d8	DECW RW3	DECW @RW3-d8	MOVW A, RW3	MOVW A, @RW3-d8	MOVW RW3, A	MOVW @RW3-d8, A	MOVW RW3, A	MOVW @RW3-d8, A	XCHW A, RW3	XCHW A, @RW3-d8
+ 4	JMP @RW4	JMP @RW4-d8	CALL @RW4	CALL @RW4-d8	INCW RW4	INCW @RW4-d8	DECW RW4	DECW @RW4-d8	MOVW A, RW4	MOVW A, @RW4-d8	MOVW RW4, A	MOVW @RW4-d8, A	MOVW RW4, A	MOVW @RW4-d8, A	XCHW A, RW4	XCHW A, @RW4-d8
+ 5	JMP @RW5	JMP @RW5-d8	CALL @RW5	CALL @RW5-d8	INCW RW5	INCW @RW5-d8	DECW RW5	DECW @RW5-d8	MOVW A, RW5	MOVW A, @RW5-d8	MOVW RW5, A	MOVW @RW5-d8, A	MOVW RW5, A	MOVW @RW5-d8, A	XCHW A, RW5	XCHW A, @RW5-d8
+ 6	JMP @RW6	JMP @RW6-d8	CALL @RW6	CALL @RW6-d8	INCW RW6	INCW @RW6-d8	DECW RW6	DECW @RW6-d8	MOVW A, RW6	MOVW A, @RW6-d8	MOVW RW6, A	MOVW @RW6-d8, A	MOVW RW6, A	MOVW @RW6-d8, A	XCHW A, RW6	XCHW A, @RW6-d8
+ 7	JMP @RW7	JMP @RW7-d8	CALL @RW7	CALL @RW7-d8	INCW RW7	INCW @RW7-d8	DECW RW7	DECW @RW7-d8	MOVW A, RW7	MOVW A, @RW7-d8	MOVW RW7, A	MOVW @RW7-d8, A	MOVW RW7, A	MOVW @RW7-d8, A	XCHW A, RW7	XCHW A, @RW7-d8
+ 8	JMP @RW0	JMP @RW0-d16	CALL @RW0	CALL @RW0-d16	INCW @RW0	INCW @RW0-d16	DECW @RW0	DECW @RW0-d16	MOVW A, @RW0	MOVW A, @RW0-d16	MOVW @RW0, A	MOVW @RW0-d16, A	MOVW @RW0, A	MOVW @RW0-d16, A	XCHW A, @RW0	XCHW A, @RW0-d16
+ 9	JMP @RW1	JMP @RW1-d16	CALL @RW1	CALL @RW1-d16	INCW @RW1	INCW @RW1-d16	DECW @RW1	DECW @RW1-d16	MOVW A, @RW1	MOVW A, @RW1-d16	MOVW @RW1, A	MOVW @RW1-d16, A	MOVW @RW1, A	MOVW @RW1-d16, A	XCHW A, @RW1	XCHW A, @RW1-d16
+ A	JMP @RW2	JMP @RW2-d16	CALL @RW2	CALL @RW2-d16	INCW @RW2	INCW @RW2-d16	DECW @RW2	DECW @RW2-d16	MOVW A, @RW2	MOVW A, @RW2-d16	MOVW @RW2, A	MOVW @RW2-d16, A	MOVW @RW2, A	MOVW @RW2-d16, A	XCHW A, @RW2	XCHW A, @RW2-d16
+ B	JMP @RW3	JMP @RW3-d16	CALL @RW3	CALL @RW3-d16	INCW @RW3	INCW @RW3-d16	DECW @RW3	DECW @RW3-d16	MOVW A, @RW3	MOVW A, @RW3-d16	MOVW @RW3, A	MOVW @RW3-d16, A	MOVW @RW3, A	MOVW @RW3-d16, A	XCHW A, @RW3	XCHW A, @RW3-d16
+ C	JMP @RW0+	JMP @RW0-RW7	CALL @RW0+	CALL @RW0-RW7	INCW @RW0+	INCW @RW0-RW7	DECW @RW0+	DECW @RW0-RW7	MOVW A, @RW0+	MOVW A, @RW0-RW7	MOVW @RW0+, A	MOVW @RW0-RW7, A	MOVW @RW0+, A	MOVW @RW0-RW7, A	XCHW A, @RW0+	XCHW A, @RW0-RW7
+ D	JMP @RW1+	JMP @RW1+RW7	CALL @RW1+	CALL @RW1+RW7	INCW @RW1+	INCW @RW1+RW7	DECW @RW1+	DECW @RW1+RW7	MOVW A, @RW1+	MOVW A, @RW1+RW7	MOVW @RW1+, A	MOVW @RW1+RW7, A	MOVW @RW1+, A	MOVW @RW1+RW7, A	XCHW A, @RW1+	XCHW A, @RW1+RW7
+ E	JMP @RW2+	JMP @PC-d16	CALL @RW2+	CALL @PC-d16	INCW @RW2+	INCW @PC-d16	DECW @RW2+	DECW @PC-d16	MOVW A, @RW2+	MOVW A, @PC-d16	MOVW @RW2+, A	MOVW @PC-d16, A	MOVW @RW2+, A	MOVW @PC-d16, A	XCHW A, @RW2+	XCHW A, @PC-d16
+ F	JMP @RW3+	JMP @addr16	CALL @RW3+	CALL @addr16	INCW @RW3+	INCW @addr16	DECW @RW3+	DECW @addr16	MOVW A, @RW3+	MOVW A, @addr16	MOVW @RW3+, A	MOVW @addr16, A	MOVW @RW3+, A	MOVW @addr16, A	XCHW A, @RW3+	XCHW A, @addr16

Table B.9-10 ea instruction 5 (first byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADD A.R0	ADD A, @RW0-d8	SUB A.R0	SUB A, @RW0-d8	ADDC A.R0	ADDC 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	ADDC A.R1	ADDC 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	ADDC A.R2	ADDC 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	ADDC A.R3	ADDC 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	ADDC A.R4	ADDC 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	ADDC A.R5	ADDC 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	ADDC A.R6	ADDC 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	ADDC A.R7	ADDC 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	ADDC A.@RW0	ADDC 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	ADDC A.@RW1	ADDC 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	ADDC A.@RW2	ADDC 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	ADDC A.@RW3	ADDC 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	ADDC A.@RW0+	ADDC 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	ADDC A.@RW1+	ADDC 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, @PC-d16	SUB A.@RW2+	SUB A, @PC-d16	ADDC A.@RW2+	ADDC A, @PC-d16	CMP A.@RW2+	CMP A, @PC-d16	AND A.@RW2+	AND A, @PC-d16	OR A.@RW2+	OR A, @PC-d16	XOR A.@RW2+	XOR A, @PC-d16	DBNZ @RW2+,r	DBNZ @R PC-d16,r
+ F	ADD A.@RW3+	ADD A, addr16	SUB A.@RW3+	SUB A, addr16	ADDC A.@RW3+	ADDC A, addr16	CMP A.@RW3+	CMP A, addr16	AND A.@RW3+	AND A, addr16	OR A.@RW3+	OR A, addr16	XOR A.@RW3+	XOR A, addr16	DBNZ @RW3+,r	DBNZ addr16,r

Table B.9-11 ea instruction 6 (first byte = 75_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADD R0,A	ADD @R W0-d8,A	SUB R0,A	SUB @R W0-d8,A	SUBC A,R0	SUBC A, @RW0-d8	NEG R0	NEG @RW0-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	NOT @RW0-d8
+ 1	ADD R1,A	ADD @R W1-d8,A	SUB R1,A	SUB @R W1-d8,A	SUBC 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	NOT @RW1-d8
+ 2	ADD R2,A	ADD @R W2-d8,A	SUB R2,A	SUB @R W2-d8,A	SUBC 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	NOT @RW2-d8
+ 3	ADD R3,A	ADD @R W3-d8,A	SUB R3,A	SUB @R W3-d8,A	SUBC 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	NOT @RW3-d8
+ 4	ADD R4,A	ADD @R W4-d8,A	SUB R4,A	SUB @R W4-d8,A	SUBC 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	NOT @RW4-d8
+ 5	ADD R5,A	ADD @R W5-d8,A	SUB R5,A	SUB @R W5-d8,A	SUBC 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	NOT @RW5-d8
+ 6	ADD R6,A	ADD @R W6-d8,A	SUB R6,A	SUB @R W6-d8,A	SUBC 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	NOT @RW6-d8
+ 7	ADD R7,A	ADD @R W7-d8,A	SUB R7,A	SUB @R W7-d8,A	SUBC 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	NOT @RW7-d8
+ 8	ADD @RW0,A	ADD @R W0-d16,A	SUB @RW0,A	SUB @R W0-d16,A	SUBC 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	NOT @RW0-d16
+ 9	ADD @RW1,A	ADD @R W1-d16,A	SUB @RW1,A	SUB @R W1-d16,A	SUBC 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	NOT @RW1-d16
+ A	ADD @RW2,A	ADD @R W2-d16,A	SUB @RW2,A	SUB @R W2-d16,A	SUBC 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	NOT @RW2-d16
+ B	ADD @RW3,A	ADD @R W3-d16,A	SUB @RW3,A	SUB @R W3-d16,A	SUBC 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	NOT @RW3-d16
+ C	ADD @RW0+ ,A	ADD @R W0-RW7,A	SUB @RW0+ ,A	SUB @R W0-RW7,A	SUBC 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+	NOT @RW0-RW7
+ D	ADD @RW1+ ,A	ADD @R W1-RW7,A	SUB @RW1+ ,A	SUB @R W1-RW7,A	SUBC 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+	NOT @RW1-RW7
+ E	ADD @RW2+ ,A	ADD @P C-d16,A	SUB @RW2+ ,A	SUB @P C-d16,A	SUBC A,@RW2+	SUBC A, @PC-d16	NEG @RW2+	NEG @PC-d16	AND @RW2+ ,A	AND @P C-d16,A	OR @RW2+ ,A	OR @P C-d16,A	XOR @RW2+ ,A	XOR @P C-d16,A	NOT @RW2+	NOT @PC-d16
+ F	ADD @RW3+ ,A	ADD addr16,A	SUB @RW3+ ,A	SUB addr16,A	SUBC A,@RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+ ,A	AND addr16,A	OR @RW3+ ,A	OR addr16,A	XOR @RW3+ ,A	XOR addr16,A	NOT @RW3+	NOT addr16

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, A, RW0 @RW0-d8	SUBW A, A, RW0 @RW0-d8	SUBW A, A, RW0 @RW0-d8	SUBW A, A, RW0 @RW0-d8	ADDCW A, A, RW0 @RW0-d8	ADDCW A, A, RW0 @RW0-d8	CMPW A, A, RW0 @RW0-d8	CMPW A, A, RW0 @RW0-d8	ANDW A, A, RW0 @RW0-d8	ANDW A, A, RW0 @RW0-d8	ORW A, A, RW0 @RW0-d8	ORW A, A, RW0 @RW0-d8	XORW A, A, RW0 @RW0-d8	XORW A, A, RW0 @RW0-d8	DWBNZ @RW0-d8	DWBNZ @RW0-d8
+ 1	ADDW A, A, RW1 @RW1-d8	SUBW A, A, RW1 @RW1-d8	SUBW A, A, RW1 @RW1-d8	SUBW A, A, RW1 @RW1-d8	ADDCW A, A, RW1 @RW1-d8	ADDCW A, A, RW1 @RW1-d8	CMPW A, A, RW1 @RW1-d8	CMPW A, A, RW1 @RW1-d8	ANDW A, A, RW1 @RW1-d8	ANDW A, A, RW1 @RW1-d8	ORW A, A, RW1 @RW1-d8	ORW A, A, RW1 @RW1-d8	XORW A, A, RW1 @RW1-d8	XORW A, A, RW1 @RW1-d8	DWBNZ @RW1-d8	DWBNZ @RW1-d8
+ 2	ADDW A, A, RW2 @RW2-d8	SUBW A, A, RW2 @RW2-d8	SUBW A, A, RW2 @RW2-d8	SUBW A, A, RW2 @RW2-d8	ADDCW A, A, RW2 @RW2-d8	ADDCW A, A, RW2 @RW2-d8	CMPW A, A, RW2 @RW2-d8	CMPW A, A, RW2 @RW2-d8	ANDW A, A, RW2 @RW2-d8	ANDW A, A, RW2 @RW2-d8	ORW A, A, RW2 @RW2-d8	ORW A, A, RW2 @RW2-d8	XORW A, A, RW2 @RW2-d8	XORW A, A, RW2 @RW2-d8	DWBNZ @RW2-d8	DWBNZ @RW2-d8
+ 3	ADDW A, A, RW3 @RW3-d8	SUBW A, A, RW3 @RW3-d8	SUBW A, A, RW3 @RW3-d8	SUBW A, A, RW3 @RW3-d8	ADDCW A, A, RW3 @RW3-d8	ADDCW A, A, RW3 @RW3-d8	CMPW A, A, RW3 @RW3-d8	CMPW A, A, RW3 @RW3-d8	ANDW A, A, RW3 @RW3-d8	ANDW A, A, RW3 @RW3-d8	ORW A, A, RW3 @RW3-d8	ORW A, A, RW3 @RW3-d8	XORW A, A, RW3 @RW3-d8	XORW A, A, RW3 @RW3-d8	DWBNZ @RW3-d8	DWBNZ @RW3-d8
+ 4	ADDW A, A, RW4 @RW4-d8	SUBW A, A, RW4 @RW4-d8	SUBW A, A, RW4 @RW4-d8	SUBW A, A, RW4 @RW4-d8	ADDCW A, A, RW4 @RW4-d8	ADDCW A, A, RW4 @RW4-d8	CMPW A, A, RW4 @RW4-d8	CMPW A, A, RW4 @RW4-d8	ANDW A, A, RW4 @RW4-d8	ANDW A, A, RW4 @RW4-d8	ORW A, A, RW4 @RW4-d8	ORW A, A, RW4 @RW4-d8	XORW A, A, RW4 @RW4-d8	XORW A, A, RW4 @RW4-d8	DWBNZ @RW4-d8	DWBNZ @RW4-d8
+ 5	ADDW A, A, RW5 @RW5-d8	SUBW A, A, RW5 @RW5-d8	SUBW A, A, RW5 @RW5-d8	SUBW A, A, RW5 @RW5-d8	ADDCW A, A, RW5 @RW5-d8	ADDCW A, A, RW5 @RW5-d8	CMPW A, A, RW5 @RW5-d8	CMPW A, A, RW5 @RW5-d8	ANDW A, A, RW5 @RW5-d8	ANDW A, A, RW5 @RW5-d8	ORW A, A, RW5 @RW5-d8	ORW A, A, RW5 @RW5-d8	XORW A, A, RW5 @RW5-d8	XORW A, A, RW5 @RW5-d8	DWBNZ @RW5-d8	DWBNZ @RW5-d8
+ 6	ADDW A, A, RW6 @RW6-d8	SUBW A, A, RW6 @RW6-d8	SUBW A, A, RW6 @RW6-d8	SUBW A, A, RW6 @RW6-d8	ADDCW A, A, RW6 @RW6-d8	ADDCW A, A, RW6 @RW6-d8	CMPW A, A, RW6 @RW6-d8	CMPW A, A, RW6 @RW6-d8	ANDW A, A, RW6 @RW6-d8	ANDW A, A, RW6 @RW6-d8	ORW A, A, RW6 @RW6-d8	ORW A, A, RW6 @RW6-d8	XORW A, A, RW6 @RW6-d8	XORW A, A, RW6 @RW6-d8	DWBNZ @RW6-d8	DWBNZ @RW6-d8
+ 7	ADDW A, A, RW7 @RW7-d8	SUBW A, A, RW7 @RW7-d8	SUBW A, A, RW7 @RW7-d8	SUBW A, A, RW7 @RW7-d8	ADDCW A, A, RW7 @RW7-d8	ADDCW A, A, RW7 @RW7-d8	CMPW A, A, RW7 @RW7-d8	CMPW A, A, RW7 @RW7-d8	ANDW A, A, RW7 @RW7-d8	ANDW A, A, RW7 @RW7-d8	ORW A, A, RW7 @RW7-d8	ORW A, A, RW7 @RW7-d8	XORW A, A, RW7 @RW7-d8	XORW A, A, RW7 @RW7-d8	DWBNZ @RW7-d8	DWBNZ @RW7-d8
+ 8	ADDW A, A, RW0 @RW0-d16	SUBW A, A, RW0 @RW0-d16	SUBW A, A, RW0 @RW0-d16	SUBW A, A, RW0 @RW0-d16	ADDCW A, A, RW0 @RW0-d16	ADDCW A, A, RW0 @RW0-d16	CMPW A, A, RW0 @RW0-d16	CMPW A, A, RW0 @RW0-d16	ANDW A, A, RW0 @RW0-d16	ANDW A, A, RW0 @RW0-d16	ORW A, A, RW0 @RW0-d16	ORW A, A, RW0 @RW0-d16	XORW A, A, RW0 @RW0-d16	XORW A, A, RW0 @RW0-d16	DWBNZ @RW0-d16	DWBNZ @RW0-d16
+ 9	ADDW A, A, RW1 @RW1-d16	SUBW A, A, RW1 @RW1-d16	SUBW A, A, RW1 @RW1-d16	SUBW A, A, RW1 @RW1-d16	ADDCW A, A, RW1 @RW1-d16	ADDCW A, A, RW1 @RW1-d16	CMPW A, A, RW1 @RW1-d16	CMPW A, A, RW1 @RW1-d16	ANDW A, A, RW1 @RW1-d16	ANDW A, A, RW1 @RW1-d16	ORW A, A, RW1 @RW1-d16	ORW A, A, RW1 @RW1-d16	XORW A, A, RW1 @RW1-d16	XORW A, A, RW1 @RW1-d16	DWBNZ @RW1-d16	DWBNZ @RW1-d16
+ A	ADDW A, A, RW2 @RW2-d16	SUBW A, A, RW2 @RW2-d16	SUBW A, A, RW2 @RW2-d16	SUBW A, A, RW2 @RW2-d16	ADDCW A, A, RW2 @RW2-d16	ADDCW A, A, RW2 @RW2-d16	CMPW A, A, RW2 @RW2-d16	CMPW A, A, RW2 @RW2-d16	ANDW A, A, RW2 @RW2-d16	ANDW A, A, RW2 @RW2-d16	ORW A, A, RW2 @RW2-d16	ORW A, A, RW2 @RW2-d16	XORW A, A, RW2 @RW2-d16	XORW A, A, RW2 @RW2-d16	DWBNZ @RW2-d16	DWBNZ @RW2-d16
+ B	ADDW A, A, RW3 @RW3-d16	SUBW A, A, RW3 @RW3-d16	SUBW A, A, RW3 @RW3-d16	SUBW A, A, RW3 @RW3-d16	ADDCW A, A, RW3 @RW3-d16	ADDCW A, A, RW3 @RW3-d16	CMPW A, A, RW3 @RW3-d16	CMPW A, A, RW3 @RW3-d16	ANDW A, A, RW3 @RW3-d16	ANDW A, A, RW3 @RW3-d16	ORW A, A, RW3 @RW3-d16	ORW A, A, RW3 @RW3-d16	XORW A, A, RW3 @RW3-d16	XORW A, A, RW3 @RW3-d16	DWBNZ @RW3-d16	DWBNZ @RW3-d16
+ C	ADDW A, A, RW0+ @RW0-RW7	SUBW A, A, RW0+ @RW0-RW7	SUBW A, A, RW0+ @RW0-RW7	SUBW A, A, RW0+ @RW0-RW7	ADDCW A, A, RW0+ @RW0-RW7	ADDCW A, A, RW0+ @RW0-RW7	CMPW A, A, RW0+ @RW0-RW7	CMPW A, A, RW0+ @RW0-RW7	ANDW A, A, RW0+ @RW0-RW7	ANDW A, A, RW0+ @RW0-RW7	ORW A, A, RW0+ @RW0-RW7	ORW A, A, RW0+ @RW0-RW7	XORW A, A, RW0+ @RW0-RW7	XORW A, A, RW0+ @RW0-RW7	DWBNZ @RW0+ @RW0-RW7	DWBNZ @RW0+ @RW0-RW7
+ D	ADDW A, A, RW1+ @RW1-RW7	SUBW A, A, RW1+ @RW1-RW7	SUBW A, A, RW1+ @RW1-RW7	SUBW A, A, RW1+ @RW1-RW7	ADDCW A, A, RW1+ @RW1-RW7	ADDCW A, A, RW1+ @RW1-RW7	CMPW A, A, RW1+ @RW1-RW7	CMPW A, A, RW1+ @RW1-RW7	ANDW A, A, RW1+ @RW1-RW7	ANDW A, A, RW1+ @RW1-RW7	ORW A, A, RW1+ @RW1-RW7	ORW A, A, RW1+ @RW1-RW7	XORW A, A, RW1+ @RW1-RW7	XORW A, A, RW1+ @RW1-RW7	DWBNZ @RW1+ @RW1-RW7	DWBNZ @RW1+ @RW1-RW7
+ E	ADDW A, A, RW2+ @PC-d16	SUBW A, A, RW2+ @PC-d16	SUBW A, A, RW2+ @PC-d16	SUBW A, A, RW2+ @PC-d16	ADDCW A, A, RW2+ @PC-d16	ADDCW A, A, RW2+ @PC-d16	CMPW A, A, RW2+ @PC-d16	CMPW A, A, RW2+ @PC-d16	ANDW A, A, RW2+ @PC-d16	ANDW A, A, RW2+ @PC-d16	ORW A, A, RW2+ @PC-d16	ORW A, A, RW2+ @PC-d16	XORW A, A, RW2+ @PC-d16	XORW A, A, RW2+ @PC-d16	DWBNZ @RW2+ @PC-d16	DWBNZ @RW2+ @PC-d16
+ F	ADDW A, A, RW3+ addr16	SUBW A, A, RW3+ addr16	SUBW A, A, RW3+ addr16	SUBW A, A, RW3+ addr16	ADDCW A, A, RW3+ addr16	ADDCW A, A, RW3+ addr16	CMPW A, A, RW3+ addr16	CMPW A, A, RW3+ addr16	ANDW A, A, RW3+ addr16	ANDW A, A, RW3+ addr16	ORW A, A, RW3+ addr16	ORW A, A, RW3+ addr16	XORW A, A, RW3+ addr16	XORW A, A, RW3+ addr16	DWBNZ @RW3+ addr16	DWBNZ @RW3+ addr16

Table B.9-13 ea instruction 8 (first byte = 77_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW @RW0.A	ADDW @R W0-d8.A	SUBW @RW0.A	SUBW @R W0-d8.A	SUBCW A, RW0.A	SUBCW A, @RW0-d8	NEGW RW0	NEGW @RW0-d8	ANDW RW0.A	ANDW @R W0-d8.A	ORW RW0.A	ORW @R W0-d8.A	XORW RW0.A	XORW @R W0-d8.A	NOTW RW0	NOTW @RW0-d8
+ 1	ADDW RW1.A	ADDW @R W1-d8.A	SUBW RW1.A	SUBW @R W1-d8.A	SUBCW A, RW1	SUBCW A, @RW1-d8	NEGW RW1	NEGW @RW1-d8	ANDW RW1.A	ANDW @R W1-d8.A	ORW RW1.A	ORW @R W1-d8.A	XORW RW1.A	XORW @R W1-d8.A	NOTW RW1	NOTW @RW1-d8
+ 2	ADDW RW2.A	ADDW @R W2-d8.A	SUBW RW2.A	SUBW @R W2-d8.A	SUBCW A, RW2	SUBCW A, @RW2-d8	NEGW RW2	NEGW @RW2-d8	ANDW RW2.A	ANDW @R W2-d8.A	ORW RW2.A	ORW @R W2-d8.A	XORW RW2.A	XORW @R W2-d8.A	NOTW RW2	NOTW @RW2-d8
+ 3	ADDW RW3.A	ADDW @R W3-d8.A	SUBW RW3.A	SUBW @R W3-d8.A	SUBCW A, RW3	SUBCW A, @RW3-d8	NEGW RW3	NEGW @RW3-d8	ANDW RW3.A	ANDW @R W3-d8.A	ORW RW3.A	ORW @R W3-d8.A	XORW RW3.A	XORW @R W3-d8.A	NOTW RW3	NOTW @RW3-d8
+ 4	ADDW RW4.A	ADDW @R W4-d8.A	SUBW RW4.A	SUBW @R W4-d8.A	SUBCW A, RW4	SUBCW A, @RW4-d8	NEGW RW4	NEGW @RW4-d8	ANDW RW4.A	ANDW @R W4-d8.A	ORW RW4.A	ORW @R W4-d8.A	XORW RW4.A	XORW @R W4-d8.A	NOTW RW4	NOTW @RW4-d8
+ 5	ADDW RW5.A	ADDW @R W5-d8.A	SUBW RW5.A	SUBW @R W5-d8.A	SUBCW A, RW5	SUBCW A, @RW5-d8	NEGW RW5	NEGW @RW5-d8	ANDW RW5.A	ANDW @R W5-d8.A	ORW RW5.A	ORW @R W5-d8.A	XORW RW5.A	XORW @R W5-d8.A	NOTW RW5	NOTW @RW5-d8
+ 6	ADDW RW6.A	ADDW @R W6-d8.A	SUBW RW6.A	SUBW @R W6-d8.A	SUBCW A, RW6	SUBCW A, @RW6-d8	NEGW RW6	NEGW @RW6-d8	ANDW RW6.A	ANDW @R W6-d8.A	ORW RW6.A	ORW @R W6-d8.A	XORW RW6.A	XORW @R W6-d8.A	NOTW RW6	NOTW @RW6-d8
+ 7	ADDW RW7.A	ADDW @R W7-d8.A	SUBW RW7.A	SUBW @R W7-d8.A	SUBCW A, RW7	SUBCW A, @RW7-d8	NEGW RW7	NEGW @RW7-d8	ANDW RW7.A	ANDW @R W7-d8.A	ORW RW7.A	ORW @R W7-d8.A	XORW RW7.A	XORW @R W7-d8.A	NOTW RW7	NOTW @RW7-d8
+ 8	ADDW @RW0.A	ADDW @R W0-d16.A	SUBW @RW0.A	SUBW @R W0-d16.A	SUBCW A, @RW0	SUBCW A, @RW0-d16	NEGW @RW0	NEGW @RW0-d16	ANDW @RW0.A	ANDW @R W0-d16.A	ORW @RW0.A	ORW @R W0-d16.A	XORW @RW0.A	XORW @R W0-d16.A	NOTW @RW0	NOTW @RW0-d16
+ 9	ADDW @RW1.A	ADDW @R W1-d16.A	SUBW @RW1.A	SUBW @R W1-d16.A	SUBCW A, @RW1	SUBCW A, @RW1-d16	NEGW @RW1	NEGW @RW1-d16	ANDW @RW1.A	ANDW @R W1-d16.A	ORW @RW1.A	ORW @R W1-d16.A	XORW @RW1.A	XORW @R W1-d16.A	NOTW @RW1	NOTW @RW1-d16
+ A	ADDW @RW2.A	ADDW @R W2-d16.A	SUBW @RW2.A	SUBW @R W2-d16.A	SUBCW A, @RW2	SUBCW A, @RW2-d16	NEGW @RW2	NEGW @RW2-d16	ANDW @RW2.A	ANDW @R W2-d16.A	ORW @RW2.A	ORW @R W2-d16.A	XORW @RW2.A	XORW @R W2-d16.A	NOTW @RW2	NOTW @RW2-d16
+ B	ADDW @RW3.A	ADDW @R W3-d16.A	SUBW @RW3.A	SUBW @R W3-d16.A	SUBCW A, @RW3	SUBCW A, @RW3-d16	NEGW @RW3	NEGW @RW3-d16	ANDW @RW3.A	ANDW @R W3-d16.A	ORW @RW3.A	ORW @R W3-d16.A	XORW @RW3.A	XORW @R W3-d16.A	NOTW @RW3	NOTW @RW3-d16
+ C	ADDW @RW0+.A	ADDW @R W0-RW7.A	SUBW @RW0+.A	SUBW @R W0-RW7.A	SUBCW A, @RW0+	SUBCW A, @RW0-RW7	NEGW @RW0+	NEGW @RW0-RW7	ANDW @RW0+.A	ANDW @R W0-RW7.A	ORW @RW0+.A	ORW @R W0-RW7.A	XORW @RW0+.A	XORW @R W0-RW7.A	NOTW @RW0+	NOTW @RW0-RW7
+ D	ADDW @RW1+.A	ADDW @R W1-RW7.A	SUBW @RW1+.A	SUBW @R W1-RW7.A	SUBCW A, @RW1+	SUBCW A, @RW1-RW7	NEGW @RW1+	NEGW @RW1-RW7	ANDW @RW1+.A	ANDW @R W1-RW7.A	ORW @RW1+.A	ORW @R W1-RW7.A	XORW @RW1+.A	XORW @R W1-RW7.A	NOTW @RW1+	NOTW @RW1-RW7
+ E	ADDW @RW2+.A	ADDW @P C-d16.A	SUBW @RW2+.A	SUBW @P C-d16.A	SUBCW A, @RW2+	SUBCW A, @PC-d16	NEGW @RW2+	NEGW @PC-d16	ANDW @RW2+.A	ANDW @P C-d16.A	ORW @RW2+.A	ORW @P C-d16.A	XORW @RW2+.A	XORW @P C-d16.A	NOTW @RW2+	NOTW @PC-d16
+ F	ADDW @RW3+.A	ADDW addr16.A	SUBW @RW3+.A	SUBW addr16.A	SUBCW A, @RW3+	SUBCW A, addr16	NEGW @RW3+	NEGW addr16	ANDW @RW3+.A	ANDW addr16.A	ORW @RW3+.A	ORW addr16.A	XORW @RW3+.A	XORW addr16.A	NOTW @RW3+	NOTW 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	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0	MUL A.R0
+ 1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1	MUL A.R1
+ 2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2	MUL A.R2
+ 3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3	MUL A.R3
+ 4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4	MUL A.R4
+ 5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5	MUL A.R5
+ 6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6	MUL A.R6
+ 7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7	MUL A.R7
+ 8	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0	MUL A.RW0
+ 9	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1	MUL A.RW1
+ A	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2	MUL A.RW2
+ B	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3	MUL A.RW3
+ C	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+	MUL A.RW0+
+ D	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+	MUL A.RW1+
+ E	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+	MUL A.RW2+
+ F	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+	MUL A.RW3+

Table B.9-15 MOVEA RWi, ea instruction (first byte = 79_H)

[illegible]

Table B.9-16 MOV Ri, ea instruction (first byte = 7Ah)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0,R0 @RW0-d8	MOV R0,R0 @RW0-d8	MOV R1,R0 @RW0-d8	MOV R1,R1 @RW0-d8	MOV R2,R0 @RW0-d8	MOV R2,R1 @RW0-d8	MOV R3,R0 @RW0-d8	MOV R3,R3 @RW0-d8	MOV R4,R0 @RW0-d8	MOV R4,R4 @RW0-d8	MOV R5,R0 @RW0-d8	MOV R5,R5 @RW0-d8	MOV R6,R0 @RW0-d8	MOV R6,R6 @RW0-d8	MOV R7,R0 @RW0-d8	MOV R7,R7 @RW0-d8
+ 1	MOV R0,R1 @RW1-d8	MOV R0,R1 @RW1-d8	MOV R1,R1 @RW1-d8	MOV R1,R1 @RW1-d8	MOV R2,R1 @RW1-d8	MOV R2,R1 @RW1-d8	MOV R3,R1 @RW1-d8	MOV R3,R1 @RW1-d8	MOV R4,R1 @RW1-d8	MOV R4,R1 @RW1-d8	MOV R5,R1 @RW1-d8	MOV R5,R5 @RW1-d8	MOV R6,R1 @RW1-d8	MOV R6,R6 @RW1-d8	MOV R7,R1 @RW1-d8	MOV R7,R7 @RW1-d8
+ 2	MOV R0,R2 @RW2-d8	MOV R0,R2 @RW2-d8	MOV R1,R2 @RW2-d8	MOV R1,R2 @RW2-d8	MOV R2,R2 @RW2-d8	MOV R2,R2 @RW2-d8	MOV R3,R2 @RW2-d8	MOV R3,R2 @RW2-d8	MOV R4,R2 @RW2-d8	MOV R4,R2 @RW2-d8	MOV R5,R2 @RW2-d8	MOV R5,R5 @RW2-d8	MOV R6,R2 @RW2-d8	MOV R6,R6 @RW2-d8	MOV R7,R2 @RW2-d8	MOV R7,R7 @RW2-d8
+ 3	MOV R0,R3 @RW3-d8	MOV R0,R3 @RW3-d8	MOV R1,R3 @RW3-d8	MOV R1,R3 @RW3-d8	MOV R2,R3 @RW3-d8	MOV R2,R3 @RW3-d8	MOV R3,R3 @RW3-d8	MOV R3,R3 @RW3-d8	MOV R4,R3 @RW3-d8	MOV R4,R4 @RW3-d8	MOV R5,R3 @RW3-d8	MOV R5,R5 @RW3-d8	MOV R6,R3 @RW3-d8	MOV R6,R6 @RW3-d8	MOV R7,R3 @RW3-d8	MOV R7,R7 @RW3-d8
+ 4	MOV R0,R4 @RW4-d8	MOV R0,R4 @RW4-d8	MOV R1,R4 @RW4-d8	MOV R1,R4 @RW4-d8	MOV R2,R4 @RW4-d8	MOV R2,R4 @RW4-d8	MOV R3,R4 @RW4-d8	MOV R3,R4 @RW4-d8	MOV R4,R4 @RW4-d8	MOV R4,R4 @RW4-d8	MOV R5,R4 @RW4-d8	MOV R5,R5 @RW4-d8	MOV R6,R4 @RW4-d8	MOV R6,R6 @RW4-d8	MOV R7,R4 @RW4-d8	MOV R7,R7 @RW4-d8
+ 5	MOV R0,R5 @RW5-d8	MOV R0,R5 @RW5-d8	MOV R1,R5 @RW5-d8	MOV R1,R5 @RW5-d8	MOV R2,R5 @RW5-d8	MOV R2,R5 @RW5-d8	MOV R3,R5 @RW5-d8	MOV R3,R5 @RW5-d8	MOV R4,R5 @RW5-d8	MOV R4,R5 @RW5-d8	MOV R5,R5 @RW5-d8	MOV R5,R5 @RW5-d8	MOV R6,R5 @RW5-d8	MOV R6,R6 @RW5-d8	MOV R7,R5 @RW5-d8	MOV R7,R7 @RW5-d8
+ 6	MOV R0,R6 @RW6-d8	MOV R0,R6 @RW6-d8	MOV R1,R6 @RW6-d8	MOV R1,R6 @RW6-d8	MOV R2,R6 @RW6-d8	MOV R2,R6 @RW6-d8	MOV R3,R6 @RW6-d8	MOV R3,R6 @RW6-d8	MOV R4,R6 @RW6-d8	MOV R4,R6 @RW6-d8	MOV R5,R6 @RW6-d8	MOV R5,R5 @RW6-d8	MOV R6,R6 @RW6-d8	MOV R6,R6 @RW6-d8	MOV R7,R6 @RW6-d8	MOV R7,R7 @RW6-d8
+ 7	MOV R0,R7 @RW7-d8	MOV R0,R7 @RW7-d8	MOV R1,R7 @RW7-d8	MOV R1,R7 @RW7-d8	MOV R2,R7 @RW7-d8	MOV R2,R7 @RW7-d8	MOV R3,R7 @RW7-d8	MOV R3,R7 @RW7-d8	MOV R4,R7 @RW7-d8	MOV R4,R7 @RW7-d8	MOV R5,R7 @RW7-d8	MOV R5,R5 @RW7-d8	MOV R6,R7 @RW7-d8	MOV R6,R6 @RW7-d8	MOV R7,R7 @RW7-d8	MOV R7,R7 @RW7-d8
+ 8	MOV R0,R0 @RW0-d16	MOV R0,R0 @RW0-d16	MOV R1,R0 @RW0-d16	MOV R1,R1 @RW0-d16	MOV R2,R0 @RW0-d16	MOV R2,R1 @RW0-d16	MOV R3,R0 @RW0-d16	MOV R3,R3 @RW0-d16	MOV R4,R0 @RW0-d16	MOV R4,R4 @RW0-d16	MOV R5,R0 @RW0-d16	MOV R5,R5 @RW0-d16	MOV R6,R0 @RW0-d16	MOV R6,R6 @RW0-d16	MOV R7,R0 @RW0-d16	MOV R7,R7 @RW0-d16
+ 9	MOV R0,R1 @RW1-d16	MOV R0,R1 @RW1-d16	MOV R1,R1 @RW1-d16	MOV R1,R1 @RW1-d16	MOV R2,R1 @RW1-d16	MOV R2,R1 @RW1-d16	MOV R3,R1 @RW1-d16	MOV R3,R1 @RW1-d16	MOV R4,R1 @RW1-d16	MOV R4,R1 @RW1-d16	MOV R5,R1 @RW1-d16	MOV R5,R5 @RW1-d16	MOV R6,R1 @RW1-d16	MOV R6,R6 @RW1-d16	MOV R7,R1 @RW1-d16	MOV R7,R7 @RW1-d16
+ A	MOV R0,R2 @RW2-d16	MOV R0,R2 @RW2-d16	MOV R1,R2 @RW2-d16	MOV R1,R2 @RW2-d16	MOV R2,R2 @RW2-d16	MOV R2,R2 @RW2-d16	MOV R3,R2 @RW2-d16	MOV R3,R2 @RW2-d16	MOV R4,R2 @RW2-d16	MOV R4,R2 @RW2-d16	MOV R5,R2 @RW2-d16	MOV R5,R5 @RW2-d16	MOV R6,R2 @RW2-d16	MOV R6,R6 @RW2-d16	MOV R7,R2 @RW2-d16	MOV R7,R7 @RW2-d16
+ B	MOV R0,R3 @RW3-d16	MOV R0,R3 @RW3-d16	MOV R1,R3 @RW3-d16	MOV R1,R3 @RW3-d16	MOV R2,R3 @RW3-d16	MOV R2,R3 @RW3-d16	MOV R3,R3 @RW3-d16	MOV R3,R3 @RW3-d16	MOV R4,R3 @RW3-d16	MOV R4,R4 @RW3-d16	MOV R5,R3 @RW3-d16	MOV R5,R5 @RW3-d16	MOV R6,R3 @RW3-d16	MOV R6,R6 @RW3-d16	MOV R7,R3 @RW3-d16	MOV R7,R7 @RW3-d16
+ C	MOV R0,R4 @RW4-d16	MOV R0,R4 @RW4-d16	MOV R1,R4 @RW4-d16	MOV R1,R4 @RW4-d16	MOV R2,R4 @RW4-d16	MOV R2,R4 @RW4-d16	MOV R3,R4 @RW4-d16	MOV R3,R4 @RW4-d16	MOV R4,R4 @RW4-d16	MOV R4,R4 @RW4-d16	MOV R5,R4 @RW4-d16	MOV R5,R5 @RW4-d16	MOV R6,R4 @RW4-d16	MOV R6,R6 @RW4-d16	MOV R7,R4 @RW4-d16	MOV R7,R7 @RW4-d16
+ D	MOV R0,R5 @RW5-d16	MOV R0,R5 @RW5-d16	MOV R1,R5 @RW5-d16	MOV R1,R5 @RW5-d16	MOV R2,R5 @RW5-d16	MOV R2,R5 @RW5-d16	MOV R3,R5 @RW5-d16	MOV R3,R5 @RW5-d16	MOV R4,R5 @RW5-d16	MOV R4,R5 @RW5-d16	MOV R5,R5 @RW5-d16	MOV R5,R5 @RW5-d16	MOV R6,R5 @RW5-d16	MOV R6,R6 @RW5-d16	MOV R7,R5 @RW5-d16	MOV R7,R7 @RW5-d16
+ E	MOV R0,R6 @RW6-d16	MOV R0,R6 @RW6-d16	MOV R1,R6 @RW6-d16	MOV R1,R6 @RW6-d16	MOV R2,R6 @RW6-d16	MOV R2,R6 @RW6-d16	MOV R3,R6 @RW6-d16	MOV R3,R6 @RW6-d16	MOV R4,R6 @RW6-d16	MOV R4,R6 @RW6-d16	MOV R5,R6 @RW6-d16	MOV R5,R5 @RW6-d16	MOV R6,R6 @RW6-d16	MOV R6,R6 @RW6-d16	MOV R7,R6 @RW6-d16	MOV R7,R7 @RW6-d16
+ F	MOV R0,R7 @RW7-d16	MOV R0,R7 @RW7-d16	MOV R1,R7 @RW7-d16	MOV R1,R7 @RW7-d16	MOV R2,R7 @RW7-d16	MOV R2,R7 @RW7-d16	MOV R3,R7 @RW7-d16	MOV R3,R7 @RW7-d16	MOV R4,R7 @RW7-d16	MOV R4,R7 @RW7-d16	MOV R5,R7 @RW7-d16	MOV R5,R5 @RW7-d16	MOV R6,R7 @RW7-d16	MOV R6,R6 @RW7-d16	MOV R7,R7 @RW7-d16	MOV R7,R7 @RW7-d16

Table B.9-17 MOVW R_Wi, ea instruction (first byte = 7B_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW RW0 @RW0-d8	MOVW RW1,RW0	MOVW RW1 @RW0-d8	MOVW RW2,RW0	MOVW RW2 @RW0-d8	MOVW RW3,RW0	MOVW RW3 @RW0-d8	MOVW RW4,RW0	MOVW RW4 @RW0-d8	MOVW RW5,RW0	MOVW RW5 @RW0-d8	MOVW RW6,RW0	MOVW RW6 @RW0-d8	MOVW RW7,RW0	MOVW RW7 @RW0-d8
+ 1	MOVW RW0,RW1	MOVW RW0 @RW1+d8	MOVW RW1,RW1	MOVW RW1 @RW1+d8	MOVW RW2,RW1	MOVW RW2 @RW1+d8	MOVW RW3,RW1	MOVW RW3 @RW1+d8	MOVW RW4,RW1	MOVW RW4 @RW1+d8	MOVW RW5,RW1	MOVW RW5 @RW1+d8	MOVW RW6,RW1	MOVW RW6 @RW1+d8	MOVW RW7,RW1	MOVW RW7 @RW1+d8
+ 2	MOVW RW0,RW2	MOVW RW0 @RW2-d8	MOVW RW1,RW2	MOVW RW1 @RW2-d8	MOVW RW2,RW2	MOVW RW2 @RW2-d8	MOVW RW3,RW2	MOVW RW3 @RW2-d8	MOVW RW4,RW2	MOVW RW4 @RW2-d8	MOVW RW5,RW2	MOVW RW5 @RW2-d8	MOVW RW6,RW2	MOVW RW6 @RW2-d8	MOVW RW7,RW2	MOVW RW7 @RW2-d8
+ 3	MOVW RW0,RW3	MOVW RW0 @RW3-d8	MOVW RW1,RW3	MOVW RW1 @RW3-d8	MOVW RW2,RW3	MOVW RW2 @RW3-d8	MOVW RW3,RW3	MOVW RW3 @RW3-d8	MOVW RW4,RW3	MOVW RW4 @RW3-d8	MOVW RW5,RW3	MOVW RW5 @RW3-d8	MOVW RW6,RW3	MOVW RW6 @RW3-d8	MOVW RW7,RW3	MOVW RW7 @RW3-d8
+ 4	MOVW RW0,RW4	MOVW RW0 @RW4-d8	MOVW RW1,RW4	MOVW RW1 @RW4-d8	MOVW RW2,RW4	MOVW RW2 @RW4-d8	MOVW RW3,RW4	MOVW RW3 @RW4-d8	MOVW RW4,RW4	MOVW RW4 @RW4-d8	MOVW RW5,RW4	MOVW RW5 @RW4-d8	MOVW RW6,RW4	MOVW RW6 @RW4-d8	MOVW RW7,RW4	MOVW RW7 @RW4-d8
+ 5	MOVW RW0,RW5	MOVW RW0 @RW5-d8	MOVW RW1,RW5	MOVW RW1 @RW5-d8	MOVW RW2,RW5	MOVW RW2 @RW5-d8	MOVW RW3,RW5	MOVW RW3 @RW5-d8	MOVW RW4,RW5	MOVW RW4 @RW5-d8	MOVW RW5,RW5	MOVW RW5 @RW5-d8	MOVW RW6,RW5	MOVW RW6 @RW5-d8	MOVW RW7,RW5	MOVW RW7 @RW5-d8
+ 6	MOVW RW0,RW6	MOVW RW0 @RW6-d8	MOVW RW1,RW6	MOVW RW1 @RW6-d8	MOVW RW2,RW6	MOVW RW2 @RW6-d8	MOVW RW3,RW6	MOVW RW3 @RW6-d8	MOVW RW4,RW6	MOVW RW4 @RW6-d8	MOVW RW5,RW6	MOVW RW5 @RW6-d8	MOVW RW6,RW6	MOVW RW6 @RW6-d8	MOVW RW7,RW6	MOVW RW7 @RW6-d8
+ 7	MOVW RW0,RW7	MOVW RW0 @RW7-d8	MOVW RW1,RW7	MOVW RW1 @RW7-d8	MOVW RW2,RW7	MOVW RW2 @RW7-d8	MOVW RW3,RW7	MOVW RW3 @RW7-d8	MOVW RW4,RW7	MOVW RW4 @RW7-d8	MOVW RW5,RW7	MOVW RW5 @RW7-d8	MOVW RW6,RW7	MOVW RW6 @RW7-d8	MOVW RW7,RW7	MOVW RW7 @RW7-d8
+ 8	MOVW RW0,@RW0	MOVW RW0, @RW0-d16	MOVW RW1,@RW0	MOVW RW1 @RW0-d16	MOVW RW2,@RW0	MOVW RW2 @RW0-d16	MOVW RW3,@RW0	MOVW RW3 @RW0-d16	MOVW RW4,@RW0	MOVW RW4 @RW0-d16	MOVW RW5,@RW0	MOVW RW5 @RW0-d16	MOVW RW6,@RW0	MOVW RW6 @RW0-d16	MOVW RW7,@RW0	MOVW RW7 @RW0-d16
+ 9	MOVW RW0,@RW1	MOVW RW0, @RW1-d16	MOVW RW1,@RW1	MOVW RW1 @RW1-d16	MOVW RW2,@RW1	MOVW RW2 @RW1-d16	MOVW RW3,@RW1	MOVW RW3 @RW1-d16	MOVW RW4,@RW1	MOVW RW4 @RW1-d16	MOVW RW5,@RW1	MOVW RW5 @RW1-d16	MOVW RW6,@RW1	MOVW RW6 @RW1-d16	MOVW RW7,@RW1	MOVW RW7 @RW1-d16
+ A	MOVW RW0,@RW2	MOVW RW0, @RW2-d16	MOVW RW1,@RW2	MOVW RW1 @RW2-d16	MOVW RW2,@RW2	MOVW RW2 @RW2-d16	MOVW RW3,@RW2	MOVW RW3 @RW2-d16	MOVW RW4,@RW2	MOVW RW4 @RW2-d16	MOVW RW5,@RW2	MOVW RW5 @RW2-d16	MOVW RW6,@RW2	MOVW RW6 @RW2-d16	MOVW RW7,@RW2	MOVW RW7 @RW2-d16
+ B	MOVW RW0,@RW3	MOVW RW0, @RW3-d16	MOVW RW1,@RW3	MOVW RW1 @RW3-d16	MOVW RW2,@RW3	MOVW RW2 @RW3-d16	MOVW RW3,@RW3	MOVW RW3 @RW3-d16	MOVW RW4,@RW3	MOVW RW4 @RW3-d16	MOVW RW5,@RW3	MOVW RW5 @RW3-d16	MOVW RW6,@RW3	MOVW RW6 @RW3-d16	MOVW RW7,@RW3	MOVW RW7 @RW3-d16
+ C	MOVW R W0,@RW0+	MOVW RW0, @RW0+RW7	MOVW R W1,@RW0+	MOVW RW1, @RW0+RW7	MOVW R W2,@RW0+	MOVW RW2, @RW0+RW7	MOVW R W3,@RW0+	MOVW RW3, @RW0+RW7	MOVW W4,@RW0+	MOVW RW4, @RW0+RW7	MOVW W5,@RW0+	MOVW RW5, @RW0+RW7	MOVW W6,@RW0+	MOVW A@R @RW0+RW7	MOVW W7,@RW0+	MOVW A@R @RW0+RW7
+ D	MOVW R W0,@RW1+	MOVW RW0, @RW1+RW7	MOVW R W1,@RW1+	MOVW RW1, @RW1+RW7	MOVW R W2,@RW1+	MOVW RW2, @RW1+RW7	MOVW R W3,@RW1+	MOVW RW3, @RW1+RW7	MOVW W4,@RW1+	MOVW RW4, @RW1+RW7	MOVW W5,@RW1+	MOVW RW5, @RW1+RW7	MOVW W6,@RW1+	MOVW R @RW1+RW7	MOVW W7,@RW1+	MOVW R @RW1+RW7
+ E	MOVW R W0,@RW2+	MOVW RW0, @PC-d16	MOVW R W1,@RW2+	MOVW RW1, @PC-d16	MOVW R W2,@RW2+	MOVW RW2, @PC-d16	MOVW R W3,@RW2+	MOVW RW3, @PC-d16	MOVW W4,@RW2+	MOVW RW4, @PC-d16	MOVW W5,@RW2+	MOVW RW5, @PC-d16	MOVW W6,@RW2+	MOVW R @PC-d16	MOVW W7,@RW2+	MOVW R @PC-d16
	MOVW R W0,RW2	MOVW RW W0,RW2	MOVW R W0,RW2	MOVW RW W0,RW2	MOVW R W0,RW2	MOVW RW W0,RW2	MOVW R W0,RW2	MOVW RW W0,RW2	MOVW W0,RW2	MOVW RW W0,RW2	MOVW W0,RW2	MOVW RW W0,RW2	MOVW R W0,RW2	MOVW RW W0,RW2	MOVW W0,RW2	MOVW R W0,RW2
+ F	W0,@RW3+	0.addr16	W1,@RW3+	1.addr16	W2,@RW3+	2.addr16	W3,@RW3+	3.addr16	W4,@RW3+	4.addr16	W5,@RW3+	5.addr16	W6,@RW3+	6.addr16	W7,@RW3+	7.addr16

Table B.9-18 MOV ea, Ri instruction (first byte = 7CH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0,R0	MOV @R W0+d8,R0	MOV R0,R1	MOV @R W0+d8,R1	MOV R0,R2	MOV @R W0+d8,R2	MOV R0,R3	MOV @R W0+d8,R3	MOV R0,R4	MOV @R W0+d8,R4	MOV R0,R5	MOV @R W0+d8,R5	MOV R0,R6	MOV @R W0+d8,R6	MOV R0,R7	MOV @R W0+d8,R7
+ 1	MOV R1,R0	MOV @R W1+d8,R0	MOV R1,R1	MOV @R W1+d8,R1	MOV R1,R2	MOV @R W1+d8,R2	MOV R1,R3	MOV @R W1+d8,R3	MOV R1,R4	MOV @R W1+d8,R4	MOV R1,R5	MOV @R W1+d8,R5	MOV R1,R6	MOV @R W1+d8,R6	MOV R1,R7	MOV @R W1+d8,R7
+ 2	MOV R2,R0	MOV @R W2+d8,R0	MOV R2,R1	MOV @R W2+d8,R1	MOV R2,R2	MOV @R W2+d8,R2	MOV R2,R3	MOV @R W2+d8,R3	MOV R2,R4	MOV @R W2+d8,R4	MOV R2,R5	MOV @R W2+d8,R5	MOV R2,R6	MOV @R W2+d8,R6	MOV R2,R7	MOV @R W2+d8,R7
+ 3	MOV R3,R0	MOV @R W3+d8,R0	MOV R3,R1	MOV @R W3+d8,R1	MOV R3,R2	MOV @R W3+d8,R2	MOV R3,R3	MOV @R W3+d8,R3	MOV R3,R4	MOV @R W3+d8,R4	MOV R3,R5	MOV @R W3+d8,R5	MOV R3,R6	MOV @R W3+d8,R6	MOV R3,R7	MOV @R W3+d8,R7
+ 4	MOV R4,R0	MOV @R W4+d8,R0	MOV R4,R1	MOV @R W4+d8,R1	MOV R4,R2	MOV @R W4+d8,R2	MOV R4,R3	MOV @R W4+d8,R3	MOV R4,R4	MOV @R W4+d8,R4	MOV R4,R5	MOV @R W4+d8,R5	MOV R4,R6	MOV @R W4+d8,R6	MOV R4,R7	MOV @R W4+d8,R7
+ 5	MOV R5,R0	MOV @R W5+d8,R0	MOV R5,R1	MOV @R W5+d8,R1	MOV R5,R2	MOV @R W5+d8,R2	MOV R5,R3	MOV @R W5+d8,R3	MOV R5,R4	MOV @R W5+d8,R4	MOV R5,R5	MOV @R W5+d8,R5	MOV R5,R6	MOV @R W5+d8,R6	MOV R5,R7	MOV @R W5+d8,R7
+ 6	MOV R6,R0	MOV @R W6+d8,R0	MOV R6,R1	MOV @R W6+d8,R1	MOV R6,R2	MOV @R W6+d8,R2	MOV R6,R3	MOV @R W6+d8,R3	MOV R6,R4	MOV @R W6+d8,R4	MOV R6,R5	MOV @R W6+d8,R5	MOV R6,R6	MOV @R W6+d8,R6	MOV R6,R7	MOV @R W6+d8,R7
+ 7	MOV R7,R0	MOV @R W7+d8,R0	MOV R7,R1	MOV @R W7+d8,R1	MOV R7,R2	MOV @R W7+d8,R2	MOV R7,R3	MOV @R W7+d8,R3	MOV R7,R4	MOV @R W7+d8,R4	MOV R7,R5	MOV @R W7+d8,R5	MOV R7,R6	MOV @R W7+d8,R6	MOV R7,R7	MOV @R W7+d8,R7
+ 8	MOV @RW0,R0	MOV @RW 0+d16,R0	MOV @RW0,R1	MOV @RW 0+d16,R1	MOV @RW0,R2	MOV @RW 0+d16,R2	MOV @RW0,R3	MOV @RW 0+d16,R3	MOV @RW0,R4	MOV @RW 0+d16,R4	MOV @RW0,R5	MOV @RW 0+d16,R5	MOV @RW0,R6	MOV @RW 0+d16,R6	MOV @RW0,R7	MOV @RW 0+d16,R7
+ 9	MOV @RW1,R0	MOV @RW 1+d16,R0	MOV @RW1,R1	MOV @RW 1+d16,R1	MOV @RW1,R2	MOV @RW 1+d16,R2	MOV @RW1,R3	MOV @RW 1+d16,R3	MOV @RW1,R4	MOV @RW 1+d16,R4	MOV @RW1,R5	MOV @RW 1+d16,R5	MOV @RW1,R6	MOV @RW 1+d16,R6	MOV @RW1,R7	MOV @RW 1+d16,R7
+ A	MOV @RW2,R0	MOV @RW 2+d16,R0	MOV @RW2,R1	MOV @RW 2+d16,R1	MOV @RW2,R2	MOV @RW 2+d16,R2	MOV @RW2,R3	MOV @RW 2+d16,R3	MOV @RW2,R4	MOV @RW 2+d16,R4	MOV @RW2,R5	MOV @RW 2+d16,R5	MOV @RW2,R6	MOV @RW 2+d16,R6	MOV @RW2,R7	MOV @RW 2+d16,R7
+ B	MOV @RW3,R0	MOV @RW 3+d16,R0	MOV @RW3,R1	MOV @RW 3+d16,R1	MOV @RW3,R2	MOV @RW 3+d16,R2	MOV @RW3,R3	MOV @RW 3+d16,R3	MOV @RW3,R4	MOV @RW 3+d16,R4	MOV @RW3,R5	MOV @RW 3+d16,R5	MOV @RW3,R6	MOV @RW 3+d16,R6	MOV @RW3,R7	MOV @RW 3+d16,R7
+ C	MOV @RW0+,R0	MOV @RW 0+RW7,R0	MOV @RW0+,R1	MOV @RW 0+RW7,R1	MOV @RW0+,R2	MOV @RW 0+RW7,R2	MOV @RW0+,R3	MOV @RW 0+RW7,R3	MOV @RW0+,R4	MOV @RW 0+RW7,R4	MOV @RW0+,R5	MOV @RW 0+RW7,R5	MOV @RW0+,R6	MOV @RW 0+RW7,R6	MOV @RW0+,R7	MOV @RW 0+RW7,R7
+ D	MOV @RW1+,R0	MOV @RW 1+RW7,R0	MOV @RW1+,R1	MOV @RW 1+RW7,R1	MOV @RW1+,R2	MOV @RW 1+RW7,R2	MOV @RW1+,R3	MOV @RW 1+RW7,R3	MOV @RW1+,R4	MOV @RW 1+RW7,R4	MOV @RW1+,R5	MOV @RW 1+RW7,R5	MOV @RW1+,R6	MOV @RW 1+RW7,R6	MOV @RW1+,R7	MOV @RW 1+RW7,R7
+ E	MOV @RW2+,R0	MOV @RW 2+RW7,R0	MOV @RW2+,R1	MOV @RW 2+RW7,R1	MOV @RW2+,R2	MOV @RW 2+RW7,R2	MOV @RW2+,R3	MOV @RW 2+RW7,R3	MOV @RW2+,R4	MOV @RW 2+RW7,R4	MOV @RW2+,R5	MOV @RW 2+RW7,R5	MOV @RW2+,R6	MOV @RW 2+RW7,R6	MOV @RW2+,R7	MOV @RW 2+RW7,R7
+ F	MOV @RW3+,R0	MOV @RW 3+RW7,R0	MOV @RW3+,R1	MOV @RW 3+RW7,R1	MOV @RW3+,R2	MOV @RW 3+RW7,R2	MOV @RW3+,R3	MOV @RW 3+RW7,R3	MOV @RW3+,R4	MOV @RW 3+RW7,R4	MOV @RW3+,R5	MOV @RW 3+RW7,R5	MOV @RW3+,R6	MOV @RW 3+RW7,R6	MOV @RW3+,R7	MOV @RW 3+RW7,R7

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 :RMO,RMO	:MOVW @RW 0+8,RW0	MOVW RW0,RW1	:MOVW @RW 0+8,RW1	MOVW RW0,RW2	:MOVW @RW 0+8,RW2	MOVW RW0,RW3	:MOVW @RW 0+8,RW3	MOVW RW0,RW4	:MOVW @RW 0+8,RW4	MOVW RW0,RW5	:MOVW @RW 0+8,RW5	MOVW RW0,RW6	:MOVW @RW 0+8,RW6	MOVW RW0,RW7	:MOVW @RW 0+8,RW7
+ 1	MOVW RW1,RW0	:MOVW @RW 1+8,RW0	MOVW RW1,RW1	:MOVW @RW 1+8,RW1	MOVW RW1,RW2	:MOVW @RW 1+8,RW2	MOVW RW1,RW3	:MOVW @RW 1+8,RW3	MOVW RW1,RW4	:MOVW @RW 1+8,RW4	MOVW RW1,RW5	:MOVW @RW 1+8,RW5	MOVW RW1,RW6	:MOVW @RW 1+8,RW6	MOVW RW1,RW7	:MOVW @RW 1+8,RW7
+ 2	MOVW RW2,RW0	:MOVW @RW 2+8,RW0	MOVW RW2,RW1	:MOVW @RW 2+8,RW1	MOVW RW2,RW2	:MOVW @RW 2+8,RW2	MOVW RW2,RW3	:MOVW @RW 2+8,RW3	MOVW RW2,RW4	:MOVW @RW 2+8,RW4	MOVW RW2,RW5	:MOVW @RW 2+8,RW5	MOVW RW2,RW6	:MOVW @RW 2+8,RW6	MOVW RW2,RW7	:MOVW @RW 2+8,RW7
+ 3	MOVW RW3,RW0	:MOVW @RW 3+8,RW0	MOVW RW3,RW1	:MOVW @RW 3+8,RW1	MOVW RW3,RW2	:MOVW @RW 3+8,RW2	MOVW RW3,RW3	:MOVW @RW 3+8,RW3	MOVW RW3,RW4	:MOVW @RW 3+8,RW4	MOVW RW3,RW5	:MOVW @RW 3+8,RW5	MOVW RW3,RW6	:MOVW @RW 3+8,RW6	MOVW RW3,RW7	:MOVW @RW 3+8,RW7
+ 4	MOVW RW4,RW0	:MOVW @RW 4+8,RW0	MOVW RW4,RW1	:MOVW @RW 4+8,RW1	MOVW RW4,RW2	:MOVW @RW 4+8,RW2	MOVW RW4,RW3	:MOVW @RW 4+8,RW3	MOVW RW4,RW4	:MOVW @RW 4+8,RW4	MOVW RW4,RW5	:MOVW @RW 4+8,RW5	MOVW RW4,RW6	:MOVW @RW 4+8,RW6	MOVW RW4,RW7	:MOVW @RW 4+8,RW7
+ 5	MOVW RW5,RW0	:MOVW @RW 5+8,RW0	MOVW RW5,RW1	:MOVW @RW 5+8,RW1	MOVW RW5,RW2	:MOVW @RW 5+8,RW2	MOVW RW5,RW3	:MOVW @RW 5+8,RW3	MOVW RW5,RW4	:MOVW @RW 5+8,RW4	MOVW RW5,RW5	:MOVW @RW 5+8,RW5	MOVW RW5,RW6	:MOVW @RW 5+8,RW6	MOVW RW5,RW7	:MOVW @RW 5+8,RW7
+ 6	MOVW RW6,RW0	:MOVW @RW 6+8,RW0	MOVW RW6,RW1	:MOVW @RW 6+8,RW1	MOVW RW6,RW2	:MOVW @RW 6+8,RW2	MOVW RW6,RW3	:MOVW @RW 6+8,RW3	MOVW RW6,RW4	:MOVW @RW 6+8,RW4	MOVW RW6,RW5	:MOVW @RW 6+8,RW5	MOVW RW6,RW6	:MOVW @RW 6+8,RW6	MOVW RW6,RW7	:MOVW @RW 6+8,RW7
+ 7	MOVW RW7,RW0	:MOVW @RW 7+8,RW0	MOVW RW7,RW1	:MOVW @RW 7+8,RW1	MOVW RW7,RW2	:MOVW @RW 7+8,RW2	MOVW RW7,RW3	:MOVW @RW 7+8,RW3	MOVW RW7,RW4	:MOVW @RW 7+8,RW4	MOVW RW7,RW5	:MOVW @RW 7+8,RW5	MOVW RW7,RW6	:MOVW @RW 7+8,RW6	MOVW RW7,RW7	:MOVW @RW 7+8,RW7
+ 8	MOVW @RW0,RW0	:MOVW @RW0 +16,RW0	MOVW @RW0,RW1	:MOVW @RW0 +16,RW1	MOVW @RW0,RW2	:MOVW @RW0 +16,RW2	MOVW @RW0,RW3	:MOVW @RW0 +16,RW3	MOVW @RW0,RW4	:MOVW @RW0 +16,RW4	MOVW @RW0,RW5	:MOVW @RW0 +16,RW5	MOVW @RW0,RW6	:MOVW @RW0 +16,RW6	MOVW @RW0,RW7	:MOVW @RW0 +16,RW7
+ 9	MOVW @RW1,RW0	:MOVW @RW1 +16,RW0	MOVW @RW1,RW1	:MOVW @RW1 +16,RW1	MOVW @RW1,RW2	:MOVW @RW1 +16,RW2	MOVW @RW1,RW3	:MOVW @RW1 +16,RW3	MOVW @RW1,RW4	:MOVW @RW1 +16,RW4	MOVW @RW1,RW5	:MOVW @RW1 +16,RW5	MOVW @RW1,RW6	:MOVW @RW1 +16,RW6	MOVW @RW1,RW7	:MOVW @RW1 +16,RW7
+ A	MOVW @RW2,RW0	:MOVW @RW2 +16,RW0	MOVW @RW2,RW1	:MOVW @RW2 +16,RW1	MOVW @RW2,RW2	:MOVW @RW2 +16,RW2	MOVW @RW2,RW3	:MOVW @RW2 +16,RW3	MOVW @RW2,RW4	:MOVW @RW2 +16,RW4	MOVW @RW2,RW5	:MOVW @RW2 +16,RW5	MOVW @RW2,RW6	:MOVW @RW2 +16,RW6	MOVW @RW2,RW7	:MOVW @RW2 +16,RW7
+ B	MOVW @RW3,RW0	:MOVW @RW3 +16,RW0	MOVW @RW3,RW1	:MOVW @RW3 +16,RW1	MOVW @RW3,RW2	:MOVW @RW3 +16,RW2	MOVW @RW3,RW3	:MOVW @RW3 +16,RW3	MOVW @RW3,RW4	:MOVW @RW3 +16,RW4	MOVW @RW3,RW5	:MOVW @RW3 +16,RW5	MOVW @RW3,RW6	:MOVW @RW3 +16,RW6	MOVW @RW3,RW7	:MOVW @RW3 +16,RW7
+ C	MOVW RW0+,RW0	:MOVW @RW0 +RW7,RW0	MOVW RW0+,RW1	:MOVW @RW0 +RW7,RW1	MOVW RW0+,RW2	:MOVW @RW0 +RW7,RW2	MOVW RW0+,RW3	:MOVW @RW0 +RW7,RW3	MOVW RW0+,RW4	:MOVW @RW0 +RW7,RW4	MOVW RW0+,RW5	:MOVW @RW0 +RW7,RW5	MOVW RW0+,RW6	:MOVW @RW0 +RW7,RW6	MOVW RW0+,RW7	:MOVW @RW0 +RW7,RW7
+ D	MOVW RW1+,RW0	:MOVW @RW1 +RW7,RW0	MOVW RW1+,RW1	:MOVW @RW1 +RW7,RW1	MOVW RW1+,RW2	:MOVW @RW1 +RW7,RW2	MOVW RW1+,RW3	:MOVW @RW1 +RW7,RW3	MOVW RW1+,RW4	:MOVW @RW1 +RW7,RW4	MOVW RW1+,RW5	:MOVW @RW1 +RW7,RW5	MOVW RW1+,RW6	:MOVW @RW1 +RW7,RW6	MOVW RW1+,RW7	:MOVW @RW1 +RW7,RW7
+ E	MOVW RW2+,RW0	:MOVW @RW2 +RW7,RW0	MOVW RW2+,RW1	:MOVW @RW2 +RW7,RW1	MOVW RW2+,RW2	:MOVW @RW2 +RW7,RW2	MOVW RW2+,RW3	:MOVW @RW2 +RW7,RW3	MOVW RW2+,RW4	:MOVW @RW2 +RW7,RW4	MOVW RW2+,RW5	:MOVW @RW2 +RW7,RW5	MOVW RW2+,RW6	:MOVW @RW2 +RW7,RW6	MOVW RW2+,RW7	:MOVW @RW2 +RW7,RW7
+ F	MOVW RW3+,RW0	:MOVW @RW3 +RW7,RW0	MOVW RW3+,RW1	:MOVW @RW3 +RW7,RW1	MOVW RW3+,RW2	:MOVW @RW3 +RW7,RW2	MOVW RW3+,RW3	:MOVW @RW3 +RW7,RW3	MOVW RW3+,RW4	:MOVW @RW3 +RW7,RW4	MOVW RW3+,RW5	:MOVW @RW3 +RW7,RW5	MOVW RW3+,RW6	:MOVW @RW3 +RW7,RW6	MOVW RW3+,RW7	:MOVW @RW3 +RW7,RW7

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,R0 @RW0+d8	XCH R1, @RW0+d8	XCH R2,R0 @RW0+d8	XCH R2, @RW0+d8	XCH R3,R0 @RW0+d8	XCH R3, @RW0+d8	XCH R4,R0 @RW0+d8	XCH R4, @RW0+d8	XCH R5,R0 @RW0+d8	XCH R5, @RW0+d8	XCH R6,R0 @RW0+d8	XCH R6, @RW0+d8	XCH R7,R0 @RW0+d8	XCH R7, @RW0+d8
+ 1	XCH R0,R1 @RW1+d8	XCH R0, @RW1+d8	XCH R1,R1 @RW1+d8	XCH R1, @RW1+d8	XCH R2,R1 @RW1+d8	XCH R2, @RW1+d8	XCH R3,R1 @RW1+d8	XCH R3, @RW1+d8	XCH R4,R1 @RW1+d8	XCH R4, @RW1+d8	XCH R5,R1 @RW1+d8	XCH R5, @RW1+d8	XCH R6,R1 @RW1+d8	XCH R6, @RW1+d8	XCH R7,R1 @RW1+d8	XCH R7, @RW1+d8
+ 2	XCH R0,R2 @RW2+d8	XCH R0, @RW2+d8	XCH R1,R2 @RW2+d8	XCH R1, @RW2+d8	XCH R2,R2 @RW2+d8	XCH R2, @RW2+d8	XCH R3,R2 @RW2+d8	XCH R3, @RW2+d8	XCH R4,R2 @RW2+d8	XCH R4, @RW2+d8	XCH R5,R2 @RW2+d8	XCH R5, @RW2+d8	XCH R6,R2 @RW2+d8	XCH R6, @RW2+d8	XCH R7,R2 @RW2+d8	XCH R7, @RW2+d8
+ 3	XCH R0,R3 @RW3+d8	XCH R0, @RW3+d8	XCH R1,R3 @RW3+d8	XCH R1, @RW3+d8	XCH R2,R3 @RW3+d8	XCH R2, @RW3+d8	XCH R3,R3 @RW3+d8	XCH R3, @RW3+d8	XCH R4,R3 @RW3+d8	XCH R4, @RW3+d8	XCH R5,R3 @RW3+d8	XCH R5, @RW3+d8	XCH R6,R3 @RW3+d8	XCH R6, @RW3+d8	XCH R7,R3 @RW3+d8	XCH R7, @RW3+d8
+ 4	XCH R0,R4 @RW4+d8	XCH R0, @RW4+d8	XCH R1,R4 @RW4+d8	XCH R1, @RW4+d8	XCH R2,R4 @RW4+d8	XCH R2, @RW4+d8	XCH R3,R4 @RW4+d8	XCH R3, @RW4+d8	XCH R4,R4 @RW4+d8	XCH R4, @RW4+d8	XCH R5,R4 @RW4+d8	XCH R5, @RW4+d8	XCH R6,R4 @RW4+d8	XCH R6, @RW4+d8	XCH R7,R4 @RW4+d8	XCH R7, @RW4+d8
+ 5	XCH R0,R5 @RW5+d8	XCH R0, @RW5+d8	XCH R1,R5 @RW5+d8	XCH R1, @RW5+d8	XCH R2,R5 @RW5+d8	XCH R2, @RW5+d8	XCH R3,R5 @RW5+d8	XCH R3, @RW5+d8	XCH R4,R5 @RW5+d8	XCH R4, @RW5+d8	XCH R5,R5 @RW5+d8	XCH R5, @RW5+d8	XCH R6,R5 @RW5+d8	XCH R6, @RW5+d8	XCH R7,R5 @RW5+d8	XCH R7, @RW5+d8
+ 6	XCH R0,R6 @RW6+d8	XCH R0, @RW6+d8	XCH R1,R6 @RW6+d8	XCH R1, @RW6+d8	XCH R2,R6 @RW6+d8	XCH R2, @RW6+d8	XCH R3,R6 @RW6+d8	XCH R3, @RW6+d8	XCH R4,R6 @RW6+d8	XCH R4, @RW6+d8	XCH R5,R6 @RW6+d8	XCH R5, @RW6+d8	XCH R6,R6 @RW6+d8	XCH R6, @RW6+d8	XCH R7,R6 @RW6+d8	XCH R7, @RW6+d8
+ 7	XCH R0,R7 @RW7+d8	XCH R0, @RW7+d8	XCH R1,R7 @RW7+d8	XCH R1, @RW7+d8	XCH R2,R7 @RW7+d8	XCH R2, @RW7+d8	XCH R3,R7 @RW7+d8	XCH R3, @RW7+d8	XCH R4,R7 @RW7+d8	XCH R4, @RW7+d8	XCH R5,R7 @RW7+d8	XCH R5, @RW7+d8	XCH R6,R7 @RW7+d8	XCH R6, @RW7+d8	XCH R7,R7 @RW7+d8	XCH R7, @RW7+d8
+ 8	XCH R0,@RW0	XCH R0, @RW0+d16	XCH R1,@RW0	XCH R1, @RW0+d16	XCH R2,@RW0	XCH R2, @RW0+d16	XCH R3,@RW0	XCH R3, @RW0+d16	XCH R4,@RW0	XCH R4, @RW0+d16	XCH R5,@RW0	XCH R5, @RW0+d16	XCH R6,@RW0	XCH R6, @RW0+d16	XCH R7,@RW0	XCH R7, @RW0+d16
+ 9	XCH R0,@RW1	XCH R0, @RW1+d16	XCH R1,@RW1	XCH R1, @RW1+d16	XCH R2,@RW1	XCH R2, @RW1+d16	XCH R3,@RW1	XCH R3, @RW1+d16	XCH R4,@RW1	XCH R4, @RW1+d16	XCH R5,@RW1	XCH R5, @RW1+d16	XCH R6,@RW1	XCH R6, @RW1+d16	XCH R7,@RW1	XCH R7, @RW1+d16
+ A	XCH R0,@RW2	XCH R0, W2+d16.A	XCH R1,@RW2	XCH R1, W2+d16.A	XCH R2,@RW2	XCH R2, W2+d16.A	XCH R3,@RW2	XCH R3, W2+d16.A	XCH R4,@RW2	XCH R4, W2+d16.A	XCH R5,@RW2	XCH R5, W2+d16.A	XCH R6,@RW2	XCH R6, W2+d16.A	XCH R7,@RW2	XCH R7, W2+d16.A
+ B	XCH R0,@RW3	XCH R0, @RW3+d16	XCH R1,@RW3	XCH R1, @RW3+d16	XCH R2,@RW3	XCH R2, @RW3+d16	XCH R3,@RW3	XCH R3, @RW3+d16	XCH R4,@RW3	XCH R4, @RW3+d16	XCH R5,@RW3	XCH R5, @RW3+d16	XCH R6,@RW3	XCH R6, @RW3+d16	XCH R7,@RW3	XCH R7, @RW3+d16
+ C	XCH R0,@RW0+	XCH R0, @RW0-RW7	XCH R1,@RW0+	XCH R1, @RW0-RW7	XCH R2,@RW0+	XCH R2, @RW0-RW7	XCH R3,@RW0+	XCH R3, @RW0-RW7	XCH R4,@RW0+	XCH R4, @RW0-RW7	XCH R5,@RW0+	XCH R5, @RW0-RW7	XCH R6,@RW0+	XCH R6, @RW0-RW7	XCH R7,@RW0+	XCH R7, @RW0-RW7
+ D	XCH R0,@RW1+	XCH R0, @RW1-RW7	XCH R1,@RW1+	XCH R1, @RW1-RW7	XCH R2,@RW1+	XCH R2, @RW1-RW7	XCH R3,@RW1+	XCH R3, @RW1-RW7	XCH R4,@RW1+	XCH R4, @RW1-RW7	XCH R5,@RW1+	XCH R5, @RW1-RW7	XCH R6,@RW1+	XCH R6, @RW1-RW7	XCH R7,@RW1+	XCH R7, @RW1-RW7
+ E	XCH R0,@RW2+	XCH R0, @PC+d16	XCH R1,@RW2+	XCH R1, @PC+d16	XCH R2,@RW2+	XCH R2, @PC+d16	XCH R3,@RW2+	XCH R3, @PC+d16	XCH R4,@RW2+	XCH R4, @PC+d16	XCH R5,@RW2+	XCH R5, @PC+d16	XCH R6,@RW2+	XCH R6, @PC+d16	XCH R7,@RW2+	XCH R7, @PC+d16
+ F	XCH R0,@RW3+	XCH R0, addr16	XCH R1,@RW3+	XCH R1, addr16	XCH R2,@RW3+	XCH R2, addr16	XCH R3,@RW3+	XCH R3, addr16	XCH R4,@RW3+	XCH R4, addr16	XCH R5,@RW3+	XCH R5, addr16	XCH R6,@RW3+	XCH R6, addr16	XCH R7,@RW3+	XCH R7, addr16

Table B.9-21 XCHW RWi, ea instruction (first byte = 7F_H)

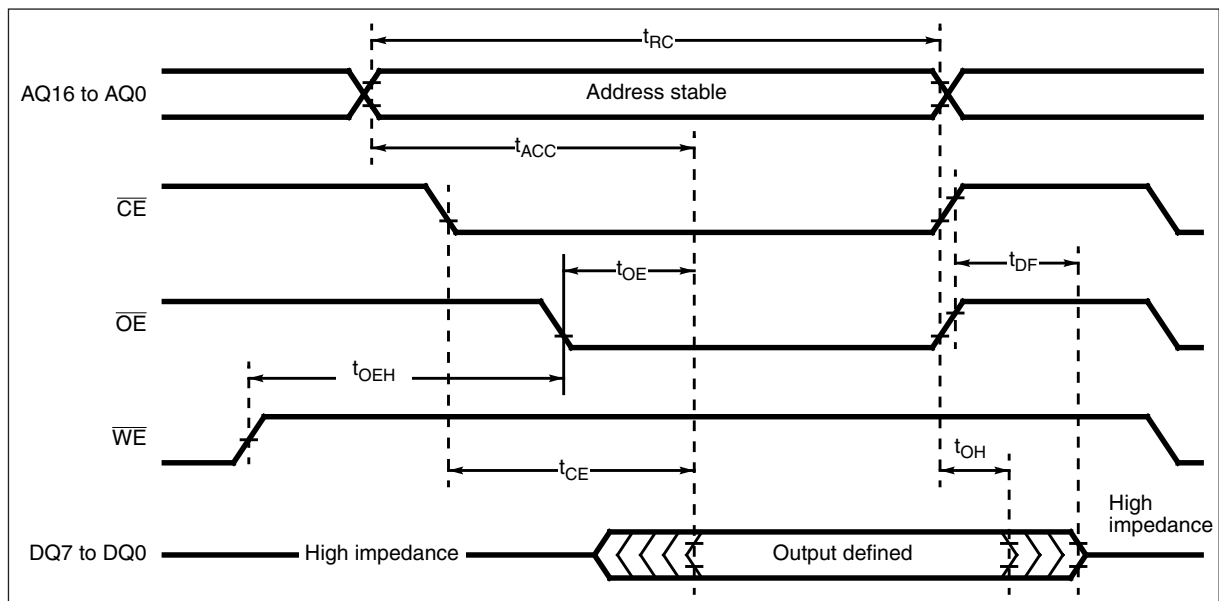
[illegible]

APPENDIX C Timing Diagrams in Flash Memory Mode

Each timing diagram for the external pins of the Flash devices in MB90340 series during Flash Memory mode is shown below.

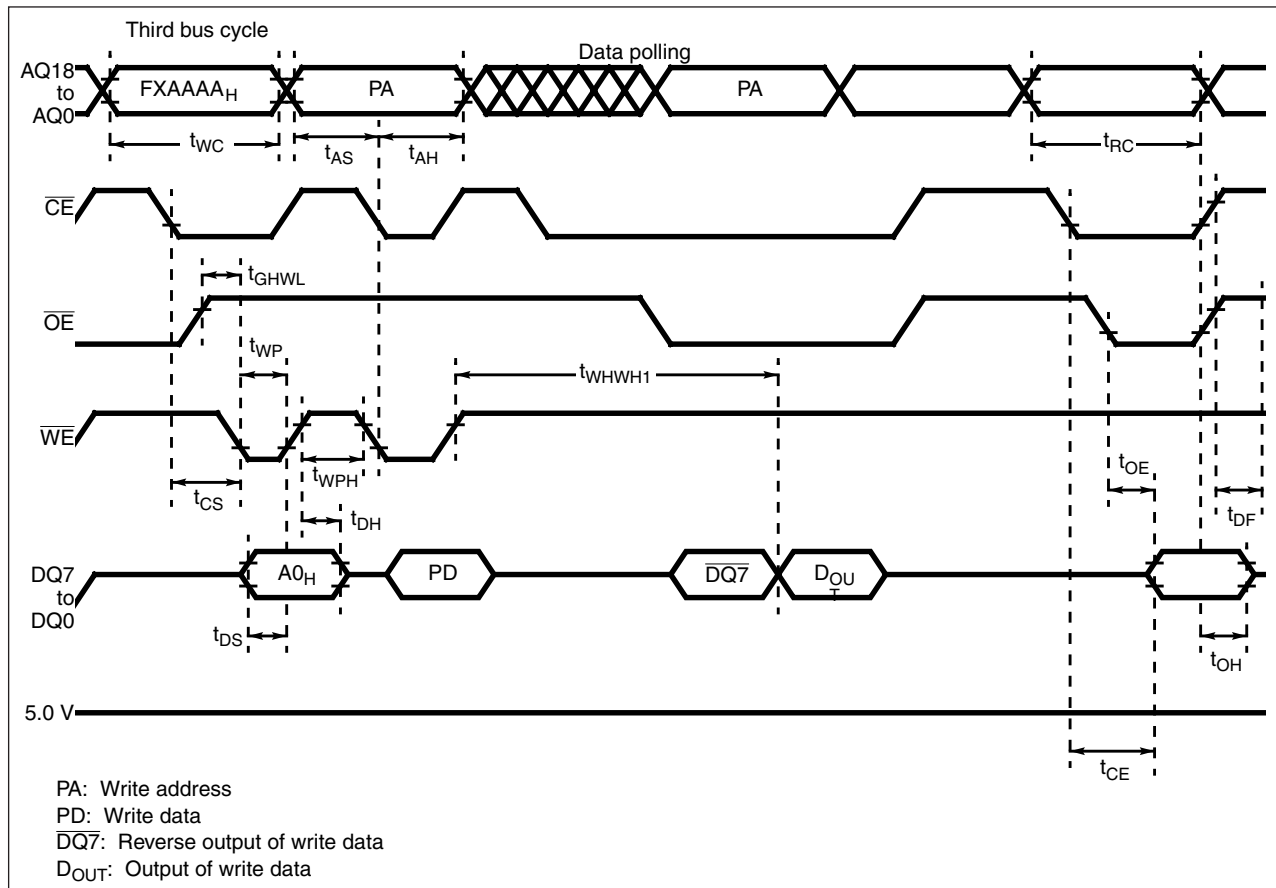
■ Data read by read access

Figure C-1 Timing diagram for read access



■ Write, data polling, read (WE control)

Figure C-2 Write, data polling, read (WE control)

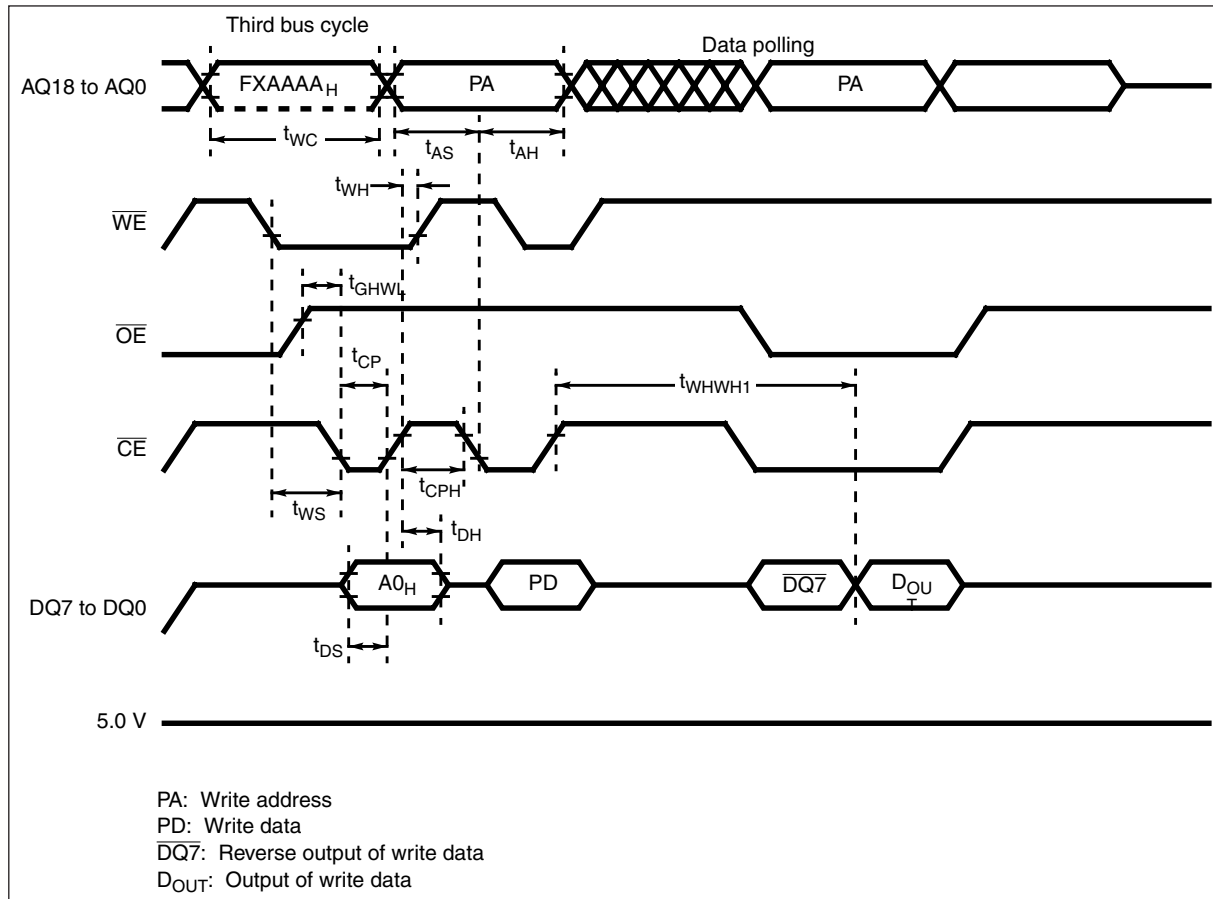


Note:

- Describes the last 2-bus cycle of 4-bus cycle sequences.
- "Fx" in "FxAAAA" described as address is any of 4/5/6/7 (MB90F342), 1/2/3/4/5/6/7 (MB90F343), 0/1/2/3/4/5/6/7 (MB90F345) or 6/7 (MB90F346A).

■ Write, data polling, read (CE control)

Figure C-3 Timing diagram for write access (CE control)

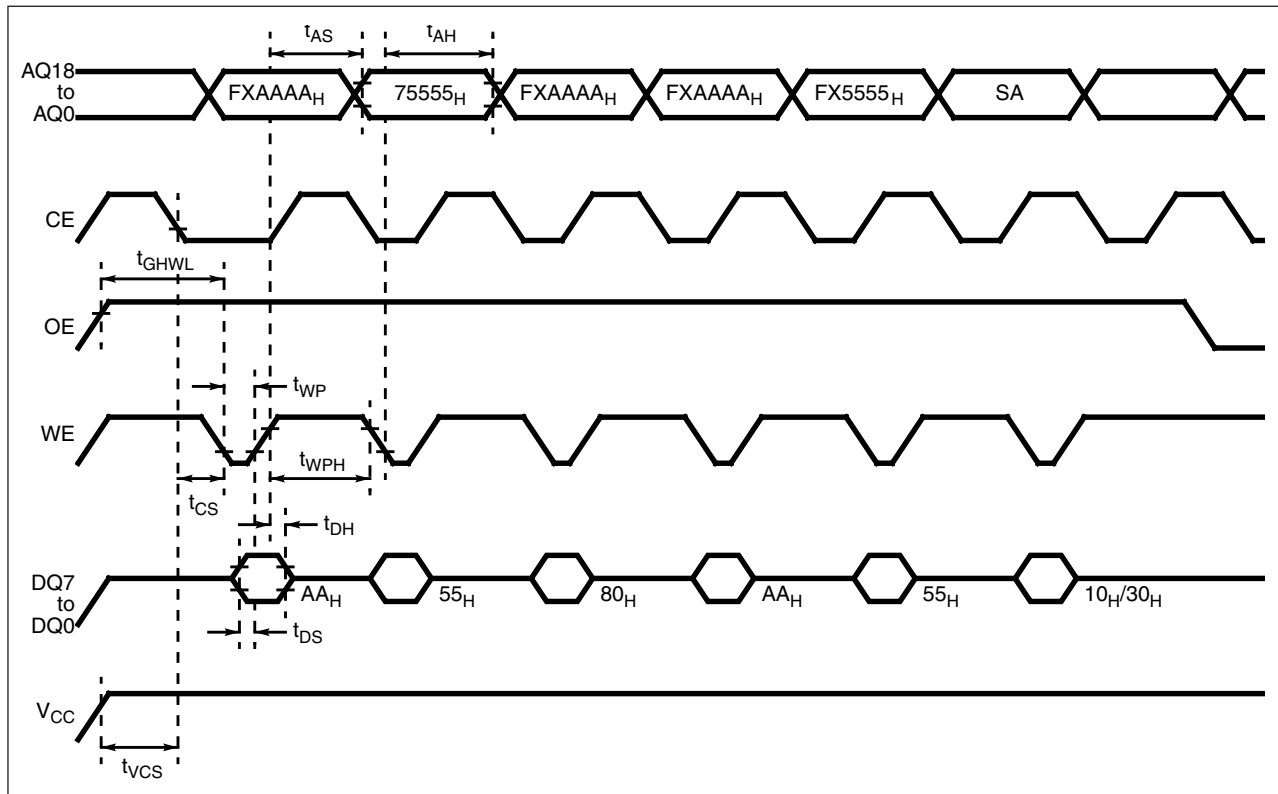


Note:

- Describes the last 2-bus cycle of 4-bus cycle sequences.
- "Fx" in "FxAAAA" described as address is any of 4/5/6/7 (MB90F342), 1/2/3/4/5/6/7 (MB90F343), 0/1/2/3/4/5/6/7 (MB90F345) or 6/7 (MB90F346A).

■ Chip erase/sector erase command sequence

Figure C-4 Timing diagram for write access (chip erasing/sector erasing)

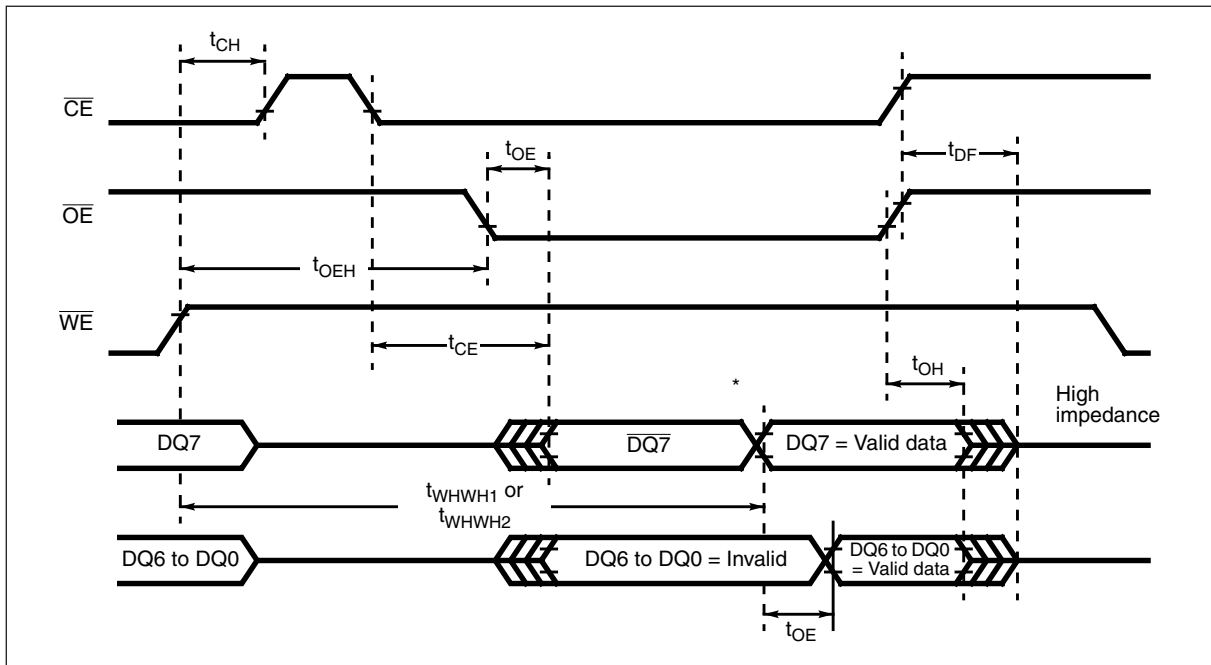


Note:

- Describes the last 2-bus cycle of 4-bus cycle sequences.
- "Fx" in "FxAAAA" described as address is any of 4/5/6/7 (MB90F342), 1/2/3/4/5/6/7 (MB90F343), 0/1/2/3/4/5/6/7 (MB90F345) or 6/7 (MB90F346A).

■ Data polling

Figure C-5 Timing diagram for data polling

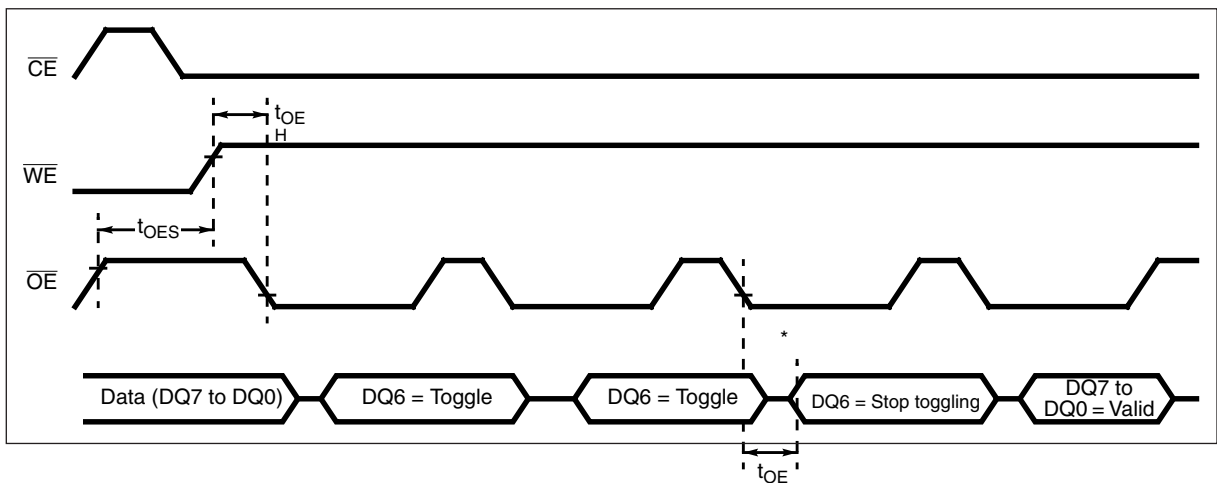


Note:

DQ7 is valid data (The device terminates automatic operation).

■ Toggle bit

Figure C-6 Timing diagram for toggle bit

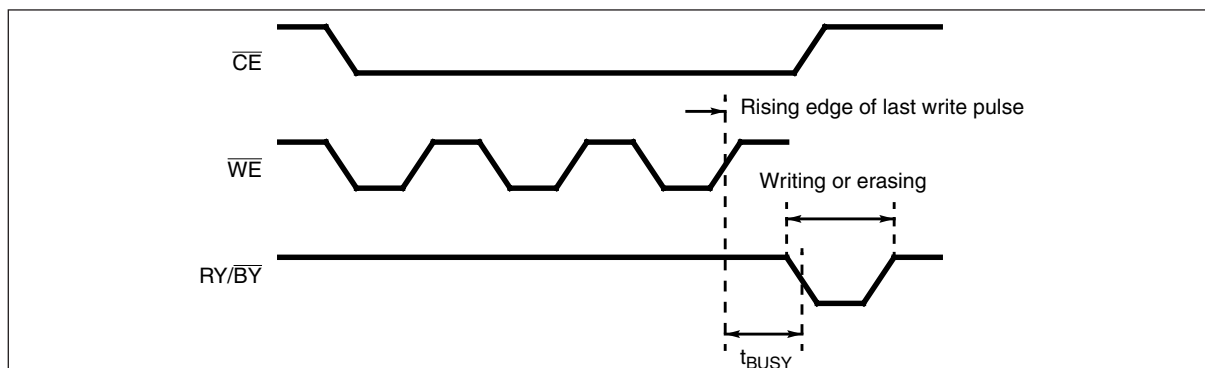


Note:

DQ6 stops toggling (The device terminates automatic operation).

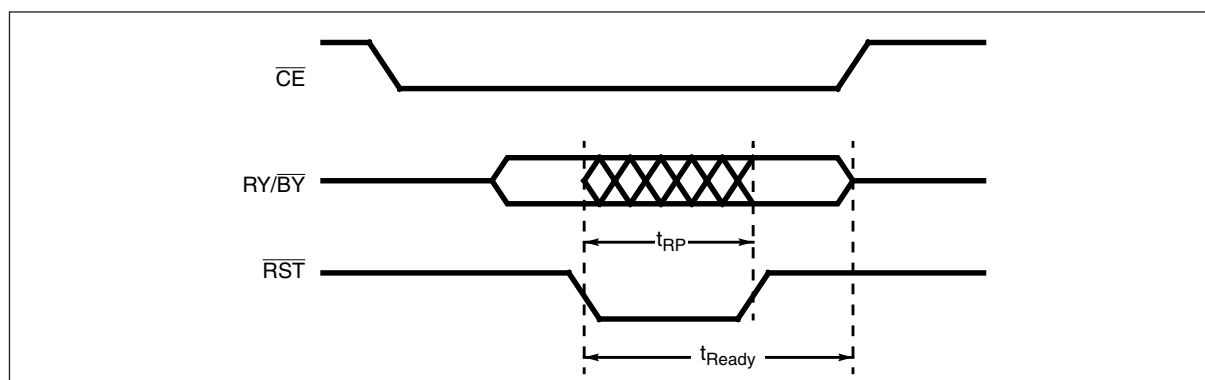
■ RY/BY timing during writing/erasing

Figure C-7 Timing diagram for output of RY/BY signal during writing/erasing



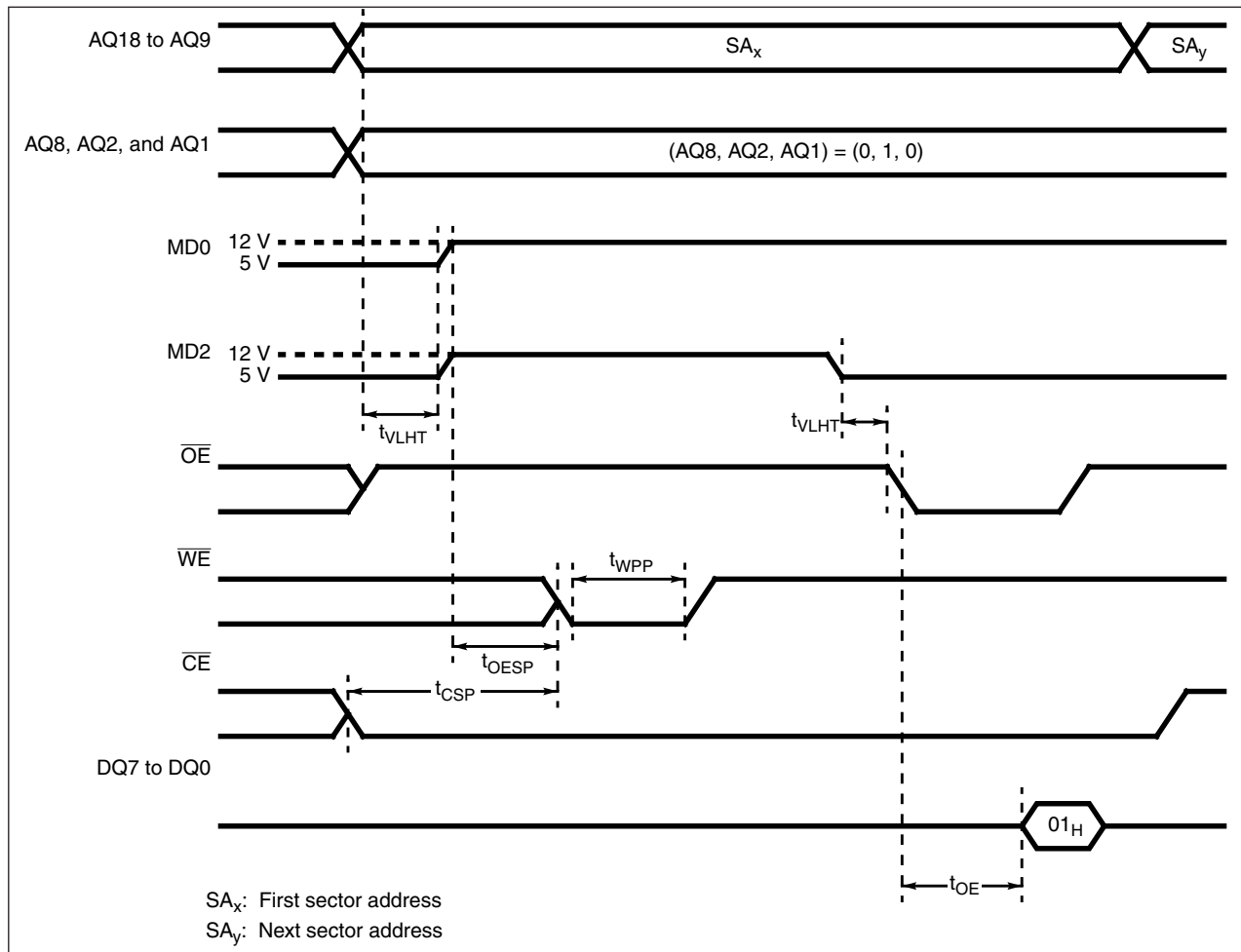
■ RST and RY/BY timing

Figure C-8 Timing diagram for output of RY/BY signal at hardware reset



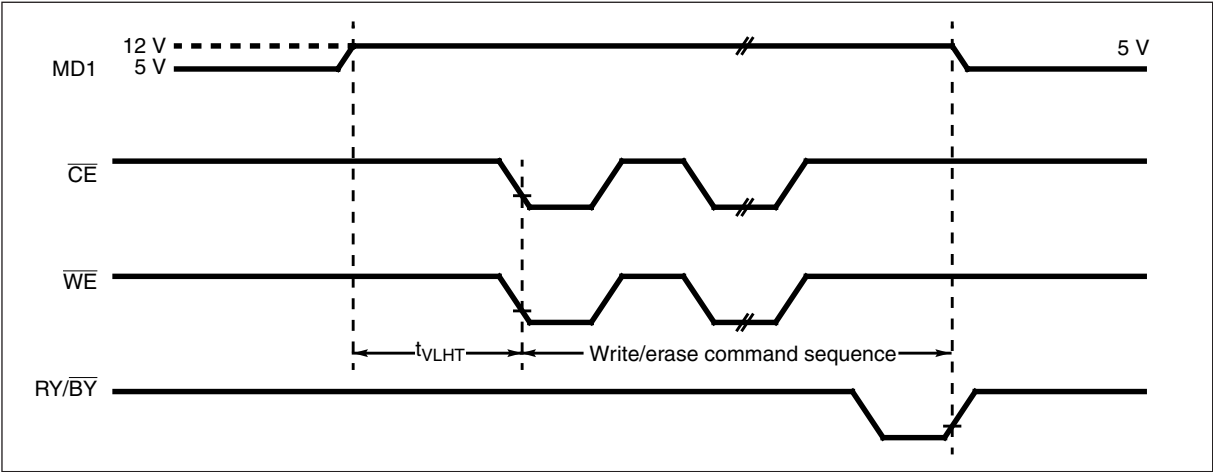
■ Enable sector protect/verify sector protect

Figure C-9 Enable sector protect/verify sector protect



■ Temporary sector protect cancellation

Figure C-10 Temporary sector protect cancellation



APPENDIX D List of Interrupt Vectors

The interrupt vector table to be referenced for interrupt processing is allocated to **FFFC00_H** to **FFFFFF_H** in the memory area and also used for software interrupts.

■ List of interrupt vectors

Table D-1 "Interrupt vectors" lists the interrupt vectors for the MB90340 series.

Table D-1 Interrupt vectors (1/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address L	Vector address H	Vector address bank	Mode register
		Number	Address				
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	Reset	--	--	FFFFDC _H	FFFFDD _H	FFFFDE _H	FFFFDF _H
INT 9	INT9 instruction	--	--	FFFFD8 _H	FFFFD9 _H	FFFFDA _H	Unused
INT 10	Exception	--	--	FFFFD4 _H	FFFFD5 _H	FFFFD6 _H	Unused
INT 11	CAN0 RX	ICR00	0000B0 _H	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Unused
INT 12	CAN0 TX/NS			FFFFCC _H	FFFFCD _H	FFFFCE _H	Unused
INT 13	CAN1 RX / Input Capture6	ICR01	0000B1 _H	FFFFC8 _H	FFFFC9 _H	FFFFCA _H	Unused
INT 14	CAN1 TX/NS /Input Caputer7			FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Unused
INT 15	CAN2 RX / I ² C0	ICR02	0000B2 _H	FFFFC0 _H	FFFFC1 _H	FFFFC2 _H	Unused
INT 16	CAN2 TX/NS			FFFFBC _H	FFFFBD _H	FFFFBE _H	Unused
INT 17	16-bit ReloadTimer0	ICR03	0000B3 _H	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Unused
INT 18	16-bit ReloadTimer1			FFFFB4 _H	FFFFB5 _H	FFFFB6 _H	Unused
INT 19	16-bit ReloadTimer2	ICR04	0000B4 _H	FFFFB0 _H	FFFFB1 _H	FFFFB2 _H	Unused
INT 20	16-bit ReloadTimer3			FFFFAC _H	FFFFAD _H	FFFFAE _H	Unused
INT 21	PPG 0/1/4/5	ICR05	0000B5 _H	FFFFA8 _H	FFFFA9 _H	FFFFAA _H	Unused
INT 22	PPG 2/3/6/7			FFFFA4 _H	FFFFA5 _H	FFFFA6 _H	Unused
INT 23	PPG 8/9/C/D	ICR06	0000B6 _H	FFFFA0 _H	FFFFA1 _H	FFFFA2 _H	Unused
INT 24	PPG A/B/E/F			FFFF9C _H	FFFF9D _H	FFFF9E _H	Unused
INT 25	Time Base Timer	ICR07	0000B7 _H	FFFF98 _H	FFFF99 _H	FFFF9A _H	Unused
INT 26	External Interrupt 0-3/8-11			FFFF94 _H	FFFF95 _H	FFFF96 _H	Unused

Table D-1 Interrupt vectors (2/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address L	Vector address H	Vector address bank	Mode register
		Number	Address				
INT 27	Watch Timer	ICR08	0000B8 _H	FFFF90 _H	FFFF91 _H	FFFF92 _H	Unused
INT 28	External Interrupt 4-7/12-15			FFFF8C _H	FFFF8D _H	FFFF8E _H	Unused
INT 29	A/D Converter	ICR09	0000B9 _H	FFFF88 _H	FFFF89 _H	FFFF8A _H	Unused
INT 30	I/O Timer 0/1			FFFF84 _H	FFFF85 _H	FFFF86 _H	Unused
INT 31	Input Capture 4/5 / I ² C1	ICR10	0000BA _H	FFFF80 _H	FFFF81 _H	FFFF82 _H	Unused
INT 32	Output Compare 0/1/4/5			FFFF7C _H	FFFF7D _H	FFFF7E _H	Unused
INT 33	Input Capture 0-3	ICR11	0000BB _H	FFFF78 _H	FFFF79 _H	FFFF7A _H	Unused
INT 34	Output Compare 2/3/6/7			FFFF74 _H	FFFF75 _H	FFFF76 _H	Unused
INT 35	UART 0 RX	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	Unused
INT 36	UART 0 TX			FFFF6C _H	FFFF6D _H	FFFF6E _H	Unused
INT 37	UART 1 RX / UART3 RX	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	Unused
INT 38	UART 1 TX / UART 3 TX			FFFF64 _H	FFFF65 _H	FFFF66 _H	Unused
INT 39	UART 2 RX / UART 4 RX	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	Unused
INT 40	UART 2 TX / UART 4 TX			FFFF5C _H	FFFF5D _H	FFFF5E _H	Unused
INT 41	Flash Memory	ICR15	0000BF _H	FFFF58 _H	FFFF59 _H	FFFF5A _H	Unused
INT 42	Delayed Interrupt			FFFF54 _H	FFFF55 _H	FFFF56 _H	Unused
INT 43	--	--	--	FFFF50 _H	FFFF51 _H	FFFF52 _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.

■ 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 MB90340 series.

Table D-2 Interrupt causes, interrupt vectors, and interrupt control registers (1/2)

Interrupt cause	EI ² OS clear	DMA channel number	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	-	-
CAN 0 RX	N	-	#11	FFFFD0 _H	ICR00	0000B0 _H
CAN 0 TX/NS	N	-	#12	FFFFCC _H		
CAN 1 RX / Input Capture 6	N	-	#13	FFFFC8 _H	ICR01	0000B1 _H
CAN 1 TX/NS / Input Capture 7	N	-	#14	FFFFC4 _H		
CAN 2 RX / I ² C0	N	-	#15	FFFFC0 _H	ICR02	0000B2 _H
CAN 2 TX/NS	N	-	#16	FFFFBC _H		
16-bit Reload Timer 0	Y1	0	#17	FFFFB8 _H	ICR03	0000B3 _H
16-bit Reload Timer 1	Y1	1	#18	FFFFB4 _H		
16-bit Reload Timer 2	Y1	2	#19	FFFFB0 _H	ICR04	0000B4 _H
16-bit Reload Timer 3	Y1	-	#20	FFFFAC _H		
PPG 0/1/4/5	N	-	#21	FFFFA8 _H	ICR05	0000B5 _H
PPG 2/3/6/7	N	-	#22	FFFFA4 _H		
PPG 8/9/C/D	N	-	#23	FFFFA0 _H	ICR06	0000B6 _H
PPG A/B/E/F	N	-	#24	FFFF9C _H		
Time Base Timer	N	-	#25	FFFF98 _H	ICR07	0000B7 _H
External Interrupt 0-3, 8-11	Y1	3	#26	FFFF94 _H		
Watch Timer	N	-	#27	FFFF90 _H	ICR08	0000B8 _H
External Interrupt 4-7, 12-15	Y1	4	#28	FFFF8C _H		
A/D Converter	Y1	5	#29	FFFF88 _H	ICR09	0000B9 _H
I/O Timer 0 / I/O Timer 1	N	-	#30	FFFF84 _H		

Table D-2 Interrupt causes, interrupt vectors, and interrupt control registers (2/2)

Interrupt cause	EI ² OS clear	DMA channel number	Interrupt vector		Interrupt control register	
			Number	Address	Number	Address
Input Capture 4/5 / I ² C1	Y1	6	#31	FFFF80 _H	ICR10	0000BA _H
Output Compare 0/1/4/5	Y1	7	#32	FFFF7C _H		
Input Capture 0-3	Y1	8	#33	FFFF78 _H	ICR11	0000BB _H
Output Compare 2/3/6/7	Y1	9	#34	FFFF74 _H		
UART 0 RX	Y2	10	#35	FFFF70 _H	ICR12	0000BC _H
UART 0 TX	Y1	11	#36	FFFF6C _H		
UART 1 RX / UART 3 RX	Y2	12	#37	FFFF68 _H	ICR13	0000BD _H
UART 1 TX / UART 3 TX	Y1	13	#38	FFFF64 _H		
UART 2 RX / UART 4 RX	Y2	14	#39	FFFF60 _H	ICR14	0000BE _H
UART 2 TX / UART 4 TX	Y1	15	#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	230
16-bit free-running timer 0 and 1	232
16-bit free-running timer block diagram.....	234
16-bit free-running timer operation	239
16-bit free-running timer timing	240
16-bit I/O timer, block diagram of	231
16-bit input capture	233
16-bit output compare	232
16-bit reload timer (in internal clock mode), input pin function of	270
16-bit reload timer (with event count function), outline of	262
16-bit reload timer register	264
16-bit reload timer, block diagram of	263
16-bit reload timer, internal clock operation of	269
16-bit reload timer, output pin function of	272
16-bit reload timer, underflow operation of	271
16-bit timer register (TMR)/16-bit reload register (TMRLR), register layout of	268
1M/2M/3M-bit flash memory feature.....	516
1M/2M/3M-bit flash memory, sector configuration of	517
1M-bit flash memory, programming example of	548
24-bit operand specification	38
8/16-bit PPG hardware, initial value of	299
8/16-bit PPG interrupt	298
8/16-bit PPG output operation.....	295
8/16-bit PPG pulse, controlling pin output of	297
8/16-bit PPG registers	286
8/16-bit PPG, block diagram of	284
8/16-bit PPG, function of	282
8/16-bit PPG, operation mode of	294
8/16-bit PPG, operation of	294
8/16-bit PPG, selecting count clock for	296

A

acceptance filter, setting	489
acceptance filtering	485
acceptance mask registers 0 and 1 (AMR0 and AMR1)	475
acceptance mask select register (AMSR)	473
accumulator (A)	45
acknowledgement	433

activation.....	227
address field, effective	586, 603
address generation type	35
address match detection function, block diagram of	504
address match detection function, operation of	507
address match detection function, system configuration example of	508
addressing	585
addressing slave	432
addressing, direct.....	587
addressing, indirect.....	593
alternative mode	520
Analog input enable register	312
analog input enable registers.....	213
arbitration	432
asynchronous LIN mode (operation mode 3), operation in	389
asynchronous mode, operation in.....	384
automatic ready function selection register (ARSR)	194

B

bank addressing type.....	39
bank select prefix	53
baud rate/reload counter register 0 and 1	367
baudrate reload counter register 0 and 1 (BGR0/1)	367
bidirectional communication function	393
bit timing register (BTR)	458
bit timing, setting	489
Block diagram of A/D converter	315
block diagram of clock generation block.....	117
Block diagram of Flash/Mask ROM version.....	6
block diagram of low-power consumption control circuit	155
block diagram of MB90V340(S)	5
block diagram of watch timer	277
buffer address pointer (BAP)	83
bus control register (IBCR)	413
bus control register (IBCR) content.....	415
bus control signal selection register (ECSR)	197
bus operation stop (HALT = 0), condition for canceling	454

bus operation stop (HALT = 1), state during	454
bus operation stop (HALT=1), condition for setting	454
bus status register (IBSR)	410
bus status register (IBSR) content	411

C

calculating baud rate	377
calculating execution cycle count	601
CAN controller, block diagram of	439
CAN controller, canceling transmission request from	483
CAN controller, completing transmission of	484
CAN controller, feature of	438
CAN controller, reception flowchart of	488
CAN controller, starting transmission of	483
CAN controller, transmission flowchart of	484
CAN direct mode register (CDMR)	499
CAN direct mode register content	499
CAN switch register (CANSWR) content	497
CAN2 RX/TX pin switching register (CANSWR)	497
Caution for disabling message buffers by BVAL bits	500
Cautions	337
CE control	645
chip erase/sector erase command sequence	646
clock	114
clock control register (ICCR)	428
clock control register (ICCR) content	429
clock generation block, block diagram of	117
clock mode	153
clock mode transition	126
clock modulator control register (CMCR)	135
clock modulator control register content	136
clock monitor, block diagram of	340
clock output enable register (CLKR)	341
clock prescaler setting	430
clock selection register (CKSCR), configuration of	120
clock selection registers	119
clock supply map	115
command sequence table	524
common machine clock frequencies	430
common register bank prefix (CMR)	54
completing reception	487
condition code register (CCR)	47
configuration of the PLL/Subclock control register (PSCCR)	124

connection of oscillator or external clock to microcontroller	131
Continuous mode	327
control status register	255
Control Status Register (ADCS0)	317
Control Status Register (ADCS1)	319
control status register (CSR)	449
control status register (CSR) (upper)	451
control status register (CSR-lower) content	450
control status register (CSR-upperr) content	452
control status register (FMCS), flash memory	522
Conversion using EI ² OS	329
Converted-data protection function	336
correspondence between reset cause bits and reset causes	148
counter operation state	273
counting example	379
CPU intermittent operating mode	153
CPU intermittent operation mode	160
CPU memory space, outline of	33
CPU operating mode and current consumption	152
current consumption	152
cycle count, execution	600

D

data counter (DCT)	82
data direction register	210
data frame and remote frame, processing for reception of	486
data polling	647
data polling flag (DQ7)	528
Data register (ADCR0, ADCR1)	322
data register (IDAR)	427
data register content	427
data register x (x = 0 to 15) (DTRx)	481
delayed interrupt cause issuance/cancellation register (DIRR)	111
delayed interrupt occurrence	112
delayed interrupt, block diagram of	110
direct addressing	587
Direct memory access (DMA)	65
DIV A, Ri and DIVW A, RWi instruction, precautions for use of	58
DIV A, Ri instruction	58
DIV A, Ri instructions without precaution	59
DIVW A, RWi instruction	58
DIVW A, RWi instructions without precaution	59
DLC register x (x = 0 to 15) (DLCRx)	480

INDEX

DTP operation	308
DTP request, switching between external interrupt and	309
DTP/external interrupt, note on using	310
DTP/external interrupt, outline of	302
DTP/external interrupts registers.....	303

E

each operating mode table, operation status in	174
effective address field	586, 603
EI ² OS operation flow	85
EI ² OS status register (ISCS).....	84
enable sector protect/verify sector protect	649
entire flash memory, block diagram of	517
erasing chips	540
erasing flash memory	516
erasing sector	541
erasing sector in flash memory	541
Example of flow of data protection function (when EI ² OS is used)	337
example of program patch processing	509
Example of starting of EI ² OS in continuous mode	332
Example of starting of EI ² OS in single mode	330
Example of starting of EI ² OS in stop mode	334
execution cycle count.....	600
execution cycle count, calculating	601
explanation of different block	349
extended communication control register (ECCR3)	365
extended intelligent I/O service (EI ² OS).....	63, 80
extended intelligent I/O service descriptor (ISD)	82
extended status/control register (ESCR3).....	363
external address output control register (HACR)	196
external clock, using.....	379
external event counter.....	270
external interrupt operation	307
external interrupt request	309
External interrupt source select register.....	306
external memory access (bus pin control circuit)	192
external memory access control signal	200
external memory access register	193
external memory access, block diagram of	192
external reset pin, block diagrams of.....	144

F

F ² MC-16LX instruction list	607
feature.....	3, 406
Features of A/D converter.....	312
flag change disable prefix (NCC)	54
flash memory (erasing chip), erasing all data in ..	540
flash memory control signal	520
flash memory mode	520
flash memory sector, restarting erasing of.....	544
flash memory sector, suspending erasing of	543
flash memory write/erase, detailed explanation of	536
flash memory, erasing optional data (erasing sectors) in	541
flash memory, note on using.....	545
flash memory, writing data to	538
flash microcomputer programmer (power supplied from programmer), example of minimum connection to	564
flash microcomputer programmer (user power supply used), example of minimum connection to	562
frame format, setting.....	489
free-running Timer (lower), control status register of	236
free-running timer (upper), control status register of	238
free-running-timer, data register of	235

G

general-purpose register.....	44

H

handling device	25
hardware interrupt.....	62, 73
hardware interrupt operation.....	74
hardware interrupt, occurrence and release of	75
hardware interrupt, structure of.....	73
hardware sequence flag.....	526
hold function.....	204

I

I/O area	34
I/O map	568
I/O map (35XX addresses)	576
I/O port.....	206
I/O port register	207
I/O register address pointer (IOA)	83
I ² C interface registers	408
ID register x (x = 0 to 15) (IDRx)	478

IDE register (IDER)	462
ILSR, initialization of	215
indirect addressing	593
input capture	253
input capture (2 channels per one module)	231
input capture block diagram	253
input capture data register	254
input capture edge register (ICE01, ICE23, ICE45, ICE67)	257
input capture fetch timing, sample of	259
input capture input timing	260
input level select register	214
input resistor register (RDR), block diagram of	212
input-output circuit	20
instruction map, structure of	621
instruction presentation item and symbol, description of	604
instruction type	584
intelligent I/O service (EI ² OS) function and interrupt	262
inter-CPU connection method	383
Interrupt cause, interrupt vector, and interrupt control register	653
interrupt control register	653
interrupt control register (ICR)	68
interrupt disable instruction	56
interrupt flow	71
interrupt level mask register (ILM)	49
interrupt vector	66, 653
interrupt vector, list of	651
interrupt, intelligent I/O service (EI ² OS) function and	262
interrupt/DTP enable register (ENIR: interrupt request enable register)	304
interrupt/DTP flags (EIRR: external interrupt request register)	304
interval interrupt function	221

L

last event indicator register (LEIR)	455
last event indicator register (LEIR) content	456
LIN-master-slave communication function	399
low-power consumption control circuit, block diagram of	155
low-power consumption mode control register (LPMCR)	157
low-power consumption mode, setting	490

M

machine clock	127

main clock mode	126
master-slave communication function	395
MB90F347 serial programming connection, basic configuration of	554
memory access mode	186
memory space in each bus mode	189
memory space map	36
memory space, multi-byte data allocation in	41
message buffer	477
message buffer (DLC register and data register), list of	445
message buffer (ID register), list of	442
message buffer (x), procedure for reception by	493
message buffer (x), procedure for transmission by	491
message buffer control registers	448
message buffer valid register (BVALR)	461
message buffers	448
micro DMAC functions	89
mode data	188
mode fetch	146
mode pin	145, 187
multi-byte data, accessing	41
multi-level message buffer, setting configuration of	495
multiple interrupt	77

O

operation enable bit	383
operation status during standby mode	161
operation, note on	110
oscillating clock frequency and serial clock input frequency	557
oscillation stabilization wait and reset state	143
oscillation stabilization wait interval	130
oscillation stabilization wait time	182
output compare	241
output compare (2 channels per one module)	230
output compare (lower), control status register of	243
output compare (upper), control status register of	245
output compare block diagram	241
output compare register	242
output compare register 0, clearing counter upon match with	240
output compare timing	252
overall control registers	448
overall control registers, list of	440
overflow, clearing counter by	239

INDEX

overview 134

P

package dimension 11
pin assignment of Flash/Mask ROM version..... 7
pin function 13
PLL clock mode..... 126
PLL clock multiplier, selection of 127
port data register 208
PPG0 operation mode control register (PPGC0)
..... 287
PPG0, 1 clock select register (PPG01) 291
PPG1 operation mode control register (PPGC1)
..... 289
prefix code..... 57
prefix code, consecutive..... 57
prefix instruction 56
prefix instruction, restriction on interrupt disable
instruction and..... 56
prescaler setting 459
Procedure for using DMAC 105
processor status (PS)..... 47
product overview 2
program address detection control status register
(PACSR) 506
program address detection registers (PADR0 to
PADR5)..... 505
program counter (PC) 50
program patch processing, example of 509
programmable restart..... 380
programming flow chart..... 434
pull-up control register (PUCR) 212

R

RAM area 34
read access, data read by 643
read state, setting flash memory to 537
ready function..... 202
receive and transmit error counter (RTEC) 457
receive and transmit error counters (RTEC) content
..... 457
receive overrun 486
receive overrun register (ROVRR) 471
received message, storing 485
reception and transmission data registers 3
(RDR3/TDR3) 361
reception complete register (RCR)..... 469
reception interrupt enable register (RIER)..... 472
Reception interrupt generation and flag set timing
..... 371

recommended set 191
register bank 51
register bank pointer (RP)..... 48
Registers for A/D converter..... 316
reload register (PRL/PRLH)..... 293
reload value and pulse width, relationship between
8/16-bit PPG 295
remote frame receiving wait register (RFWTR) ... 465
remote frame, processing for reception of 486
remote request receiving register (RRTRR) 470
request level setting register
(ELVR: External interrupt level register) 305
reset cause bit..... 147
reset cause bit, notes about..... 148
reset causes and oscillation stabilization wait times
..... 142
reset operation, overview of..... 145
reset state, setting flash memory to 537
reset, causes of..... 140
ROM area 34
ROM mirroring module, block diagram of 512
ROM mirroring register (ROMM)..... 513
RST and RY/BY timing 648
RY/BY timing during writing/erasing 648

S

sample output waveform when CMOD[1:0] = "10"
..... 250
sample output waveform when CMOD[1:0] = "11"
..... 251
sample output waveform with two compare
registers when CMOD[1:0] = "01" 249
sample output waveform when CMOD[1:0] = "00"
..... 248
SCC, MSS and INT bit competition..... 418
sector erase timer flag (DQ3)..... 532
serial clock input frequency..... 557
serial control register 3 (SCR3)..... 355
serial mode register 3 (SMR3) 357
serial programming connection (power supplied
from programmer), example of 560
serial programming connection (user power supply
used), example of 558
serial status register 3 (SSR3) 359
setting ID..... 489
Setting Register (ADSR) 323
seven bit slave address mask register (ISMK).... 425
seven bit slave address mask register content.... 426
seven bit slave address register 425
seven bit slave address register content..... 425

signal mode	383	timer control register (TMSCR), register layout of	265
single chip mode, status of each pin in	175	timing limit exceeded flag (DQ5)	531
Single mode 1 / 2	327	toggle bit	647
slave address detection	431	toggle bit flag (DQ6)	530
slave address masking	432	toggle bit-2 flag (DQ2)	534
sleep mode, release of	163	transmission cancel register (TCANR)	466
sleep mode, switching to	163	transmission complete register (TCR)	467
software interrupt	63, 78	transmission interrupt enable register (TIER)	468
software interrupt operation	78	transmission interrupt generation and flag set timing	373
special register	42	transmission interrupt request generation timing	374
standby mode	153	transmission request register (TREQR)	463
standby mode and interrupt, switching to	181	transmission RTR register (TRTRR)	464
start condition	431		
status change diagram	173	U	
stop condition	431	UART3 as master device	400
Stop mode	327	UART3 as slave device	401
stop mode, release of	171	UART3 baud rate selection	375
stop mode, switching to	170	UART3 direct pin access	392
structure	81	UART3 EI ² OS functions	370
structure of instruction map	621	UART3 functions	344
sub-clock mode	126	UART3 interrupt and EI ² OS	347
suggested division ratios for different machine speeds and baud rates	378	UART3 interrupts and EI ² OS	370
synchronization method	383	UART3 operation mode	346
synchronous mode (operation mode 2), operation in	386	UART3 pins	353
system stack pointer (SSP), user stack pointer (USP) and	46	UART3 registers	354
		UART3, block diagram of	348
T		UART3, Notes on using	402
temporary sector protect cancellation	650	UART3, operation of	382
ten bit address mask register (ITMK)	423	undefined instruction, exception due to execution of	107
ten bit address mask register (ITMK) content	424	undefined instruction, execution of	107
ten bit slave address register (ITBA)	422	user power supply	558, 562
ten bit slave address register (ITBA) content	422	user stack pointer (USP) and system stack pointer (SSP)	46
the clock modulator, recommended startup sequence for	137		
the External BUS 16-bit data bus mode, status of each pin in	177	W	
the External BUS 8-bit data bus mode, status of each pin in	179	watch mode, release of	169
timebase counter	221	watch mode, switching to	168
timebase timer control register (TBTC)	219	watch timer	280
timebase timer mode, release of	166	watch timer register	276
timebase timer mode, switching to	166	watch timer, block diagram of	277
timebase timer, block diagram of	218	watch timer, interval interrupt function of	280
timebase timer, outline of	218	watch-dog counter	227
timer control register (TMSCR), register content of	265	watch-dog stop	227
		watch-dog timer block diagram	224
		watch-dog timer control register (WDTC)	225

INDEX

watch-dog timer control register (WTC)	278	write, data polling, read (WE control).....	644
WE control.....	644	writing to flash memory	516
write, data polling, read (CE control)	645		

CM44-10125-1E

FUJITSU SEMICONDUCTOR • CONTROLLER MANUAL

F²MC-16LX

16-BIT MICROCONTROLLER

MB90340 Series

HARDWARE MANUAL

January 2004 the first edition

Published **FUJITSU LIMITED** Electronic Devices

Edited Standardization Promoting Dept.
