

Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.



F²MC-16LX 16-Bit Microcontroller MB90990 Series Hardware Manual

Doc. # 002-04564 Rev. *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intl): 408.943.2600
<http://www.cypress.com>

Copyrights

© Cypress Semiconductor Corporation, 2008-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Preface



Objectives and intended reader

Thank you very much for your continued patronage of Cypress products.

The MB90990 series has been developed as a general-purpose version of the F²MC-16LX family, 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 MB90990 series for engineers who actually use the MB90990 series to design products. Please read this manual first.

Note: F²MC is the abbreviation of Cypress Flexible Microcontroller.

Trademark

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

Structure of this manual

This manual contains the following 27 chapters and appendices.

"1. Overview"

This chapter explains the features and basic specifications.

"2. CPU"

This chapter explains the CPU.

"3. Interrupts"

This chapter explains the function and operation of the interrupts and the extended intelligent I/O service (EI²OS).

"4. Delayed Interrupt Generation Module"

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

"5. Clocks"

This chapter explains the clocks used by MB90990 series microcontrollers.

"6. Clock Supervisor"

This chapter describes the functions and operations of the clock supervisor. This function can be used by only the products with "J"-suffix.

"7. Rests"

This chapter describes resets for the MB90990 series microcontrollers.

"8. Low Power Consumption Mode"

This chapter explains the low-power consumption mode of MB90990 series microcontrollers.

"9. Memory Access Modes"

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

"10. I/O Ports"

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

"11. Time-Base Timer"

This chapter explains the functions and operations of the time-base timer.

"12. Watchdog Timer"

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

"13. 16-Bit I/O Timer"

This chapter explains the function and operation of the 16-bit I/O timer.

"14. 16-Bit Reload Timer"

This chapter describes the functions and operation of the 16-bit reload timer.

"15. Watch Timer"

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

"16. 8-/16-Bit PPG Timer"

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

"17. DTP/External Interrupt"

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

"18. 8-/10-Bit A/D Converter"

This chapter describes the functions and operations of the 8-/10-bit A/D converter.

"19. Low Voltage Detection/CPU Operating Detection Reset"

This chapter explains the function and operating the low voltage detection/CPU operating detection reset. This function can use only the product with "J" suffix of MB90990 series. Evaluation products correspond to the CPU operation detection.

"20. LIN-UART"

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

"21. Can Controller"

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

"22. Address Match Detection Function"

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

"23. ROM Mirroring Function Select Module"

This chapter describes the functions and operations of the ROM mirroring function select module.

"24. Flash Memory"

This chapter explains the functions and operation of the flash memory.

"25. Examples of Serial Programming Connection for Flash Memory Products"

This chapter shows an example of a serial programming connection when the AF220/AF210/AF120/AF110 flash microcontroller programmer made by Yokogawa Digital Computer Corporation is used.

"26. Clock Calibration Unit"

This chapter outlines the clock calibration unit and describes its register configuration and functions. This function can be used by only the products with "J"-suffix. Evaluation products don't correspond to the CR clock calibration.

"27. D/A Converter"

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

" Appendix"

The appendices provide I/O maps, instructions by F²MC-16LX, and other information.

Contents



1. Overview	13
Overview of MB90990 Series	13
Block Diagram of MB90990 Series	17
Package Dimension	20
Pin Assignment	21
Pin Functions	22
I/O Circuit	25
Notes on Handling Device	29
2. CPU	35
Overview of the CPU	35
Memory Space	36
Memory Map	37
Linear Addressing	38
Bank Addressing Types	39
Multi-byte Data in Memory Space	40
Registers	41
Register Bank	48
Prefix Codes	49
Interrupt Disable Instructions	51
3. Interrupts	53
Overview of Interrupts	53
Interrupt Vector	56
Interrupt Control Registers (ICR00 to ICR15)	57
Interrupt Flow	61
Hardware Interrupts	63
Software Interrupts	66
Extended Intelligent I/O Service (EI ² OS)	67
Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI ² OS)	71
Exceptions	73
4. Delayed Interrupt Generation Module	75
Overview of Delayed Interrupt Generation Module	75
Block Diagram of Delayed Interrupt Generation Module	75
Configuration of Delayed Interrupt Generation Module	76
Explanation of Operation of Delayed Interrupt Generation Module	77
Notes on Using Delayed Interrupt Generation Module	78
Program Example of Delayed Interrupt Generation Module	78
5. Clocks	81
Clocks	81

Block Diagram of the Clock Generation Block	82
Clock Selection Register (CKSCR)	84
PLL/Sub clock Control Register (PSCCR)	87
Clock Mode	90
Oscillation Stabilization Wait Time	94
Connection of an Oscillator	94
6. Clock Supervisor	97
Overview of Clock Supervisor	97
Configuration of Clock Supervisor	97
Registers of Clock Supervisor	99
Operations of Clock Supervisor	101
Notes on Using Clock Supervisor	105
7. Rests	107
Overview of Resets	107
Reset Sources and Oscillation Stabilization Wait Times	109
External Reset Pin	111
Reset Operation	112
Reset Source Bits	113
Status of Pins in Reset	117
8. Low Power Consumption Mode	119
Overview of Low-power Consumption Mode	119
Block Diagram of the Low-power Consumption Circuit	122
Low-power Consumption Mode Control Register (LPMCR)	124
CPU Intermittent Operation Mode	127
Standby Mode	128
Status Change in Standby Mode	135
Status of Pins in Standby Mode and during Reset	136
Notes on Using Low-power Consumption Mode	137
9. Memory Access Modes	141
Outline of Memory Access Modes	141
10. I/O Ports	145
I/O Ports	145
I/O Port Registers	145
11. Time-Base Timer	153
Overview of Time-base Timer	153
Block Diagram of Time-base Timer	154
Configuration of Time-base Timer	156
Interrupt of Time-base Timer	158
Explanation of Operations of Time-base Timer Functions	159
Notes on Using Time-base Timer	163
Program Example of Time-base Timer	163
12. Watchdog Timer	165
Overview of Watchdog Timer	165
Configuration of Watchdog Timer	167

Watchdog Timer Registers	168
Explanation of Operations of Watchdog Timer Functions	170
Notes on Using Watchdog Timer	174
Program Example of Watchdog Timer	175
13. 16-Bit I/O Timer	177
Overview of 16-bit I/O Timer	177
Block Diagram of 16-bit I/O Timer	178
Configuration of 16-bit I/O Timer	182
Interrupts of 16-bit I/O Timer	192
Explanation of Operation of 16-bit Free-run Timer	193
Explanation of Operation of Input Capture	194
Notes on Using 16-bit I/O Timer	196
Program Example of 16-bit I/O Timer	197
14. 16-Bit Reload Timer	199
Overview of the 16-bit Reload Timer	199
Block Diagram of 16-bit Reload Timer	201
Configuration of 16-bit Reload Timer	203
Interrupts of 16-bit Reload Timer	210
Explanation of Operation of 16-bit Reload Timer	210
Notes on Using 16-bit Reload Timer	219
Sample Program of 16-bit Reload Timer	222
15. Watch Timer	225
Overview of Watch Timer	225
Block Diagram of Watch Timer	227
Configuration of Watch Timer	228
Watch Timer Interrupt	230
Explanation of Operation of Watch Timer	231
Program Example of Watch Timer	232
16. 8-/16-Bit PPG Timer	235
Overview of 8-/16-bit PPG Timer	235
Block Diagram of 8-/16-bit PPG Timer	237
Configuration of 8-/16-bit PPG Timer	243
Interrupts of 8-/16-bit PPG Timer	251
Explanation of Operation of 8-/16-bit PPG Timer	252
Notes on Using 8-/16-bit PPG Timer	259
17. DTP/External Interrupt	261
Overview of DTP/External Interrupt	261
Block Diagram of DTP/External Interrupt	262
Configuration of DTP/External Interrupt	263
Explanation of Operation of DTP/External Interrupt	270
Notes on Using DTP/External Interrupt	276
Program Example of DTP/External Interrupt Circuit	277
18. 8-/10-Bit A/D Converter	281
Overview of 8-/10-bit A/D Converter	281
Block Diagram of 8-/10-bit A/D Converter	282

Configuration of 8-/10-bit A/D Converter	285
Interrupts of 8-/10-bit A/D Converter	299
Explanation of 8-/10-bit A/D Converter Operations	299
Notes on Using 8-/10-bit A/D Converter	308
19. Low Voltage Detection/CPU Operating Detection Reset	311
Overview of Low Voltage/CPU Operating Detection Reset Circuit	311
Configuration of Low Voltage/CPU Operating Detection Reset Circuit	312
Low Voltage/CPU Operating Detection Reset Circuit Register	314
Operating of Low Voltage/CPU Operating Detection Reset Circuit	317
Notes on Using Low Voltage/CPU Operating Detection Reset Circuit	318
Sample Program for Low Voltage/CPU Operating Detection Reset Circuit	319
20. LIN-UART	321
Overview of LIN-UART	321
Configuration of LIN-UART	323
LIN-UART Pins	327
LIN-UART Registers	328
LIN-UART Interrupts	341
LIN-UART Baud Rates	346
Operation of LIN-UART	352
Notes on Using LIN-UART	371
21. Can Controller	373
Features of CAN Controller	373
Block Diagram of CAN Controller	374
List of Registers	375
Classifying CAN Controller Registers	380
Transmission of CAN Controller	411
Reception of CAN Controller	413
Reception Flowchart of CAN Controller	415
How to Use CAN Controller	416
Procedure for Transmission by Message Buffer (x)	416
Procedure for Reception by Message Buffer (x)	418
Setting Configuration of Multi-level Message Buffer	419
CAN Direct Mode Register	421
Notes on Using CAN Controller	421
22. Address Match Detection Function	423
Overview of Address Match Detection Function	423
Block Diagram of Address Match Detection Function	423
Configuration of Address Match Detection Function	424
Explanation of Operation of Address Match Detection Function	432
Program Example of Address Match Detection Function	438
23. ROM Mirroring Function Select Module	439
Overview of ROM Mirroring Function Select Module	439
ROM Mirroring Function Select Register (ROMM)	440
24. Flash Memory	443
Overview of Flash Memory	443

Block Diagram of the Entire Flash Memory	444
Sector Configuration of the Flash Memory	446
Write/Erase Modes	446
Flash Memory Control Status Register (FMCS)	447
Flash Memory Write Control Register (FWR0/FWR1)	450
Starting the Flash Memory Automatic Algorithm	454
Confirming the Automatic Algorithm Execution State	455
Writing to and Erasing Flash Memory	459
Notes on Using Flash Memory	465
Flash Security Feature	466
25. Examples of Serial Programming Connection for Flash Memory Products	469
Basic Configuration of Serial Programming Connection	469
Example of Serial Programming Connection (User Power Supply Used)	472
Example of Serial Programming Connection (Power Supplied from Programmer)	473
Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)	475
Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)	476
26. Clock Calibration Unit	479
Overview of Clock Calibration Unit	479
Register of Clock Calibration Unit	481
Application Notes	487
27. D/A Converter	489
Overview of D/A Converter	489
Block Diagram of D/A Converter	490
Configuration of D/A Converter	491
D/A Converter Registers	492
Sample Program for the D/A Converter	493
28. Appendix	495
I/O Maps	495
I/O Map (Addresses:007900H to 007FFFH)	499
29. Index	589
30. Numerics	603
31. Revision History	620

1. Overview



This chapter explains the features and basic specifications.

1.1 Overview of MB90990 Series

1.2 Block Diagram of MB90990 Series

1.3 Package Dimension

1.4 Pin Assignment

1.5 Pin Functions

1.6 I/O Circuit

1.7 Notes on Handling Device

1.1 Overview of MB90990 Series

The MB90990 series is a 16-bit microcontroller designed for automotive applications and contains CAN function, capture, compare timer, A/D converter, and so on.

■ Features of MB90990 Series

MB90990 series has the following features:

- Clock

Built-in PLL clock multiplying circuit

Machine clock (PLL clock) selectable from 1/2 frequency of oscillation clock or 1 to 8-multiplied oscillation clock (4 MHz to 32 MHz when oscillation clock is 4 MHz)

Minimum instruction execution time: 31.25 ns (4-MHz oscillation clock and 8-multiplied PLL clock)

- 16-MB CPU memory space

Internal 24-bit addressing

- Instruction system optimized for controllers

Various data types (bit, byte, word, long word)

23 types of addressing modes

Enhanced signed instructions of multiplication/division and RETI instruction

High-accuracy operations enhanced by 32-bit accumulator

- Instruction system for high-level language (C language)/multi-task

System stack pointer

Enhanced pointer indirect instructions

Barrel shift instructions

- Higher execution speed

4-byte instruction queue

- Powerful interrupt function

Powerful interrupt function with 8 levels and 34 factors

Corresponds to 8-channel external interrupts

- ❑ CPU-independent automatic data transfer function

Extended intelligent I/O service (EI²OS): Maximum 16 channels

- ❑ Low-power consumption (standby) modes

Sleep mode (stops CPU operation clock)

Time-base timer mode (operates only oscillation clock and sub clock, time-base timer and watch timer)

Watch mode (operates only sub clock and watch timer)

Stop mode (stops oscillation clock and sub clock)

CPU intermittent operation mode

- ❑ Process

CMOS Technology

- ❑ I/O ports

General-purpose I/O ports (CMOS output)

- 36 ports

- ❑ Timers

Time-base timer, watch timer, watchdog timer: 1 channel

8-/16-bit PPG timer: 8 bits × 6 channels or 16 bits × 3 channels

16-bit reload timer: 2 channels

16-bit I/O timer

- 16-bit free-run timer: 1 channel (FRT0: ICU0/ICU1/ICU2/ICU3)

- 16-bit input capture (ICU): 4 channels

- ❑ Full-CAN Controller: 1 channel

Conforms to CAN Specification Ver. 2.0A and Ver. 2.0B.

Built-in 16 message buffers

CAN wake up

- ❑ UART (LIN): LIN-UART × 2 channels, UART × 1 channel

Full-duplex double buffer

Clock asynchronous or clock synchronous serial transfer

- ❑ DTP/external interrupt: 8 channels, CAN wake up: 1 channel

External input to start EI²OS and generate external interrupt

- ❑ Delayed interrupt generation module

Generates interrupt request for task switching

- ❑ 8-/10-bit A/D converter: 16 channels

8-bit and 10-bit resolutions

Start by external trigger input

Conversion time: 3 μs (including sampling time at 32MHz machine clock frequency)

- ❑ Program patch function

Detects address match for six address pointers

- ❑ Low voltage/CPU operation detection reset function (product with "J"-suffix)

Detects low voltage (set program to 2.8V to 4.2V) and reset automatically

Automatically reset when program hangs up and counter is not cleared within interval time (set approx. 16.4ms to approx. 524ms@ 4MHz external)

- ❑ Clock supervisor (product with "J"-suffix)

- ❑ Clock calibration unit (products with "J"-suffix)

Capable of improving precision of CR oscillation circuit by calculating calibration value from measurement and performing trimming.

- Changeable port input voltage level

Automotive input level/CMOS Schmitt input level (initial value in single-chip mode is Automotive level)

- 8-bit D/A converter: 1 channel

■ Product Lineup

Table 1-1. shows product lineup of MB90990 series.

Table 1-1. Product Lineup

<div>Product name</div> <div>Item</div>	MB90F997JBS, MB90F997MBS	MB90V950AJAS, MB90V950AMAS
Product type	Flash memory product	Evaluation product
CPU	F ² MC-16LX CPU	
System clock	On-chip PLL clock multiplier method (×1, ×2, ×3, ×4, ×6, ×8, 1/2 when PLL stops) Minimum instruction execution time of 31.25 ns (at original oscillation of 4 MHz, multiply-by-eight)	
ROM	Flash memory Main :128K bytes Satellite : 32K bytes	External
RAM	8K bytes	30K bytes
Package	LQFP-48	PGA-299
Emulator power supply*1	-	Included
FPGA data*2	-	050617 version
Adapter board*2	-	After MB2147-20 04C version

*1: It is a setting for the jumper switch (TOOL VCC) when the emulator (MB2147-01) is used.
Refer to the Emulator hardware manual for details.

*2: When using the FPGA data and adapter board other than the above ones, please contact their sales representative.

■ Functions

Table 1-2. Peripheral Functions (Sheet 1 of 2)

<div>Product name</div> <div>Item</div>	MB90F997JBS	MB90F997MBS	MB90V950AJAS	MB90V950AMAS
UART	LIN-UART × 2 channels UART × 1 channel		7 channels	
	A wide-range communication speed can be set with the dedicated reload timer. LIN function can be used as a LIN master and LIN slave. (Only LIN-UART is provided)			
I ² C(400 kbps)	-		2 channels	
A/D converter	16 channels		24 channels	
	10/8-bit resolution Conversion time : Minimum 3 μs including sampling time (per channel)			
16-bit reload timer	2 channels		4 channels	
	Operating clock frequency: fsys/2 ¹ , fsys/2 ³ , fsys/2 ⁵ (fsys=system clock frequency) Supports the external event count function			
16-bit I/O timer	1 channel		2 channels	
	Outputs an interrupt signal at overflowing Supports timer clear when a match with output compare (ch.0 and ch.4) Operating clock frequency: fsys/2 ¹ , fsys/2 ² , fsys/2 ³ , fsys/2 ⁴ , fsys/2 ⁵ , fsys/2 ⁶ , fsys/2 ⁷ (fsys=system clock frequency) I/O timer 0 (clock input FRCK0) corresponds to ICU0/ICU1/ICU2/ICU3, OCU0/OCU1/OCU2/OCU3 I/O timer 1 (clock input FRCK1) corresponds to ICU4/ICU5/ICU6/ICU7, OCU4/OCU5/OCU6/OCU7			
16-bit output compare	-		8 channels	
	Outputs an interrupt signal when a match with 16-bit I/O timer and 16-bit compare registers Up to 3 of compare registers can be used to generate an output signal			
16-bit input capture	4 channels		8 channels	
	Performs the retention of I/O timer value and the generation of interrupt by the pin input (rising edge, falling edge, or both edges)			
8/16-bit PPG	3 channels		8 channels	
	8-bit reload counter × 6		8-bit reload counter × 16	
	Lower 8-bit reload register × 6		Lower 8-bit reload register × 16	
	Upper 8-bit reload register × 6		Upper 8-bit reload register × 16	
	Supports 8-bit and 16-bit operating modes 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 Operating clock frequency: fsys, fsys/2 ¹ , fsys/2 ² , fsys/2 ³ , fsys/2 ⁴ or 102.4 μs when fosc = 5 MHz (fsys = system clock frequency, fosc = oscillation clock frequency)			

Table 1-2. Peripheral Functions (Sheet 2 of 2)

<div>Product name</div> <div>Item</div>	MB90F997JBS	MB90F997MBS	MB90V950AJAS	MB90V950AMAS
CAN interface	1 channel		3 channels	
	Complies with CAN specification Ver. 2.0A and Ver. 2.0B Automatic retransmission when an error occurs Automatic response to CAN remote frame requests 16 message buffers with data and ID prioritization Supports multiple messages Flexible configuration of acceptable filtering: Full-bit compare/Full-bit mask/Partial-bit mask × 2 Supports 10 kbps to 1 Mbps (When using 1 Mbps, machine clock must operate at 8 MHz or higher. When using 10 kbps, it must operate at 16 MHz or less.)			
External interrupt	8 channels		16 channels	
	Edge detection or level detection can be configured			
Clock calibration unit	Yes	No	No	No
Clock supervisor	Yes	No	Yes	No
Low-voltage detection reset	Yes	No	No	No
CPU operation detection reset	Yes	No	Yes	No
Clock modulation	-		Yes	
D/A converter	1 channel		2 channels	
Sub clock	Yes	No	Yes	No
I/O Ports	General-purpose I/O ports (CMOS output): - 36 ports Input level setting: - Selectable among CMOS /Automotive level		General-purpose I/O ports (CMOS output): - 82 ports Input level setting: - Port0 to Port3: Selectable among CMOS/Automotive/TTL levels - Port4 to PortA: Selectable among CMOS/Automotive levels	
Flash Memory (Product with the flash memory only)	Supports automatic programming, Embedded Algorithm, Write/Erase/ Erase-Suspend/Resume commands A flag indicating completion of the algorithm Boot sector configuration Erase can be performed on each sector Flash security function to protect the contents of the flash memory			

1.2 Block Diagram of MB90990 Series

Figures below show a block diagram of the MB90990 series.

■ Block Diagram of Evaluation Product

Figure 1-1. and Figure 1-2. show the block diagram of evaluation product.

Figure 1-1. Block Diagram of Evaluation Product (MB90V950AMAS)

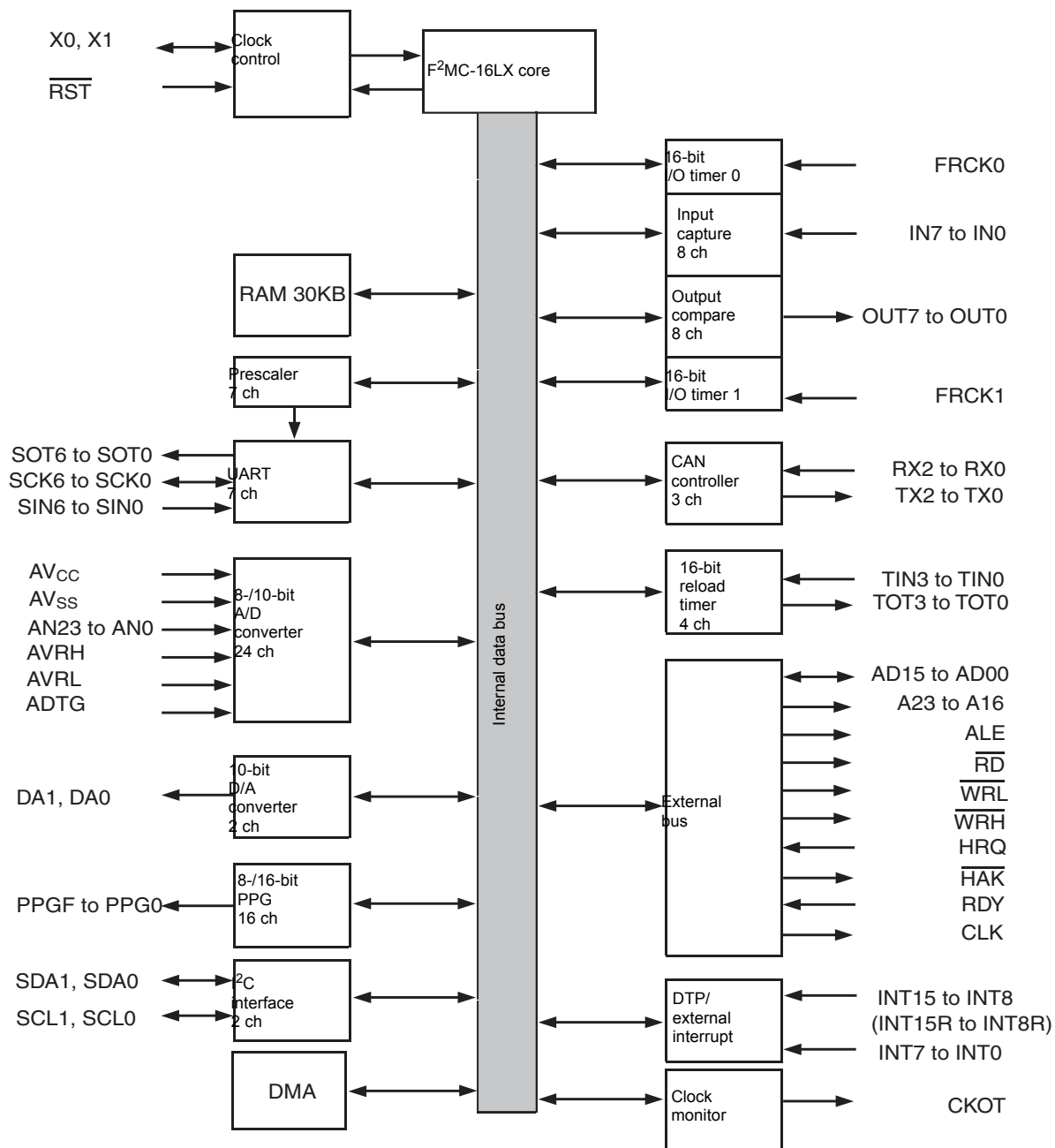
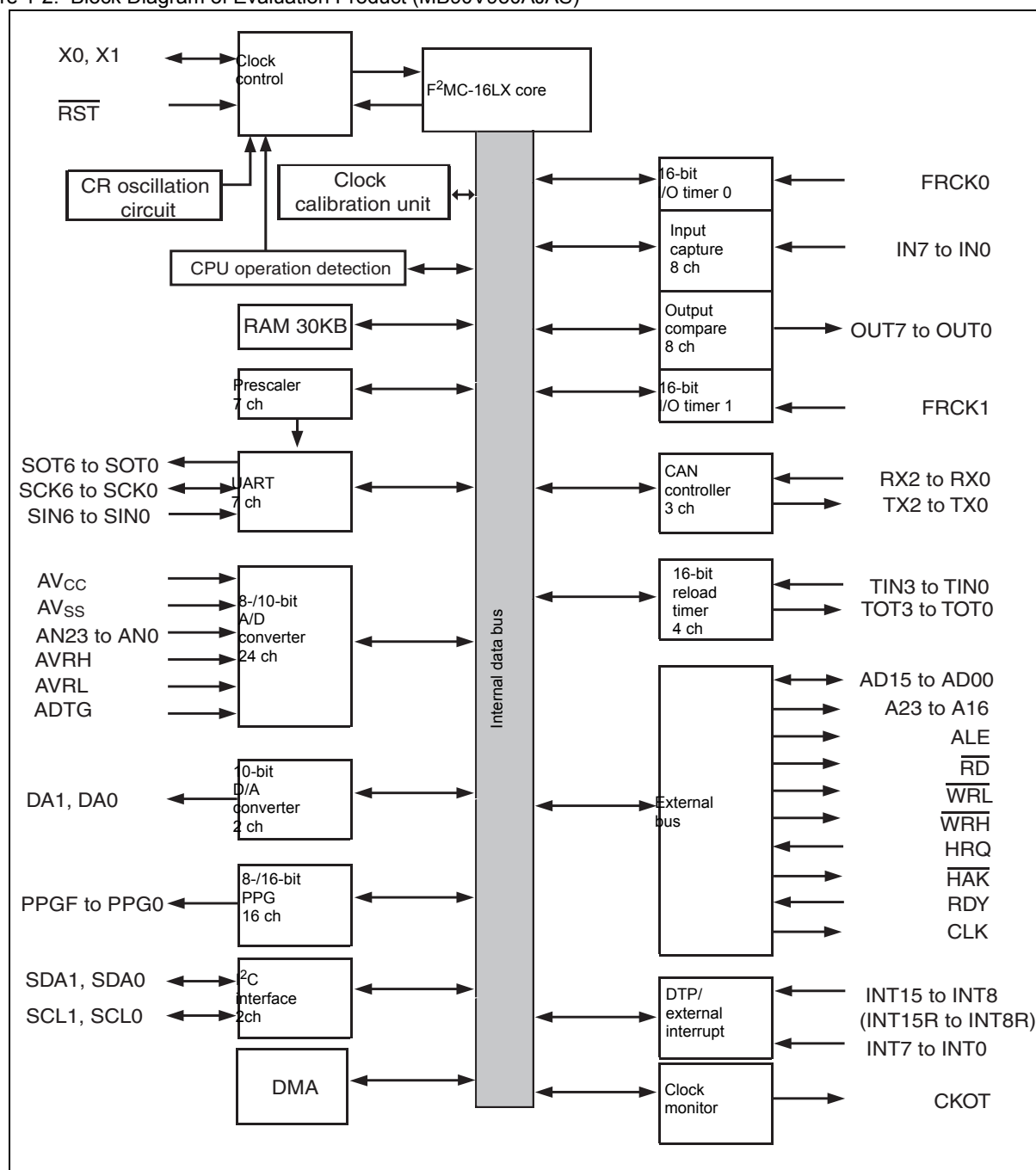


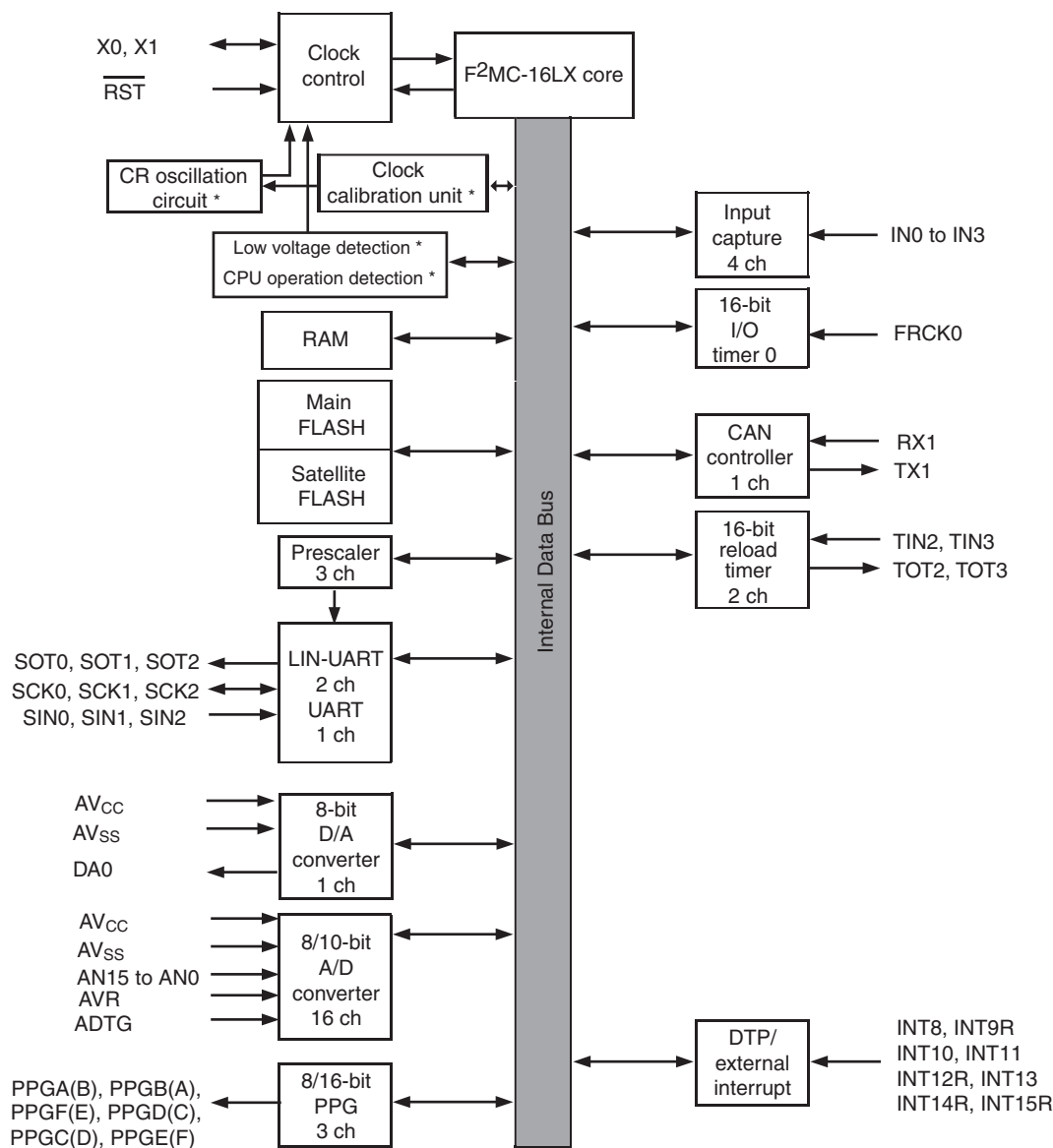
Figure 1-2. Block Diagram of Evaluation Product (MB90V950AJAS)



■ Block Diagram of Flash Memory Product

Figure 1-3. shows a block diagram of flash memory product.

Figure 1-3. Block Diagram of Flash Memory Product



*: Product with "J"-suffix

1.3 Package Dimension

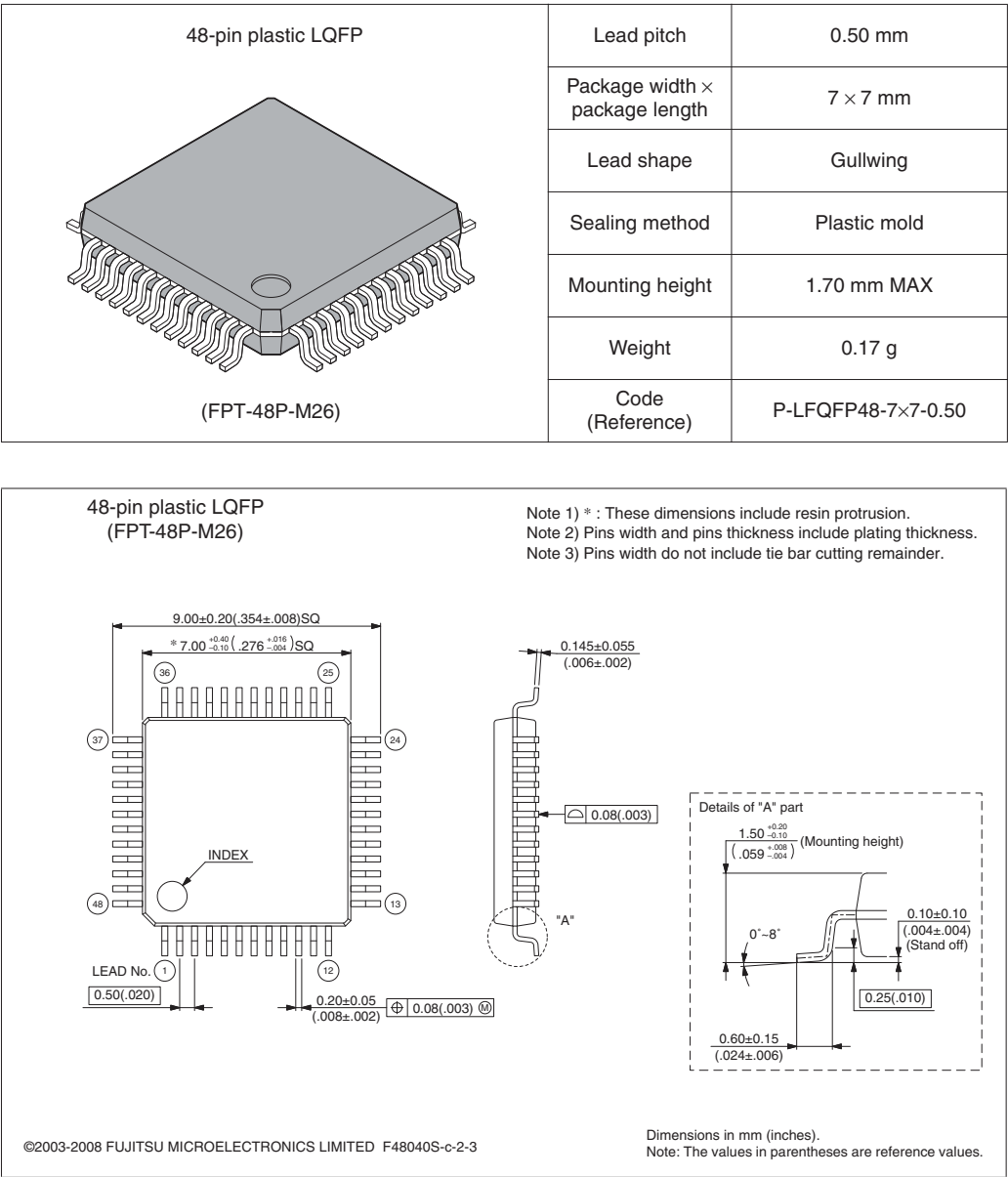
MB90990 series has a package.

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

■ Package Dimension (LQFP-48)

Figure 1-4. shows the package dimensions of LQFP-48 type.

Figure 1-4. Package Dimensions of LQFP-48 Type



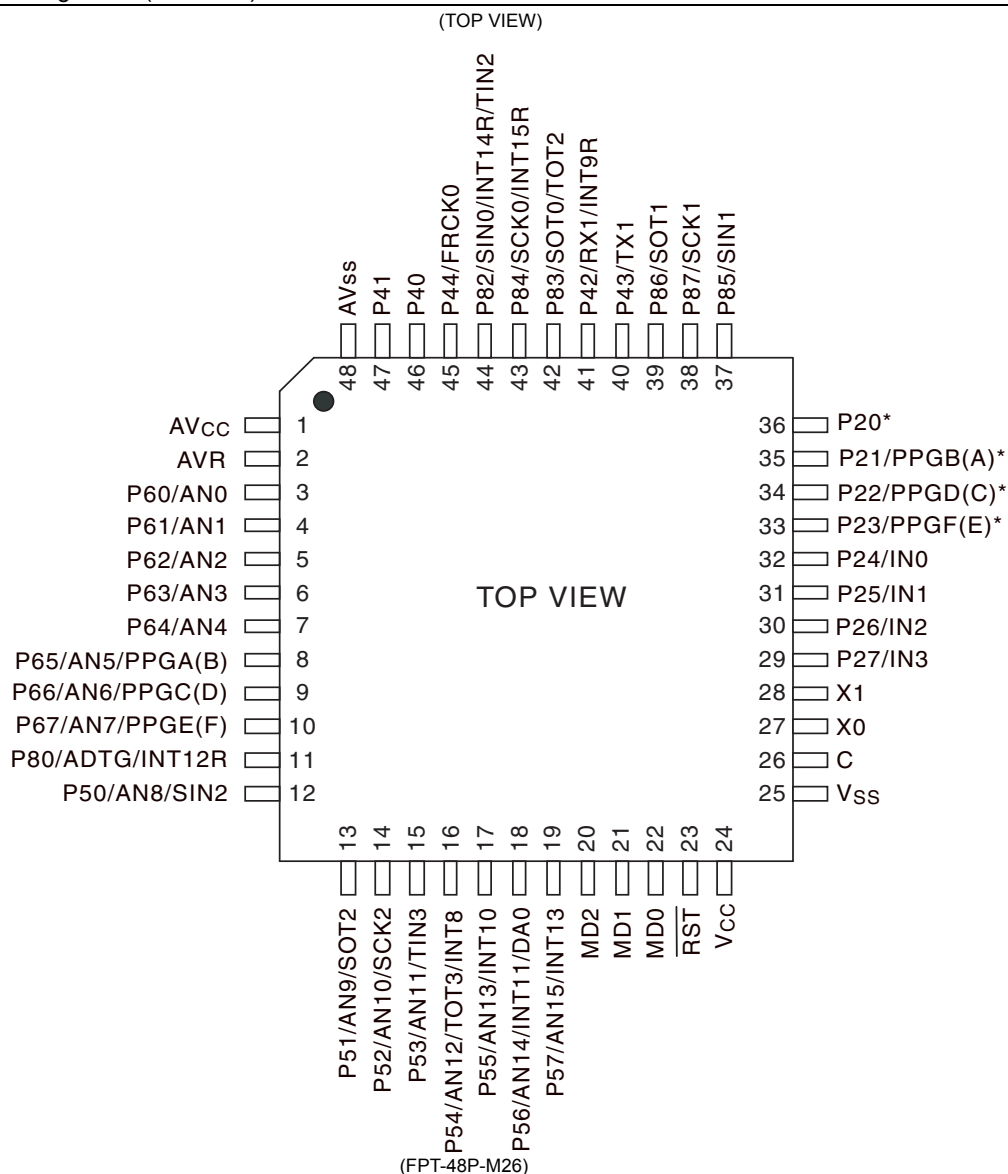
1.4 Pin Assignment

This section shows the pin assignment for the MB90990 series.

■ Pin Assignment (LQFP-48)

Figure 1-5. shows the pin assignment of LQFP-48 type.

Figure 1-5. Pin Assignment (LQFP-48)



*: High current port

1.5 Pin Functions

Table 1-3. describes the pin functions of the MB90990 series.

■ Pin Functions

Table 1-3. Pin Description (Sheet 1 of 3)

Pin number	Pin name	Circuit type	Functional description
1	AV _{CC}	I	V _{CC} power input pin for analog circuit
2	AVR	-	Power (V _{ref} +) input pin for A/D converter. The power supply should not be input V _{CC} exceeding.
3 to 7	P60 to P64	H	General-purpose I/O port
	AN0 to AN4		Analog input pin for A/D converter
8 to 10	P65 to P67	H	General-purpose I/O port
	AN5 to AN7		Analog input pin for A/D converter
	PPGA(B), PPGC(D), PPGE(F)		Output pin for PPG
11	P80	F	General-purpose I/O port
	ADTG		Trigger input pin for A/D converter
	INT12R		External interrupt request input pin for INT12R
12	P50	L	General-purpose I/O port
	AN8		Analog input pin for A/D converter
	SIN2		Serial data input pin for UART2
13	P51	H	General-purpose I/O port
	AN9		Analog input pin for A/D converter
	SOT2		Serial data output pin for UART2
14	P52	H	General-purpose I/O port
	AN10		Analog input pin for A/D converter
	SCK2		Clock I/O pin for UART2
15	P53	H	General-purpose I/O port
	AN11		Analog input pin for A/D converter
	TIN3		Event input pin for reload timer 3
16	P54	H	General-purpose I/O port
	AN12		Analog input pin for A/D converter
	TOT3		Output pin for reload timer 3
	INT8		External interrupt request input pin for INT8
17	P55	H	General-purpose I/O port
	AN13		Analog input pin for A/D converter
	INT10		External interrupt request input pin for INT10
18	P56	M	General-purpose I/O port
	AN14		Analog input pin for A/D converter
	INT11		External interrupt request input pin for INT11
	DA0		Analog output pin for D/A converter

Table 1-3. Pin Description (Sheet 2 of 3)

Pin number	Pin name	Circuit type	Functional description
19	P57	H	General-purpose I/O port (In the evaluation product, I/O circuit type is different from that in the flash memory product.)
	AN15		Analog input pin for A/D converter
	INT13		External interrupt request input pin for INT13
20	MD2	D	Input pin for selecting operation mode
21, 22	MD1, MD0	C	Input pin for selecting operation mode
23	RST	E	Reset input
24	V _{CC}	-	Power input pin (3.5 V to 5.5 V)
25	V _{SS}	-	Power input pin (0 V)
26	C	I	Capacity pin for stabilizing power supply. It should be connected to higher than or equal to 0.1 μ F ceramic capacitor.
27	X0	A	Oscillation input pin
28	X1	A	Oscillation output pin
29 to 32	P27 to P24	G	General-purpose I/O port. The register can be set to select whether to use pull-up resistor. This function is enabled in single-chip mode.
	IN3 to IN0		Event input pin for input capture 0 to 3.
33 to 35	P23 to P21	J	General-purpose I/O port. The pull-up resistor ON/OFF can be set by setting the register. This function becomes valid at single-chip mode. High current output port (In the evaluation product, I/O circuit type is different from that in the flash memory product.)
	PPGF(E), PPGD(C), PPGB(A)		Output pin for PPG
36	P20	J	General-purpose I/O port. The pull-up resistor ON/OFF can be set by setting the register. This function becomes valid at single-chip mode. High current output port (In the evaluation product, I/O circuit type is different from that in the flash memory product.)
37	P85	K	General-purpose I/O port
	SIN1		Serial data input pin for UART1
38	P87	F	General-purpose I/O port
	SCK1		Clock I/O pin for UART1
39	P86	F	General-purpose I/O port
	SOT1		Serial data output pin for UART1
40	P43	F	General-purpose I/O port
	TX1		TX output pin for CAN1 interface
41	P42	F	General-purpose I/O port
	RX1		RX input pin for CAN1 interface
	INT9R		External interrupt request input pin for INT9R

Table 1-3. Pin Description (Sheet 3 of 3)

Pin number	Pin name	Circuit type	Functional description
42	P83	F	General-purpose I/O port
	SOT0		Serial data output pin for UART0
	TOT2		Output pin for reload timer 2
43	P84	F	General-purpose I/O port
	SCK0		Clock I/O pin for UART0
	INT15R		External interrupt request input pin for INT15R
44	P82	K	General-purpose I/O port
	SIN0		Serial data input pin for UART0
	INT14R		External interrupt request input pin for INT14R
	TIN2		Event input pin for reload timer 2
45	P44	F	General-purpose I/O port (I/O circuit type of P44 is different from that of Evaluation product.)
	FRCK0		Free-run timer 0 clock pin
46, 47	P40, P41	F	General-purpose I/O port
48	AV _{SS}	I	V _{SS} power input pin for analog circuit

1.6 I/O Circuit

Table 1-4. lists the I/O Circuit of each pin in MB90990 series.

■ I/O Circuit

Table 1-4. I/O Circuit Types (Sheet 1 of 5)

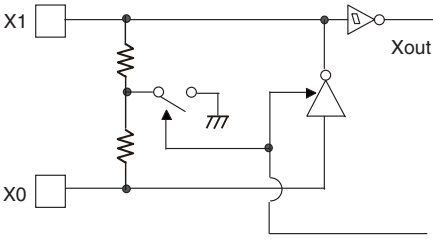
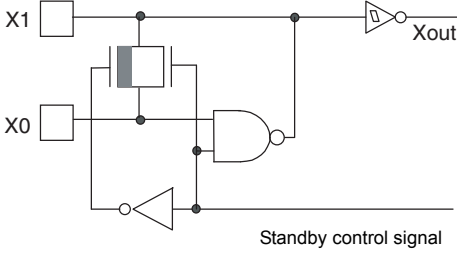
Type	Circuit	Remarks
A		Oscillation circuit High-speed oscillation feedback resistor = approx. 1 MW (Flash memory product)
		Oscillation circuit High-speed oscillation feedback resistor = approx. 1 MW (Evaluation product)

Table 1-4. I/O Circuit Types (Sheet 2 of 5)

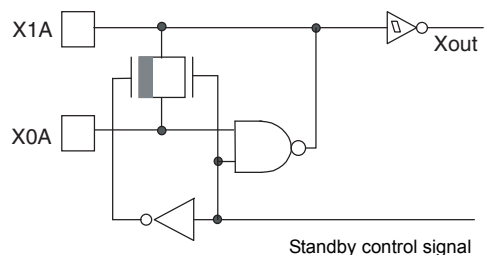

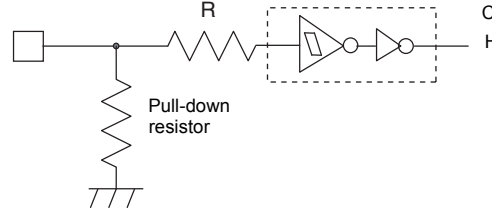
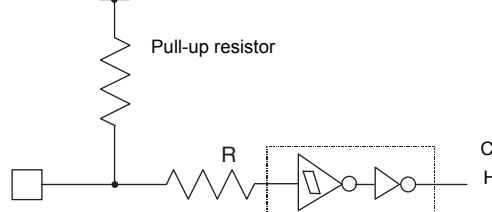
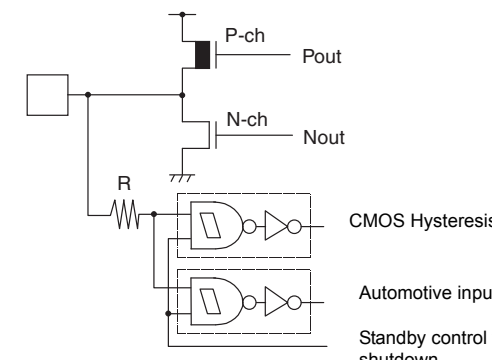
Type	Circuit	Remarks
B	 <p>Standby control signal</p>	<p>Oscillation circuit</p> <p>Low-speed oscillation feedback resistor = approx. 10 MW</p>
C	 <p>CMOS Hysteresis input</p>	<p>Evaluation product: CMOS hysteresis input</p> <p>Flash memory product: CMOS input</p>
D	 <p>CMOS Hysteresis input</p> <p>Pull-down resistor</p>	<p>Evaluation product: CMOS hysteresis input</p> <p>Pull-down resistor value: approx. 50 kW</p> <p>Flash memory product: CMOS input</p> <p>No pull-down</p>
E	 <p>CMOS Hysteresis input</p> <p>Pull-up resistor</p>	<p>CMOS hysteresis input</p> <p>Pull-up resistor value: approx. 50 kW</p>
F	 <p>CMOS Hysteresis input</p> <p>Automotive input</p> <p>Standby control for input shutdown</p> <p>P-ch</p> <p>N-ch</p> <p>Pout</p> <p>Nout</p>	<p>CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$)</p> <p>CMOS hysteresis inputs ($V_{IH} 0.8V_{CC}$ $V_{IL} 0.2V_{CC}$) (with the standby-time input shutdown function)</p> <p>Automotive input (with the standby-time input shutdown function)</p>

Table 1-4. I/O Circuit Types (Sheet 3 of 5)

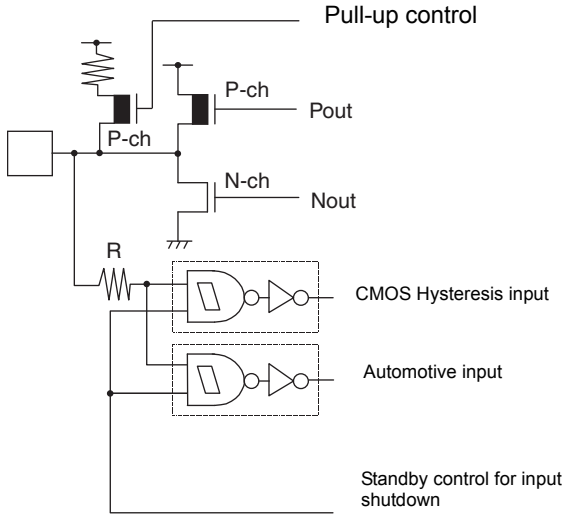
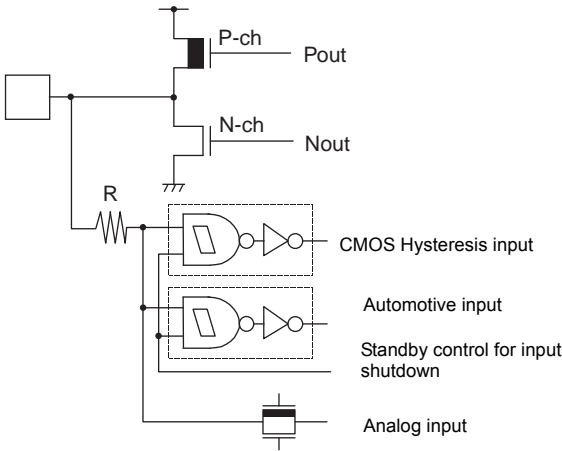
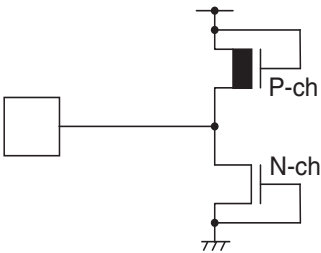
Type	Circuit	Remarks
G		<p>CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$)</p> <p>CMOS hysteresis inputs ($V_{IH}0.8V_{CC}$ $V_{IL}0.2V_{CC}$) (with the standby-time input shutdown function)</p> <p>Automotive input (with the standby-time input shutdown function)</p> <p>Programmable pull-up resistor: approx. 50 kW</p>
H		<p>CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$)</p> <p>CMOS hysteresis inputs ($V_{IH}0.8V_{CC}$ $V_{IL}0.2V_{CC}$) (with the standby-time input shutdown function)</p> <p>Automotive input (with the standby-time input shutdown function)</p> <p>A/D analog input</p>
I		<p>Power supply input protection circuit</p>

Table 1-4. I/O Circuit Types (Sheet 4 of 5)

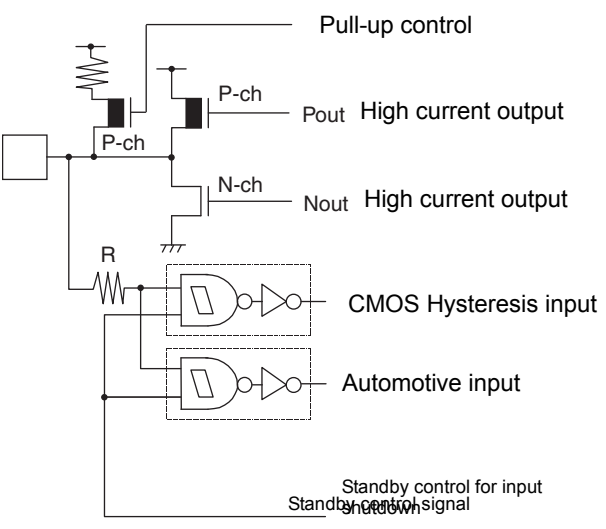
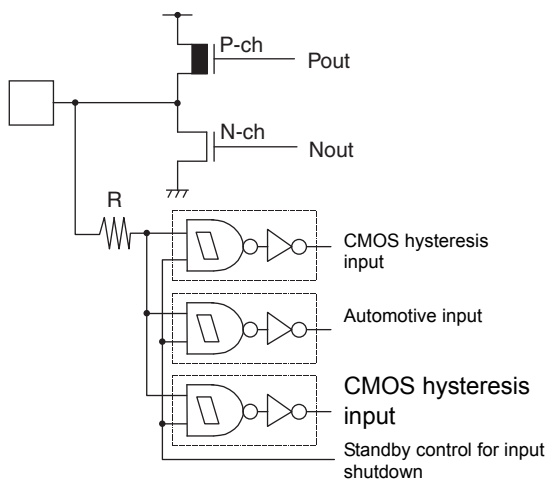
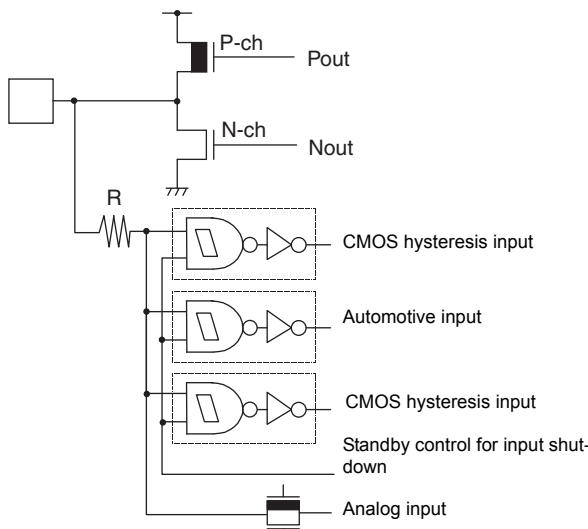
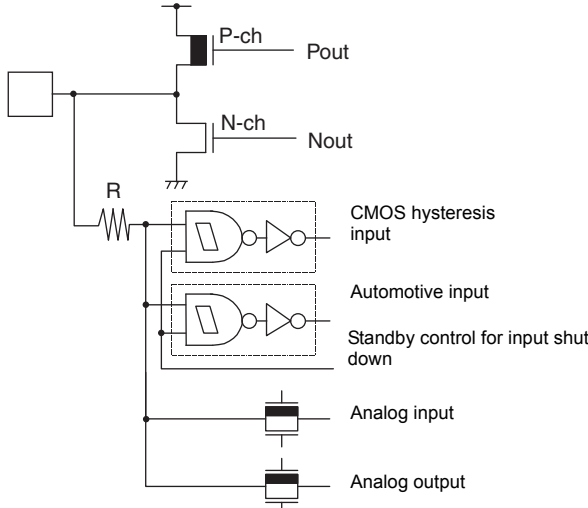
Type	Circuit	Remarks
J	 <p>Pull-up control</p> <p>P-ch Pout High current output</p> <p>N-ch Nout High current output</p> <p>R</p> <p>CMOS Hysteresis input</p> <p>Automotive input</p> <p>Standby control for input shutdown</p> <p>Standby signal</p>	<p>CMOS level output $(I_{OL} = 20 \text{ mA}, I_{OH} = -14 \text{ mA})$ (Evaluation product: $I_{OL} = 4 \text{ mA}, I_{OH} = -4 \text{ mA})$</p> <p>CMOS hysteresis inputs $(V_{IH} 0.8V_{CC} \text{ } V_{IL} 0.2V_{CC})$ (with the standby-time input shutdown function)</p> <p>Automotive inputs (with the standby-time input shutdown function)</p> <p>Programmable pull-up resistor: approx. 50 kΩ</p>
K	 <p>P-ch Pout</p> <p>N-ch Nout</p> <p>R</p> <p>CMOS hysteresis input</p> <p>Automotive input</p> <p>CMOS hysteresis input</p> <p>Standby control for input shutdown</p>	<p>CMOS level output $(I_{OL} = 4 \text{ mA}, I_{OH} = -4 \text{ mA})$</p> <p>CMOS hysteresis inputs $(V_{IH} 0.7V_{CC} \text{ } V_{IL} 0.3V_{CC})$ (with the standby-time input shutdown function)</p> <p>Automotive inputs (with the standby-time input shutdown function)</p> <p>CMOS hysteresis inputs $(V_{IH} 0.8V_{CC} \text{ } V_{IL} 0.2V_{CC})$ (with the standby-time input shutdown function)</p>

Table 1-4. I/O Circuit Types (Sheet 5 of 5)

Type	Circuit	Remarks
L		<p>CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$)</p> <p>CMOS hysteresis inputs ($V_{IH}0.8V_{CC}$ $V_{IL}0.2V_{CC}$) (with the standby-time input shutdown function)</p> <p>Automotive inputs (with the standby-time input shutdown function)</p> <p>CMOS hysteresis inputs ($V_{IH}0.7V_{CC}$ $V_{IL}0.3V_{CC}$) (with the standby-time input shutdown function)</p> <p>A/D analog input</p>
M		<p>CMOS level output ($I_{OL} = 4 \text{ mA}$, $I_{OH} = -4 \text{ mA}$)</p> <p>CMOS hysteresis inputs ($V_{IH}0.8V_{CC}$ $V_{IL}0.2V_{CC}$) (with the standby-time input shutdown function)</p> <p>Automotive inputs (with the standby-time input shutdown function)</p> <p>A/D analog input D/A analog output</p>

1.7 Notes on Handling Device

This section explains notes on handling the MB90990 series.

■ Notes on 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.

When used, note that maximum rated voltage is not exceeded.

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

❑ **Notes on Using external clock**

The high-speed oscillator pins (X0, X1) are not available for external clock inputs.

❑ **Notes during operation of PLL clock mode**

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

□ Power supply pins (V_{CC}/V_{SS})

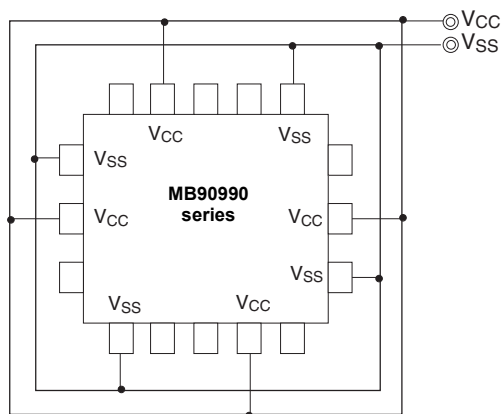
If there are multiple V_{CC} and V_{SS} pins, from the point of view of device design, pins to be of the same potential are connected the inside of the device to prevent such malfunctioning as latch up.

To reduce unnecessary radiation, prevent malfunctioning of the strobe signal due to the rise of ground level, and keep the total output current standard, be sure to connect the V_{CC} and V_{SS} pins to the power supply and ground externally (see Figure 1-6.).

Connect V_{CC} and V_{SS} to the device from the power supply source with lowest possible impedance.

It is recommended to 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 1-6. Power Supply Pins (V_{CC}/V_{SS})



□ Pull-up/down resistors

The MB90990 series does not support internal pull-up/down resistors (except Port2: programmable pull-up resistors).

Use pull-up/down handling where needed.

□ Crystal Oscillator Circuit

Noises around X0 or X1 pin 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. Please ask the crystal maker to evaluate the oscillational characteristics of the crystal and this device.

□ 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} , AVR) and analog inputs (AN0 to AN15) after turning-on the digital power supply (V_{CC}).

Turn-off the digital power supply after turning off the A/D converter power supply and analog inputs. In this case, make sure that the voltage does not exceed AVR or AV_{CC} .

□ Connection of Unused Pins of A/D Converter

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

□ Notes on Energization

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

❑ Stabilization of power supply voltage

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

❑ Note using CAN Function

When using CAN, the DIRECT bit of the CAN direct mode register (CDMR) must be set to "1" (See Table 1.7-1). If the DIRECT bit is not set correctly, the device does not operate normally.

Table 1-5. Setting of CAN Direct Mode

	CAN direct mode setting (setting of CDMR:DIRECT bit)
Required setting	1: Enable CAN direct mode
Setting disabled	0: Disable CAN direct mode (initial value)

Note:

For details on the CAN direct mode, see section "[21.12 CAN Direct Mode Register](#)".

□ Flash security Function

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 1-6](#). for the address of the security bit.

Table 1-6. Address of Security Bit

	Flash memory size	Address of security bit
MB90F997MBS, MB90F997JBS	Embedded 1Mbit flash memory	FE0001 _H

□ For the diversion of MB90950 series software assets

In programming of the MB90990 series, keep the following points in mind for the diversion of MB90950 series software assets in particular.

Access to the registers which do not exist in the MB90990 series.

As for the registers and bits which exist in MB90950 series but not in the MB90990 series, do not access them or ensure that the initial value is set. Setting any other value than the initial value may cause an abnormal operation in emulation using MB90V950.

Setting of the external interrupt source select register (EISSR).

The MB90990 series is not equipped with the external interrupt request input, INT8R, INT9, INT10R, INT11R, INT12, INT13R, INT14 and INT15.

Enabling unequipped pins causes a false operation. First set the EISSR and then set each of the registers when DTP/ external interrupt is used.

□ Notes when developing software using MB90V950

MB90V950 does not contain the following functions.

Low-voltage detection reset circuit

CR Clock calibration (Clock calibration unit)

□ Notes for serial communication

At serial communication, incorrect data may be received by noise, etc. To prevent such incorrectly receiving, the board should be designed to reduce the noise. For occurring of incorrect-data receiving by noise, etc., execute the retransmission (Example: adding the finally checksum data, etc.).

2. CPU



This chapter explains the CPU.

2.1 Overview of the CPU

2.2 Memory Space

2.3 Memory 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

2.1 Overview 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.

■ Overview 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. Up to 16M bytes 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-8L 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: 31.25 ns (at 4-MHz oscillation, 8 times clock multiplication)
- ❑ Maximum memory space: 16M bytes, accessed in linear or bank mode
- ❑ Instruction set optimized for controller applications

Rich data types: Bit, byte, word, long word

Extended addressing modes: 23 types

High-precision operation (32-bit length) based on 32-bit accumulator

- ❑ Powerful interrupt functions

Eight priority levels (programmable)

- ❑ CPU-independent automatic transfer

Up to 16 channels of the extended intelligent I/O service

- ❑ Instruction set compatible with high-level language (C)/multitasking

System stack pointer/instruction set symmetry/barrel-shift instructions

- Improved execution speed: 4-byte queue

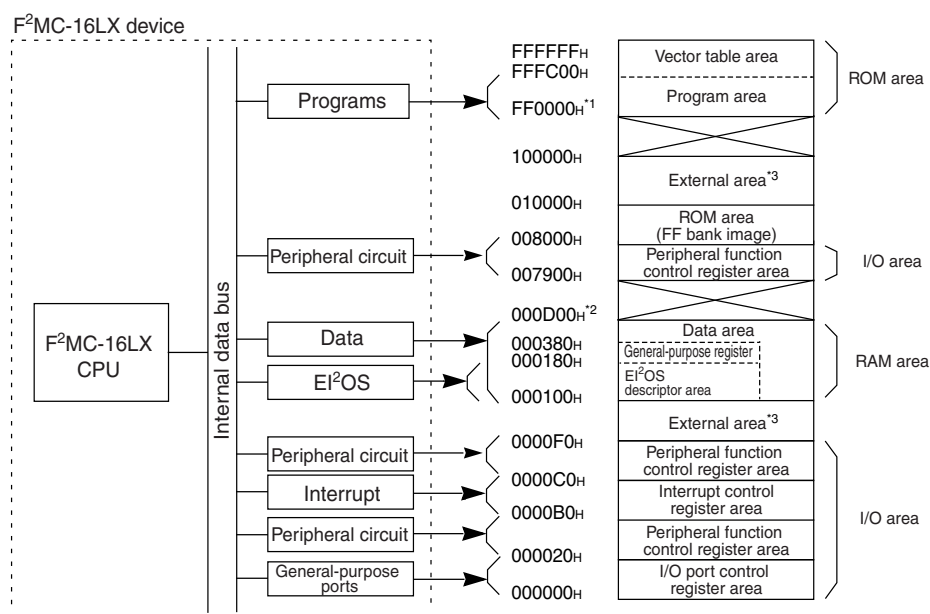
2.2 Memory Space

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

■ Overview of CPU Memory Space

Figure 2-1. shows a sample relationship between the F²MC-16LX system and memory map.

Figure 2-1. Sample Relationship Between F²MC-16LX System and Memory Map



- *1: The size of the built-in ROM differs for each model.
- *2: The size of the built-in RAM differs for each model.
- *3: Access is not possible in single-chip mode.

■ ROM Area

- Vector table area (address: FFFC00_H to FFFFFFF_H)

This area is used as a vector table for reset/interrupt and CALLV vector.

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: FE0000_H to FFBFF_H (for 128K bytes))

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 0020FF_H (for 8K bytes))

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 internal 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 2 addressing modes:

- Linear addressing

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

- Bank addressing

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

2.3 Memory Map

The memory map of the MB90990 series is shown in Figure 2-2. .

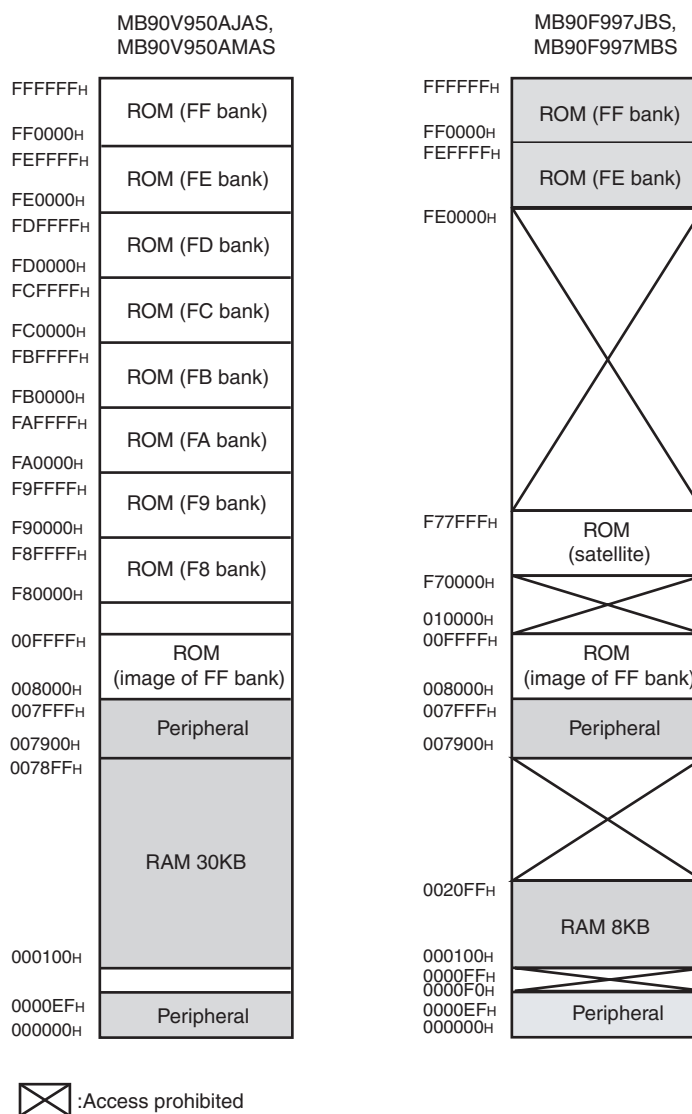
■ Memory Map

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 referred 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. The ROM area in bank FF cannot display its entire image in bank 00.

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

Figure 2-2. Memory Map



2.4 Linear Addressing

There are 2 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 2-3. shows an example of 24-bit operand specification. Figure 2-4. shows an example of 32-bit register indirect specification.

Figure 2-3. Example of Linear Method (24-bit Operand Specification)

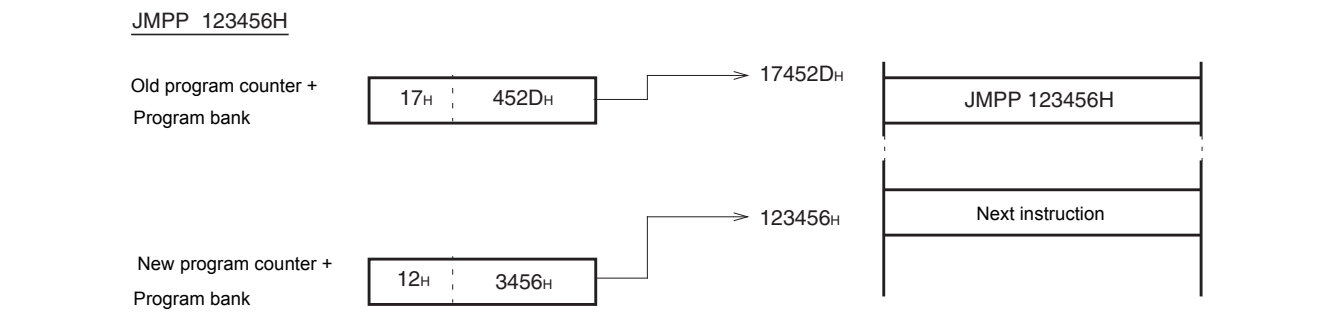
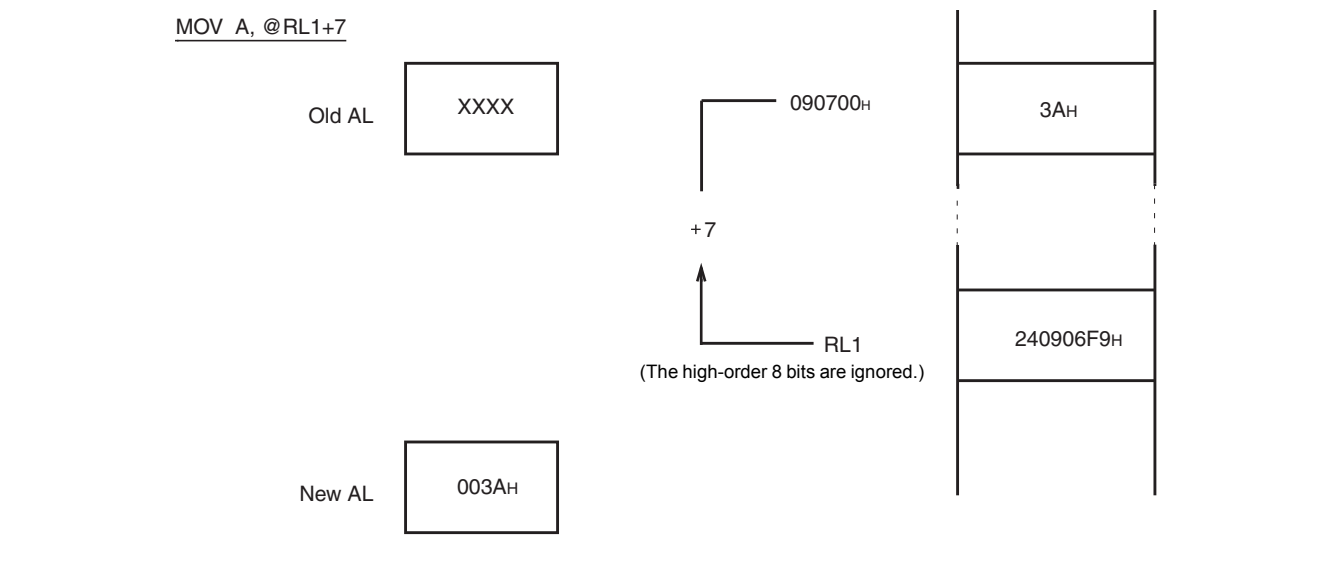


Figure 2-4. Example of Linear Method (32-bit Register Indirect Specification)



2.5 Bank Addressing Types

In the bank method, the 16M bytes space is divided into 256 for 64K bytes banks. The following 5 bank registers are used to specify the banks corresponding to each space:

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

■ Bank Addressing Types

- Program counter bank register (PCB)

The 64K bytes 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 64K bytes 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 64K bytes bank specified by the USB or SSB 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 data bank register (ADB)

The 64K bytes 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 2-1. 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.

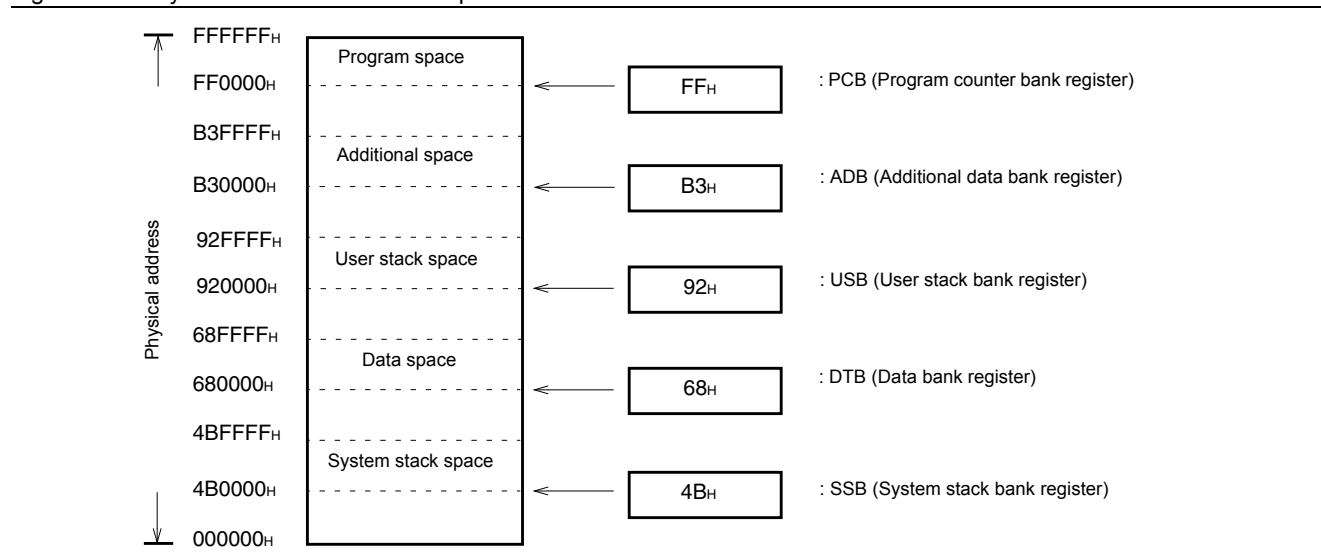
After reset, the DTB, USB, SSB, and ADB are initialized to "00_H". The PCB is initialized to a value specified by the reset vector. After reset, the DT, SP, and AD spaces are allocated in bank "00_H" (000000_H to 00FFFF_H), and the PC space is allocated in the bank specified by the reset vector.

Table 2-1. 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. is an example of a memory space divided into register banks.

Figure 2-5. Physical Addresses of Each Space



2.6 Multi-byte Data in Memory Space

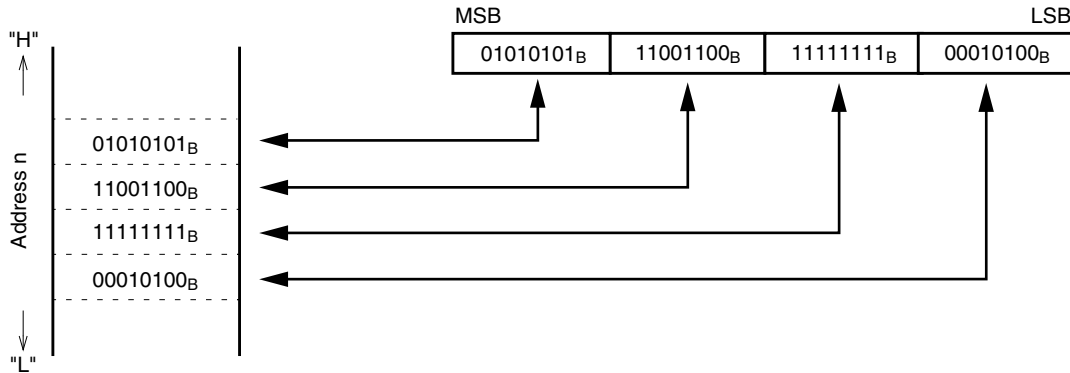
Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the low-order 16 bits are transferred before the high-order 16 bits.

If a reset signal is inputted 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. 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.

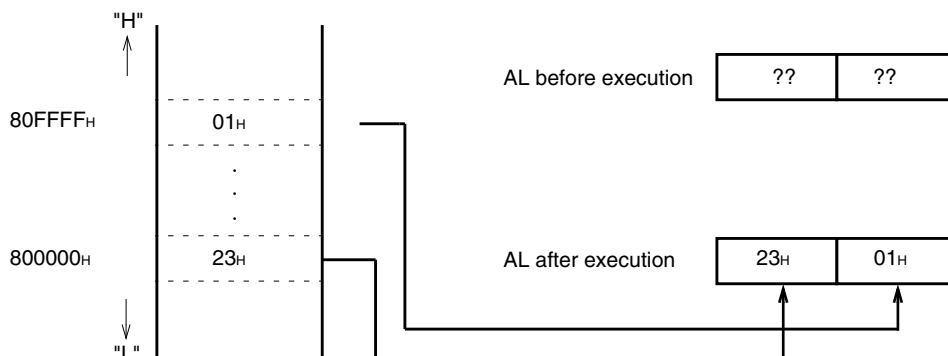
Figure 2-6. Sample Allocation of Multi-byte Data in Memory



■ 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 2-7. is an example of an instruction accessing multi-byte data.

Figure 2-7. Execution of MOVW A, FFFFH



2.7 Registers

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

■ Dedicated Registers

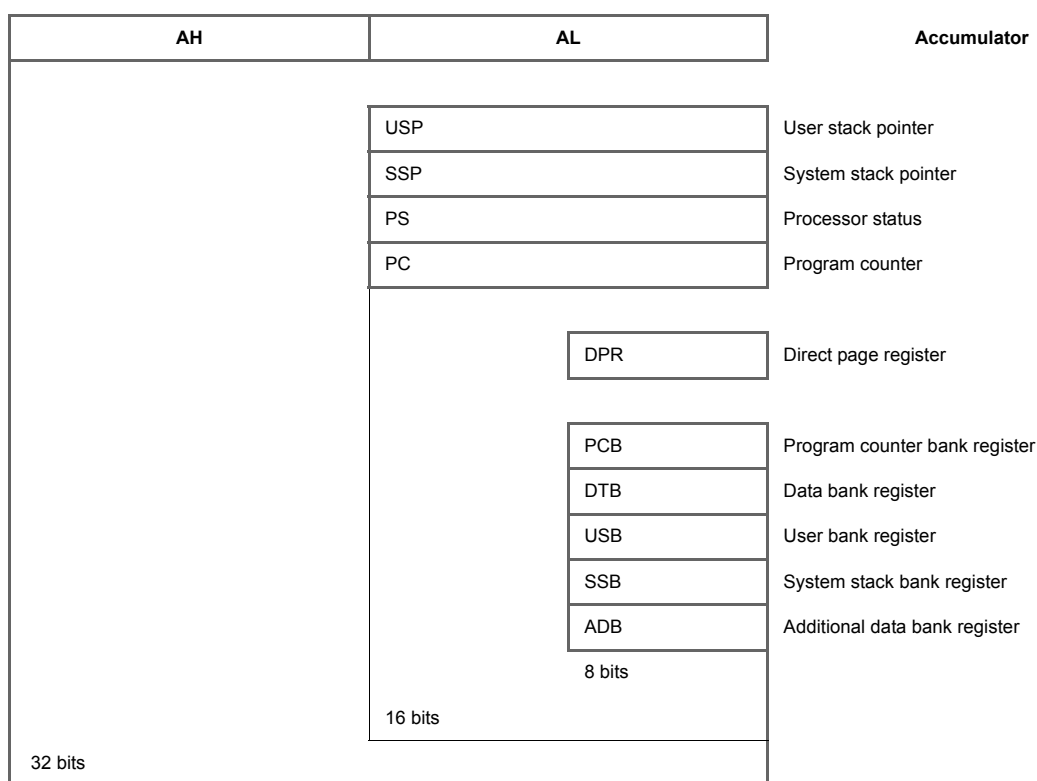
The F²MC-16LX CPU core has the following dedicated registers:

- 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 counter 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 data bank register (ADB) : 8-bit register indicating the AD space
- Direct page register (DPR) : 8-bit register indicating a direct page

Figure 2-8. is a diagram of the dedicated registers.

Figure 2-8. Dedicated Registers

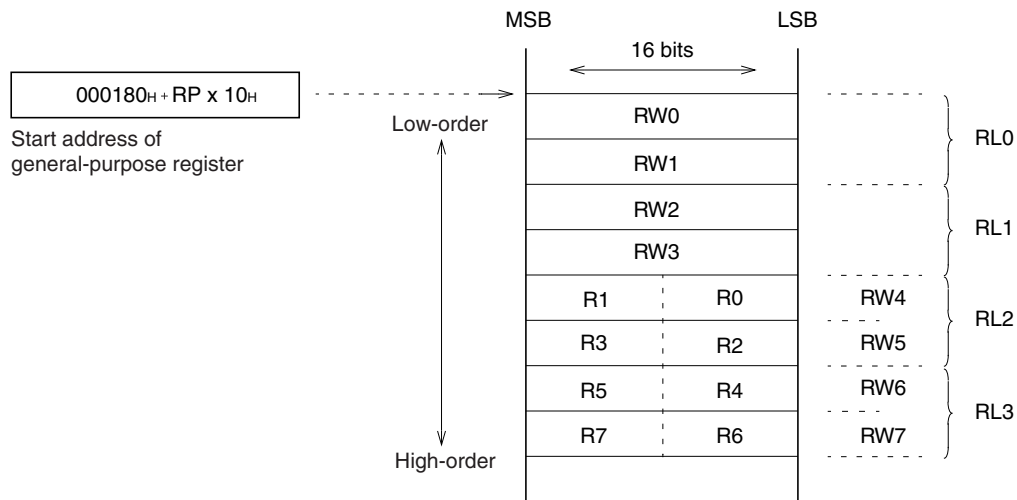


General-purpose Registers

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 2-9. .

- 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 2-9. General-purpose Registers



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

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

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

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

2.7.1 Accumulator (A)

The accumulator (A) register consists of 2 16-bit arithmetic operation registers (AH and AL), and is used as a temporary storage for operation results and transfer data.

■ Accumulator (A)

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 2-10. and Figure 2-11.). The data stored in the A register can be operated upon with the data in memory or registers (R_i, RW_i, or RL_i). In the same manner as with the F²MC-8, 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 reset. The register holds an undefined value immediately after reset.

Figure 2-10. 32-bit Data Transfer

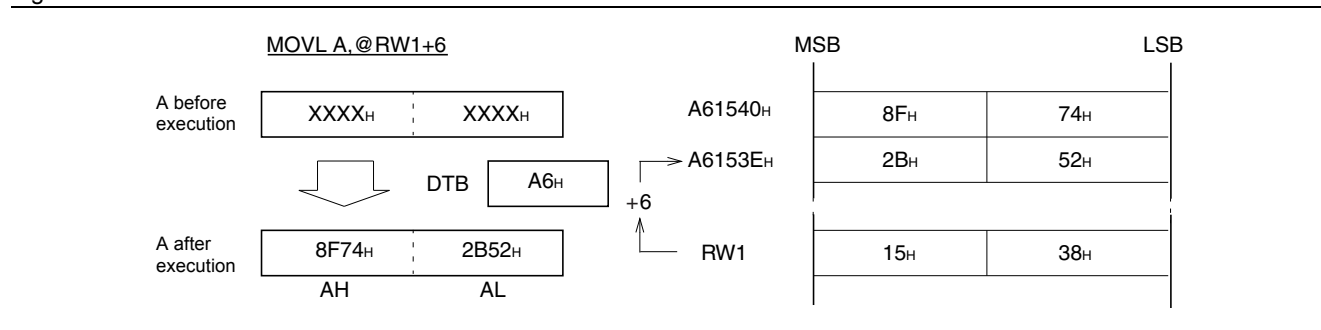
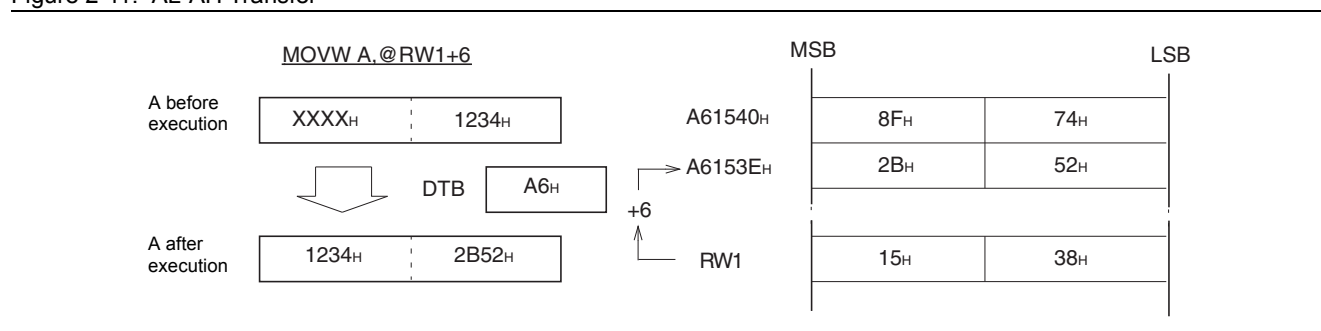


Figure 2-11. AL-AH Transfer



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

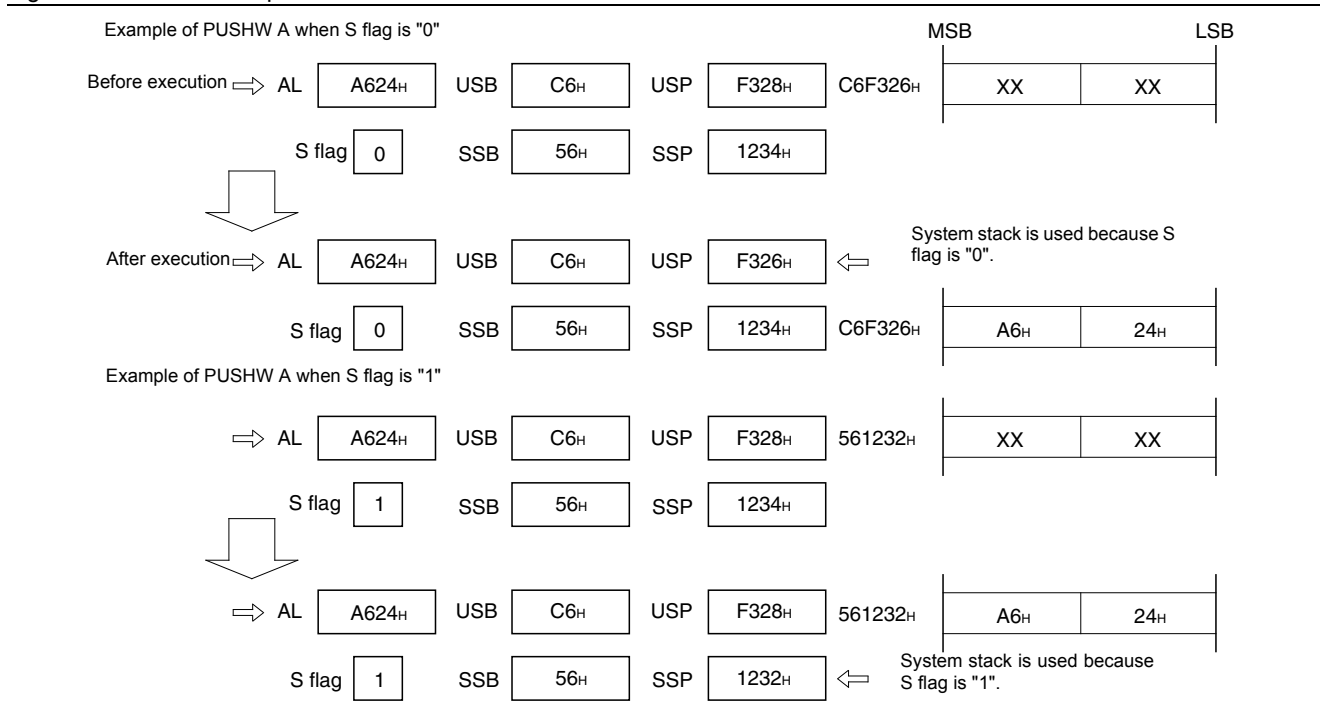
USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data when a push/pop instruction or subroutine is executed.

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

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 2-12](#)). 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 reset. Instead, they hold undefined values.

Figure 2-12. Stack Manipulation Instruction and Stack Pointer

**Note:**

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

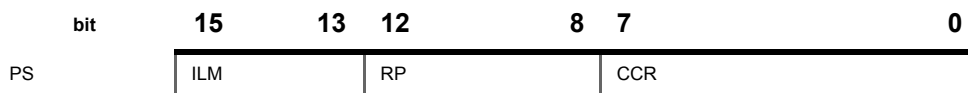
2.7.3 Processor Status (PS)

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 2-13](#), the high-order byte of the PS register consists of a register bank pointer (RP) and an interrupt level mask register (ILM). The ILM 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 2-13. Processor Status (PS) Structure



■ Condition Code Register (CCR)

[Figure 2-14](#) is a diagram of condition code register (CCR) configuration.

Figure 2-14. Condition Code Register (CCR) Configuration

bit	7	6	5	4	3	2	1	0	
	-	I	S	T	N	Z	V	C	: CCR
Initial value	-	0	1	*	*	*	*	*	

*: Undefined

□ I: Interrupt enable flag

Interrupt requests 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 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 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: $[000180_H + (RP) \times 10_H]$ (see Figure 2-15.). 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 "00037F_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 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 2-15. Register Bank Pointer (RP)

bit	12	11	10	9	8	
	B4	B3	B2	B1	B0	: RP
Initial value	0	0	0	0	0	

■ 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 2-2](#).). 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 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 2-16. Interrupt Level Mask Register (ILM)

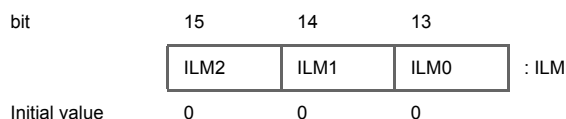


Table 2-2. Levels Indicated by the Interrupt Level Mask (ILM) Register

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

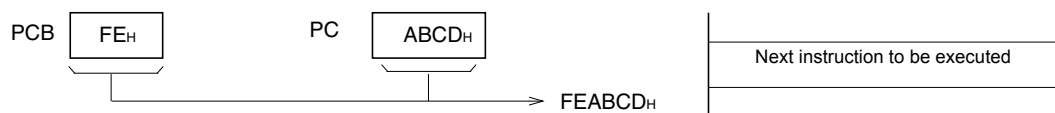
2.7.4 Program Counter (PC)

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 2-17. shows the program counter.

Figure 2-17. 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.

RL0 to RL3 are used as the linear pointer that directly accesses entire space.

■ Register Bank

Table 2-3. lists the functions of the registers. Table 2-4. indicates the relationship between the registers.

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

Table 2-3. Register Functions

R0 to R7	Used as operands of instructions. Note: R0 is used as a counter for barrel shift and 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.

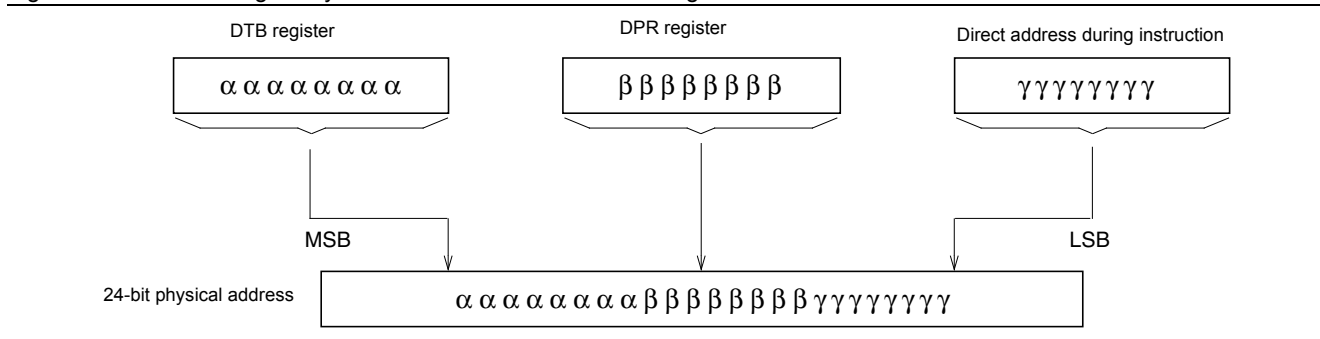
Table 2-4. Relationship between Registers

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

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

DPR specifies addr8 to addr15 of the instruction operands in direct addressing mode as shown in Figure 2-18. . DPR is eight bits long, and is initialized to "01_H" by reset.

Figure 2-18. 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 reset vector when reset occurs. 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 16M bytes 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 2-5. lists the bank select prefixes and the corresponding memory spaces.

Table 2-5. Bank Select Prefix

Bank select prefix	Selected space
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

MOV A, io/ MOV io, A/ MOVX A, io/ MOVW A, io/ MOVW io, A/ MOV io, #imm8

MOVW io, #imm16/ MOVB A, io:bp/ MOVB io:bp, A/ SETB io:bp/ CLRB io:bp

BBC io:bp, rel/ BBS io:bp, rel/ WBTC, WBTS

The I/O space of 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, the register accessed by that instruction can be changed to 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 that suppresses unnecessary flag change 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

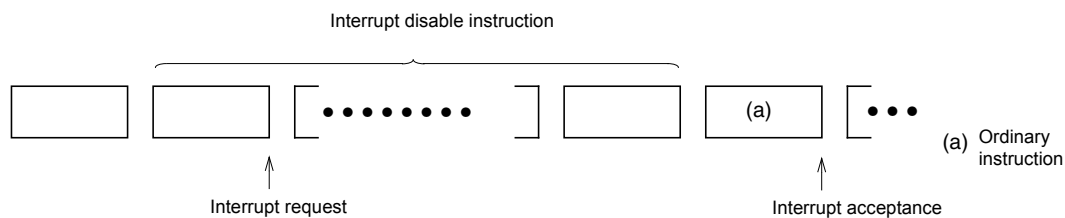
Interrupt requests are not sampled for the following ten instructions:

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

■ Interrupt Disable Instructions

If a valid hardware 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 2-19](#).

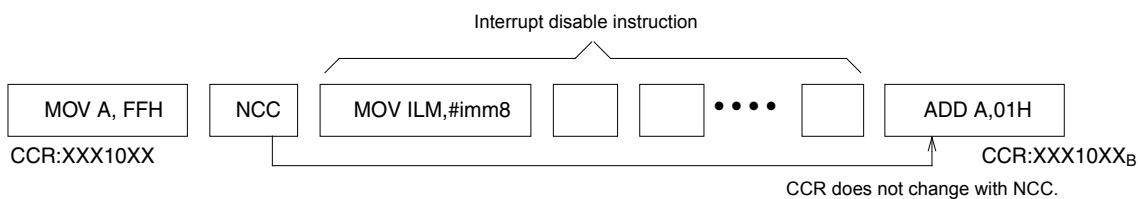
Figure 2-19. Interrupt Disable Instruction



■ 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 2-20](#).

Figure 2-20. Interrupt Disable Instructions and Prefix Codes

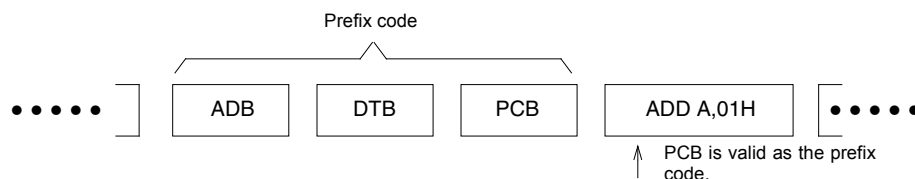


■ Consecutive Prefix Codes

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

In the figure below, competitive prefix codes are PCB, ADB, and DTB. For details, see [Figure 2-21](#).

Figure 2-21. Consecutive Prefix Codes



3. Interrupts



This chapter explains the function and operation of the interrupts and the extended intelligent I/O service (EI²OS).

3.1 Overview of Interrupts

3.2 Interrupt Vector

3.3 Interrupt Control Registers (ICR00 to ICR15)

3.4 Interrupt Flow

3.5 Hardware Interrupts

3.6 Software Interrupts

3.7 Extended Intelligent I/O Service (EI²OS)

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

3.9 Exceptions

3.1 Overview of Interrupts

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

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

■ Hardware Interrupts

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

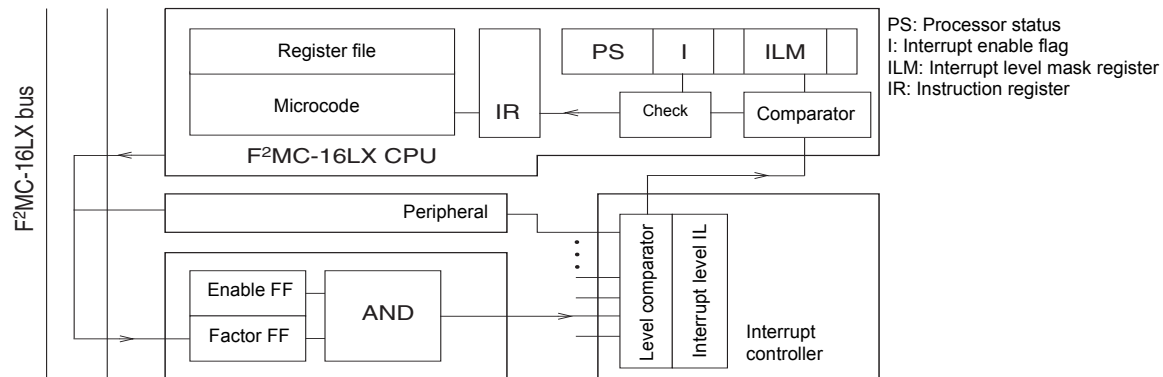
□ Specifying an interrupt level

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 (ILM0, ILM1, and ILM2). When an unmasked interrupt request occurs, the CPU saves 12 bytes of data that consists of registers PS, PC, PCB, DTB, ADB, DPR, and AH, AL in the memory area indicated by the SSB and SSP registers.

Figure 3-1. Overview of Hardware Interrupts



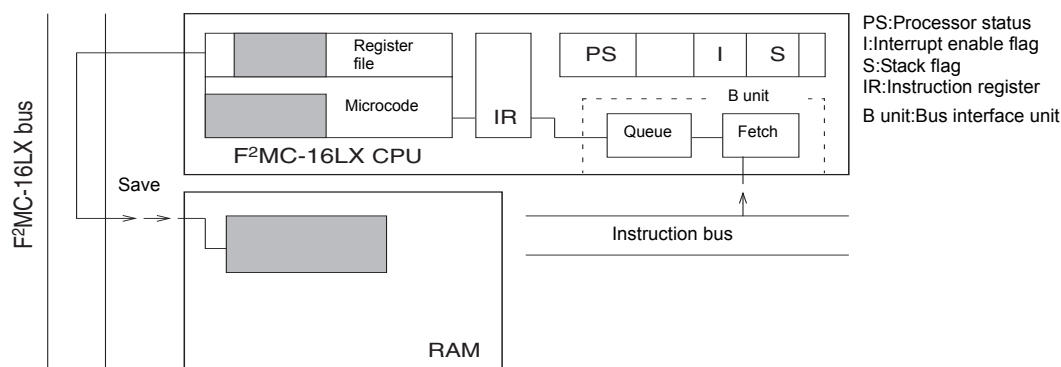
■ Software Interrupts

Software interrupt is the function which transfers control to the interrupt processing program defined by the user from CPU current program execution.

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 3-2. Overview of Software Interrupts

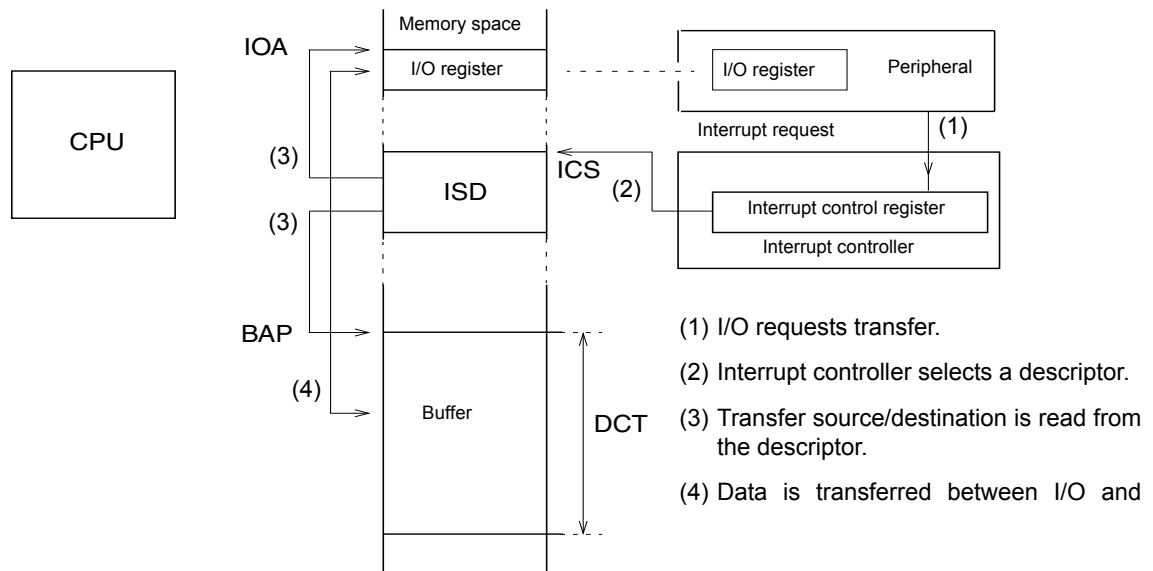


■ Extended Intelligent I/O Service (EI²OS)

The extended intelligent I/O service automatically transfers data between a built-in peripheral 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 peripheral, 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, clear the ISE flag to "0".

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


■ Exceptions processing

Exception processing is basically the same as interrupt processing. When an exception is detected between instructions, ordinary processing is suspended, and exception processing is performed. In general, exception processing occurs as a result of an unexpected operation. Therefore, use exception processing 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 "FFFFFF_H" as shown in Table 3-1.

■ Interrupt Vector

Table 3-1. Interrupt Vector (Sheet 1 of 2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address lower	Vector address middle	Vector address upper	Mode register
		Number	Address				
INT 0 *	—	—	—	FFFFC _H	FFFFD _H	FFFFE _H	Unused
INT 1 *	—	—	—	FFFF8 _H	FFFF9 _H	FFFFA _H	Unused
.	—	—	—
INT 7 *	—	—	—	FFFE0 _H	FFFE1 _H	FFFE2 _H	Unused
INT 8	Reset	—	—	FFFD0 _H	FFFD1 _H	FFFD2 _H	FFFD3 _H
INT 9	INT9 instruction	—	—	FFFD8 _H	FFFD9 _H	FFFDA _H	Unused
INT 10	Exception processing	—	—	FFFD4 _H	FFFD5 _H	FFFD6 _H	Unused
INT 11	Reserved	ICR00	000B0 _H	FFFD0 _H	FFFD1 _H	FFFD2 _H	Unused
INT 12	Reserved			FFFD4 _H	FFFD5 _H	FFFD6 _H	Unused

Table 3-1. Interrupt Vector (Sheet 2 of 2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address lower	Vector address middle	Vector address upper	Mode register
		Number	Address				
INT 13	CAN1 reception	ICR01	0000B1 _H	FFFFC8 _H	FFFFC9 _H	FFFFCA _H	Unused
INT 14	CAN1 transmission/node status			FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Unused
INT 15	Reserved	ICR02	0000B2 _H	FFFFC0 _H	FFFFC1 _H	FFFFC2 _H	Unused
INT 16	Clock Calibration Unit			FFFFBC _H	FFFFBD _H	FFFFBE _H	Unused
INT 17	Reserved	ICR03	0000B3 _H	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Unused
INT 18	Reserved			FFFFB4 _H	FFFFB5 _H	FFFFB6 _H	Unused
INT 19	16-bit reload timer 2	ICR04	0000B4 _H	FFFFB0 _H	FFFFB1 _H	FFFFB2 _H	Unused
INT 20	16-bit reload timer 3			FFFFAC _H	FFFFAD _H	FFFFAE _H	Unused
INT 21	Reserved	ICR05	0000B5 _H	FFFFA8 _H	FFFFA9 _H	FFFFAA _H	Unused
INT 22	Reserved			FFFFA4 _H	FFFFA5 _H	FFFFA6 _H	Unused
INT 23	PPG C/D	ICR06	0000B6 _H	FFFFA0 _H	FFFFA1 _H	FFFFA2 _H	Unused
INT 24	PPG 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 8 to 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 12 to 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			FFFF84 _H	FFFF85 _H	FFFF86 _H	Unused
INT 31	Reserved	ICR10	0000BA _H	FFFF80 _H	FFFF81 _H	FFFF82 _H	Unused
INT 32	Reserved			FFFF7C _H	FFFF7D _H	FFFF7E _H	Unused
INT 33	Input capture 0 to 3	ICR11	0000BB _H	FFFF78 _H	FFFF79 _H	FFFF7A _H	Unused
INT 34	Reserved			FFFF74 _H	FFFF75 _H	FFFF76 _H	Unused
INT 35	UART 0 reception	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	Unused
INT 36	UART 0 transmission			FFFF6C _H	FFFF6D _H	FFFF6E _H	Unused
INT 37	UART 1 reception	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	Unused
INT 38	UART 1 transmission			FFFF64 _H	FFFF65 _H	FFFF66 _H	Unused
INT 39	UART 2 reception	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	Unused
INT 40	UART 2 transmission			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 generation module			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 overlaps that for INT #vct8 (#0 to #7). Care must be taken when using the CALLV instruction.

3.3 Interrupt Control Registers (ICR00 to ICR15)

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 3 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 interrupt control registers by using a read-modify-write instruction, as doing so causes a misoperation.

■ Interrupt Control Registers (ICR00 to ICR15)

Figure 3-4. is a diagram of the bit configuration of an interrupt control registers.

Figure 3-4. Interrupt Control Registers (ICR)

bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	Interrupt control register
	ICS3	ICS2	ICS1 or S1	ICS0 or S0	ISE	IL2	IL1	IL0	Initial value 00000111 _B
	W	W	*	*	R/W	R/W	R/W	R/W	

*: "1" is always read. ICS1 and ICS0 are only for writing, and S1 and S0 are only for reading.

Supplement:

The extended intelligent I/O service channel select bits (ICR:ICS3 to ICS0) are valid for write only. The extended intelligent I/O service status bits (ICR:S1, S0) are valid for read only. In read operation, "1" is read from bit6, bit7/bit14, and bit15 (ICS2, ICS3).

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.

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

These bits are readable and writable and specify the interrupt level of the corresponding built-in peripheral. Upon a reset, these bits are initialized to level 7 (no interrupt). Table 3-2. describes the relationship between the interrupt level setting bits and interrupt levels.

Table 3-2. Interrupt Level Setting Bits and Interrupt Levels

IL2	IL1	IL0	Level
0	0	0	0 (Highest interrupt)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (Lowest interrupt)
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 (When completed by count and by a request from the built-in peripheral), the ISE bit is cleared to zero. If the corresponding peripheral does not have the EI²OS function, the ISE bit must be set to "0" on the software side.

Upon reset, the ISE bit is initialized to "0".

[bit 15 to bit 12, bit 7 to bit 4] ICS3 to ICS0 (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 determine the extended intelligent I/O service descriptor addresses in memory, which is explained later. The ICS3 to ICS0 bits are initialized to "0000_B" by reset.

Table 3-3. describes the correspondence between the ICS bits, channel numbers, and descriptor addresses.

Table 3-3. ICS Bits, Channel Numbers, and Descriptor Address

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

[bit13, bit12, bit5, bit4] S1 and S0 (extended intelligent I/O service status bits)

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

Table 3-4. shows the relationship between the S bits and the end conditions.

Table 3-4. S Bits and End Conditions

S1	S0	End conditions
0	0	EI ² OS running or not activated
0	1	Stop status by count end
1	0	Reserved
1	1	Stop status by request from built-in peripheral

3.4 Interrupt Flow

Figure 3-5. shows the interrupt flow.

■ Interrupt Flow

Figure 3-5. Interrupt Flow

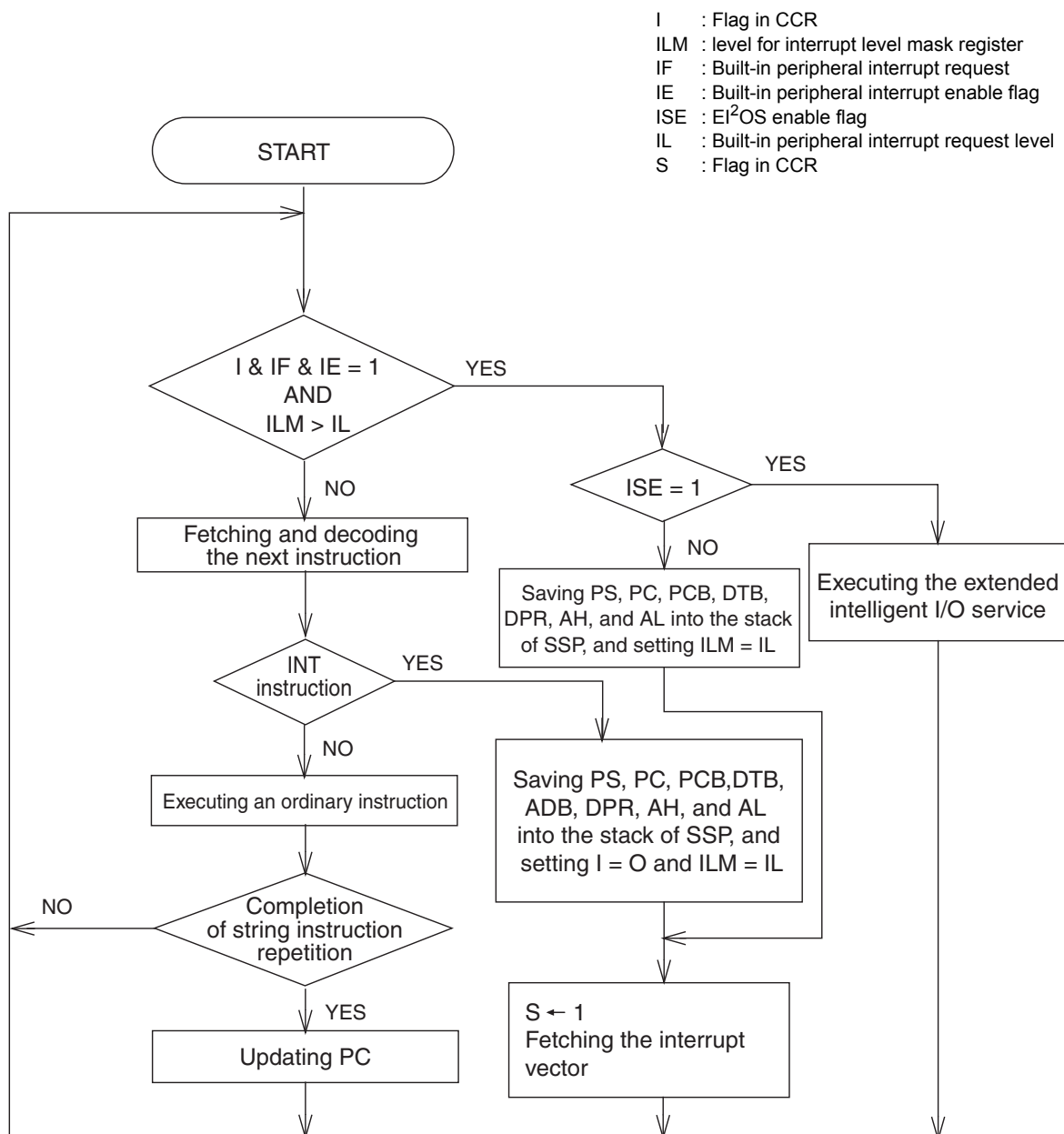
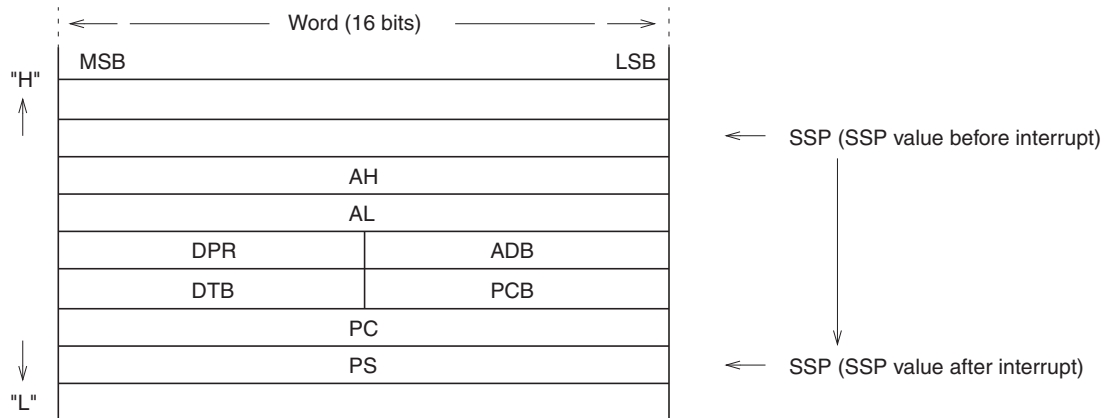


Figure 3-6. Register Saving during Interrupt Processing



3.5 Hardware Interrupts

In response to an interrupt request signal from a built-in peripheral, 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 3 sections:

- Built-in peripheral

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

- 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 peripheral control registers for built-in peripheral, 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 referred during interrupt processing is assigned to addresses "FFFC00_H" to "FFFFFF_H" in memory. These addresses are shared with software interrupts.

Table C-2 in "[APPENDIX C List of MB90990 Series Interrupt Vectors](#)" shows the assignment of the MB90990 series.

3.5.1 Hardware Interrupt Operation

A built-in peripheral 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 that is unique to the built-in peripheral occurs. When the interrupt enable flag indicates "enable", the peripheral 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 (IL) and the ILM in the PS register. If the interrupt level (IL) 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 refers 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.

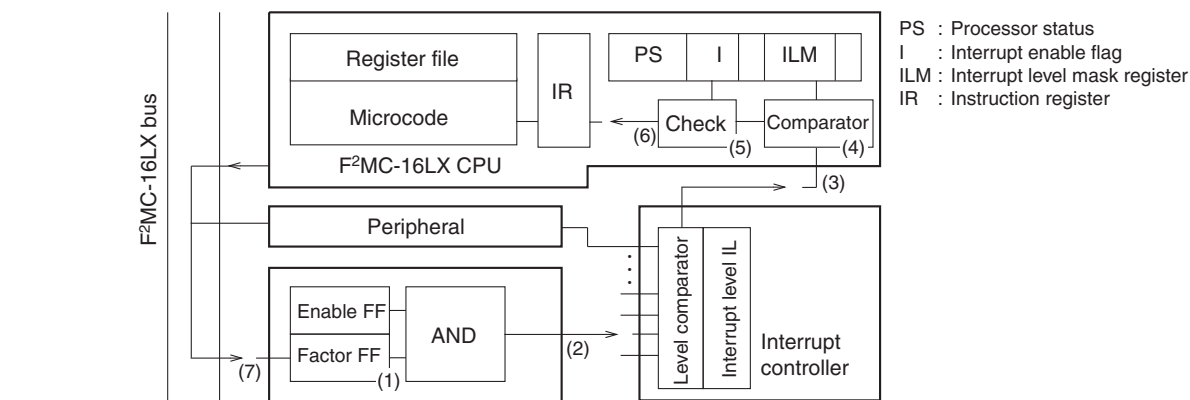
The interrupt processing body saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, AH and AL) to the memory area indicated by SSB and SSP, fetches 3 bytes of interrupt vector, loads them onto PC and PCB, updates the ILM of PS to a level value of the received interrupt request, sets the S flag to "1", then performs branch processing. As a result, the interrupt processing program defined by the user is executed next.

3.5.2 Occurrence and Release of Hardware Interrupt

Figure 3-7. 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 3-7. Occurrence and Release of Hardware Interrupt



- (1) An interrupt cause occurs in a peripheral.
- (2) The interrupt enable bit in the peripheral is referred. 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.

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

See [Table 3-5.](#) for the cycle count compensation value.

Interrupt start: $24 + 6 \times (\text{Cycle count compensation value})$

Interrupt return: $15 + 6 \times (\text{Cycle count compensation value})$ (RETI instruction)

Table 3-5. Compensation Values for Interrupt Processing Cycle Count

Address indicated by 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 peripheral is being updated. If an interrupt occurs during interrupt processing, a higher-level interrupt is processed first.

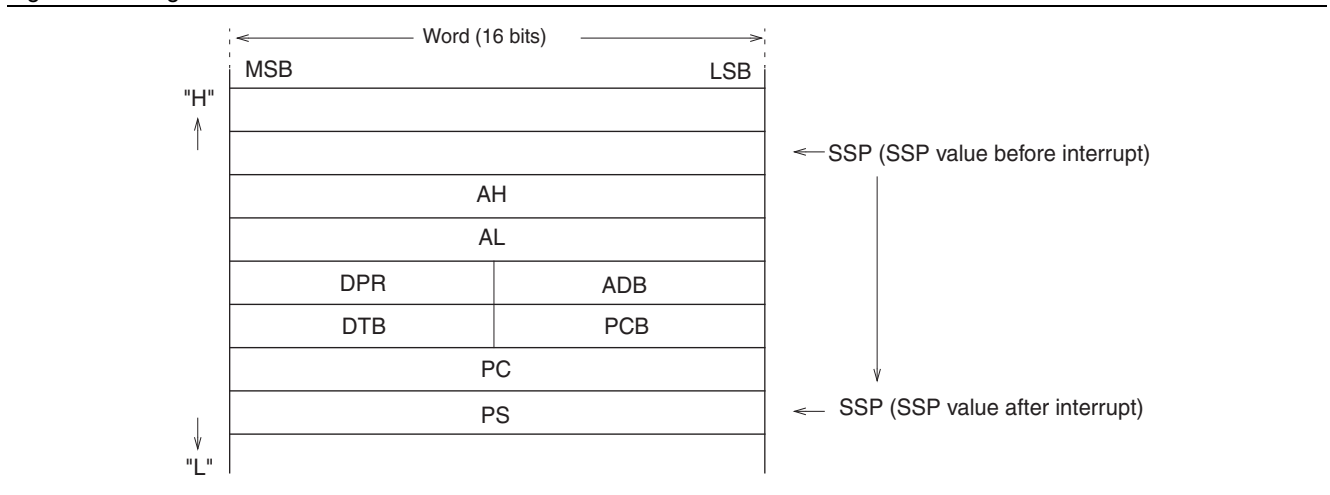
■ Multiple Interrupts

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 3-8.](#) shows the order of the registers saved in the stack.

Figure 3-8. 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.
- The interrupt enable flag (I) of the CCR register in PS is set to "1". 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 "0" 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 C-1 in [APPENDIX C](#) lists the interrupt vectors of the MB90990 series.

Software interrupts share the same interrupt vector area with hardware interrupts.

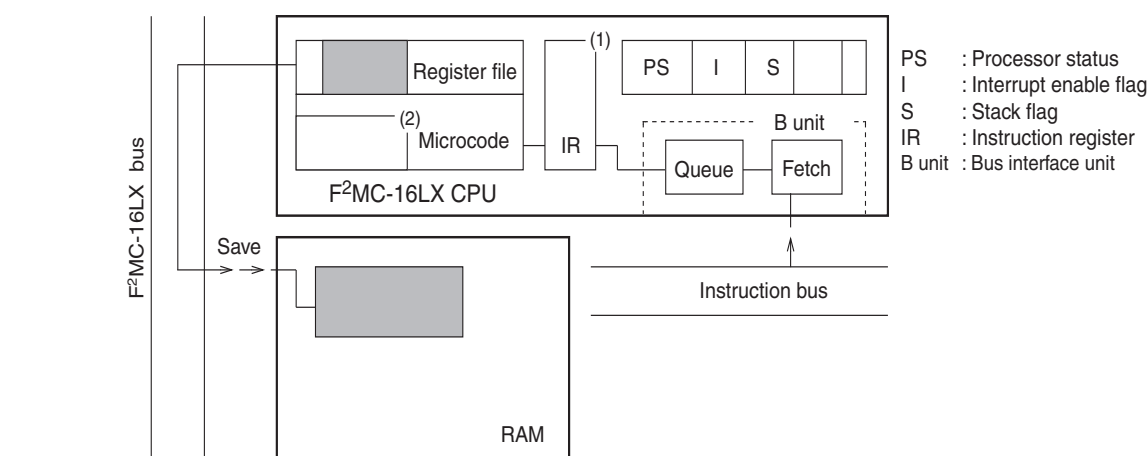
For example, interrupt request number INT 12 is used for external interrupt of a hardware interrupt as well as for INT #12 of a software interrupt. Therefore, external interrupt and INT #12 call the same interrupt processing routine.

■ 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, AH and AL) to the memory area indicated by SSB and SSP. The microcode then fetches 3 bytes of interrupt vector and loads them onto PC and PCB, resets the I flag to "0", and sets the S flag to "1". Then, the microcode performs branch processing. As a result, the interrupt processing program defined by the user application program is executed next.

[Figure 3-9.](#) illustrates the flow from the occurrence of a software interrupt until there is no interrupt request in the interrupt processing program.

Figure 3-9. Occurrence and Release of Software Interrupt



(1) The software interrupt instruction is executed.

- (2) Dedicated 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 counter bank register (PCB) is "FF_H", 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 INT #vct8 instruction.

Table C-2 in "APPENDIX C " shows the relationship of interrupt cause, interrupt vector, and interrupt control register in the MB90990 series.

3.7 Extended Intelligent I/O Service (EI²OS)

The EI²OS function, a kind of hardware interrupt operation, 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 (buffer address is updated).

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

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

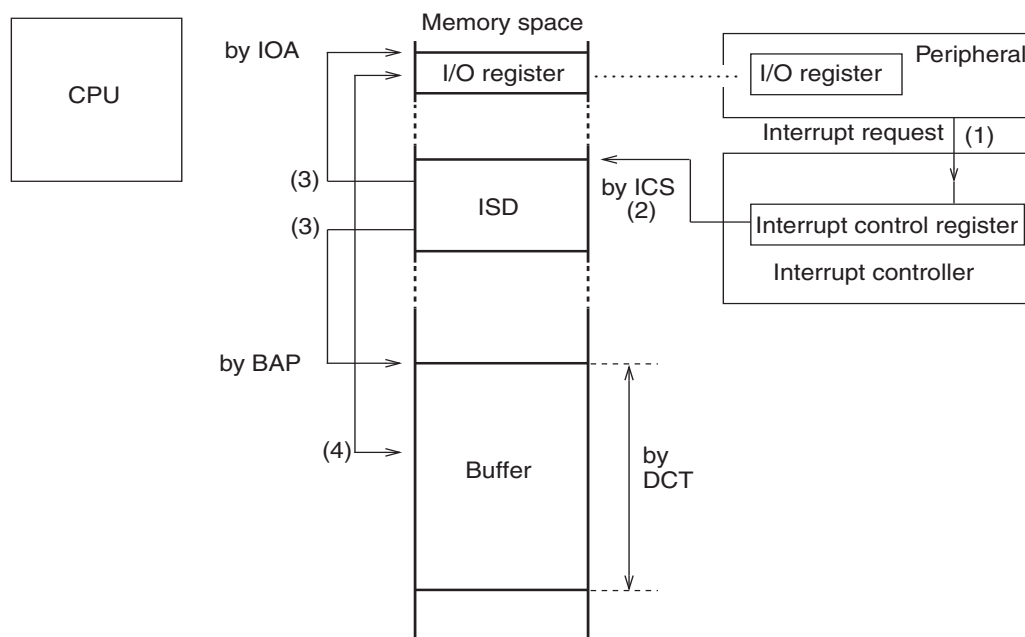
- Interrupt control register: Exists in the interrupt controller and indicates the ISD address.
- Extended intelligent I/O service descriptor (ISD): Exists in RAM and holds the transfer mode, I/O address, number of transfers, and buffer address.

Note:

When using REALOS, it is impossible to use the extended intelligent I/O service (EI²OS).

Figure 3-10. outlines the extended intelligent I/O service.

Figure 3-10. Outline of Extended Intelligent I/O Service



- (1) I/O requests transfer.
- (2) Interrupt controller selects descriptor.
- (3) Transfer source and destination are read from descriptor.
- (4) Data is transferred between I/O and memory.

Notes:

- 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 "FFFFFF_H".
- The maximum transfer count that can be specified by DCT is 65536.

■ Structure of Extended Intelligent I/O Service (EI²OS)

EI²OS is handled by the following 4 sections:

Built-in peripheral

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

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)

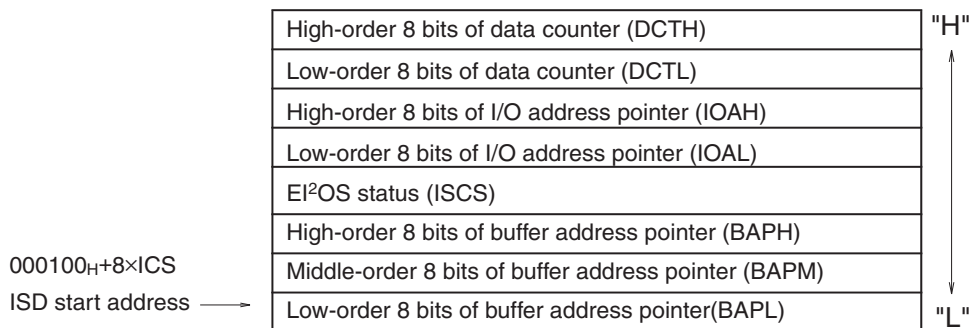
The extended intelligent I/O service descriptor exists between "000100_H" and "00017F_H" in internal RAM and consists of the following items:

- Data counter
- I/O address pointer
- Status data
- Buffer address pointer

■ Extended Intelligent I/O Service Descriptor (ISD)

Figure 3-11. shows the configuration of the extended intelligent I/O service descriptor.

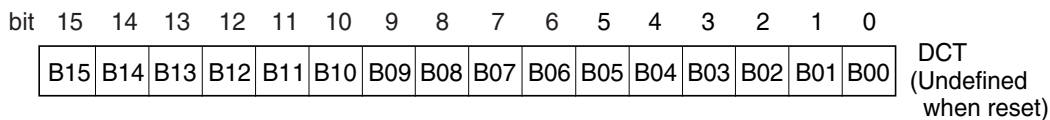
Figure 3-11. Extended Intelligent I/O Service Descriptor Configuration



■ 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 after data transfer. EI²OS is terminated when this counter reaches "0". Figure 3-12. is a diagram of the data counter configuration.

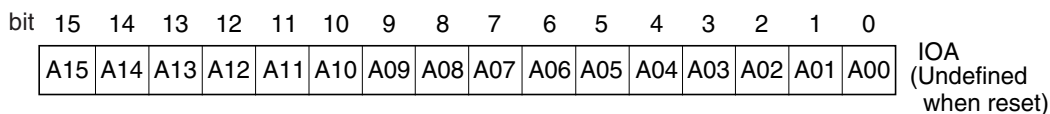
Figure 3-12. Data Counter Configuration



■ I/O Register Address Pointer (IOA)

This is a 16-bit register that indicates the low-order address (A15 to A00) of the buffer and I/O register used for data transfer. The high-order addresses (A23 to A16) are all zeroes, and any I/O between addresses "000000_H" and "00FFFF_H" can be specified. Figure 3-13. is a diagram of the IOA configuration.

Figure 3-13. 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 16M bytes space. If the BF bit of ISCS is cleared to "0" (update enabled), only the low-order 16 bits of BAP changes and BAPH does not change.

■ EI²OS Status Register (ISCS)

EI²OS status register (ISCS) is an 8-bit length register, indicating buffer address pointer, I/O register address pointer update/fix, transfer data length (bytes/words) and transfer direction.

Figure 3-14. is a diagram of the ISCS configuration.

Be sure to write "0" in bit 7 to bit 5 of ISCS.

Figure 3-14. ISCS Configuration

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

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 (increment) 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 (increment) 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 pointer are updated.

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

0: I/O address pointer --> Buffer address pointer

1: Buffer address pointer --> I/O address pointer

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

0: The extended intelligent I/O service is not terminated by a built-in peripheral request.

1: The extended intelligent I/O service is terminated by a built-in peripheral request.

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

Figure 3-15. is a diagram of the EI²OS operation flow. Figure 3-16. is a diagram of the EI²OS use procedure.

■ EI²OS Operation Flow

Figure 3-15. EI²OS Operation Flow

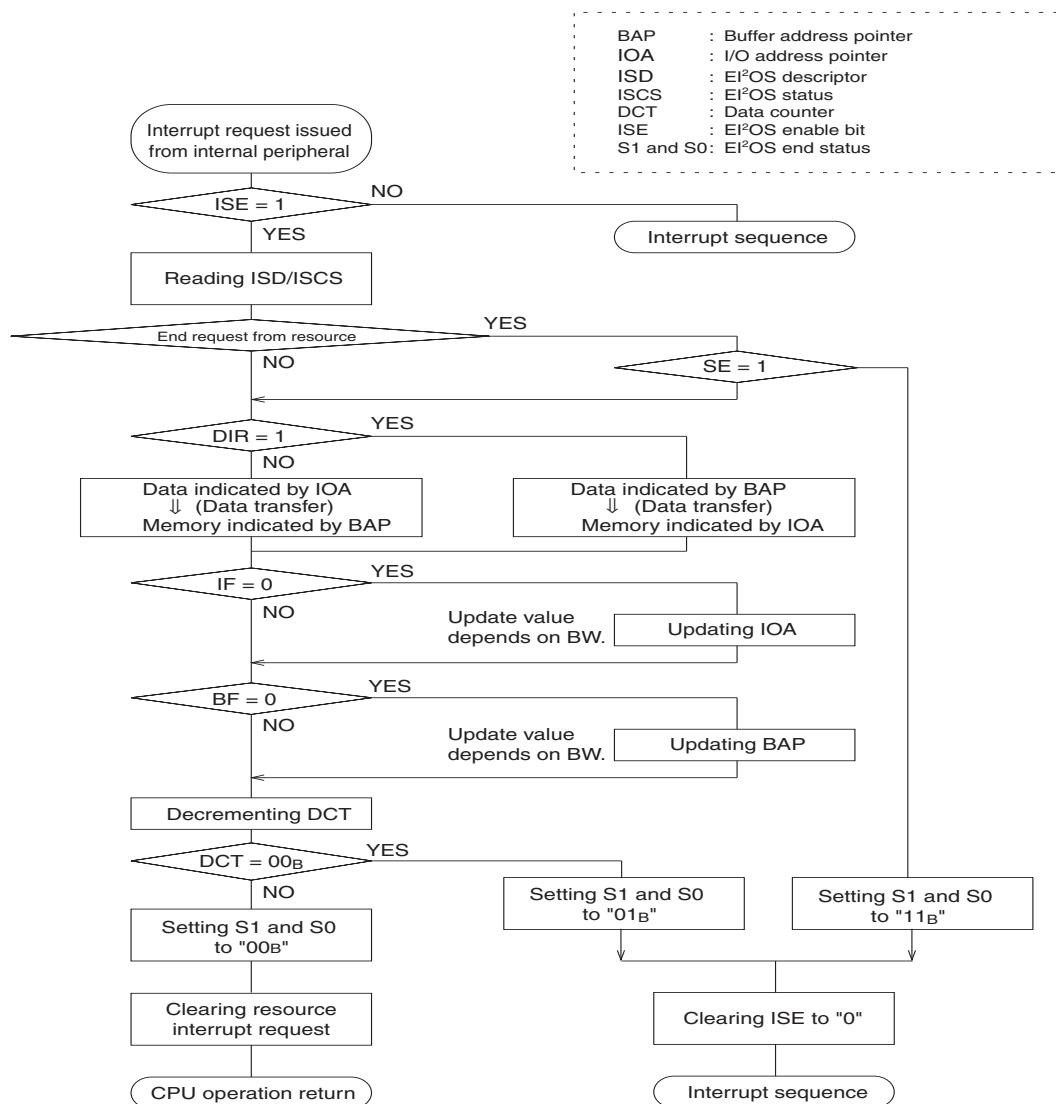
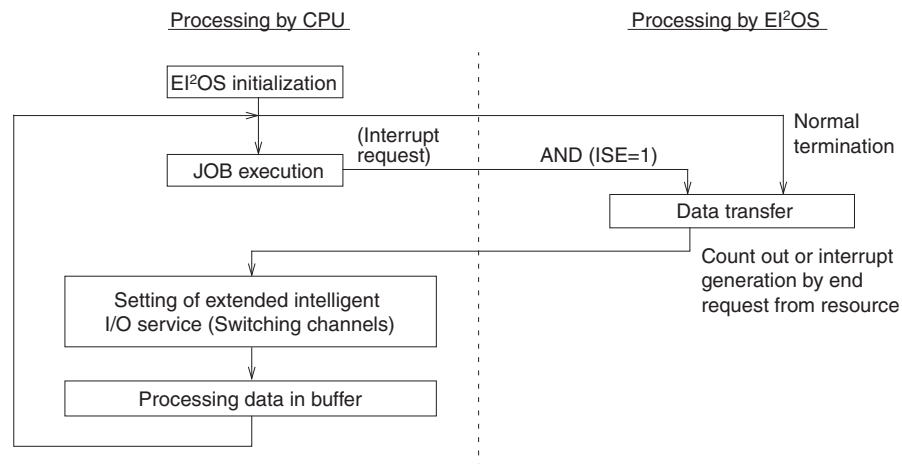


Figure 3-16. EI²OS Use Flow


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

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

(Table 3-6. + Table 3-7.) machine cycles

- When a stop request is issued from a resource

(36 + 6 × Table 3-8.) machine cycles

- When the counting is completed

(Table 3-6. + Table 3-7. + (21 + 6 × Table 3-8.) machine cycles

Table 3-6. Execution Time when the 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 3-7. Data Transfer Compensation Values for EI²OS Execution Time

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

Table 3-7. Data Transfer Compensation Values for EI²OS Execution Time

I/O address pointer	Internal access	
	B/E	O
B: Byte data transfer E: Even address word transfer O: Odd address word transfer		

Table 3-8. Compensation Value of Interrupt Handling Time

Address pointed to by the stack pointer	Compensation value [cycle]
External 8 bits	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

3.9 Exceptions

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

■ Execution of an Undefined Instruction

Exception processing is fundamentally the same as interrupt processing. When an exception is detected between instructions, exception processing is performed separately from ordinary processing. In general, exception processing is performed as a result of an unexpected operation. Cypress recommends using exception processing 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 INT10 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 to "0" and the S flag is set to "1". The PC value saved in the stack is the address at which the undefined instruction is stored. In the instruction code 2 bytes or more, the value will be the address at which the recognized code as an undefined instruction is stored. Processing can be restored by the RETI instruction, but is of no use, however, because the same exception occurs again.

4. Delayed Interrupt Generation Module



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

- [4.1 Overview of Delayed Interrupt Generation Module](#)
- [4.2 Block Diagram of Delayed Interrupt Generation Module](#)
- [4.3 Configuration of Delayed Interrupt Generation Module](#)
- [4.4 Explanation of Operation of Delayed Interrupt Generation Module](#)
- [4.5 Notes on Using Delayed Interrupt Generation Module](#)
- [4.6 Program Example of Delayed Interrupt Generation Module](#)

4.1 Overview of Delayed Interrupt Generation Module

The delayed interrupt generation module generates the interrupt for task switching.

The hardware interrupt request can be generated/canceled by software.

■ Overview of Delayed Interrupt Generation Module

By using the delayed interrupt generation module, a hardware interrupt request can be generated or canceled by software.

[Table 4-1.](#) shows the overview of the delayed interrupt generation module.

Table 4-1. Overview of Delayed Interrupt Generation Module

	Function and control
Interrupt factor	An interrupt request is generated by setting the R0 bit in the delayed interrupt request generate/cancel register to "1" (DIRR: R0 = 1). An interrupt request is canceled by setting the R0 bit in the delayed interrupt request generate/cancel register to "0" (DIRR: R0 = 0).
Interrupt number	#42 (2A _H)
Interrupt control	An interrupt is not enabled by the DIRR register.
Interrupt flag	The interrupt flag is held in the R0 bit in the DIRR register.
EI ² OS	The DIRR register does not correspond to the EI ² OS.

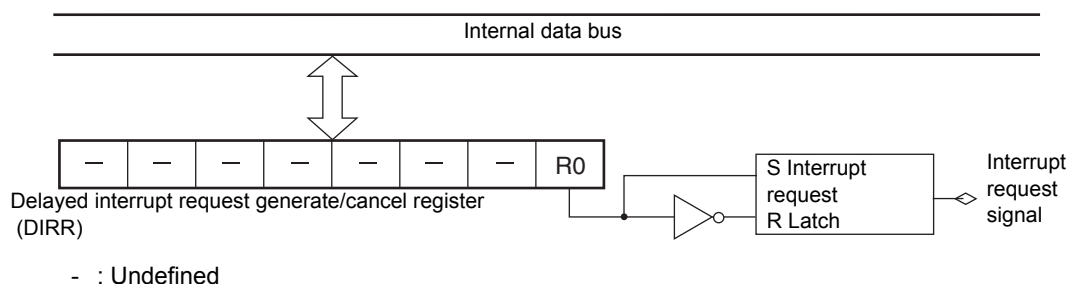
4.2 Block Diagram of Delayed Interrupt Generation Module

The delayed interrupt generation module consists of the following blocks:

- Interrupt request latch
- Delayed interrupt request generate/cancel register (DIRR)

■ Block Diagram of Delayed Interrupt Generation Module

Figure 4-1. Block Diagram of Delayed Interrupt Generation Module



□ Interrupt request latch

This latch keeps the settings (delayed interrupt request generation or cancellation) of the delayed interrupt request generate/cancel register (DIRR).

□ Delayed interrupt request generate/cancel register (DIRR)

This register generates or cancels a delayed interrupt request.

■ Interrupt Number

The interrupt number used in the delayed interrupt generation module is as follows:

Interrupt number #42(2A_H)

4.3 Configuration of Delayed Interrupt Generation Module

This section lists registers and initial values in the delayed interrupt generation module.

■ List of Registers and Initial Values

Figure 4-2. List of Registers and Initial Values in Delayed Interrupt Generation Module

Delayed interrupt request generate/cancel register (DIRR)	bit	15	14	13	12	11	10	9	8
Address: 00009F _H		1	1	1	1	1	1	1	0

4.3.1 Delayed Interrupt Request Generate/Cancel Register (DIRR)

The delayed interrupt request generate/cancel register (DIRR) generates or cancels a delayed interrupt request.

■ Delayed Interrupt Request Generate/Cancel Register (DIRR)

Figure 4-3. Delayed Interrupt Request Generate/Cancel Register (DIRR)

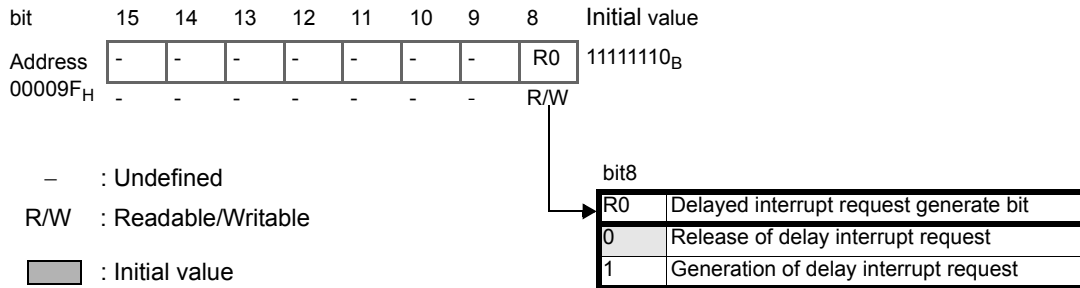


Table 4-2. Functions of Delayed Interrupt Request Generate/Cancel Register (DIRR)

Bit name		Function
bit15 to bit9	Undefined bits	Read: The value is undefined. Write: No effect
bit8	R0: Delayed interrupt request generate bit	This bit generates or cancels a delayed interrupt request. When set to "0": Cancels delayed interrupt request When set to "1": Generates delayed interrupt request

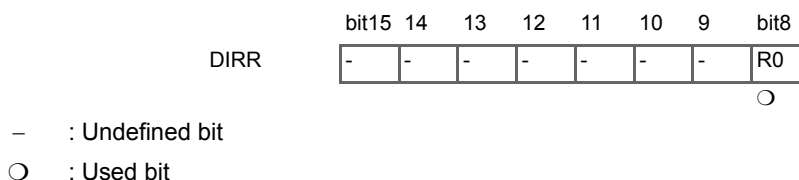
4.4 Explanation of Operation of Delayed Interrupt Generation Module

The delayed interrupt generation module has a function for generating or canceling an interrupt request by software.

■ Explanation of Operation of Delayed Interrupt Generation Module

Using the delayed interrupt generation module requires the setting shown in [Figure 4-4](#).

Figure 4-4. Setting for Delayed Interrupt Generation Module



When the R0 bit in the delayed interrupt request generate/cancel register (DIRR) is set to "1" (DIRR: R0 = 1), an interrupt request is generated. There is no interrupt request enable bit.

□ Operation of delayed interrupt generation module

When the R0 bit in the delayed interrupt request generate/cancel register (DIRR) is set to "1", the interrupt request latch is set to "1" and an interrupt request is generated to the interrupt controller.

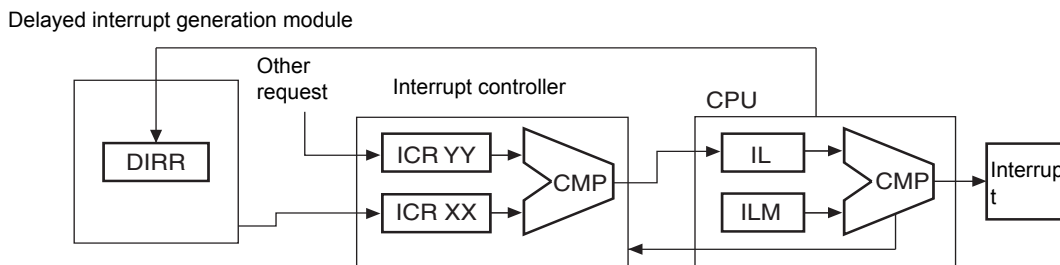
An interrupt request is generated to the CPU when the interrupt controller prioritizes the interrupt request over other requests.

When the interrupt level mask bit of the condition code register (CCR: ILM) is compared to the interrupt request level (ICR: IL), and the interrupt request level is higher than ILM, CPU executes the delayed interrupt processing after the instruction currently being executed is completed.

At interrupt processing, the user program sets the R0 bit to "0" to cancel the interrupt request and performs task switching.

Figure 4-5. shows the operation of the delayed interrupt generation module.

Figure 4-5. Operation of Delayed Interrupt Generation Module



4.5 Notes on Using Delayed Interrupt Generation Module

This section explains the notes on using the delayed interrupt generation module.

■ Notes on Using Delayed Interrupt Generation Module

- The interrupt processing is restarted at return from interrupt processing without setting the R0 bit in the delayed interrupt request generate/cancel register (DIRR) to "0" within the interrupt processing routine.
- Unlike software interrupts, interrupts in the delayed interrupt generation module are delayed.

4.6 Program Example of Delayed Interrupt Generation Module

This section gives a program example of the delayed interrupt generation module.

■ Program Example of Delayed Interrupt Generation Module

- Processing specification

The main program writes "1" to the R0 bit in the delayed interrupt request generate/cancel register (DIRR) to generate a delayed interrupt request and performs task switching.

□ Coding example

```
ICR15 EQU    0000BFH        ;Interrupt control register
DIRR    EQU    00009FH        ;Delayed interrupt factor generate/
                                cancel register
DIRR_R0 EQU    DIRR:0        ;Delay interrupt request generating bit
;-----Main program-----
CODE     CSEG
START:
    AND     CCR,#0BFH        ;Interrupt disabled
    MOV     I:ICR15,#00H     ;Interrupt level 0 (strong)
    MOV     ILM,#07H        ;Setting ILM in PS to level 7
    OR      CCR,#40H        ;Interrupt enabled
    SETB    I:DIRR_R0        ;Delay interrupt request generating
LOOP     MOV     A,#00H        ;No limit loop
         MOV     A,#01H
         BRA     LOOP
;-----Interrupt program-----
WARI:
    CLRB    I:DIRR_R0        ;Clear interrupt request flag
    :
```

```

;      User processing
;      :
;      RETI                      ;Recovery from interrupt
CODE   ENDS
;-----Vector setting-----
VECT   CSEG      ABS=0FFH
;      ORG        0FF54H          ;Setting vector to interrupt #42 (2AH)
;      DSL        WARI
;      ORG        0FFDCH          ;Reset vector setting
;      DSL        START
;      DB         00H            ;Setting to single-chip mode
VECT   ENDS
;      END          START

```


5. Clocks



This chapter explains the clocks used by MB90990 series microcontrollers.

5.1 Clocks

5.2 Block Diagram of the Clock Generation Block

5.3 Clock Selection Register (CKSCR)

5.4 PLL/Sub clock Control Register (PSCCR)

5.5 Clock Mode

5.6 Oscillation Stabilization Wait Time

5.7 Connection of an Oscillator

5.1 Clocks

The clock generation block controls the operation of the internal clock that controls operation of the CPU and peripheral functions. The clock generated by the clock generation block is called the machine clock. One cycle of machine clock is called one machine cycle. The clock to be supplied from a high-speed oscillator is called an oscillation clock, and the 2-frequency division of the oscillation clock is called a main clock. The 4- or 8-frequency division of the clock supplied from a CR oscillator is called a sub clock, and the clock by the PLL oscillation is called PLL clock.

■ Clocks

The clock generation block contains the oscillation circuit that generates the oscillation clock by connecting oscillator to oscillation pin. The clock generation block also contains the PLL clock multiplier circuit, which generates six clocks whose frequencies are multiplication of the oscillation clock frequency. The clock generation block controls the oscillation stabilization wait time 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 by connecting the oscillator to high-speed oscillator pins (X0, X1).

- Main clock (MCLK)

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

- Sub clock (SCLK)

The sub clock is the CR oscillation clock divided by 4 or 8. The division ratio of sub clock is determined by SCDS bit of PLL/sub clock control register (PSCCR). The sub clock can be used as operation clock of the watch timer or the low-speed machine clock.

- PLL clock (PCLK)

The PLL clock is obtained by multiplying the oscillation clock frequency with the PLL clock multiplier circuit (PLL oscillation circuit). One of six types of clocks can be selected by setting the multiplication ratio selection bits (CKSCR: CS1, CS0, PSCCR: CS2).

- Machine clock

The machine clock controls the operation of the CPU and peripheral functions. One cycle of machine clock is regarded as one machine cycle ($1/\phi$). An operating machine clock can be selected from among the main clock, sub clock, and six types of PLL clock.

Note:

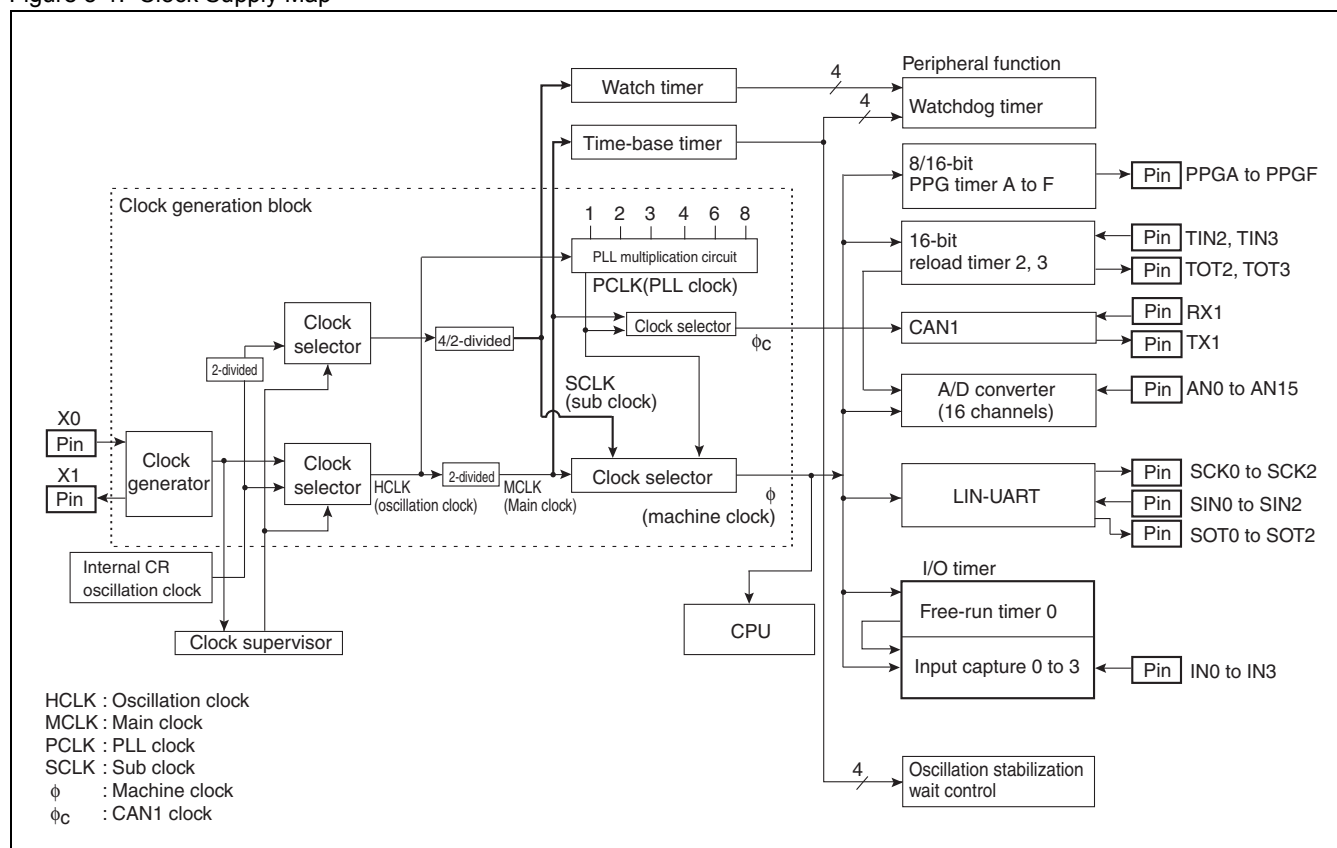
When the operating voltage is 5 V, the oscillation clock can be between 3 MHz and 16 MHz. The highest operating frequency for the CPU and peripheral resource is 32 MHz. However, normal operation is not guaranteed if a multiplication ratio resulting in a higher frequency than 32 MHz is specified.

The PLL oscillation operates at the range of 4 MHz to 32 MHz, but the PLL oscillation range differs by operation voltage and multiplication ratio. See the data sheet for the details.

■ 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 sub clock (clock mode) and by a change in the PLL clock multiplication ratio. Since some peripheral functions receive frequency-divided output from the time-base timer, a peripheral unit can select the clock best suited for this operation. [Figure 5-1.](#) shows the clock supply map.

Figure 5-1. Clock Supply Map



5.2 Block Diagram of the Clock Generation Block

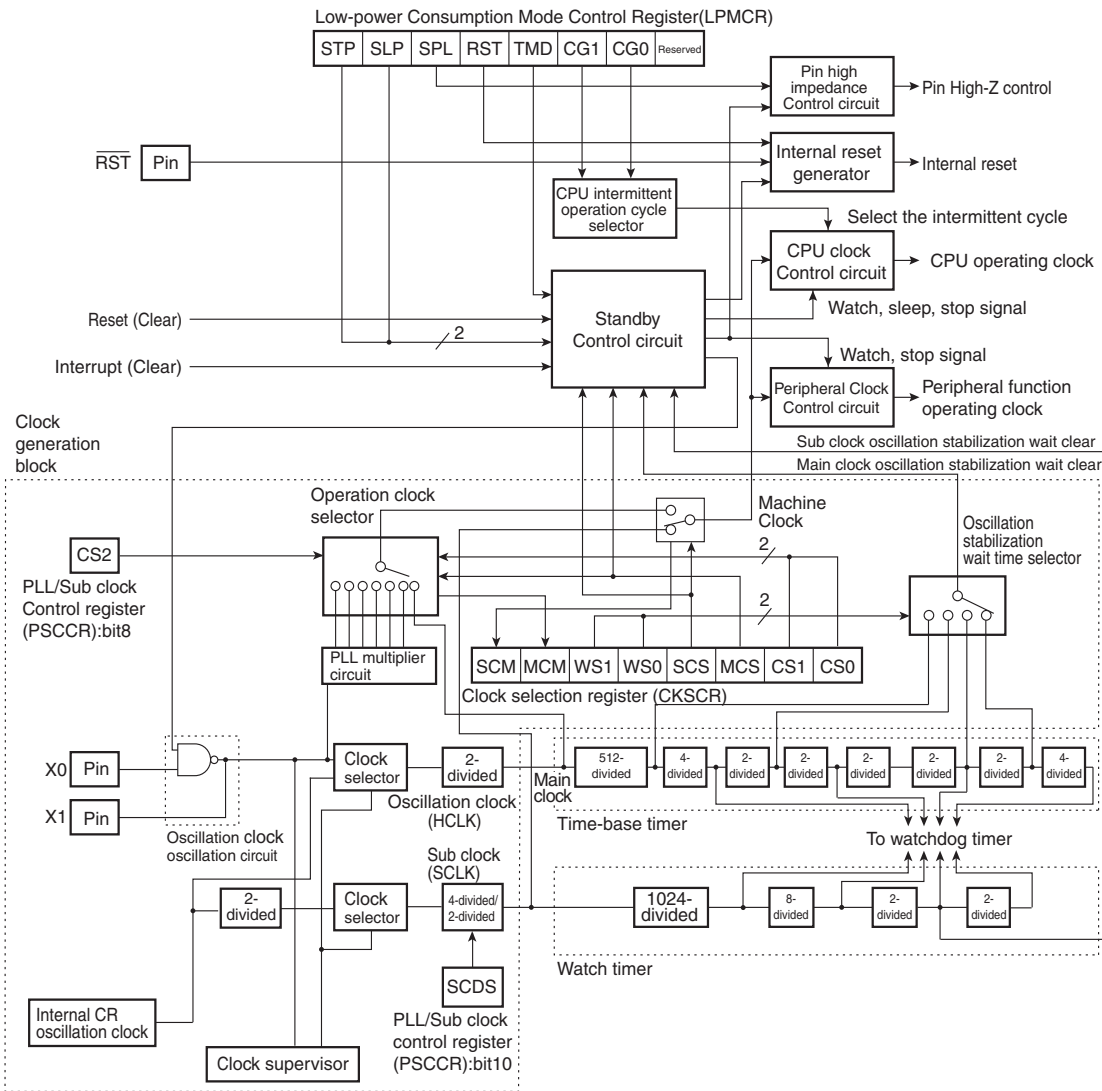
The clock generation block consists of six blocks:

- Oscillation clock generation circuit/sub clock generation circuit
- PLL multiplier circuit
- Clock selector
- Clock selection register (CKSCR)
- PLL/sub clock control register (PSCCR)
- Oscillation stabilization wait time selector

■ Block Diagram of the Clock Generation Block

Figure 5-2. shows a block diagram of the clock generation block.
 The figure also includes the standby control circuit and time-base timer circuit.

Figure 5-2. Block Diagram of the Clock Generation Block



□ Main clock oscillation circuit

This circuit generates an oscillation clock (HCLK) by connecting an oscillator to the high-speed oscillation pins.

□ PLL multiplier circuit

The oscillation clock is multiplied by the PLL oscillation and supplies it as a PLL clock (PCLK) to the clock selector.

□ Clock selector

From among the main clock, six different PLL clocks and sub clock, the clock selector selects the clock that is supplied to the CPU and peripheral function.

□ Clock selection register (CKSCR)

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

□ PLL/sub clock control register (PSCCR)

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

□ Oscillation stabilization wait time selector

This oscillation stabilization wait time selector selects an oscillation stabilization wait time for the oscillation clock. Selection is made from among four different time-base timer outputs.

5.2.1 Register of Clock Generation Block

This section explains the register of the clock generation block.

■ List of Register of Clock Generation Block and Initial Value

Figure 5-3. List of Clock Selection Register and Initial Value

Clock selection register (CKSCR)

Address
0000A1H

bit	15	14	13	12	11	10	9	8
	1	1	1	1	1	1	0	0

PLL/sub clock control register (PSCCR)

Address
0000CFH

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

5.3 Clock Selection Register (CKSCR)

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

■ Configuration of the Clock Selection Register (CKSCR)

Figure 5-4. Configuration of the Clock Selection Register (CKSCR)

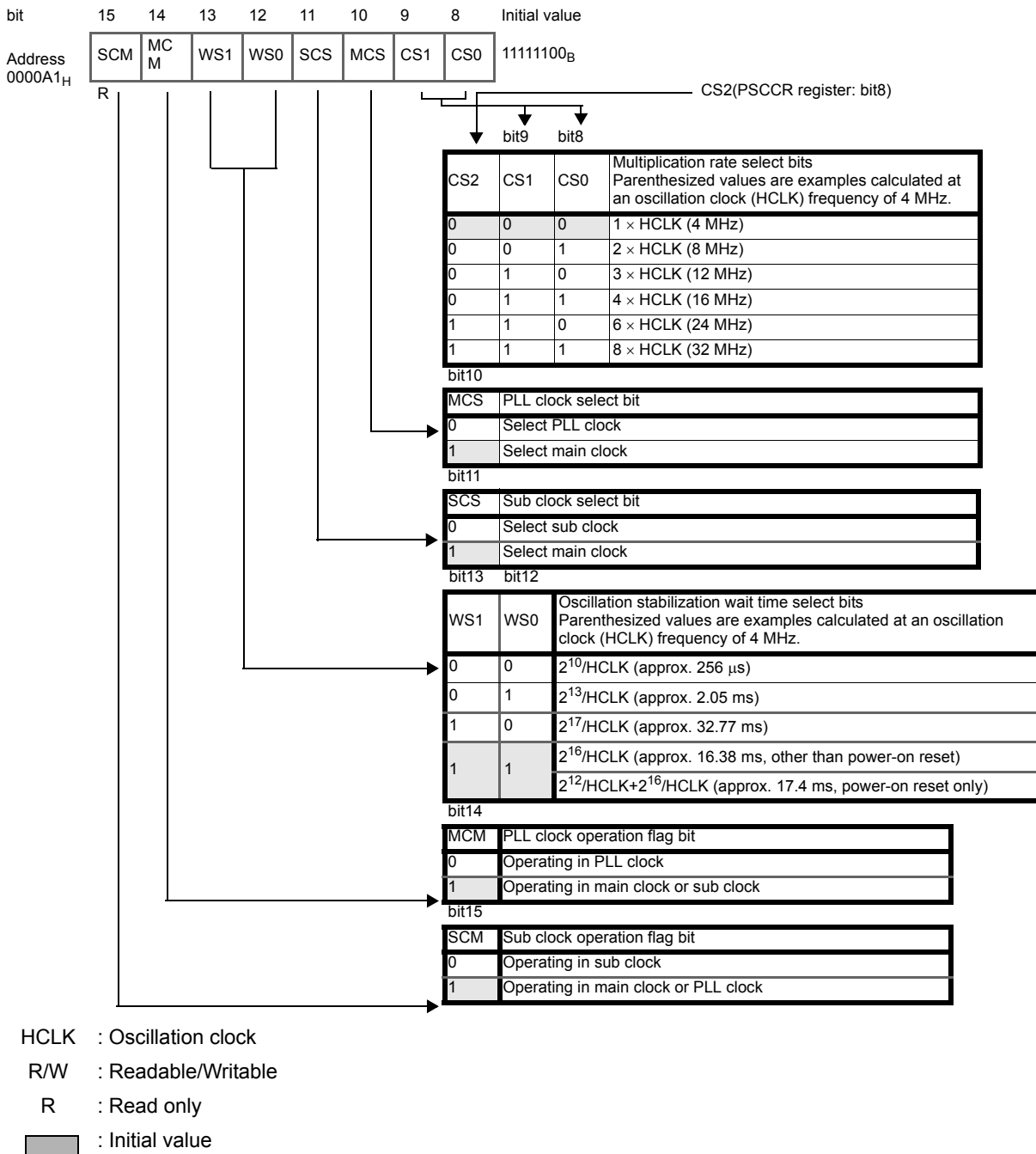


Table 5-1. Functions of Clock Selection Register (CKSCR) (Sheet 1 of 2)

Bit name		Function
bit15	SCM: Sub clock operation flag bit	<p>The bit indicates the main clock or sub clock currently selected as the machine clock.</p> <p>When the sub clock operation flag bit (CKSCR: SCM) is "0" and the sub clock select bit (CKSCR: SCS) is "1", it indicates that the machine clock is currently switching from sub clock to main clock. When the sub clock operation flag bit (CKSCR: SCM) is "1" and the sub clock select bit (CKSCR: SCS) is "0", it indicates that the machine clock is currently switching from main clock to sub clock. The writing will not be affected to the operation.</p>
bit14	MCM: PLL clock operation flag bit	<p>The bit indicates the main clock or PLL clock currently selected as the machine clock.</p> <p>When the PLL clock operation flag bit (CKSCR: MCM) is "1" and the PLL clock select bit (CKSCR: MCS) is "0", it indicates that the oscillation stabilization wait time of the PLL clock is currently being taken. The writing will not be affected to the operation.</p>
bit13, bit12	WS1, WS0: Oscillation stabilization wait time select bits	<p>These bits are used to select an oscillation stabilization wait time required for the oscillation clock when the stop mode is canceled, when transition occurs from sub clock mode to main clock mode, or when transition occurs from sub clock mode to PLL clock mode.</p> <p>These bits are used to select one from four time-base timer outputs.</p> <p>Any reset causes the bits to return to the initial value.</p> <p>Note:</p> <p>Set the oscillation stabilization wait time to an appropriate value depending on the oscillator used. See Section "7.2 Reset Sources and Oscillation Stabilization Wait Times". The oscillation stabilization wait time taken when the clock mode is switched from main clock to PLL clock is fixed at $2^{14}/\text{HCLK}$ (about 4.1 ms during operation at an oscillation clock frequency of 4 MHz). When the CPU switches from sub clock mode to PLL clock mode or when it returns from PLL stop mode to PLL clock mode, the oscillation stabilization wait time follows the values specified in these bits.</p> <p>The PLL clock requires an oscillation stabilization wait time of at least $2^{14}/\text{HCLK}$. For switching from sub clock mode to PLL clock mode and transiting to the PLL stop mode, therefore, set these bits to "10_B" or "11_B".</p>
bit11	SCS: Sub clock select bit	<p>This bit indicates the main clock or sub clock to be selected as the machine clock.</p> <p>When the machine clock is switched from the main clock to the sub clock (CKSCR: SCS = 1 → 0), the main clock mode changes to the sub clock mode of 1/SCLK (100 kHz oscillation clock frequency, operating at 8 division: approx. 80 μs) in synchronization with the sub clock.</p> <p>When the machine clock is switched from the sub clock to the main clock (CKSCR: SCS = 0 → 1), the clock mode changes from sub clock mode to main clock mode after the main clock oscillation stabilization wait time is generated. Time-base timer is cleared automatically.</p> <p>Any reset causes the bit to return to the initial value.</p> <p>Notes:</p> <p>When both of the MCS and SCS bits contain "0", the SCS bit supersedes the MCS bit, thereby setting the sub clock mode.</p> <p>If both the sub clock select bit (CKSCR: SCS) and PLL clock select bit (CKSCR: MCS) contain "0", the sub clock is preferred.</p> <p>When switching from the main clock to sub clock (CKSCR: SCS = 1 → 0), use the time-base timer interrupt enable bit (TBTC: TBIE) or interrupt level mask register (ILM: ILM2 to ILM0) to disable time-base timer interrupts before writing "0" to the sub clock select bit.</p> <p>The $2^{14}/\text{SCLK}$ sub clock oscillation stabilization wait time (100 kHz oscillation clock frequency, operating at 8 division: approx. 1.3 s) is generated at power on or at cancellation of the stop mode. If the clock mode is switched from main clock mode to sub clock mode, therefore, the oscillation stabilization wait time is generated.</p>
bit10	MCS: PLL clock select bit	<p>This bit indicates the main clock or PLL clock to be selected as the machine clock.</p> <p>When the machine clock is switched from the main clock to the PLL clock (CKSCR: MCS = 1 → 0), the clock mode changes from main clock mode to PLL clock mode after the PLL clock oscillation stabilization wait time is generated. The time-base timer is cleared automatically. The oscillation stabilization wait time taken when the clock mode is switched from main clock to PLL clock is fixed at $2^{14}/\text{HCLK}$ (about 4.1 ms during operation at an oscillation clock frequency of 4 MHz). The oscillation stabilization wait time taken when the machine clock is switched from sub clock mode to PLL clock mode follows the values specified in the oscillation stabilization wait time select bits (CKSCR: WS1, WS0).</p> <p>Any reset causes the bit to return to the initial value.</p> <p>Notes:</p> <p>When both of the MCS and SCS bits contain "0", the SCS bit supersedes the MCS bit, thereby setting the sub clock mode.</p> <p>When switching from the main clock to PLL clock (CKSCR: MCS = 1 → 0), use the time-base timer interrupt enable bit (TBTC: TBIE) or interrupt level mask register (ILM: ILM2 to ILM0) to disable time-base timer interrupts before writing "0" to the PLL clock select bit.</p>

Table 5-1. Functions of Clock Selection Register (CKSCR) (Sheet 2 of 2)

Bit name		Function					
bit9, bit8	CS1, CS0: Multiplication rate select bits	These bits select the PLL clock multiplication rate with the CS2 bit in the PLL/sub clock control register (PSCCR). One of six types of PLL clock multiplication rate can be selected. Any reset causes the bits to return to the initial value. Setting of CS0, CS1, and CS2					
			CS2	CS1	CS0	PLL clock multiplication rate	
			0	0	0	$\times 1$	
			0	0	1	$\times 2$	
			0	1	0	$\times 3$	
			0	1	1	$\times 4$	
			1	1	0	$\times 6$	
			1	1	1	$\times 8$	
		Note: When the PLL clock is selected (CKSCR: MCS=0), writing is controlled. When rewriting the multiplication rate, first write "1" into the PLL clock selection bit (CKSCR: MCS) temporarily, then rewrite the multiplication rate select bits (CKSCR: CS1, CS0) and return the PLL clock selection bit (CKSCR: MCS) to "0".					

5.4 PLL/Sub clock Control Register (PSCCR)

PLL/sub clock control register selects the PLL multiplication rate and sub clock division rate. This register is write only. Read value of all bits is set to "1".

■ Configuration of the PLL/Sub clock Control Register (PSCCR)

Figure 5-5. shows the configuration of the PLL/sub clock control register (PSCCR). Table 5-2. shows the function of each bit in the PLL/sub clock control register (PSCCR).

Figure 5-5. Configuration of the PLL/Sub clock Control Register (PSCCR)

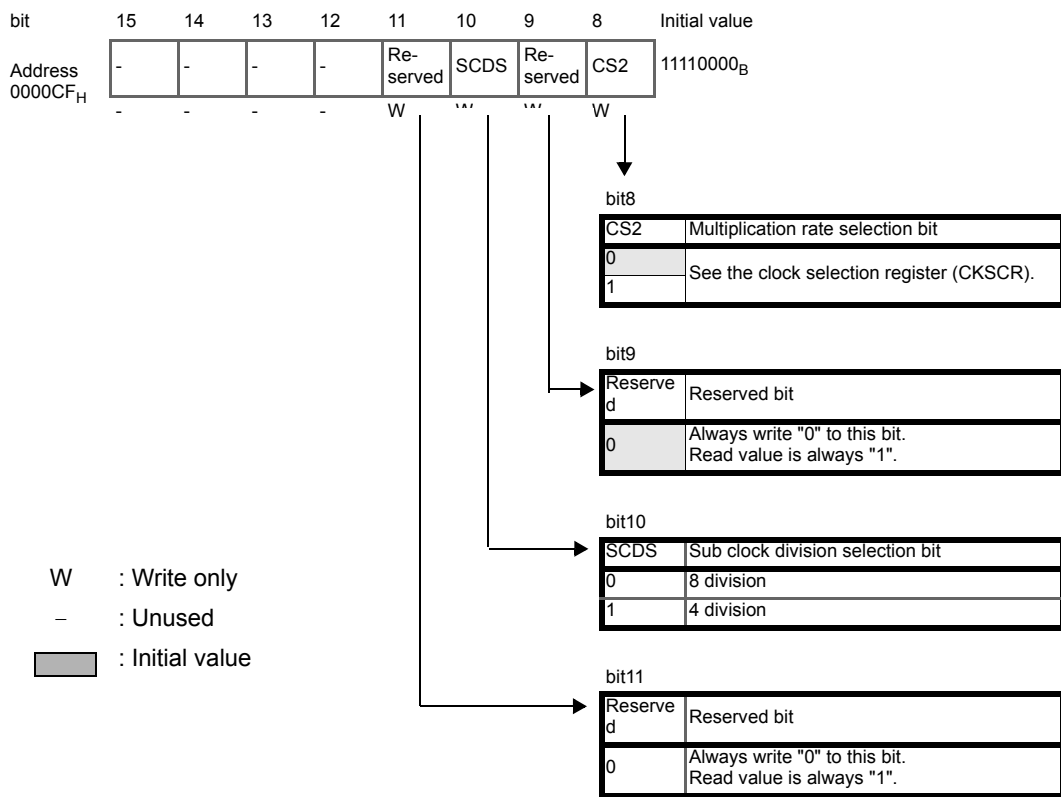


Table 5-2. Functional Description of Each Bit in the PLL/Sub clock Control Register (PSCCR)

Bit name		Function					
bit15 to bit12	Unused bits	These bits are not used. Writing to these bits has no effect on operation. Read value is always "1".					
bit11	Reserved bit	Always write "0" to this bit. Read value is always "1".					
bit10	SCDS: Sub clock division selection bit	The division ratio of the sub clock is selected. When "0" is written to this bit, 8 division is selected. When "1" is written to this bit, 4 division is selected. Read value is always "1". This bit is initialized to "0" by all reset sources. Note: Do not rewrite this bit while operating the sub clock.					
bit9	Reserved bit	Always write "0" to this bit. Read value is always "1".					
bit8	CS2: Multiplication rate selection bit	This bit and CS1 and CS0 bits of the clock selection register (CKSCR) determine the PLL multiplication rate.					
			CS2	CS1	CS0	PLL clock multiplication rate	
			0	0	0	× 1	
			0	0	1	× 2	
			0	1	0	× 3	
			0	1	1	× 4	
			1	1	0	× 6	
			1	1	1	× 8	
		Read value is always "1". This bit is initialized to "0" by all reset sources. Note: When MCS or MCM bit is "0", it is prohibited to change a value of this bit. Change the value in the main clock mode.					

Note:

PSCCR register is write-only register. Read value is different from writing value. Do not use the RMW instruction (SETB/CLRB instruction).

5.5 Clock Mode

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

■ Clock Mode

□ Main clock mode

In main clock mode, a clock with 2-frequency division of the clock generated by connecting on oscillator to the high-speed oscillation pins (X0, X1), as the operating clock for the CPU and peripheral functions.

□ Sub clock mode

The sub clock mode uses the divided-by-four or divided-by-eight clock, generated by the CR oscillator, as the operating clock for the CPU and peripheral functions. The division ratio for the sub clock can be set with SCDS bit in the PLL/sub clock control register (PSCCR).

□ PLL clock mode

The PLL clock mode uses the oscillation clock, multiplied by the PLL clock multiplier circuit (PLL oscillation circuit), 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/sub clock control register (PSCCR: CS2).

■ 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 time ($2^{14}/\text{HCLK}$).

□ 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 sub clock.

□ 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 time.

□ 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 time.

■ Selection of a PLL Clock Multiplier

Writing the value from "000_B" to "011_B", "110_B" and "111_B" to the CS1 and CS0 bits of the clock selection register (CKSCR) and CS2 bit of the PLL/sub clock control register (PSCCR) can select six types (1 to 4 multiplication, 6 multiplication and 8 multiplication) of PLL clock multiplier.

■ Machine Clock

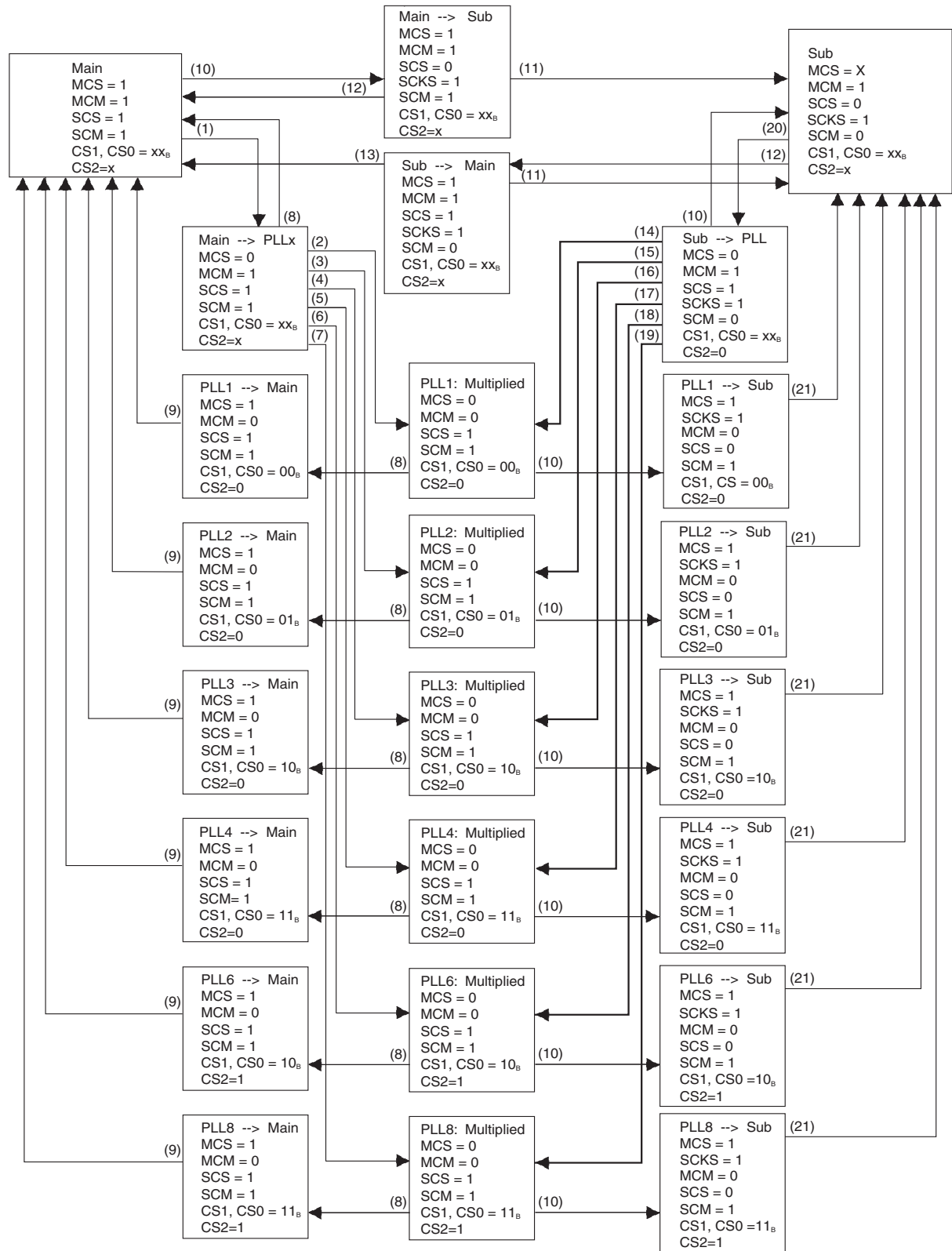
PLL clock, main clock, and sub clock outputted from the PLL multiplier circuit are used as machine clock. This machine clock is supplied to the CPU and peripheral functions. The main clock, PLL clock, or sub clock can be selected by writing to the MCS or SCS bit of the clock selection register (CKSCR).

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 MCS bit of the clock selection register (CKSCR) is "0" (PLL clock mode) and when the SCS bit of the clock selection register (CKSCR) is "0" (sub clock mode), the SCS bit is prioritized, and a transition to the sub clock mode is occurred.
- 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 switching to other clock mode and low-power consumption mode is performed before a transition is completed, the mode may not be switched.

Figure 5-6. shows the status change diagram caused by machine clock switching.

Figure 5-6. Status Change Diagram for Machine Clock Selection



- (1) Write "0" to MCS bit
- (2) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 00_B& CS2= 0
- (3) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 01_B& CS2= 0
- (4) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 0
- (5) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 11_B& CS2= 0
- (6) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 1
- (7) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 11_B& CS2= 1
- (8) Write "1" to MCS bit (include reset)
- (9) Synchronous timing of PLL clock and main clock
- (10) Write "0" to SCS bit
- (11) Synchronous timing of main clock and sub clock
- (12) Write "1" to SCS bit (MCS= 1)
- (13) Termination of main clock oscillation stabilization wait time
- (14) Termination of main clock oscillation stabilization wait time & CS1, CS0= 00_B& CS2= 0
- (15) Termination of main clock oscillation stabilization wait time & CS1, CS0= 01_B& CS2= 0
- (16) Termination of main clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 0
- (17) Termination of main clock oscillation stabilization wait time & CS1, CS0= 11_B& CS2= 0
- (18) Termination of main clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 1
- (19) Termination of main clock oscillation stabilization wait time & CS1, CS0= 11_B& CS2= 1
- (20) Write "1" to SCS bit (MCS= 0)
- (21) Synchronous timing of PLL clock and sub clock

MCS	:	PLL clock select bit of clock selection register (CKSCR)
MCM	:	PLL clock operation flag bit of clock selection register (CKSCR)
SCS	:	Sub clock select bit of clock selection register (CKSCR)
SCM	:	Sub clock operation flag bit of clock selection register (CKSCR)
CS1, CS0	:	Multiplication rate select bits of clock selection register (CKSCR)
CS2	:	Multiplication rate selection bit of PLL/ sub clock control register (PSCCR)
SCKS	:	Sub clock selection bit of clock supervisor control register (CSVCR)

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".

5.6 Oscillation Stabilization Wait Time

When the power is turned on during the oscillation clock is stopped or when stop mode is released, a time until the oscillation clock stabilizes (oscillation stabilization wait time) is required immediately after oscillation starts. Also, the oscillation stabilization wait time is required when the clock mode is switched from main clock to PLL clock, main clock to sub clock, sub clock to main clock, and sub clock to PLL clock.

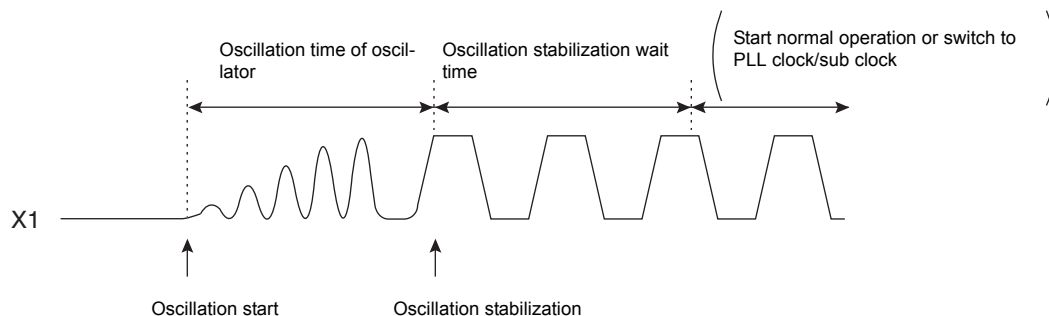
■ Operation of Oscillation Stabilization Wait Time

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 time has elapsed, the machine clock is supplied to the CPU. Because the oscillation stabilization wait time depends on the type of oscillator (crystal, ceramic, etc.), the proper oscillation stabilization wait time for the oscillator used must be selected. An oscillation stabilization wait time is selected by setting the clock selection register (CKSCR).

When clock mode is switched from main clock to PLL clock, main clock to sub clock, sub clock to main clock, or sub clock 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 5-7. shows the operation immediately after oscillation starts.

Figure 5-7. Operation Immediately after Oscillation Stabilization Wait Time



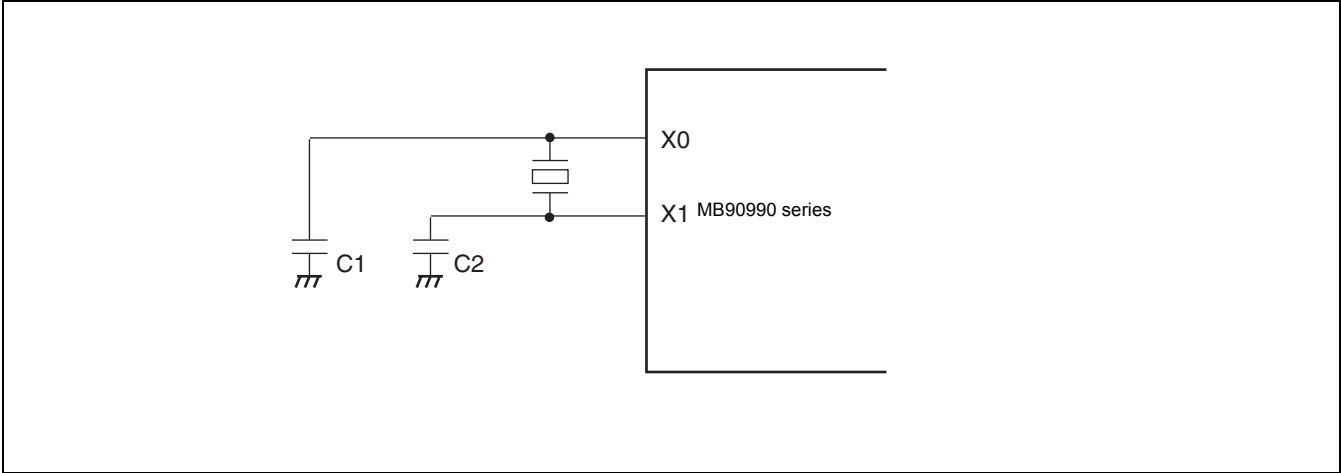
5.7 Connection of an Oscillator

The MB90990 series microcontroller contains a system clock generation circuit. Connecting an external oscillator to the oscillation pin generates the internal clock.

■ Connection of an Oscillator to the Microcontroller

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

Figure 5-8. Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller



6. Clock Supervisor



This chapter describes the functions and operations of the clock supervisor. **This function can be used by only the products with "J"-suffix.**

[6.1 Overview of Clock Supervisor](#)

[6.2 Configuration of Clock Supervisor](#)

[6.3 Registers of Clock Supervisor](#)

[6.4 Operations of Clock Supervisor](#)

[6.5 Notes on Using Clock Supervisor](#)

6.1 Overview of Clock Supervisor

The clock supervisor prevents the situation which is out of control, when oscillation clock oscillations have halted. This function switches to an CR clock generated in internal CR oscillator circuit, if oscillation clock oscillations have halted.

■ Overview of Clock Supervisor

- ❑ The clock supervisor monitors the oscillation clock oscillation and generates an internal reset if it detects that the oscillation has halted. In this case, the clock supervisor switches to the internal CR clock.

Reset source bits of watchdog timer control register (WDTC) can be used to determine whether a reset was triggered by the clock supervisor.

- ❑ A oscillation clock oscillation halt is detected if the rising edge of the oscillation clock is not detected for 4 CR clock cycles. The clock supervisor may detect incorrectly, if oscillation clock is longer than 4 CR clock cycles.
- ❑ While the clock stops in the stop mode, clock monitoring is disabled.
- ❑ The CR clock is continuously operated even if the oscillation is returned once it has halted.

Note:

Refer to the data sheet for the period and other details about the CR clock.

6.2 Configuration of Clock Supervisor

The clock supervisor consists of the following blocks:

Control circuit

CR oscillator circuit

Oscillation clock monitor

Main clock selector

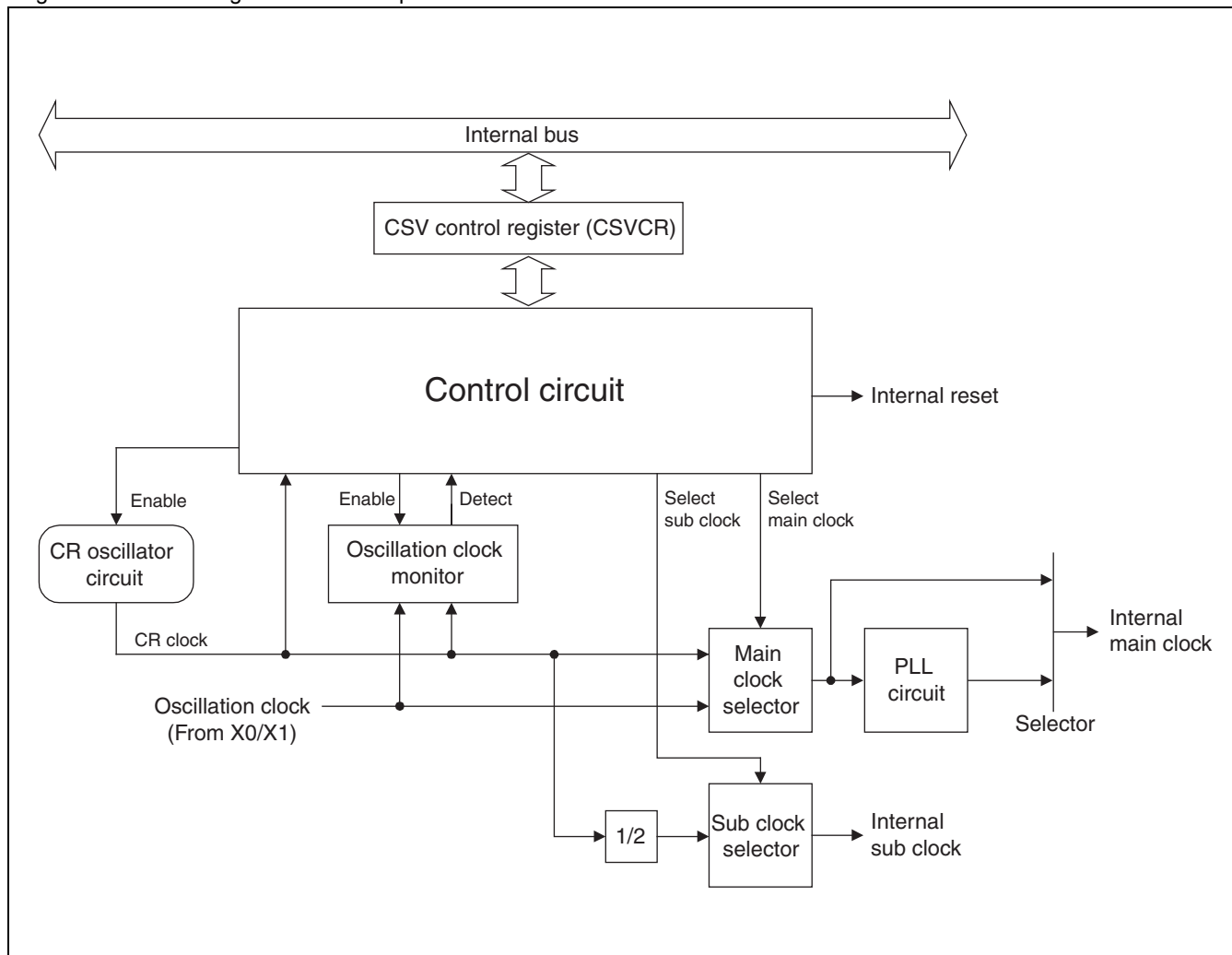
Sub clock selector

CSV control register (CSVCR)

■ Block Diagram of Clock Supervisor

Figure 6-1. shows a block diagram of the clock supervisor.

Figure 6-1. Block Diagram of Clock Supervisor



□ Control circuit

This block controls the clocks, resets, and other settings based on the information in the CSV control register (CSVCR).

□ CR oscillator circuit

This block is a internal CR oscillator circuit. The oscillation can be turned on or off via a control signal from the control circuit. This also serves as an internal clock after a clock halt is detected.

□ Oscillation clock monitor

This block monitors whether the oscillation clock halts.

□ Main clock selector

This block outputs the CR clock as the internal main clock upon detection of a oscillation clock halt.

□ Sub clock selector

This block outputs the clock obtained by dividing the CR clock as the internal sub clock.

□ CSV control register (CSVCR)

This block is used to control clock monitoring and CR clock and to check information on halt detection.

6.3 Registers of Clock Supervisor

This section describes the clock supervisor registers.

■ Clock Supervisor Register

Figure 6-2. shows the register of the clock supervisor.

Figure 6-2. Clock Supervisor Register

Clock supervisor control register (CSVCR)								Initial value
Address	bit 7	6	5	4	3	2	1	
0007960 _H	SCKS	MM	Reserved	RCE	MSVE	Reserved	Reserved	00011100 _B
	R/W	R	R	R/W	R/W	R/W	R/W	

R/W: Readable/writable
 R: Read only

6.3.1 Clock Supervisor Control Register (CSVCR)

The clock supervisor control register (CSVCR) is used to enable the various functions and to check the status.

■ Clock Supervisor Control Register (CSVCR)

Figure 6-3. Clock Supervisor Control Register (CSVCR)

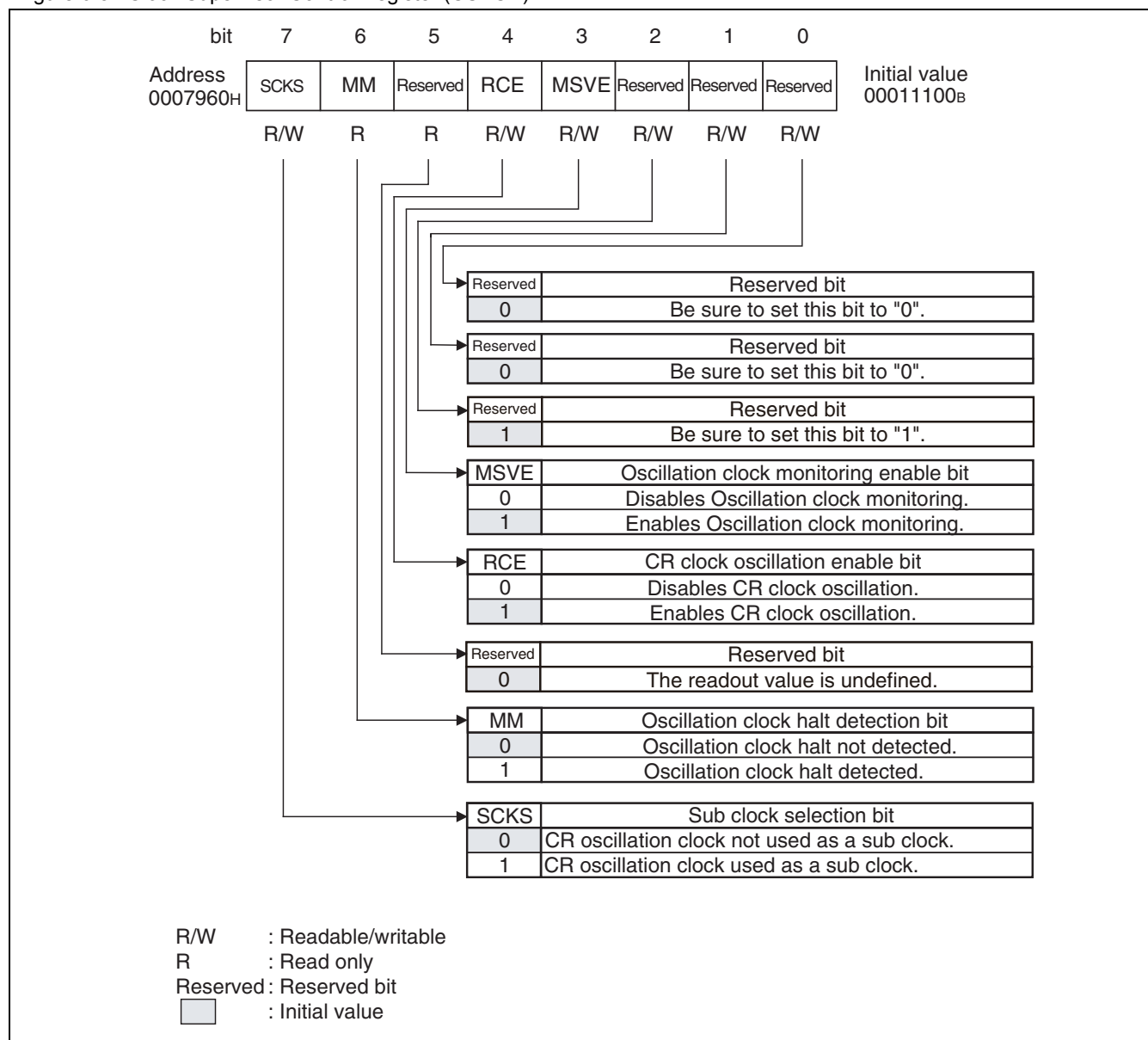


Table 6-1. Functions of Bits in Clock Supervisor Control Register (CSVCR)

Bit name		Function
bit7	SCKS: Sub clock selection bit	<p>This bit enables the built-in CR oscillation clock to be used as a sub clock.</p> <p>When set to "0": Disables transition to a sub clock.</p> <p>When set to "1": Enables transition to a sub clock, using the built-in CR oscillation clock.</p> <p>This bit is initialized on power-on reset, external reset, or low-voltage detection reset.</p> <p>When the bit is cleared to "0" in sub clock mode, it stops supplying the sub clock, and then the program stops. In this case, the bit can be restored upon power-on reset, external reset, or low-voltage detection reset.</p>
bit6	MM: Oscillation clock halt detection bit	<p>This bit is read-only, and this bit indicates that a oscillation clock halt has been detected.</p> <p>When set to "1": The bit indicates that a oscillation clock halt has been detected.</p> <p>When set to "0": The bit indicates that no oscillation clock oscillation halt has been detected.</p> <p>This bit is initialized to "0" by a power-on reset, external reset, or low-voltage detection reset.</p>
bit5	Reserved bit	<p>This bit is reserved.</p> <p>This bit is read-only, the read value is undefined.</p>
bit4	RCE: CR clock oscillation enable bit	<p>This bit enables CR oscillation.</p> <p>When set to "1": The bit enables oscillation.</p> <p>When set to "0": The bit disables oscillation.</p> <p>Before writing "0" to this bit, make sure that the clock monitor function has been disabled with the MM bit set to "0".</p> <p>This bit is initialized to "1" by a power-on reset, external reset, or low-voltage detection reset.</p>
bit3	MSVE: Oscillation clock monitoring enable bit	<p>This bit enables the monitoring of oscillation clock.</p> <p>When set to "1": The bit enables oscillation clock monitoring.</p> <p>When set to "0": The bit disables oscillation clock monitoring.</p> <p>On products with "J"-suffix, this bit is initialized to "1" by a power-on reset or low-voltage detection reset. On products with M-suffix, this bit is initialized to "1" by a power-on reset or external reset.</p>
bit2	Reserved bit	<p>This bit is reserved.</p> <p>Write "1" to this bit. The read value is always "1".</p>
bit1	Reserved bit	<p>This bit is reserved.</p> <p>Write "0" to this bit. The read value is always "0".</p>
bit0	Reserved bit	<p>This bit is reserved.</p> <p>Write "0" to this bit. The read value is always "0".</p>

Note:

When the power is turned on, the clock supervisor starts monitoring after the oscillation stabilization wait time for the oscillation clock elapses. The oscillation stabilization wait time of the oscillation clock must therefore be longer than the time required for the clock supervisor to start operating.

6.4 Operations of Clock Supervisor

This section describes the operations of the clock supervisor.

■ Operations of Clock Supervisor

The clock supervisor monitors the oscillation clock oscillations. If main clock and sub clock oscillations have halted, the device switches to an CR clock and generates a reset.

The following describes the operation in each clock mode.

- ❑ Oscillation clock oscillation halt in main clock mode

The clock supervisor detects that oscillation clock oscillation has halted, if no rising edge is detected on the oscillation clock for 4 CR clock cycles in main clock mode.

If a oscillation clock halt is detected, a reset is generated and the main clock switches to the CR clock.

The clock supervisor may detect incorrectly, if oscillation clock is a low speed (longer than 4 CR clock cycles). It results from using the CR clock for detecting that oscillation clock oscillation have halted.

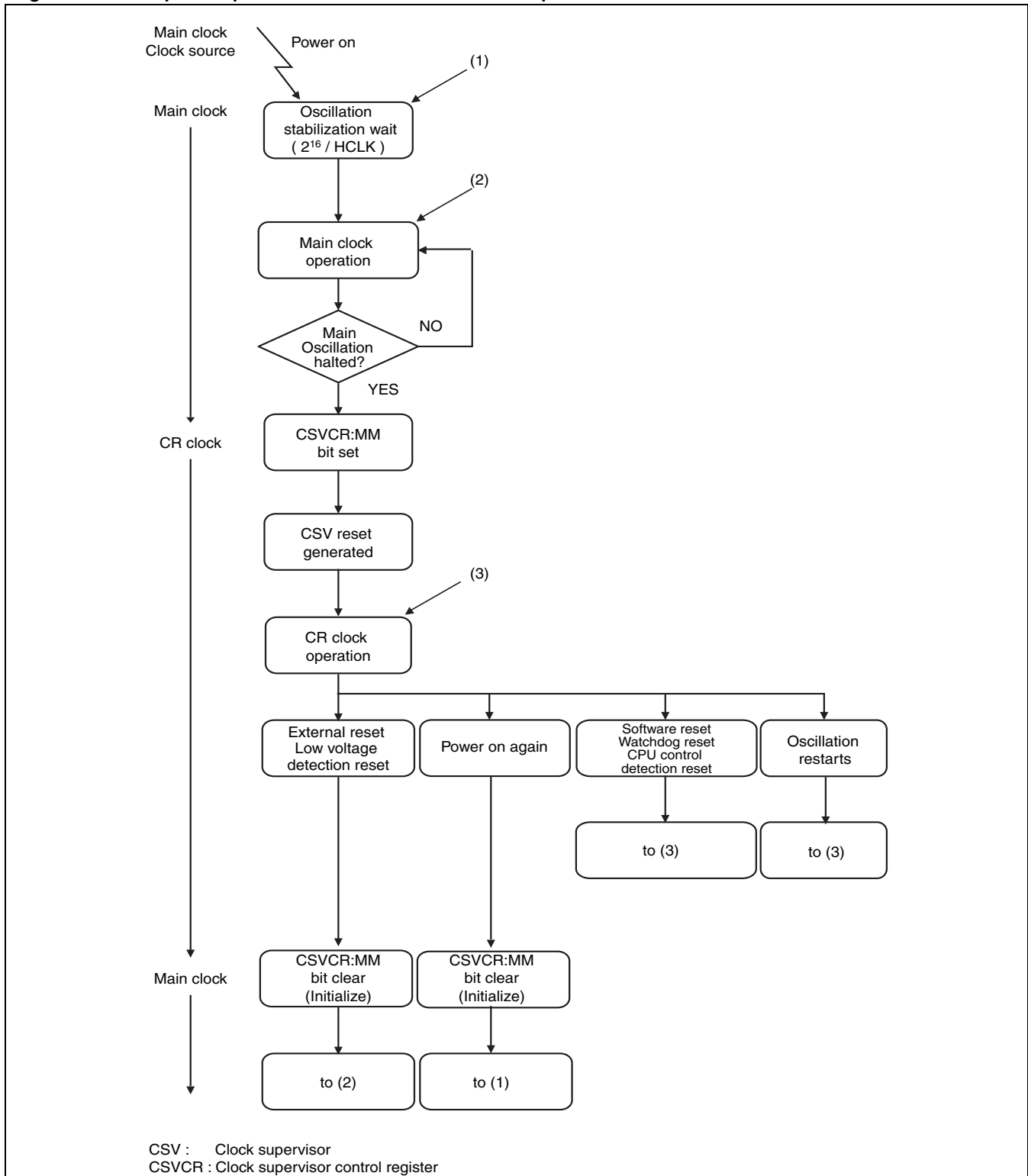
The clock supervisor does not detect the oscillation clock during stop mode.

- ❑ Oscillation clock halt in sub clock mode

In sub clock mode, the oscillation clock remains halted and is therefore not detected by the clock supervisor.

■ Example of Operation Flowchart for the Clock Supervisor

Figure 6-4. Example of Operation Flowchart for the Clock Supervisor



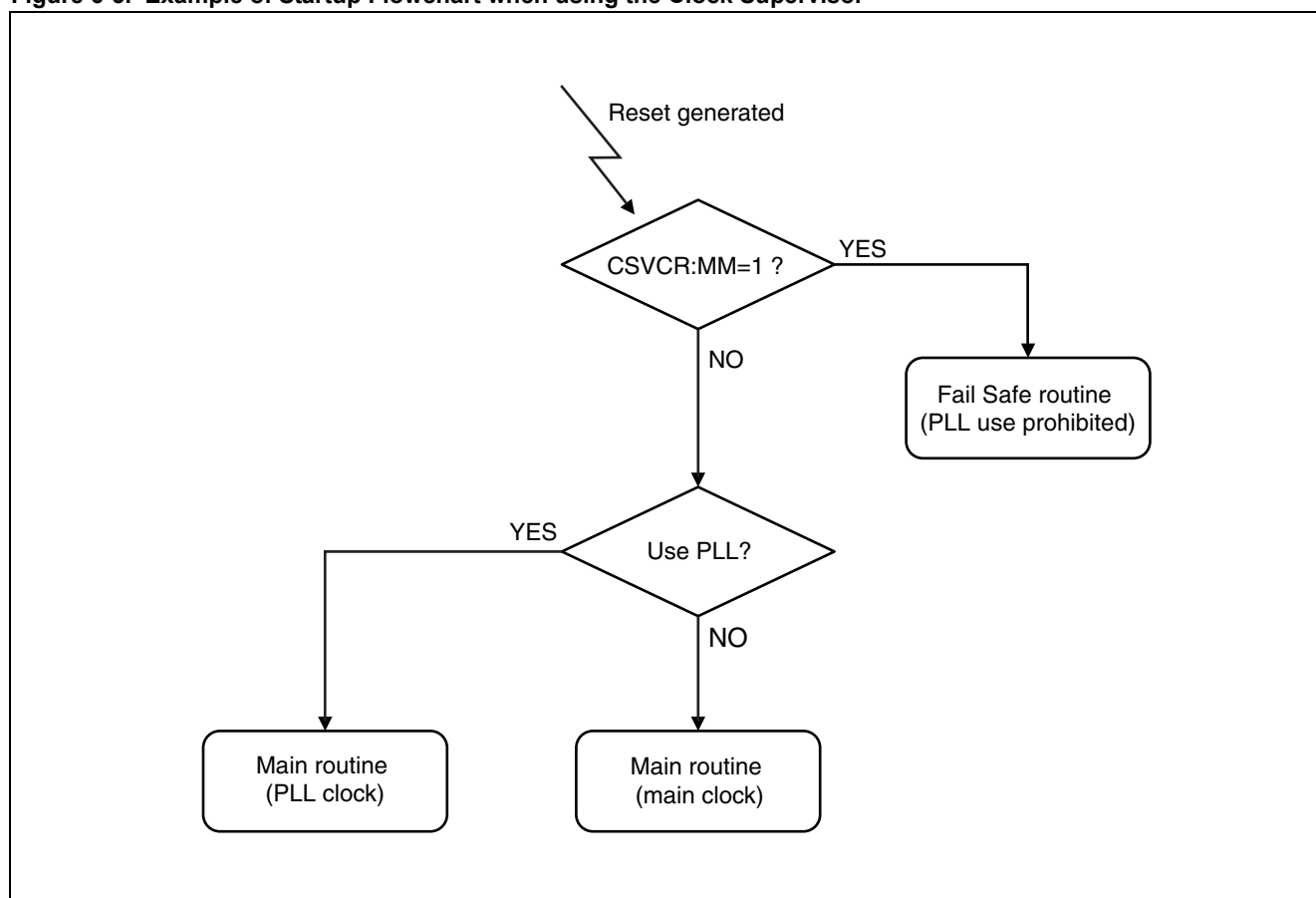
- ❑ After the power is turned on, the main clock operation starts after the oscillation stabilization wait time generated by the oscillation clock has elapsed.
- ❑ If the oscillation clock halts at power on, the device remains in the reset state (oscillation stabilization wait state). The operation changes to the main clock, after the oscillation restarts and the oscillation stabilization wait time elapsed.
- ❑ If an oscillation halt is detected during main clock operation, the operating clock is switched to the CR clock and a reset is generated.
- ❑ If the main oscillation continues (oscillation does not halt), the device continues to run using the main clock.
- ❑ If an external reset occurs during the CR clock operation, operation changes to the main clock. However, if the oscillation is halted at this time, another CSV reset is generated and the device returns to CR clock operation.

■ Example of Startup Flowchart when using the Clock Supervisor

Inserting checking process of the main clock halt detection bit (CSVCR:MM) enables user programs to control the Fail Safe routine.

Figure 6-5. shows the example of startup flowchart when using the clock supervisor.

Figure 6-5. Example of Startup Flowchart when using the Clock Supervisor



6.5 Notes on Using Clock Supervisor

Take note of the following points when using the clock supervisor.

■ Notes on using the Clock Supervisor

When using the clock supervisor, the following points should be checked.

- ❑ Operation of the clock supervisor at power on

When the power is turned on, the clock supervisor starts monitoring after the oscillation stabilization wait time for the oscillation clock has elapsed. Therefore, unless the operation continues for longer than the oscillation stabilization wait time for the oscillation clock, the clock supervisor will not operate.

- ❑ Transition to CR clock mode

Do not turn on the PLL after changing to CR clock mode.

As the frequency is below the lower limit for the input frequency of the PLL circuit, the PLL operation will not be guaranteed.

- ❑ Disabling the CR oscillation

Do not use the CR oscillation enable bit (CSVCR:RCE) to disable the CR oscillation during CR clock mode.

As this halts the internal clock, it may result in deadlock.

- ❑ Initializing the oscillation clock halt detection bit

The oscillation clock halt detection bit (CSVCR:MM) is initialized by a power-on reset, a low-voltage detection reset, or external reset only. The bit is not initialized by neither a watchdog reset, software reset, CPU operation detection reset, nor CSV reset. Accordingly, the device remains in CR clock mode if one of these resets occurs during CR clock mode.

7. Rests



This chapter describes resets for the MB90990 series microcontrollers.

[7.1 Overview of Resets](#)

[7.2 Reset Sources and Oscillation Stabilization Wait Times](#)

[7.3 External Reset Pin](#)

[7.4 Reset Operation](#)

[7.5 Reset Source Bits](#)

[7.6 Status of Pins in Reset](#)

7.1 Overview of 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 the processing at the address indicated by the reset vector.

The seven factors of a reset are as follows:

- Power-on reset
- External reset request via the $\overline{\text{RST}}$ pin
- Software reset request
- Watchdog timer overflow
- Low voltage detection reset request
- CPU operation detection reset request
- Clock supervisor reset request

■ Factors of Reset

Table 7-1. lists the factors of reset.

Table 7-1. Cause of Reset

Reset	Factor	Machine clock	Watchdog timer	Oscillation stabilization wait
Power-on	At power on	Main clock (MCLK)	Stop	Yes
External pin	"L" level input to $\overline{\text{RST}}$ pin	Main clock (MCLK)	Stop	None
Software	Write "0" to internal reset signal generation bit (RST) of low-power consumption mode control register (LPMCR)	Main clock (MCLK)	Stop	None
Watchdog timer	Watchdog timer overflow	Main clock (MCLK)	Stop	None
Low voltage detection reset	When low voltage* is detected	Main clock (MCLK)	Stop	None
CPU operation detection reset	When CPU operation detection counter overflows	Main clock (MCLK)	Stop	None
Clock supervisor reset	When some trouble of oscillation clock is detected	Internal CR oscillation clock	Stop	None

MCLK: Main clock (oscillation clock frequency divided by 2)

*: For details, see "19. Low Voltage Detection/CPU Operating Detection Reset".

□ Power-on reset

A power-on reset is generated when the power is turned-on. The oscillation stabilization wait time is fixed to $2^{12}+2^{16}$ oscillation clock cycles ($2^{12}/\text{HCLK}+2^{16}/\text{HCLK}$) (about 17.4 ms: at 4 MHz oscillation). When the oscillation stabilization wait time has elapsed, the reset is executed.

□ 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 at least 500 ns. The oscillation stabilization wait time is not required for an external reset.

Notes:

- If the reset source is generated during a write operation, 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 inputted concurrently. However, note that the following two points.
Note that a reset may prevent the data transfer requested by a string-processing instruction from being completed because the reset is accepted before a specified number of counters are transferred.
At external bus access, if the cycle is exceeded a certain period by RDY input, the reset is accepted forcibly without waiting the completion of instruction. Forcible reset is accepted within 16 machine cycles.
- When returning to the main clock mode by the external reset pin ($\overline{\text{RST}}$ pin) from the stop mode, sub clock mode, sub-sleep mode, and watch mode, input "L" level for at least oscillation time of oscillator* + 100 μs .
- *: Oscillation time of oscillator is the time that amplitude reaches 90%. It takes several to dozens of ms for crystal oscillators, hundreds of μs to several ms for ceramic oscillators.
- When returning to the main clock mode by the external reset pin ($\overline{\text{RST}}$ pin) from the time-base timer mode, input "L" level for at least 100 μs .

□ Software reset

A software reset is generated an internal reset 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 reset

A watchdog reset is generated by a watchdog timer overflow that occurs when "0" is not 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 is not required for watchdog reset.

□ Low voltage detection reset

The low-voltage detection reset is generated when the low voltage* is detected.

The oscillation stabilization wait time is not required for the low-voltage detection reset.

* : For details, see "[19. Low Voltage Detection/CPU Operating Detection Reset](#)".

□ CPU operation detection reset

The CPU operation detection reset is 20-bit counter that the oscillation clock is a count clock. If the CL bit of the low voltage/CPU operation detection reset control register (LVRC) is not cleared within a specified time after activation, the reset is generated.

The oscillation stabilization wait time is not required for the CPU operation detection reset.

□ Clock supervisor reset

Generates a reset when the failure of a oscillation clock is detected. Clock supervisor reset does not wait for the oscillation stabilization wait time to elapse.

Definition of clocks

HCLK: Oscillation clock frequency

MCLK: Main clock frequency

SCLK: Sub clock frequency

ϕ : Machine clock (CPU operating clock) frequency

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

See Section "[5.1 Clocks](#)", for details.

Note:

When a reset is generated in stop or sub clock mode, $2^{16}/\text{HCLK}$ oscillation stabilization wait time is required (approximately 16.28 ms, using HCLK=4 MHz oscillation).

See Section "[5.6 Oscillation Stabilization Wait Time](#)" for details.

7.2 Reset Sources and Oscillation Stabilization Wait Times

The MB90990 series has seven reset sources. The oscillation stabilization wait time for reset depends on the reset source.

■ Reset Sources and Oscillation Stabilization Wait Times

Table 7-2. summarizes reset sources and oscillation stabilization wait times.

Table 7-2. Reset Sources and Oscillation Stabilization Wait Times

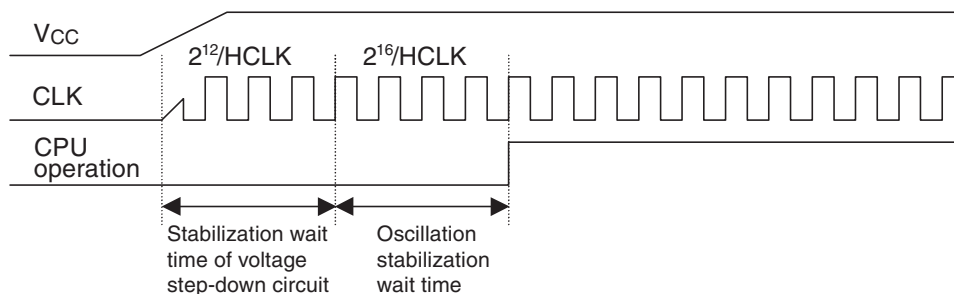
Reset	Reset source	Oscillation stabilization wait time The parenthesized values are provided when oscillation clock frequency operates at 4 MHz
Power-on	Power-on	$2^{12}/\text{HCLK} + 2^{16}/\text{HCLK}$ (approx. 17.4 ms)
Watchdog	Watchdog timer overflow	None Note:However, the WS1 and WS0 bits are initialized to "11 _B ".
External	L input from $\overline{\text{RST}}$ pin	None Note:However, the WS1 and WS0 bits are initialized to "11 _B ".
Software	Write "0" to RST bit of low-power consumption mode control register (LPMCR)	None Note:However, the WS1 and WS0 bits are initialized to "11 _B ".
Low voltage detection	When low voltage is detected	None Note:However, the WS1 and WS0 bits are initialized to "11 _B ".
CPU operation detection	When CPU operation detection counter overflows	None Note:However, the WS1 and WS0 bits are initialized to "11 _B ".
Clock supervisor reset	When some trouble of oscillation clock is detected	None Note:However, the WS1 and WS0 bits are initialized to "11 _B ".

HCLK:Oscillation clock frequency

WS1, WS0:Oscillation stabilization wait time select bits of clock selection register (CKSCR)

Figure 7-1. shows the oscillation stabilization wait times at power-on reset.

Figure 7-1. Oscillation Stabilization Wait Times at Power-on Reset



Note:

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

See Section "5.6 Oscillation Stabilization Wait Time" for details about oscillation stabilization wait times.

■ Oscillation Stabilization Wait Reset State

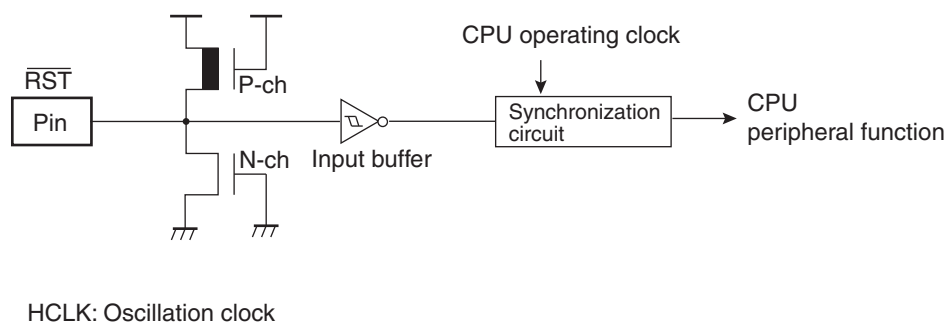
A reset operation in response to power-on reset and other resets during stop mode or sub clock mode is performed after the oscillation stabilization wait time has elapsed. This time interval is generated by the time-base timer. If the external reset has not been cleared after the interval, the reset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

7.3 External Reset Pin

The external reset pin (\overline{RST} pin) is an input pin used exclusively for reset. Inputting an L level signal generates an internal reset. For the MB90990 series, resets are generated in synchronization with the CPU operating clock. However, only the external pin generates a reset in asynchronous manner.

■ Block Diagrams of the External Reset Pin

Figure 7-2. Block Diagram of the External Reset Pin



Note:

Inputs to the \overline{RST} pin are accepted during cycles in which memory is not affected to prevent memory from being destroyed by reset during a write operation. A clock is required to initialize the internal circuit.

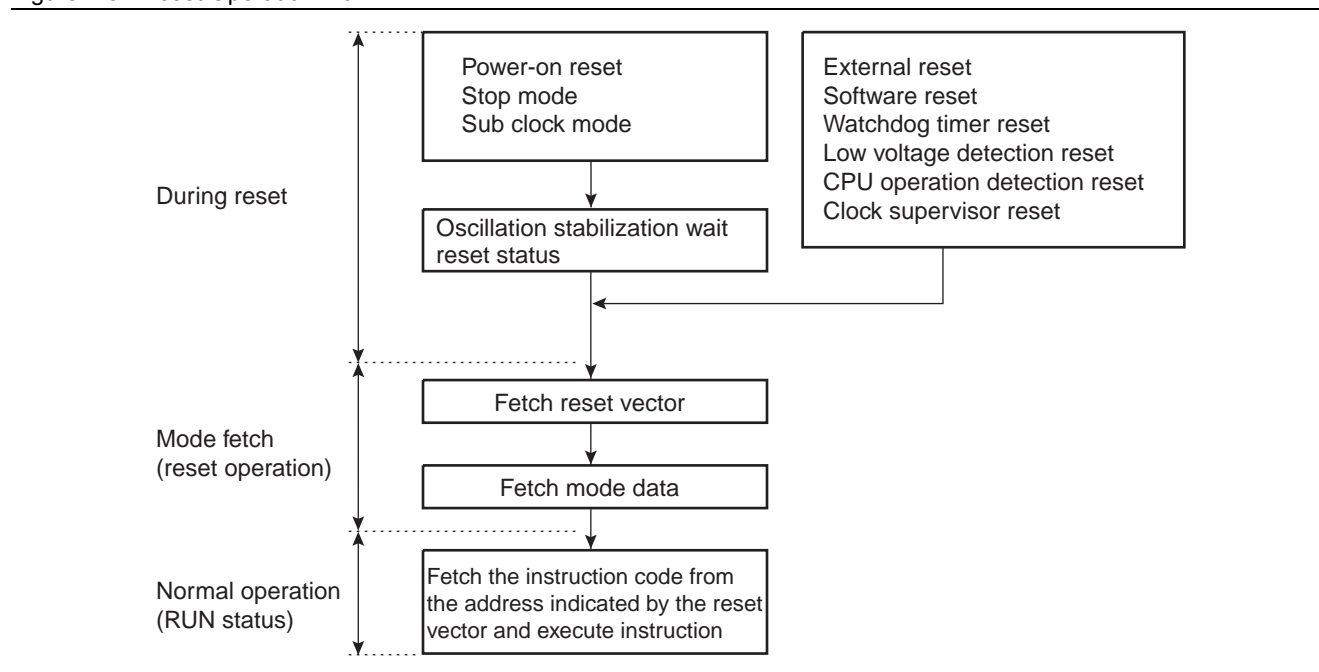
7.4 Reset Operation

When the reset signal is inactivated, the reset vector and mode data are 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 after the reset operation. For the power on reset, reset from the stop mode or sub clock mode, the mode fetch is performed after the oscillation stabilization wait time is elapsed.

■ Overview of Reset Operation

Figure 7-3. shows the reset operation flow.

Figure 7-3. Reset Operation Flow



■ Mode Pins

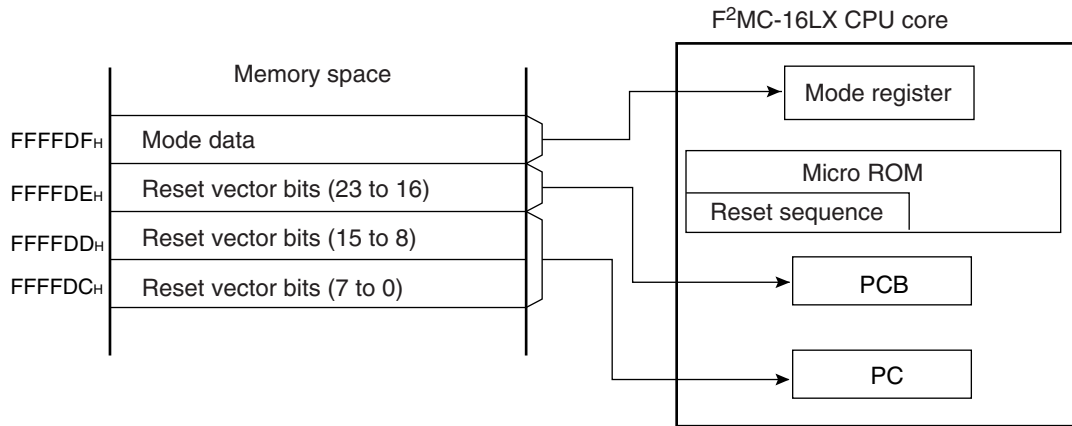
Setting the mode pins (MD0 to MD2) specifies how to fetch the reset vector and the mode data. Fetching the reset vector and the mode data is performed in the reset sequence. See Section "9.1.1 Mode Pins", for details on mode pins.

■ 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. shows the transfer of the reset vector and mode data.

Figure 7-4. 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: FFFFDC_H to FFFFDE_H)

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

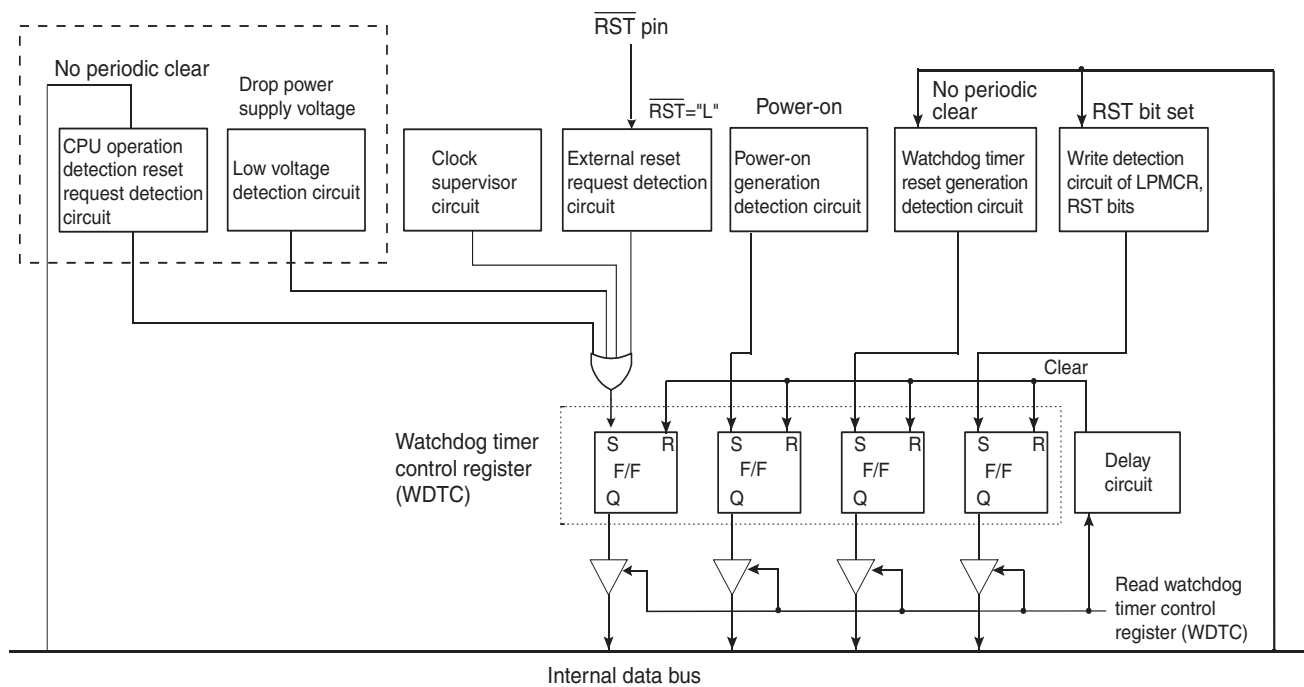
7.5 Reset Source Bits

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

■ Reset Source Bits

As shown in Figure 7-5. , a flip-flop is associated with each reset source. The contents of the flip-flops are obtained by reading the watchdog timer control register (WDTC). If the source of 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. Block Diagram of Reset Source Bits



S : Set
R : Reset
Q : Output
F/F : Flip Flop

■ Correspondence between Reset Source Bits and Reset Sources

Figure 7-6. shows the configuration of the reset source bits of the watchdog timer control register (WDTC). Table 7-3. maps the correspondence between the reset source bits and reset sources. See Section "12.3.1 Watchdog Timer Control Register (WDTC)" for details.

Figure 7-6. Configuration of Reset Source Bits (Watchdog Timer Control Register)

Watchdog timer control register (WDTC)

Address	bit15	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000A8 _H	(TBTC)			PONR	Reserved	WRST	ERST	SRST	WTE	WT1	WT0	X1XXX111 _B
				R	R	R	R	R	W	W	W	

R : Read only
W : Write only
X : Undefined

Table 7-3. Correspondence between Reset Source Bits and Reset Sources

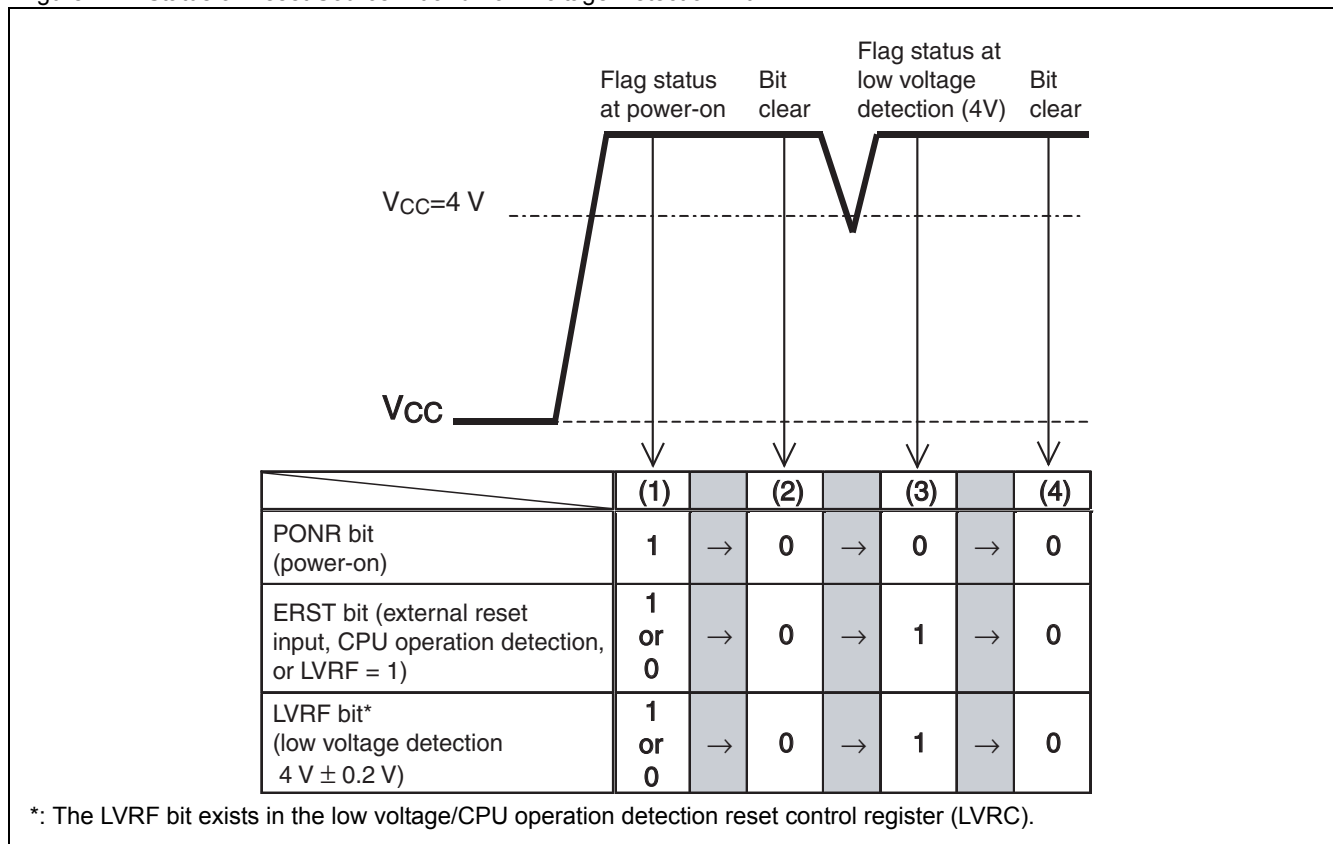
Reset source	PONR	WRST	ERST	SRST	LVRC		CSVCR
					LVRF	CPUF	MM
Generation of power-on reset request	1	X	X	X	Δ	Δ	0
Generation of reset request due to watchdog timer overflow	Δ	1	Δ	Δ	Δ	Δ	Δ
External reset request from $\overline{\text{RST}}$ pin	Δ	Δ	1	Δ	0	0	0
Low voltage detection reset	Δ	Δ	1	Δ	1	Δ	0
CPU operation detection reset	Δ	Δ	1	Δ	Δ	1	Δ
Clock supervisor (oscillation clock halt)	Δ	Δ	1	Δ	Δ	Δ	1
Generation of software reset request	Δ	Δ	Δ	1	Δ	Δ	Δ

Δ: Previous state retained

X: Undefined

■ Status of Reset Source Bit and Low Voltage Detection Bit

Figure 7-7. Status of Reset Source Bit and Low Voltage Detection Bit



(1) At power-on

Power-on reset bit (PONR), ERST, and LVRF are set to "1" at power on. However, ERST and LVRF are set to "0" when there is a steep rise at power-on.

(2) Bit clear

Bit is cleared by reading the WDTC register and by writing "0" to LVRF.

(3) At low voltage detection (4.0 V ± 0.2 V)

The LVRF and ERST bits are set to "1" at low voltage detection of $V_{CC} = 4.0 \text{ V} \pm 0.2 \text{ V}$.

(4) Bit clear

Bit is cleared by reading the WDTC register and by writing "0" to LVRF.

■ Notes about Reset Source Bits

- Multiple reset sources generated at the same time

When multiple reset sources are generated at the same time, the corresponding reset source bits of the watchdog timer control register (WDTC) are also set to "1". If, for example, an external reset request via the 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 power-on reset, because the PONR bit is set to "1" but all other reset source bits are undefined, the software should be programmed so that it will ignore all reset source bits except the PONR bit if it is "1".

- Clearing the reset source bits

The reset source bits are cleared only when the watchdog timer control register (WDTC) is read. Any bit corresponding to a reset source 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 Reset

This section describes the status of pins when reset occurs.

■ Status of Pins during Reset

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

About status of each pins during reset, please see Section "8.7 Status of Pins in Standby Mode and during Reset".

- 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.

■ 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).

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

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

Note:

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

8. Low Power Consumption Mode



This chapter explains the low-power consumption mode of MB90990 series microcontrollers.

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

8.1 Overview of Low-power Consumption Mode

The MB90990 series has the following CPU operating modes, any of which can be used depending on operating clock selection and clock oscillation control:

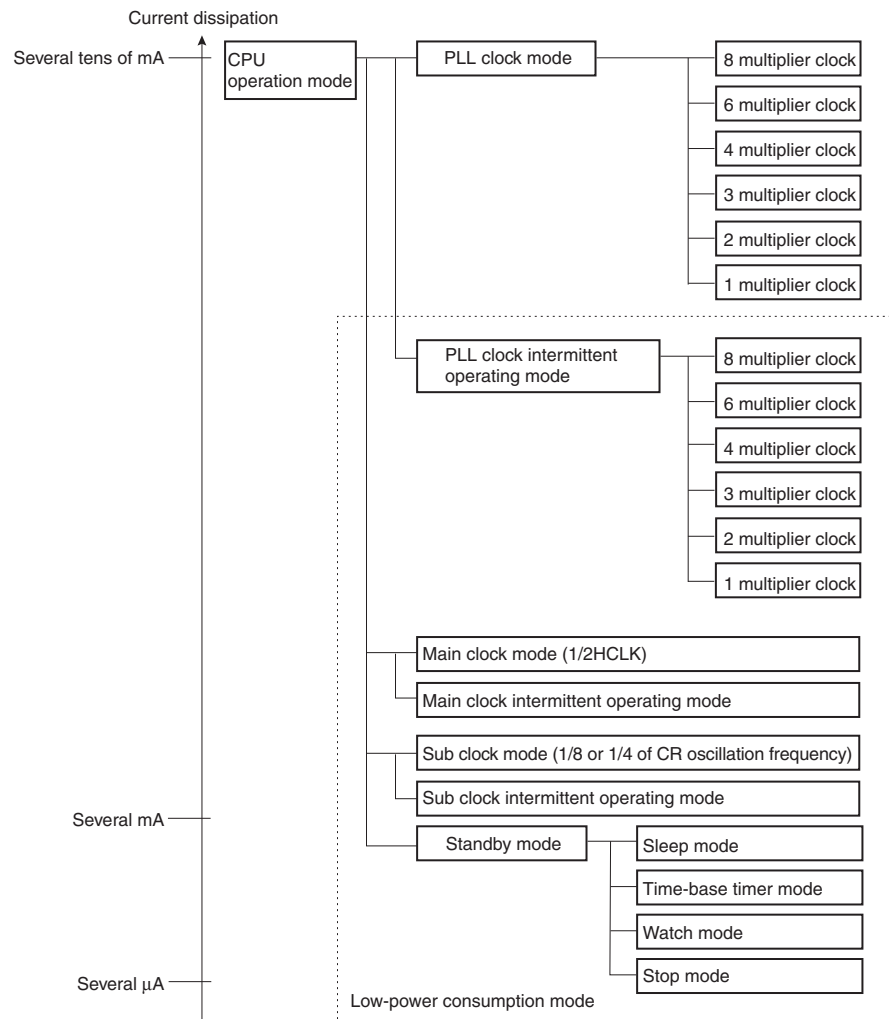
The modes other than the PLL clock mode are low-power consumption mode.

- Clock mode: main clock mode, PLL clock mode, or sub clock mode
- CPU intermittent operating mode: main clock intermittent operating mode, PLL clock intermittent operating mode, or sub clock intermittent operating mode
- Standby mode: sleep mode, stop mode, watch mode, or time-base timer mode

■ CPU Operating Modes and Current Dissipation

Figure 8-1. shows the relationship between the CPU operating modes and current dissipation.

Figure 8-1. CPU Operating Mode and Current Dissipation



This figure shows the image of operating mode and has some difference from actual current dissipation.

■ 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 (SCLK) is used to operate the CPU and peripheral functions. The sub clock can select a clock frequency divided by 8 or 4 of clocks from the CR oscillation.

In the sub clock mode, the main clock and PLL multiplier circuit are inactive.

The sub clock oscillation stabilization wait time of $2^{14}/\text{SCLK}$ (Approx. 1.3 s @100 kHz oscillation clock frequency, 1/8 division) takes place when power-on and reactivation from stop mode. If a transition from main clock mode to sub clock mode is performed that period, an oscillation stabilization wait time occurs.

Reference:

For the clock mode, see "5. Clocks".

■ CPU Intermittent Operating Mode

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

■ Standby Mode

In this mode, the standby control circuit stops supplying the operation clock to the CPU or peripheral functions or stops the oscillation clock itself (HCLK), thereby reducing power dissipation.

□ Sleep mode

The sleep mode stops the operation clock to the CPU during operation in each clock mode. The CPU stops, and the peripheral function operates the clock before the transition to the sleep mode. The sleep mode is divided into the main sleep mode, PLL sleep mode and sub-sleep mode before the transition to sleep mode.

□ Watch mode

The watch mode operates the sub clock (SCLK), watch timer, and low voltage detection circuit only. The main clock and PLL clock stop. All peripheral functions other than the watch timer and low voltage detection circuit stop. When the WDCS bit of watch timer control register (WTC) is "0", the watchdog timer continues operating.

- Time-base timer mode

The time-base timer mode operates the oscillation clock (HCLK), sub clock (SCLK), watchdog timer, time-base timer, watch timer, and low voltage detection circuit only. All peripheral functions other than the time-base timer, watchdog timer, watch timer, and low voltage detection circuit stop.

- Stop mode

The stop mode stops the oscillation clock (HCLK) and sub clock (SCLK) during operation in each clock mode, and all functions other than low voltage detection circuit stop. Data can be retained at the lowest power dissipation.

Note:

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).

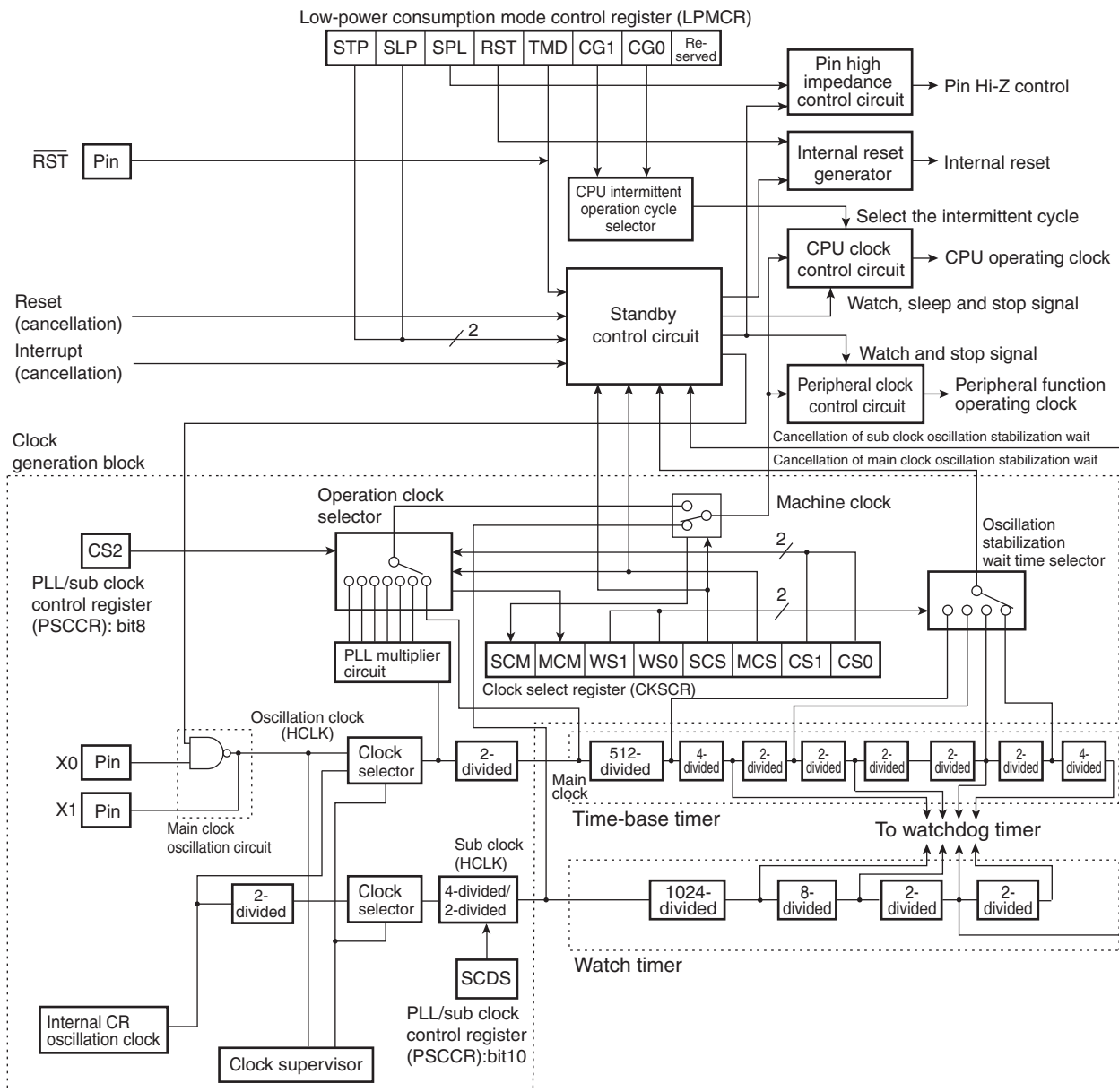
If the mode is switched to other clock mode and low-power consumption mode before completion of switching, the mode may not be switched.

8.2 Block Diagram of the Low-power Consumption Circuit

This section shows the block diagram of the low-power consumption circuit.

■ Block Diagram of the Low-power Consumption Circuit

Figure 8-2. Block Diagram of the Low-power Consumption Circuit



- ❑ CPU intermittent operation cycle selector

This selector selects the halt cycle count of the CPU clock during the CPU intermittent operation mode.

- ❑ Standby control circuit

CPU clock control and peripheral clock control circuits switch the CPU operation clock and peripheral function operation clock to transit to and cancel the standby mode.

- ❑ CPU clock control circuit

This circuit controls clocks supplied to the CPU.

- ❑ Pin high-impedance control circuit

This circuit makes I/O pins high-impedance in the watch mode, time-base timer mode and stop mode.

- ❑ Internal reset generator

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 mode.

8.3 Low-power Consumption Mode Control Register (LPMCR)

This register switches to or releases the low-power consumption mode. This register also generates the internal reset signal and sets the halt cycle count during the CPU intermittent operation mode.

■ Low-power Consumption Mode Control Register (LPMCR)

Figure 8-3. Configuration of the Low-power Consumption Mode Control Register (LPMCR)

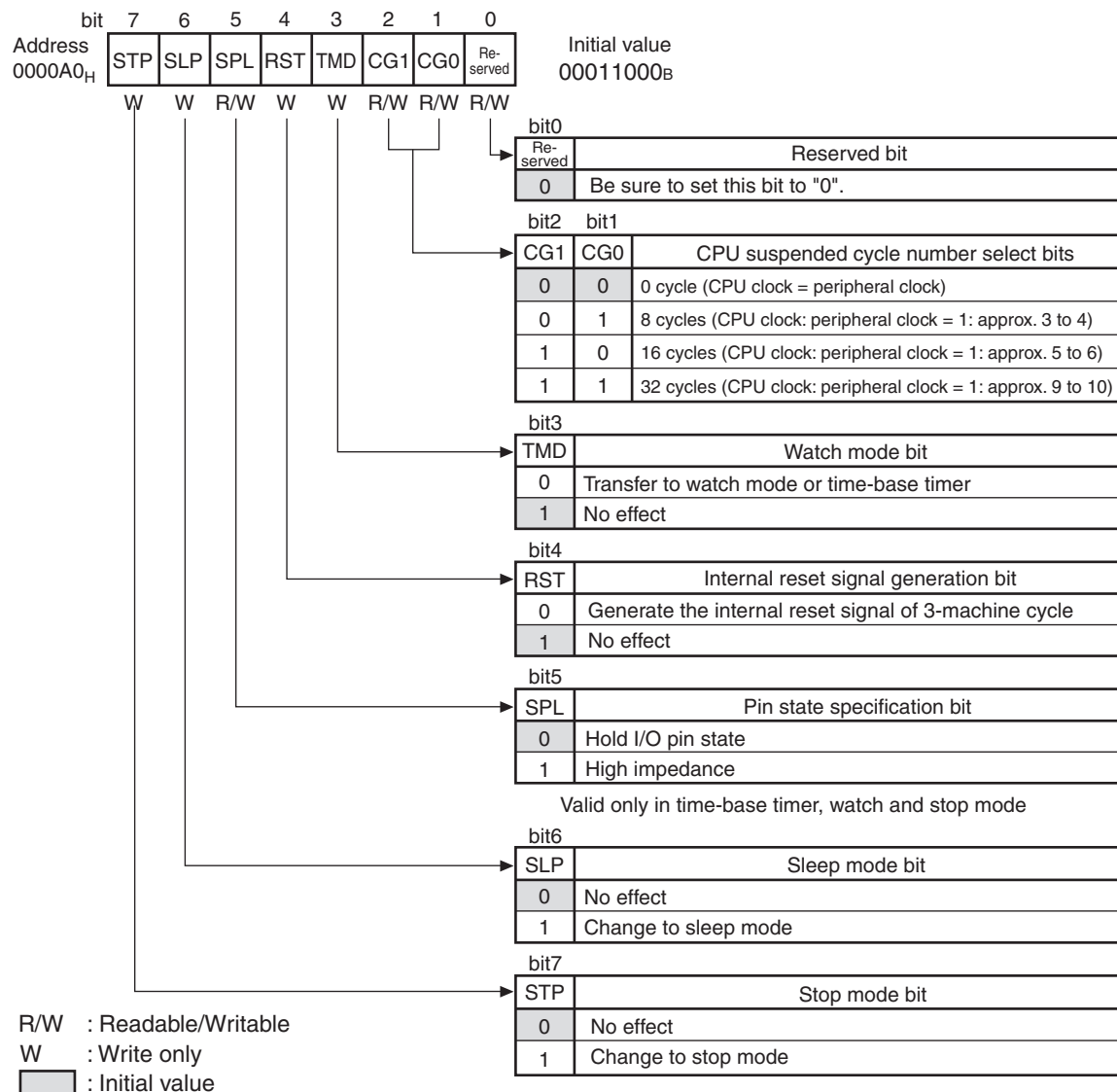


Table 8-1. Functions of Low-power Consumption Mode Control Register (LPMCR)

Bit name		Function
bit7	STP: Stop mode bit	<p>This bit transits to the stop mode.</p> <p>When the bit is set to "0": No effect.</p> <p>When the bit is set to "1": The CPU enters the stop mode.</p> <p>Read: "0" is always read.</p> <p>Initialized to "0" by a reset or an external interrupt.</p>
bit6	SLP: Sleep mode bit	<p>This bit shifts to sleep mode</p> <p>When the bit is set to "0": No effect.</p> <p>When the bit is set to "1": The CPU enters the sleep mode.</p> <p>Read: "0" is always read.</p> <p>Initialized to "0" by a reset or an external interrupt.</p> <p>When the STP and SLP bits are set to "1" at the same time, the STP bit supersedes the SLP bit, causing a transition to stop mode.</p>
bit5	SPL: Pin state specification bit	<p>The bit is used to set the state of input/output pins after transition to the stop mode, watch mode, or time-base timer mode.</p> <p>When the bit is set to "0": The current level of input/output pins is held.</p> <p>When the bit is set to "1": The I/O pins enter a high impedance state.</p> <p>The bit is initialized to "0" at a reset.</p>
bit4	RST: Internal reset signal generation bit	<p>This bit generates software reset.</p> <p>When the bit is set to "0": An internal reset signal for three machine cycles is generated.</p> <p>When the bit is set to "1": No effect</p> <p>Read: "1" is always read.</p>
bit3	TMD: Watch mode bit	<p>This bit shifts to watch mode or time-base timer mode.</p> <p>When the bit is set to "0": If the main clock mode or PLL clock mode is used, the bit transits to the time-base timer mode. If the sub clock mode is used, the bit transits to the watch mode.</p> <p>When the bit is set to "1": No effect</p> <p>The bit is set to "1" when reset or interrupt occurs.</p> <p>Read: "1" is always read.</p>
bit2, bit1	CG1, CG0: CPU suspended cycle number select bits	<p>These bits are used to set the halt cycle count of the CPU clock in the CPU intermittent operation mode.</p> <p>Any reset causes the bits to return to the initial value.</p>
bit0	Reserved: reserved bit	Always set this bit to "0".

Notes:

- Switching to a low-power consumption mode is performed by writing the low-power consumption mode control register (LPMCR). Only the instructions listed in [Table 8-2](#). should be used for this purpose. If other instructions are used for switching to a low-power consumption mode, operation cannot be assured.
- The standby mode transition instruction in [Table 8-2](#). must always be followed by an instructions highlighted by a line below.
 MOV LPMCR, #H' xx ; The low-power consumption mode transition instruction in [Table 8-2](#).
 NOP
 NOP
 JMP \$+3; jump to next instruction
 MOV A, #H' 10; any instruction
 The devices does not guarantee its operation after returning from the standby mode if you place an instructions other than the one enclosed in the dotted 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 Section "8.8 Notes on Using Low-power Consumption Mode".
- When word-length is used for writing the low-power consumption mode control register (LPMCR), 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 time-base timer mode, disable the output of peripheral functions, and set the STP bit of the LPMCR register to "1" or set the TMD bit of the LPMCR register to "0".

Table 8-2. Instructions to be Used for Switching to a Low-power Consumption Mode

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

8.4 CPU Intermittent Operation Mode

This mode is used for intermittent operation of the CPU while operation clock is supplied to the CPU and peripheral functions. The purpose of this mode is to reduce power dissipation.

■ CPU Intermittent Operation Mode

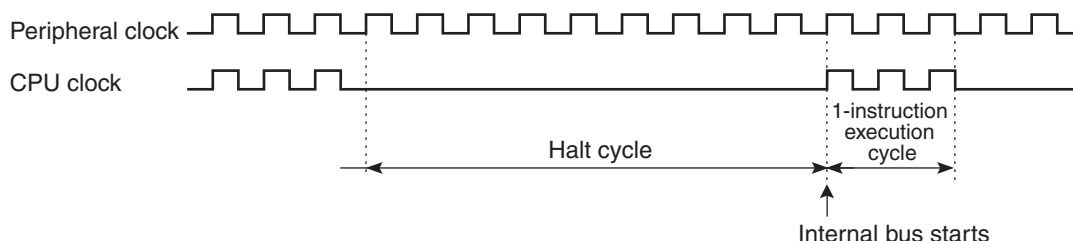
This mode halts the supply of the clock pulse to the CPU for a certain period. The halt occurs after the execution of every instruction that accesses a register, internal memory, I/O, peripheral functions, or the external bus. Internal bus cycle activation is therefore delayed. While high-speed clock 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 machine cycles that halts the clock supplied to the CPU.

- The instruction execution time of the CPU intermittent operation mode can be calculated by adding the normal execution time to the adjusted value that is the number of accesses to the register, internal memory, and peripheral functions multiplied by the number of pause cycle count.

Figure 8-4. shows the operating clock pulses during the CPU intermittent operation mode.

Figure 8-4. Clock Pulses during the CPU Intermittent Operation Mode



8.5 Standby Mode

The standby mode causes the standby control circuit to either stop supplying an operation clock to the CPU or peripheral functions or to stop the oscillation clock, reducing power dissipation.

■ Type of Standby Mode and Operation Status

Table 8-3. shows the type of standby mode and operation status.

Table 8-3. Type of Standby Mode and Operation Status (Sheet 1 of 2)

Mode name		Transition conditions	Oscillation clock (HCLK)	Sub clock (SCLK)	Machine clock	CPU	Watchdog timer	Peripheral function	Pin	Release method
Sleep mode	Main sleep mode	MCS=1 SCS=1 SLP=1	○	○	○	×	○*1	○	○	External reset or interrupt
	Sub-sleep mode	SCKS=1 MCS=X SCS=0 SLP=1 WDOS=0	×	○	○	×	○*1	○	○	External reset or interrupt
	Sub-sleep mode	SCKS=1 MCS=X SCS=0 SLP=1 WDOS=1	×	○	○	×	*2	○	○	External reset or interrupt
	PLL sleep mode	MCS=0 SCS=1 SLP=1	○	○	○	×	○*1	○	○	External reset or interrupt
Time-base timer mode	SPL=0	MCS=X SCS=1 TMD=0	○	○	×	×	○*1	×	◆	External reset or interrupt*4
	SPL=1	MCS=X SCS=1 TMD=0	○	○	×	×	○*1	×	Hi-Z*5	External reset or interrupt*4

Table 8-3. Type of Standby Mode and Operation Status (Sheet 2 of 2)

Mode name		Transition conditions	Oscillation clock (HCLK)	Sub clock (SCLK)	Machine clock	CPU	Watchdog timer	Peripheral function	Pin	Release method
Watch mode	SPL=0	SCKS=1 MCS=X SCS=0 TMD=0 WDSCS=0	×	○	×	×	○ ^{*1}	× ^{*6}	◆	External reset or interrupt ^{*7}
	SPL=1	SCKS=1 MCS=X SCS=0 TMD=0 WDSCS=0	×	○	×	×	○ ^{*1}	× ^{*6}	Hi-Z ^{*5}	External reset or interrupt ^{*7}
	SPL=0	SCKS=1 MCS=X SCS=0 TMD=0 WDSCS=1	×	○	×	×	- ^{*2}	× ^{*6}	◆	External reset or interrupt ^{*7}
	SPL=1	SCKS=1 MCS=X SCS=0 TMD=0 WDSCS=1	×	○	×	×	- ^{*2}	× ^{*6}	Hi-Z ^{*5}	External reset or interrupt ^{*7}
Stop mode	SPL=0	STP=1	×	×	×	×	×	×	◆	External reset or interrupt ^{*8}
	SPL=1	STP=1	×	×	×	×	×	×	Hi-Z ^{*5}	External reset or interrupt ^{*8}

○:Operation, ×: Stop, ◆: Held in the state before transiting, Hi-Z: High impedance

^{*1}:When the mode changes, the watchdog timer is cleared once.

^{*2}:The watchdog timer cannot be used.

^{*3}:The time-base timer, watch timer, and low voltage detection function operate.

^{*4}:Watch timer, time-base timer, and external interrupts

^{*5}:The DTP/external interrupt input pin operates.

^{*6}:The watch timer operates.

^{*7}:Watch timer and external interrupts

^{*8}:External interrupt

MCS:PLL clock select bit in clock selection register (CKSCR)

SCS:Sub clock select bit in the clock selection register (CKSCR)

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

SCKS:Sub clock select bit in the clock supervisor control register (CSVCR)

Note:

For the pins shared between port functions and peripheral functions in the stop mode, watch mode, or time-base timer mode, 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 during operation in each clock mode. The CPU stops, and peripheral function operates.

■ Switching to Sleep Mode

Writing "1" in the SLP bit and "0" in the STP bit of the low-power consumption mode control register (LPMCR) triggers a switch to a sleep mode according to setting of the MCS and SCS bits in the clock selection register (CKSCR). [Table 8-4](#) shows the correspondence between MCS and SCS bits in the clock selection register (CKSCR) and sleep mode.

Table 8-4. Setting of Clock Selection Register (CKSCR) and Sleep Mode

Clock selection register (CKSCR)		Sleep mode to be switched
MCS	SCS	
1	1	Main sleep mode
0	1	PLL sleep mode
1	0	Sub-sleep mode
0	0	

Note:

When "1" is written to the SLP and STP bits of the low-power consumption mode control register (LPMCR) at the same time, the STP bit setting overrides the SLP bit setting and the mode switches to the stop mode. When "1" is set to the SLP bit and "0" is set to the TMD bit at the same time, the TMD bit setting overrides the SLP bit setting and the mode switches to the time-base 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.

- Operation during an interrupt request

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

- Status of pins

During a sleep mode, all pins retain their previous status.

■ Return from Sleep Mode

The sleep mode is canceled by a reset source or when an interrupt is generated.

- Return by reset source

When the sleep mode is canceled by a reset source, the mode transits to the main clock mode after the sleep mode is canceled, transiting to the reset sequence.

Note:

For returning from sub sleep mode to main clock mode using the external reset pin ($\overline{\text{RST}}$ pin), input the "L" level for at least oscillation time for the oscillator* + 100 μs + 16 machine cycles (main clock).

*:The oscillation time for the oscillator is the period of time taken until its amplitude reaches 90%. It takes several to dozens of ms for crystal oscillators, hundreds of μs to several ms for ceramic oscillators.

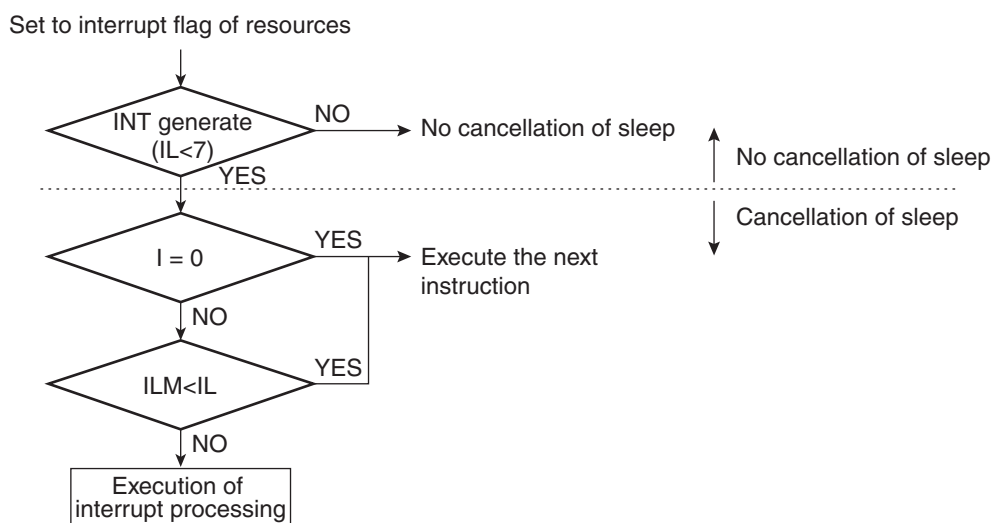
□ Return by interrupt

When a higher interrupt request than the interrupt level (IL) of 7 is generated from the resources in the sleep mode, the sleep mode is canceled. After the sleep mode is canceled, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR).

- When the CPU is not ready to accept any interrupt request, the next instruction to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.

Figure 8-5. shows the release of a sleep mode when an interrupt occurs.

Figure 8-5. 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 a sleep mode has been specified. The CPU then proceeds to interrupt processing.

8.5.2 Watch Mode

This mode causes all functions, excluding the sub clock (SCLK), watch timer, and low voltage detection circuit, to stop. Main clock and PLL clock stop.

When the WDCS bit of watch timer control register (WTC) is "0", the watchdog timer continues operating.

■ Switching to the Watch Mode

When "0" is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the sub clock run mode, 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.

- 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 watch mode.

If the CPU is not ready to accept any interrupt request, the instruction next to currently executing instruction is executed. If the CPU is ready to accept any interrupt request, an interrupt operation immediately branches to the interrupt processing routine.

- Status of pins

Whether the I/O 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).

Note:

To set the pin that is shared the peripheral function and port to the high impedance in the watch mode, disable the output of peripheral function, then set the TMD bit of the low-power consumption mode control register (LPMCR) to "0".

■ **Return from Watch Mode**

The watch mode is canceled by a reset source or when an interrupt is generated.

- Return by reset source

When the watch mode is canceled by a reset source, the mode transits to the main clock mode after the watch mode is canceled, transiting to the reset sequence.

- Return by interrupt

When an interrupt request higher than the interrupt level (IL) of 7 is generated from the watch timer and external interrupt in the watch mode, the watch mode is canceled. After the watch mode is canceled, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR). In the sub clock mode, no oscillation stabilization wait time is generated and the interrupt request is identified immediately after return from the watch mode.

- When the CPU is not ready to accept any interrupt request, the next instruction to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.

Note:

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which the watch mode has been specified. The CPU then proceeds to interrupt processing.

8.5.3 Time-base Timer Mode

This mode causes all functions, excluding oscillation clock (HCLK), sub clock (SCLK), the time-base timer, the watchdog timer, the watch timer, and low voltage detection circuit, to stop. In this mode, only the peripheral function of the time-base timer, watchdog timer, watch timer, and low voltage detection circuit, operate.

■ **Switching to the Time-base 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: SCM = 1), switching to the time-base timer mode occurs.

- Data retention function

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

- 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 time-base timer mode.

If the CPU is not ready to accept any interrupt request, the instruction next to currently executing instruction is executed. If the CPU is ready to accept any interrupt request, an interrupt operation immediately branches to the interrupt processing routine.

□ Status of pins

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

Note:

To set the pin that is shared the peripheral function and port to the high impedance in the time-base timer mode, disable the output of the peripheral function, then set the TMD bit of the low-power consumption mode control register (LPMCR) to "0".

■ Return from Time-base Timer Mode

The time-base timer mode is canceled by a reset source or when an interrupt is generated.

□ Return by reset source

When the time-base timer mode is canceled by a reset source, the mode transits to the main clock mode after the time-base timer mode is canceled, transiting to the reset sequence.

□ Return by interrupt

When an interrupt request higher than interrupt level (IL) of 7 is generated from the watch timer, time-base timer, and external interrupt in the time-base timer mode, the time-base timer mode is canceled. After the time-base timer mode is canceled, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR).

- When the CPU is not ready to accept any interrupt request, the next instruction to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.
- The following two time-base timer modes are available:
 - Main clock \longleftrightarrow time-base timer mode
 - PLL clock \longleftrightarrow time-base timer mode

Note:

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which the time-base timer mode has been specified. The CPU then proceeds to interrupt processing.

8.5.4 Stop Mode

Because this mode causes oscillation clock (HCLK) and sub clock (SCLK) to stop during operation in each clock mode, data can be retained by the lowest power dissipation.

■ Stop Mode

When "1" is written to the STP bit of the low-power consumption mode control register (LPMCR) during operation in the PLL clock mode (CKSCR: MCS=1, SCS=0), the mode transits to the stop mode according to the settings of the MCS bit and SCS bit in the clock selection register (CKSCR).

Table 8-5. shows the settings of the MCS and SCS bits in the clock selection register (CKSCR) and the stop modes.

Table 8-5. Clock Selection Register (CKSCR) Settings and Stop Modes

Clock selection register (CKSCR)		Stop Mode to be Transited
MCS	SCS	
1	1	Main stop mode
0	1	PLL stop mode
1	0	Sub-stop mode
0	0	

Note:

If both the STP and SLP bits in the low-power consumption mode control register (LPMCR) are set to "1" simultaneously, the STP bit is preferred and the mode transits to the stop mode.

□ Data retention function

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

□ Operation during an interrupt request

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

If the CPU is not ready to accept any interrupt request, the instruction next to currently executing instruction is executed. If the CPU is ready to accept any interrupt request, an interrupt operation immediately branches to the interrupt processing routine.

□ Status of pins

Whether the I/O 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:

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

■ Return from Stop Mode

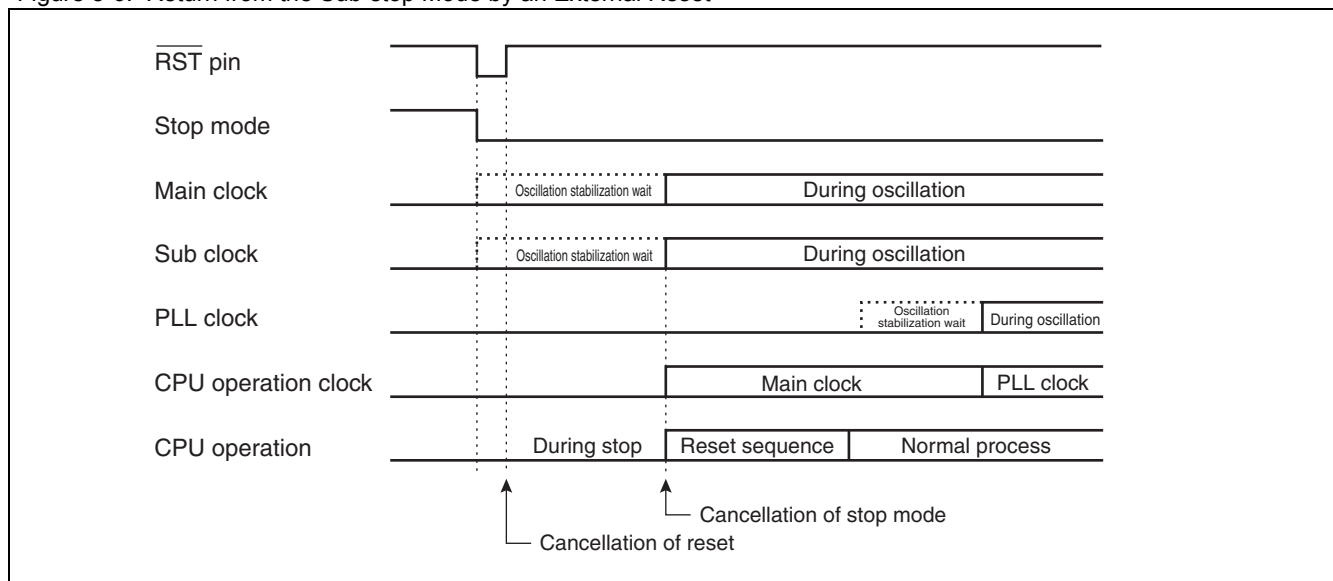
The stop mode is canceled by a reset source or when an interrupt is generated. At return from the stop mode, the oscillation clock (HCLK) and the sub clock (SCLK) stop, so the stop mode is canceled after the elapse of the main clock oscillation stabilization wait time or the sub clock oscillation stabilization wait time.

□ Return by reset source

When the stop mode is canceled by a reset source, the main clock oscillation stabilization wait time is generated. After the termination of the main clock oscillation stabilization wait time, the stop mode is canceled, transiting to the reset sequence.

Figure 8-6. shows the return from the sub-stop mode by an external reset.

Figure 8-6. Return from the Sub-stop Mode by an External Reset



□ Return by interrupt

When an interrupt request higher than the interrupt level (IL) of 7 is generated from external interrupt in the stop mode, the stop mode is canceled. In the stop mode, the main clock oscillation stabilization wait time or the sub clock oscillation stabilization wait time is generated after the stop mode is canceled. After the termination of the main clock oscillation stabilization wait time or sub clock oscillation stabilization wait time, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR).

- When the CPU is not ready to accept any interrupt request, the instruction next to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.

Notes:

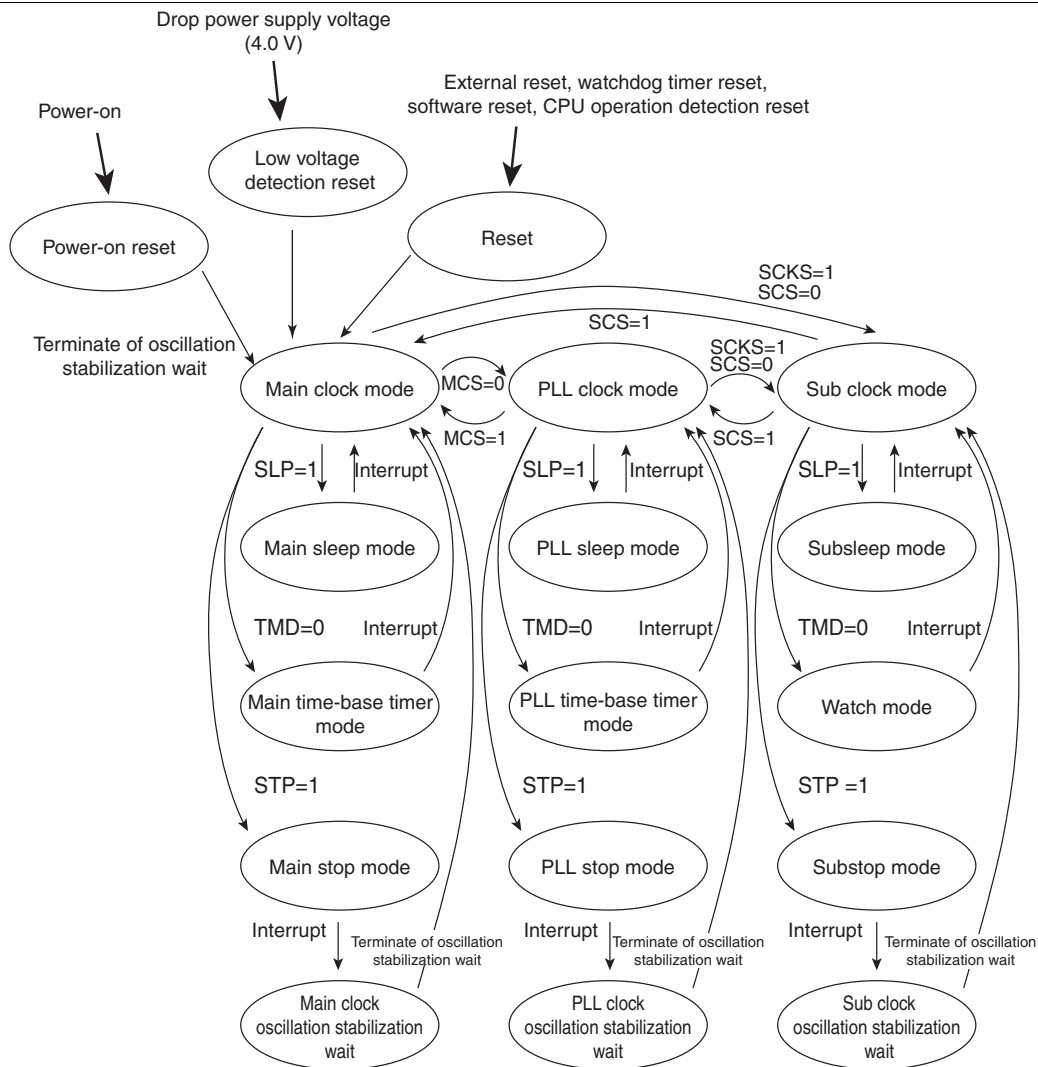
- When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which the stop mode has been specified. The CPU then proceeds to interrupt processing.
When transiting to the PLL stop mode, set the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) 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 in the clock selection register (CKSCR: WS1, WS0). The oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) 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 in the clock selection register (CKSCR: WS1, WS0) to "10_B" or "11_B".

8.6 Status Change in Standby Mode

Figure 8-7. shows the operation status and status transition in the clock mode and standby mode of the MB90990 series.

■ Status Change Diagram

Figure 8-7. Status Change Diagram



8.7 Status of Pins in Standby Mode and during Reset

The status of I/O pins in the standby mode and during reset are described for each access mode.

■ Status of I/O Pins (Single-chip Mode)

Table 8-6. Status of I/O Pins (Single-chip Mode)

Pin Name	At sleep	At stop/watch/time-base timer ^{*6}		At reset
		SPL=0	SPL=1	
P27 to P20 P44, P43, P41, P40 P53 to P50 P67 to P60 P87 to P85, P83	Immediately preceding state held ^{*2}	Input cut off ^{*4} /immediately preceding state held ^{*2}	Input cut off ^{*4} /output Hi-Z ^{*5}	Input disabled ^{*3} /output Hi-Z ^{*5}
P42 ^{*7} P54 to P57 P80, P82, P84 ^{*7}		Input enabled ^{*1}		

*1: Input enabled means that input function can be used. When the pin is set as input port, handle the pull-up/pull-down or input the external signal. When the pin is set as output port, the pin is set to the same state as other pins.

*2: Indicates that either the output pins output their state as it is immediately before entering each standby mode or the input pins are input-disabled. Output of the output state as it is means that when the resource with an output is in operation, the state of pins is output according to the state of the resource and, when the state of output pins is output, it is held.

*3: Input disabled means that no pin value can be accepted internally because the internal circuit is off while the operation of the input gates of pins is enabled.

*4: In input cut off status, input data is masked. When selecting CMOS/Automotive, "L" level is transmitted to the inside. When selecting TTL, "H" level is transmitted to the inside.

*5: Output Hi-Z means that the driving of pin driving transistors is disabled to place the pins in a high impedance state.

*6: In these modes, the pull-up function of Port2 is invalid.

*7: In stop/watch/time-base timer modes, input is enabled when the INTxR bit of the external interrupt source select register (EISSR) is "1" and the DTP/external interrupt is enabled (ENIR:EN=1). In case of the other settings, input data is masked. When selecting CMOS/Automotive, "L" level is transmitted to the inside. When selecting TTL, "H" level is transmitted to the inside.

Note:

To set that pin to high impedance which serves either for a peripheral resource or as a port in stop mode, watch mode, or time-base timer mode, disable the output of the peripheral resource, then set the STP bit to "1" or set the TMD bit to "0" in the low-power consumption mode control register (LPMCR).

8.8 Notes on Using Low-power Consumption Mode

This section explains the notes on using the low-power consumption modes.

■ Transition to Standby Mode

When an interrupt request is generated from the resource to the CPU, the mode does not transit to each standby mode even after setting the STP and SLP bits to "1" and the TMD bit to "0" in the low-power consumption mode control register (LPMCR) (and also even after interrupt processing).

If the CPU is servicing an interrupt, the interrupt-service-time interrupt request flag is cleared and the CPU can enter the standby mode unless any other interrupt request has been generated.

■ Notes on the Transition to Standby Mode

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 time-base timer mode, use the following procedure:

1. Disable the output of peripheral functions.
2. Set the SPL bit to "1", STP bit to "1", or TMD bit to "0" in the low-power consumption mode control register (LPMCR).

■ Cancellation of Standby Mode by Interrupt

When an interrupt request higher than the interrupt level (IL) of 7 is generated from the resource and external interrupt during operation in the sleep mode, watch mode, time-base timer mode, or stop mode, the standby mode is canceled. The standby mode is canceled by an interrupt regardless of whether the CPU accepts interrupts or not.

Note:

To prevent the CPU from causing a branch to interrupt servicing immediately after returning from standby mode, take measures, such as disabling interrupts before setting the standby mode.

■ **Note on Canceling Standby Mode**

The standby mode can be canceled by an input according to the settings of an input source of an external interrupt. The input source can be selected from "H" level, "L" level, rising edge, and falling edge.

■ **Oscillation Stabilization Wait Time**

- Oscillation stabilization wait time of main clock

In the sub clock mode, watch mode, or stop mode, the oscillation of the main clock stops and the oscillation stabilization wait time of the main clock is required. The oscillation stabilization wait time of the main clock is set by the WS1 and WS0 bits in the clock selection register (CKSCR).

- Oscillation stabilization wait time of sub clock

In the sub-stop mode, the oscillation of the sub clock (SCLK) stops and the oscillation stabilization wait time of the sub clock is required. The oscillation stabilization wait time of the sub clock is fixed at $2^{14}/\text{SCLK}$ (SCLK: sub clock).

- 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 time for main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0). The oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) must be selected accordingly to account for the longer of the 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 in the clock selection register (CKSCR: WS1, WS0) 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 time for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0). The oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) 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 in the clock selection register (CKSCR: WS1, WS0) 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).

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

■ 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 8-2](#).

The standby mode transition instruction in [Table 8-2](#) must always be followed by an instructions highlighted by a line below.

MOV LPMCR, #H' xx ; The low-power consumption mode transition instruction in [Table 8-2](#).

```

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

```

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

- To access the low-power consumption mode control register (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 time-base 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("NOP");
__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 STP bit to "1" */
    NOP
    NOP
    JMP $+3 /* Jump to the next instruction */
#pragma endasm

```


9. Memory Access Modes



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

9.1 Outline of Memory Access Modes

9.1 Outline of Memory Access Modes

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

■ Outline of Memory Access Modes

Table 9-1. Mode Pin and Mode

Operating mode	Bus mode
RUN mode	Single-chip
Flash programming	-

□ Operating mode

Operating mode means the mode for controlling the device operation status. The operating mode is specified by the MD2 to MD0 mode setting pins and the M1 and M0 bits in mode data. By selecting an operating mode, normal operation activation or flash memory 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 MD2 to MD0 mode setting pins and the M1 and M0 bits in mode data. The MD2 to MD0 mode setting pins specifies the bus mode for reading the reset vector and mode data, and the M1 and M0 bits in mode data specifies the bus mode for normal operation.

□ RUN mode

RUN mode means the CPU operating mode. The RUN mode has the main clock mode, PLL clock mode, and various low-power consumption mode. See "8. Low Power Consumption Mode" for details.

9.1.1 Mode Pins

Table 9-2. lists the operations that can be specified by combining the three external pins MD2 to MD0.

■ Mode Pins

Table 9-2. Mode Pin and Mode

Mode pin setting			Mode name	Reset vector access area	External data bus width	Remarks
MD2	MD1	MD0				
0	0	0	Setting disabled			
0	0	1				
0	1	0				
0	1	1	Internal vector mode	Internal	(Mode data)	Reset sequence and the following processing are controlled by mode data.
1	0	0	Setting disabled			
1	0	1				
1	1	0	Flash serial programming*	-	-	-
1	1	1	Flash memory	-	-	Mode when parallel writer is used

*:The serial programming of the flash memory cannot be written only by setting the mode pin. Other pin also need to be set. See "25. Examples of Serial Programming Connection for Flash Memory Products" for details.

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. Configuration of Mode Data

bit	7	6	5	4	3	2	1	0
Address: FFFFDF _H	M1	M0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

[bit7, bit6] M1, M0 (bus mode setting bits)

M1 and M0 bits are used to specify the operating mode after the reset sequence is completed. Table 9-3. shows the relationship between M1 and M0 bits and functions.

Table 9-3. Function of M1, M0 (Bus Mode Setting Bits)

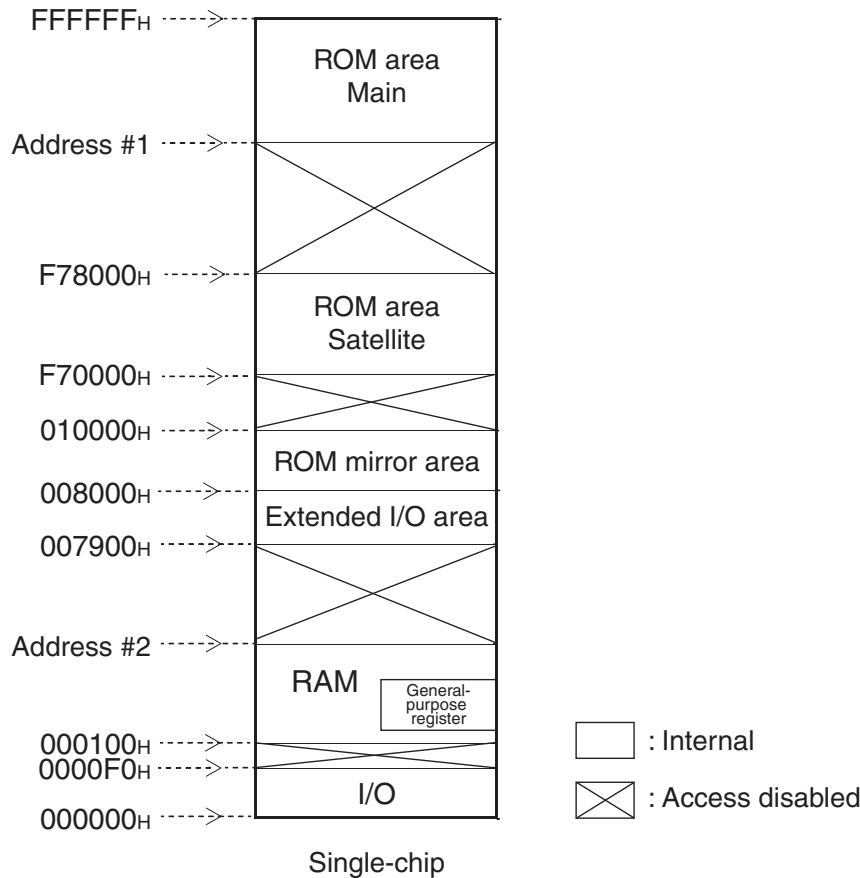
M1	M0	Function
0	0	Single-chip mode
0	1	Setting disabled
1	0	
1	1	

9.1.3 Memory Space in Each Bus Mode

Figure 9-2. shows the correspondence between the access areas and physical addresses for each bus mode.

■ Memory Space in Each Bus Mode

Figure 9-2. Relationship between Access Areas and Physical Addresses for Each Bus Mode



Product type	Address #1	Address #2
MB90F997JBS, MB90F997AMBS	FE0000 _H	0020FF _H
MB90V950AJAS, MB90V950AMAS	F80000 _H	007900 _H

■ Recommended Setting

Table 9-4. lists an example of recommended settings for mode pins and mode data.

Table 9-4. Recommended Setting Example of Mode Pin and Mode Data

Setting example	MD2	MD1	MD0	M1	M0
Single-chip	0	1	1	0	0

10. I/O Ports



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

10.1 I/O Ports

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 port direction register (DDR) 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 port data register value is read. The above also applies to a read operation for the read-modify-write (RMW) instructions.

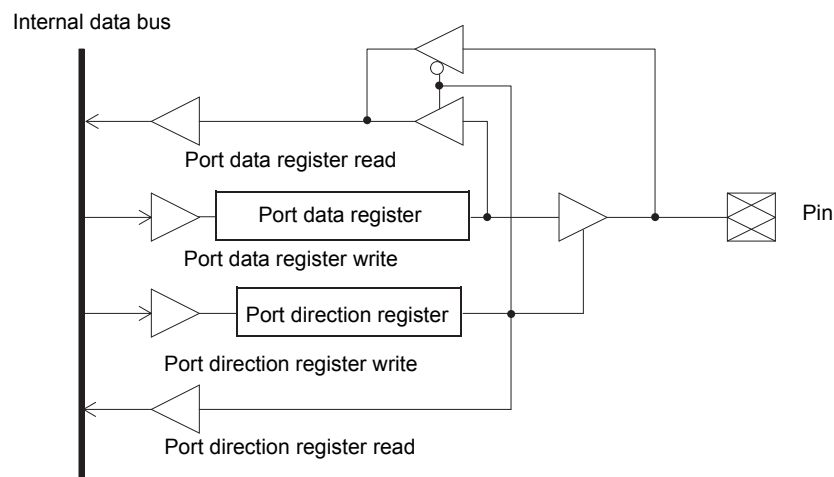
■ Overview of I/O Ports

When a pin is used as an output of other peripheral function, the logic level at the pin is read regardless of the port data register value.

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

Figure 10-1. is a block diagram of the I/O ports.

Figure 10-1. I/O Port Block Diagram



10.2 I/O Port Registers

There are five types of I/O port registers:

- Port data register (PDR2, PDR4 to PDR6, PDR8)
- Port direction register (DDR2, DDR4 to DDR6, DDR8, DDRA)

- Pull-up control register (PUCR2)
- Analog input enable register (ADER5, ADER6)
- Input level select register (ILSR0, ILSR1)

■ I/O Port Registers

Figure 10-2. shows the I/O port registers.

Figure 10-2. I/O Port Registers

	bit 7	6	5	4	3	2	1	0	
Address: 000002 _H	P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address: 000004 _H	–	–	–	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address: 000005 _H	P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address: 000006 _H	P67	P66	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address: 000008 _H	P87	P86	P85	P84	P83	P82	–	P80	Port 8 data register (PDR8)
	bit 7	6	5	4	3	2	1	0	
Address: 000012 _H	D27	D26	D25	D24	D23	D22	D21	D20	Port 2 direction register (DDR2)
Address: 000014 _H	–	–	–	D44	D43	D42	D41	D40	Port 4 direction register (DDR4)
Address: 000015 _H	D57	D56	D55	D54	D53	D52	D51	D50	Port 5 direction register (DDR5)
Address: 000016 _H	D67	D66	D65	D64	D63	D62	D61	D60	Port 6 direction register (DDR6)
Address: 000018 _H	D87	D86	D85	D84	D83	D82	–	D80	Port 8 direction register (DDR8)
Address: 00001A _H	–	–	–	SIL1	SIL0	–	–	–	Port A direction register (DDRA)
	bit 7	6	5	4	3	2	1	0	
Address: 00001E _H	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20	Port 2 pull-up control register (PUCR2)
	bit 15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00000B _H	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8	Port 5 analog input enable register (ADER5)
Address: 00000C _H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 6 analog input enable register (ADER6)
	bit 15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00000E _H	–	IL6	IL5	IL4	–	IL2	–	–	Input level select register (ILSR0)
Address: 00000F _H	–	–	–	–	–	–	–	IL8	Input level select register (ILSR1)

10.2.1 Port Data Register (PDR)

Note that R/W for I/O ports differ from Read/Write 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 port data register latch value is read.
 - Write: Data is written to an output latch and outputted to the corresponding pin.

Figure 10-3. shows the detailed bit configuration of port data registers (PDR).

■ Port Data Register (PDR)

Figure 10-3. Port Data Registers (PDR)

	bit7	6	5	4	3	2	1	0	Initial value	Access
PDR2 Address: 000002 _H	P27	P26	P25	P24	P23	P22	P21	P20	XXXXXXXX _B	R/W
PDR4 Address: 000004 _H	—	—	—	P44	P43	P42	P41	P40	XXXXXXXX _B	R/W
PDR5 Address: 000005 _H	P57	P56	P55	P54	P53	P52	P51	P50	XXXXXXXX _B	R/W
PDR6 Address: 000006 _H	P67	P66	P65	P64	P63	P62	P61	P60	XXXXXXXX _B	R/W
PDR8 Address: 000008 _H	P87	P86	P85	P84	P83	P82	—	P80	XXXXXXXX _B	R/W

□ Reading the port data register

The value obtained when reading the port data register (PDR) depends on the status of the port direction register (DDR) and status of the peripheral function connected to the pin.

The following shows the value obtained by each combination.

Value of DDR	Output state of peripheral function	Reading value
0 (input)	Enabled	Output value from peripheral function
1 (output)	Enabled	Output value from peripheral function
0 (input)	Disabled	Pin state
1 (output)	Disabled	Value of output latch

Further, when using as input by peripheral function, set the DDR of the connected pin to "0" (input).

10.2.2 Port Direction Register (DDR)

This register has the following functions:

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

■ Port Direction Register (DDR)

Figure 10-4. shows the port direction registers (DDR).

Figure 10-4. Port Direction Registers (DDR)

	bit7	6	5	4	3	2	1	0	Initial value	Access
DDR2										
Address: 000012 _H	D27	D26	D25	D24	D23	D22	D21	D20	00000000 _B	R/W
DDR4										
Address: 000014 _H	—	—	—	D44	D43	D42	D41	D40	00000000 _B	R/W
DDR5										
Address: 000015 _H	D57	D56	D55	D54	D53	D52	D51	D50	00000000 _B	R/W
DDR6										
Address: 000016 _H	D67	D66	D65	D64	D63	D62	D61	D60	00000000 _B	R/W
DDR8										
Address: 000018 _H	D87	D86	D85	D84	D83	D82	—	D80	00000000 _B	R/W
DDRA										
Address: 00001A _H	—	—	SIL2	SIL1	SIL0	—	—	—	00000111 _B	W

Bits Dxx (DDR2, DDR4 to DDR6, DDR8)

These bits set to the I/O direction of the port. When each pin is used as port, the corresponding pin is controlled as mentioned below.

When set to "0": The corresponding pin is set to input mode.

When set to "1": The corresponding pin is set to output mode.

Bit SIL0, SIL1, SIL2 (DDRA)

These bits set the input level of the corresponding SIN (serial data input for LIN-UART) pin forcibly. SIL0, SIL1 and SIL2 correspond to SIN0 (LIN-UART0), SIN1 (LIN-UART1) and SIN2 (UART2), respectively.

When set to "0": CMOS or Automotive is selected for the input level depending on the setting of the corresponding ILx bit in the ILSR. (See Section "10.2.5 Input Level Select Register (ILSR0, ILSR1)" for ILSR.)

When set to "1": CMOS is selected for the input level regardless of the setting of the corresponding ILx bit in ILSR.

The initial value of these bits is "0".

Table 10-1. SIN0/SIN1 Input Level Setting

DDRA	ILSR1	SIN0(P82) / SIN1(P85) input level
SIL0/SIL1 bit	IL8 bit	
0	0	Automotive level
0	1	CMOS level (0.8V _{CC} /0.2V _{CC})
1	x	CMOS level (0.7V _{CC} /0.3V _{CC})

Table 10-2. SIN2 Input Level Setting

DDRA	ILSR0	SIN2(P50) input level
SIL2 bit	IL5 bit	
0	0	Automotive level
0	1	CMOS level ($0.8V_{CC}/0.2V_{CC}$)
1	x	CMOS level ($0.7V_{CC}/0.3V_{CC}$)

Note:

SIL0, SIL1 and SIL2 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.

- DDRA: bit0 to bit2, bit6, bit7 (Undefined bits)

"1" is always read from these bits.

Writing to these bits is no effect.

10.2.3 Pull-up Control Register (PUCR2)

Each pin of port2 has programmable pull-up resistor. Each bit of this register controls corresponding pull-up resistor whether to be used or not.

Figure 10-5. shows the pull-up control register (PUCR2), and Figure 10-6. is the block diagram.

■ Pull-up Control Register (PUCR2)

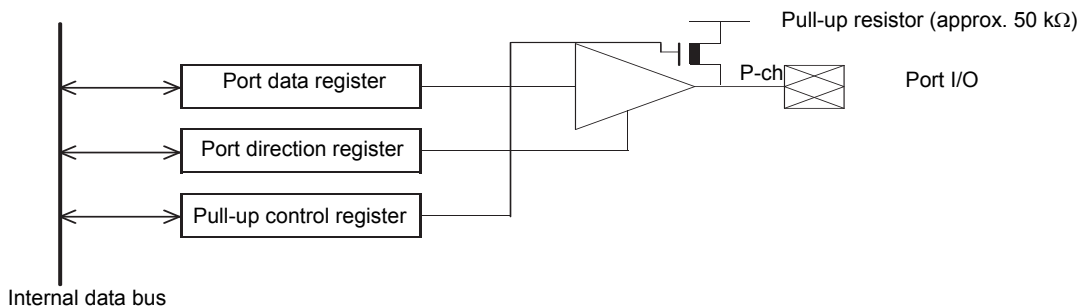
Figure 10-5. Pull-up Control Register (PUCR2)

PUCR2	bit 7	6	5	4	3	2	1	0	Initial value	Access
Address : 00001E _H	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20	00000000 _B	R/W

R/W: Readable/Writable

■ Block Diagram of Pull-up Control Register (PUCR2)

Figure 10-6. Block Diagram of Pull-up Control Register (PUCR2)



In input mode, the pull-up resistor is controlled.

When set to "0": No pull-up resistor in input mode

When set to "1": Pull-up resistor in input mode

Note:

In output mode, this register has no meaning (no pull-up resistor).

The port direction register (DDR) determines the input-output mode.

In stop mode (SPL=1), the state with no pull-up resistor is entered (high impedance).

10.2.4 Analog Input Enable Register (ADER)

Figure 10-7. shows the analog input enable register (ADER).

■ Analog Input Enable Registers (ADER)

Figure 10-7. Analog Input Enable Registers (ADER6, ADER5)

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

R/W: Readable/Writable

Each bit of ADER6/ADER5 sets to enable/disable the analog input of each pin in the port 6 and port 5. ADER6 and ADER5 correspond to the port 6 and port 5, respectively.

When set to "0": The corresponding pin is set to disable the analog input. A pin set to disable the analog input can be used as I/O pin for peripheral function other than I/O port and A/D converter.

When set to "1": The corresponding pin is set to the analog input mode. A pin set to the analog input mode is the dedicated analog input pin for the A/D converter. The pin cannot be used as I/O pin of I/O port and other peripheral function.

Note:

When the analog input enable bit (ADE_x) is set to "1", each pin of the port 6 and port 5 is the analog input pin for the A/D converter. Initial value of the ADE_x bit is "1". Therefore, the corresponding pins cannot be used as I/O pin of peripheral function other than I/O port and A/D converter at the initial setting. When the pin is used as I/O pin of other peripheral function and I/O port, the ADE_x bit is set to "0".

10.2.5 Input Level Select Register (ILSR0, ILSR1)

The input level select register allows to switch from Automotive Hysteresis input levels to CMOS Hysteresis input levels or to the TTL input level.

■ Input Level Select Register (ILSR0, ILSR1)

Figure 10-8. Input Level Select Register (ILSR1, ILSR0)

ILSR1								
Address	bit15	14	13	12	11	10	9	8
00000FH	–	–	–	–	–	–	–	IL8
Initial value :	–	–	–	–	–	–	–	R/W
	X	X	X	X	X	X	X	0/1
ILSR0								
Address	bit7	6	5	4	3	2	1	0
00000EH	–	IL6	IL5	IL4	–	IL2	–	–
Initial value :	–	R/W	R/W	R/W	–	R/W	–	–
	X	0/1	0/1	0/1	X	0/1	X	X

R/W: Readable/Writable
 –: Unused
 X: Undefined

bit2, bit4 to bit6 and bit8:IL2, IL4 to IL6, IL8

These bits set the input level of the corresponding port. IL2, IL4 to IL6, IL8 correspond to the port 2, port 4 to port 6, port 8, respectively.

When set to "0": Automotive input level is selected.

When set to "1": CMOS hysteresis input level is selected.

The initial value of these bits depends on the operation mode (mode pin) setting:

- For the flash memory mode, the initial value is "1" (TTL input).
- For all other modes, the initial value is "0" (Automotive).

bit0, bit1, bit3, bit7 and bit9 to bit15: Undefined bits

"0" is always read from these bits. Writing to these bits is no effect.

Note:

The threshold of the corresponding input pin varies immediately after the setting of the input level select register is changed.

Therefore, do not use the read value from the pin until 2 machine cycles are elapsed after the setting is changed.

When the setting is changed, be sure to disable the corresponding resource.

■ Initial Value of Input Level Select Register (ILSR0, ILSR1)

Initial value of each bit of ILSR is determined when external reset signal is released depending on the value of MD2, MD1 and MD0 pin input, as shown in following table.

About detail of each mode, see "[9. Memory Access Modes](#)".

Table 10-3. Relationship between Mode Pin and Initial Value of Input Level Select Register (ILSR)

MD2	MD1	MD0	Operation mode	Initial value	Port input level
				ILx	Port 2, Port 4 to Port 6, Port 8
0	0	0	Reserved		
0	0	1			
0	1	0			
0	1	1	Internal vector mode	0	Automotive
1	0	0	Reserved		
1	0	1			
1	1	0	Flash serial write	0	Automotive
1	1	1	Flash memory	1	TTL

11. Time-Base Timer



This chapter explains the functions and operations of the time-base timer.

11.1 Overview of Time-base Timer

11.2 Block Diagram of Time-base Timer

11.3 Configuration of Time-base Timer

11.4 Interrupt of Time-base Timer

11.5 Explanation of Operations of Time-base Timer Functions

11.6 Notes on Using Time-base Timer

11.7 Program Example of Time-base Timer

11.1 Overview of Time-base Timer

The time-base timer is an 18-bit free-run counter (time-base timer counter) that increments in synchronization with the main clock (half frequency of main oscillation clock).

- Four interval times can be selected and an interrupt request can be generated for each interval time.
- An operation clock is supplied to the oscillation stabilization wait time timer and other peripherals.

■ Interval Timer Function

- When the time-base timer counter reaches the interval time set by the interval time select bits (TBTC: TBC1, TBC0), an overflow (carrying) occurs (TBTC: TBOF = 1) and an interrupt request is generated.
- When an interrupt is enabled when an overflow occurs (TBTC: TBIE = 1), an overflow occurs (TBTC: TBOF = 1) and an interrupt is generated.
- The time-base timer has four interval times that can be selected. [Table 11-1](#) shows the interval times of the time-base timer.

Table 11-1. Interval Times of Time-base Timer

Count clock	Interval time
2/HCLK(0.5 μ s)	2^{12} /HCLK (approx. 1.0 ms)
	2^{14} /HCLK (approx. 4.1 ms)
	2^{16} /HCLK (approx. 16.4 ms)
	2^{19} /HCLK (approx. 131.1 ms)

HCLK: Oscillation clock

The parenthesized values are provided at 4 MHz oscillation clock.

■ Clock Supply

The time-base timer supplies an operation clock to the resources such as an oscillation stabilization wait time timer, PPG timer, and watchdog timer. Table 11-2. shows the clock cycles supplied from the time-base timer to each resource.

Table 11-2. Clock Cycles Supplied from Time-base Timer

Where to supply clock	Clock cycle
Oscillation stabilization wait time*	$2^{10}/\text{HCLK}$ (approx. 256 μs)
	$2^{13}/\text{HCLK}$ (approx. 2.0 ms)
	$2^{16}/\text{HCLK}$ (approx. 16.4 ms)
	$2^{17}/\text{HCLK}$ (approx. 32.8 ms)
Watchdog timer	$2^{12}/\text{HCLK}$ (approx. 1.0 ms)
	$2^{14}/\text{HCLK}$ (approx. 4.1 ms)
	$2^{16}/\text{HCLK}$ (approx. 16.4 ms)
	$2^{19}/\text{HCLK}$ (approx. 131.1 ms)
PPG timer	$2^9/\text{HCLK}$ (approx. 128 μs)

HCLK: Oscillation clock

The parenthesized values are provided at 4 MHz oscillation clock.

*:As the oscillation cycle is unstable immediately after oscillation starts, standard oscillation stabilization wait time values are given as a guide.

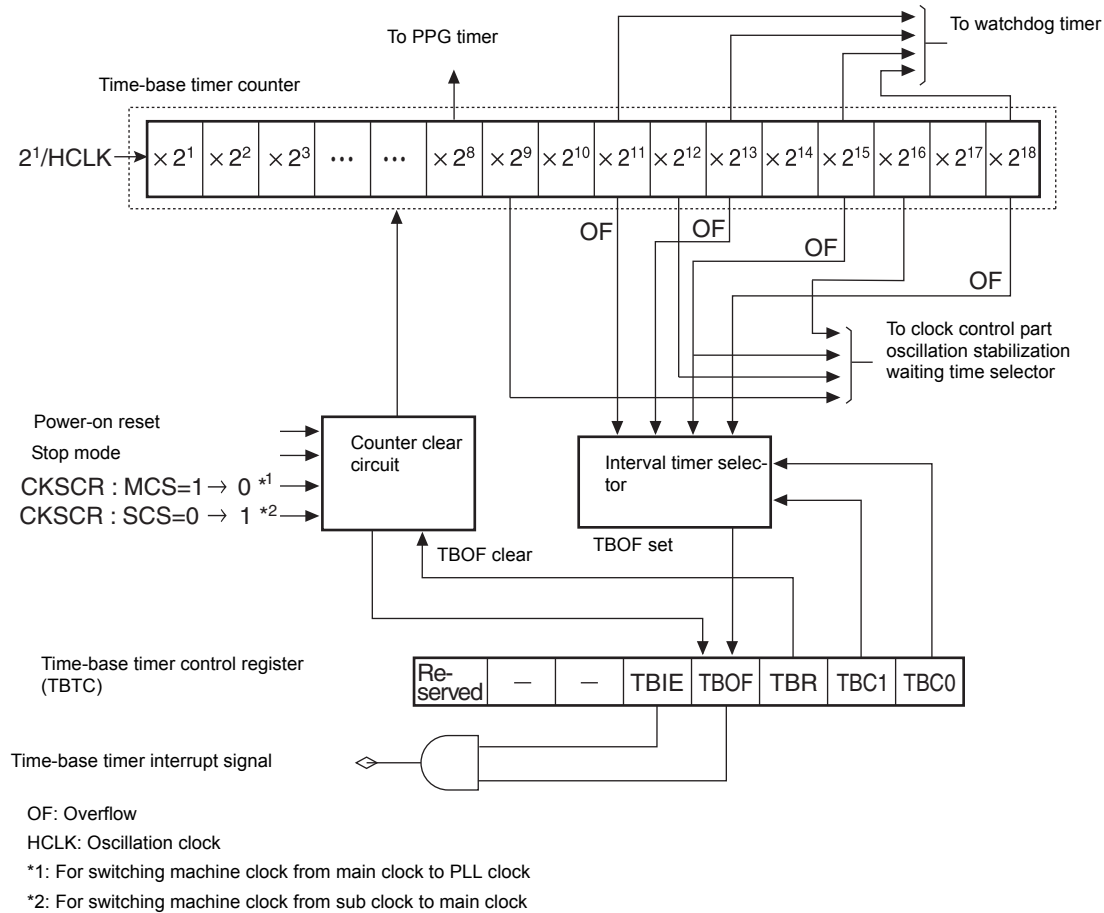
11.2 Block Diagram of Time-base Timer

The time-base timer consists of the following blocks:

- Time-base timer counter
- Counter clear circuit
- Interval timer selector
- Time-base timer control register (TBTC)

■ Block Diagram of Time-base Timer

Figure 11-1. Block Diagram of Time-base Timer



The actual interrupt request number of the time-base timer is as follows:

Interrupt request number: #25 (19_H)

□ Time-base timer counter

The time-base timer counter is an 18-bit up counter that uses a clock with a half frequency of the oscillation clock (HCLK) as a count clock.

□ Counter clear circuit

The counter clear circuit clears the value of the time-base timer counter by the following sources:

- Time-base timer counter clear bit in the time-base timer control register (TBTC: TBR=0)
- Power-on reset
- Transition to main stop mode or PLL stop mode (CKSCR:SCS=1, LPMCR: STP=1)
- Switching the clock mode (from main clock mode to PLL clock mode, from sub clock mode to PLL clock mode, or from sub clock mode to main clock mode)

□ Interval timer selector

The interval timer selector selects the output of the time-base timer counter from four types.

When incrementing causes the selected interval time bit to overflow (carrying), an interrupt request is generated.

□ Time-base timer control register (TBTC)

The time-base timer control register (TBTC) selects the interval time, clears the time-base timer counter, enables or disables interrupts, and checks and clears the state of an interrupt request.

11.3 Configuration of Time-base Timer

This section explains the registers and interrupt factors of the time-base timer.

■ List of Registers and Initial Values of Time-base Timer

Figure 11-2. List of Registers and Initial Values of Time-base Timer

Time-base timer control register (TBTC)

bit	15	14	13	12	11	10	9	8
	1	1	1	0	0	1	0	0

■ Generation of Interrupt Request from Time-base Timer

When the selected interval timer counter bit reaches the interval time, the overflow interrupt request flag bit in the time-base timer control register (TBTC: TBOF) is set to "1". If the overflow interrupt request flag bit is set (TBTC: TBOF = 1) when the interrupt is enabled (TBTC: TBIE = 1), the time-base timer generates an interrupt request.

11.3.1 Time-base Timer Control Register (TBTC)

The time-base timer control register (TBTC) provides the following settings:

- Selecting the interval time of the time-base timer
- Clearing the counter value of the time-base timer
- Enabling or disabling the interrupt request when an overflow occurs
- Checking and clearing the state of the interrupt request flag when an overflow occurs

■ Time-base Timer Control Register (TBTC)

Figure 11-3. Time-base Timer Control Register (TBTC)

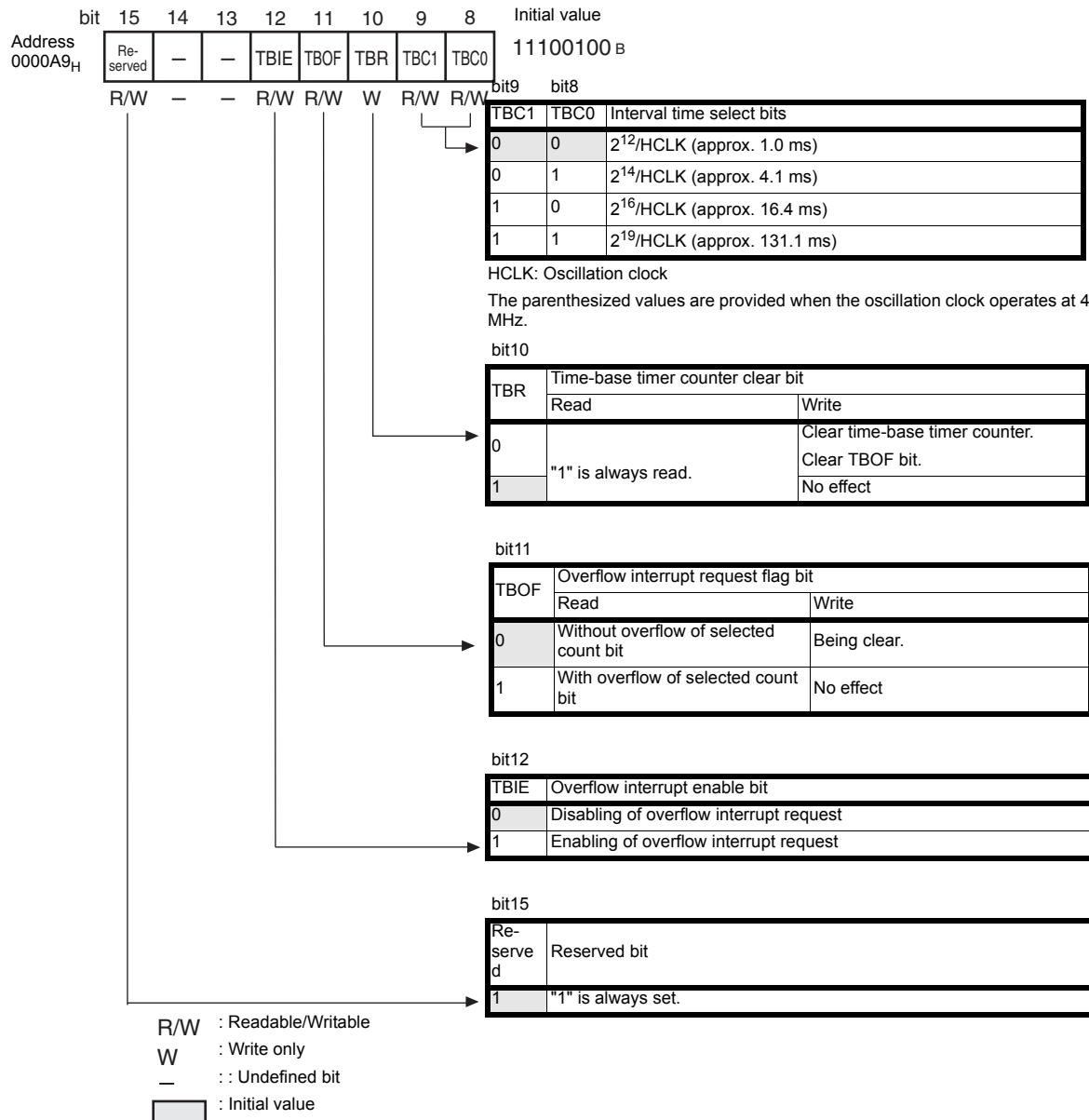


Table 11-3. Functions of Time-base Timer Control Register (TBTC)

Bit name		Function
bit15	Reserved: reserved bit	Always set this bit to "1".
bit14, bit13	Undefined bits	Read: The value is undefined. Write: No effect
bit12	TBIE: Overflow interrupt enable bit	This bit enables or disables an interrupt when the interval timer bit in the time-base timer counter overflows. When set to "0": No interrupt request is generated at an overflow (TBOF = 1). When set to "1": An interrupt request is generated at an overflow (TBOF = 1).
bit11	TBOF: Overflow interrupt request flag bit	This bit indicates an overflow (carrying) in the interval timer bit in the time-base timer counter. When an overflow (carrying) occurs (TBOF = 1) with interrupts enabled (TBIE = 1), an interrupt request is generated. When set to "0": The bit is cleared. When set to "1": Disabled. The state remains unchanged. Read by read-modify-write (RMW) instructions: "1" is read. Notes: <ul style="list-style-type: none"> ■ To clear the TBOF bit, disable interrupts (TBIE = 0) or mask interrupts using the interrupt mask register (ILM) in the processor status. ■ The TBOF bit is cleared at a write of "0", transition to main stop mode or to PLL stop mode, transition from sub clock mode to main clock mode or to PLL clock mode, transition from main clock mode to PLL clock mode, at a write of "0" to the time-base timer counter clear bit (TBR), or at reset.
bit10	TBR: Time-base timer counter clear bit	This bit clears all the bits in the time-base timer counter. When set to "0": All the bits in the time-base timer counter are cleared to "0". The TBOF bit is also cleared. When set to "1": Disabled. The state remains unchanged. Read: "1" is always read.
bit9, bit8	TBC1, TBC0: Interval time select bits	These bits set the cycle of the interval timer in the time-base timer counter. <ul style="list-style-type: none"> ■ The interval time of the time-base timer is set according to the setting of the TBC1 and TBC0 bits. ■ One of four time intervals can be selected.

11.4 Interrupt of Time-base Timer

The time-base timer generates an interrupt request (interval timer function) when the interval time bit in the time-base timer counter corresponding to the interval time set by the time-base timer control register carries (overflows).

■ Interrupt of Time-base Timer

- ❑ The time-base timer continues incrementing for as long as the main clock (with a half frequency of the oscillation clock) is inputted.
- ❑ When the interval time set by the interval time select bits in the time-base timer control register (TBTC: TBC1, TBC0) is reached, the interval time select bit corresponding to the interval time selected in the time-base timer counter carries and an overflow generates.
- ❑ When the interval time select bit overflows, the overflow interrupt request flag bit in the time-base timer control register (TBTC: TBOF) is set to "1".
- ❑ When the overflow interrupt request flag bit in the time-base timer control register is set (TBTC: TBOF = 1) with an interrupt enabled (TBTC: TBIE = 1), an interrupt request is generated.
- ❑ When the selected interval time is reached, the overflow interrupt request flag bit in the time-base timer control register (TBTC: TBOF) is set regardless of whether an interrupt is enabled or disabled (TBTC: TBIE)
- ❑ To clear the overflow interrupt request flag bit (TBTC: TBOF), disable a time-base timer interrupt at interrupt processing (TBTC: TBIE = 0) or mask a time-base timer interrupt by using the ILM bit in the processor status (PS) to write "0" to the TBOF bit.

Note:

An interrupt request is issued immediately if you enable interrupts (TBTC:TBIE = 1) with the overflow interrupt request flag bit in the time-base timer control register set (TBTC:TBOF = 1).

■ Correspondence Between Time-base Timer Interrupt and EI²OS

- The time-base timer does not correspond to EI²OS.
- For details of the interrupt number, interrupt control register, and interrupt vector address, see Section "3.2 Interrupt Vector".

11.5 Explanation of Operations of Time-base Timer Functions

The time-base timer operates as an interval timer or an oscillation stabilization wait time timer. It also supplies a clock to peripherals.

■ Interval Timer Function

Interrupt generation at every interval time enables the time-base timer to be used as an interval timer.

Operating the time-base timer as an interval timer requires the settings shown in Figure 11-4. .

- Setting of time-base timer

Figure 11-4. Setting of Time-base Timer

Time-base timer control register (TBTC)		bit15	14	13	12	11	10	9	bit8
Address: 0000A9 _H		Re-served	—	—	TBIE	TBOF	TBR	TBC1	TBC0
		1			⊙	0	0	⊙	⊙

— : Undefined bit
 ⊙ : Used bit
 0 : Set to "0".
 1 : Set to "1".

- Operations of the Interval Timer Functions

The time-base timer can be used as an interval timer by generating an interrupt at every set interval time.

- The time-base timer continues incrementing in synchronization with the main clock (a half frequency of the oscillation clock) while the oscillation clock is active.
- When the time-base timer counter reaches the interval time set by the interval time select bits in the time-base timer control register (TBTC: TBC1, TBC0), it causes an overflow (carrying) and the overflow interrupt request flag bit (TBTC: TBOF) is set to "1".
- When the overflow interrupt request flag bit is set (TBTC: TBOF = 1) with interrupts enabled (TBTC: TBIE = 1), an interrupt request is generated.

Note:

The interval time may be longer than the one set by clearing the time-base timer counter.

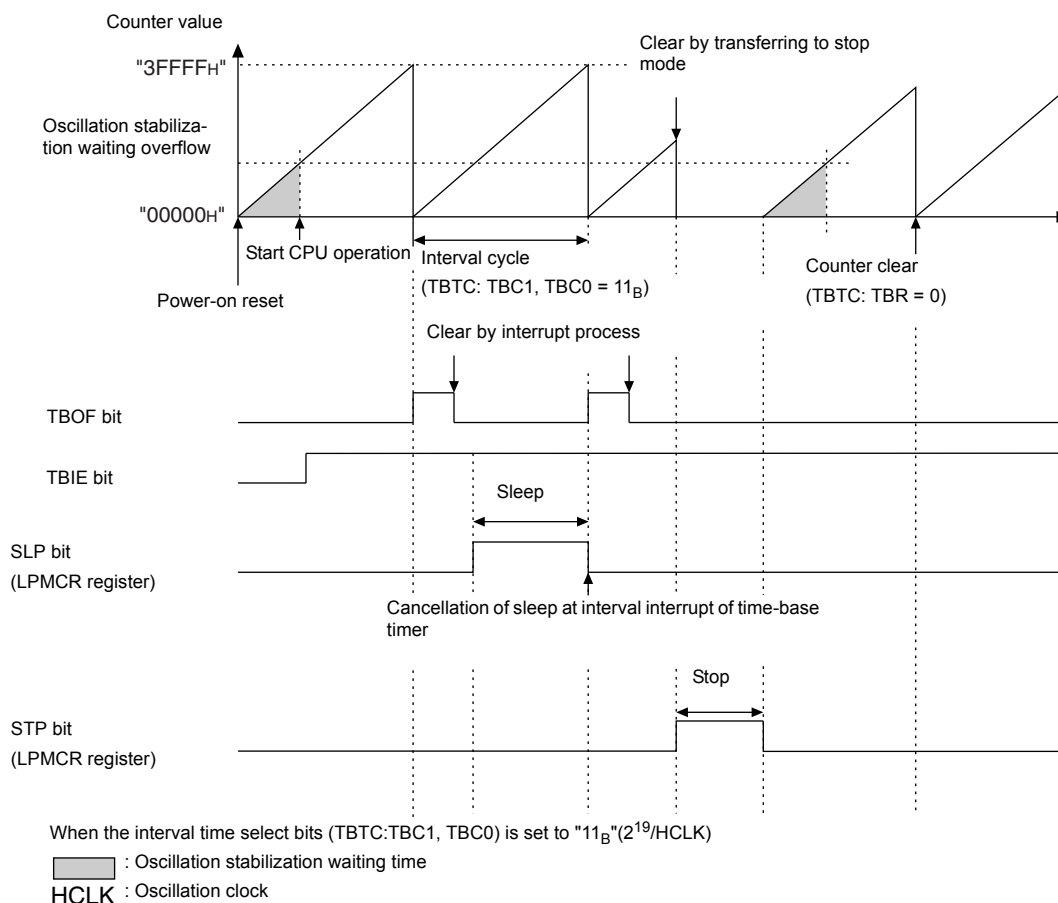
□ Example of operation for time-base timer

Interval timer operations in the following states are shown in [Figure 11-5](#).

- A power-on reset occurs.
- The mode transits to the sleep mode during the operation of the interval timer.
- The mode transits to the stop mode during the operation of the interval timer.
- A request to clear the time-base timer counter is issued.

At transition to the stop mode, the time-base timer counter is cleared to stop counting up. At return from the stop mode, the time-base timer counts the oscillation stabilization wait time of the main clock.

Figure 11-5. Example of Operation for Time-base Timer



■ Operation as Oscillation Stabilization Wait Time Timer

The time-base timer can be used as the oscillation stabilization wait time timer for the main clock and PLL clock.

The oscillation stabilization wait time is the time elapsed from when the time-base timer counter increments from "0" until the set oscillation stabilization wait time select bit overflows (carrying).

Table 11-4. shows clearing conditions and oscillation stabilization wait time of time-base timer.

Table 11-4. Clearing Conditions and Oscillation Stabilization Wait Time of Time-base Timer (1/2)

Operation	Counter clear	TBOF clear	Oscillation stabilization wait time
Writing "0" to time-base timer counter clear bit (TBTC: TBR)	○	○	¾
Reset			
Power-on reset	○	○	Transition to main clock mode after oscillation stabilization wait time of main clock completed
Watchdog reset	×	○	None
External reset Low voltage detection reset CPU operation detection reset Clock supervisor reset	×	○	None
Software reset	×	○	None
Switching clock mode			
Main clock → PLL clock (CKSCR: MCS=1 → 0)	○	○	Transition to PLL clock mode after oscillation stabilization wait time of PLL clock completed
Main clock → sub clock (CKSCR: SCS=1 → 0)	×	×	Transition to sub clock mode after oscillation stabilization wait time of sub clock completed
Sub clock → main clock (CKSCR: SCS=0 → 1)	○	○	Transition to main clock mode after oscillation stabilization wait time of main clock completed
Sub clock → PLL clock (CKSCR: MCS=0, SCS=0 → 1)	○	○	Transition to PLL clock mode after oscillation stabilization wait time of main clock completed
PLL clock → main clock (CKSCR: MCS=0 → 1)	×	×	None
PLL clock → sub clock (CKSCR: MCS=0, SCS=1 → 0)	×	×	None

Table 4.0-0 Clearing Conditions and Oscillation Stabilization Wait Time of Time-base Timer (2/2)

Operation	Counter clear	TBOF clear	Oscillation stabilization wait time
Cancellation of stop modes			
Cancellation of main stop mode	○	○	Transition to PLL clock mode after oscillation stabilization wait time of main clock completed
Cancellation of PLL stop mode	○	○	Transition to PLL clock mode after oscillation stabilization wait time of main clock completed
Cancellation of sub-stop mode	×	×	Transition to sub clock mode after oscillation stabilization wait time of sub clock completed
Cancellation of watch mode			
Cancellation of sub-watch mode	×	×	None
Cancellation of time-base timer modes			
Return to main clock mode	×	×	None
Return to sub clock mode	×	×	None
Return to PLL clock mode	×	×	None
Cancellation of sleep modes			
Cancellation of main sleep mode	×	×	None
Cancellation of sub-sleep mode	×	×	None
Cancellation of PLL sleep mode	×	×	None

■ Supply of Operation Clock

The time-base timer supplies an operation clock to the PPG timers and the watchdog timer.

Note:

Clearing the time-base timer counter may affect the operation of the resources such as the watchdog timer and PPG timers using the output of the time-base timer.

References:

- For details on the PPG timer, see "[16. 8-/16-Bit PPG Timer](#)".
- For details on the watchdog timer, see "[12. Watchdog Timer](#)".

11.6 Notes on Using Time-base Timer

Notes on using the time-base timer are shown below.

■ Notes on Using Time-base Timer

- Clearing interrupt request

To clear the overflow interrupt request flag bit in the time-base timer control register (TBTC: TBOF = 0), disable interrupts (TBTC: TBIE = 0) or mask the time-base timer interrupt by using the interrupt level mask register in the processor status.

- Clearing time-base timer counter

Clearing the time-base timer counter affects the following operations:

- When the time-base timer is used as the interval timer (interval interrupt).
- When the watchdog timer is used.
- When the clock supplied from the time-base timer is used as the operation clock of the PPG timer.
- Using time-base timer as oscillation stabilization wait time timer

After power on or in the main stop mode, PLL stop mode, and sub clock mode, the oscillation clock stops. Therefore, when oscillation starts, the time-base timer requires the oscillation stabilization wait time of the main clock. An appropriate oscillation stabilization wait time must be selected according to the types of oscillators connected to high-speed oscillation input pins.

Reference:

For details on the oscillation stabilization wait time, see Section "5.6 Oscillation Stabilization Wait Time".

- Resources to which time-base timer supplies clock
- At transition to operation modes (PLL stop mode, sub clock mode, and main stop mode) in which the oscillation clock stops, the time-base timer counter is cleared and the time-base timer stops.
- When the time-base timer counter is cleared, an after-clearing interval time is needed. It may cause the clock supplied from the time-base timer to have a short High level or a 1/2 cycle longer Low level.
- The watchdog timer performs normal counting because the watchdog timer counter and time-base timer counter are cleared simultaneously.

11.7 Program Example of Time-base Timer

Programming examples for the time-base timer are shown below.

■ Program Example of Time-base Timer

- Processing specification

The $2^{12}/\text{HCLK}$ (HCLK: oscillation clock) interval interrupt is generated repeatedly. In this case, the interval time is approximately 1.0 ms (at 4 MHz operation).

- Coding example

```
ICR07 EQU 0000B7H ;Time-base timer interrupt control register
TBTC EQU 0000A9H ;Time-base timer control register
TBOF EQU TBTC:3 ;Interrupt request flag bit
TBIE EQU TBTC:2 ;Interrupt enable bit
;-----Main program-----
CODE CSEG
START: ;Stack pointer (SP), already initialized
        AND CCR,#0BFH ;Interrupt disable
        MOV I:ICR07,#00H ;Interrupt level 0(highest)
        MOV I:TBTC,#10000000B
```

```

                                ;Upper 3 bits are fixed
                                ;TBOF clear,
                                ;Counter clear interval time
                                ;212/HCLK selection
                                ;Interrupt enable
                                ;Setting ILM in PS to level 7
                                ;Interrupt enable
                                ;No limit loop
LOOP:  MOV     A,#00H
        MOV     A,#01H
        BRA     LOOP
;-----Interrupt program-----
WARI:
        CLRB    I:TBIE          ;Clear interrupt enable bit
        CLRB    I:TBOF         ;Clear interrupt request flag
        :
        User processing
        :
        SETB    I:TBIE          ;Interrupt enable
        RETI     ;Recovery from interrupt processing
CODE   ENDS
;-----Vector setting-----
VECT   CSEG     ABS=0FFH
        ORG      0FF98H         ;Vector setting to interrupt number
                                #25(19H)
        DSL      WARI
        ORG      0FFDCH         ;Reset vector setting
        DSL      START
        DB       00H           ;Setting to single-chip mode
VECT   ENDS
        END      START

```

12. Watchdog Timer



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

[12.1 Overview of Watchdog Timer](#)

[12.2 Configuration of Watchdog Timer](#)

[12.3 Watchdog Timer Registers](#)

[12.4 Explanation of Operations of Watchdog Timer Functions](#)

[12.5 Notes on Using Watchdog Timer](#)

[12.6 Program Example of Watchdog Timer](#)

12.1 Overview of Watchdog Timer

The watchdog timer is a 2-bit counter that uses the time-base timer or watch timer as a count clock. If the counter is not cleared within a set interval time, the CPU is reset.

■ Functions of Watchdog Timer

- ❑ The watchdog timer is a timer counter that is used to prevent program malfunction. When the watchdog timer is started, the watchdog timer counter must continue to be cleared within a set interval time. If the set interval time is reached without clearing the watchdog timer counter, the CPU is reset. This is called watchdog timer.
- ❑ The interval time of the watchdog timer depends on the clock cycle input as a count clock and a watchdog reset occurs between the minimum and maximum times.
- ❑ The clock source output destination is set by the watchdog clock select bit in the watch timer control register (WTC: WDSCS).
- ❑ The interval time of the watchdog timer is set by the time-base timer output select bit/watch timer output select bit in the watchdog timer control register (WDTC: WT1, WT0).

[Table 12-1.](#) lists the interval times of the watchdog timer.

Table 12-1. Interval Time of Watchdog Timer

Main clock

Clock cycle	Examples calculated	
	External clock(@4 MHz)	
	Min.	Max.
$(2^{14} \pm 2^{11})/\text{HCLK}$	Approx. 0.287s	Approx. 0.369s
$(2^{16} \pm 2^{13})/\text{HCLK}$	Approx. 2.294s	Approx. 2.949s
$(2^{18} \pm 2^{15})/\text{HCLK}$	Approx. 4.588s	Approx. 5.898s
$(2^{21} \pm 2^{18})/\text{HCLK}$	Approx. 9.175s	Approx. 11.796s

Sub clock

Clock cycle	Examples calculated	
	External clock (@100 kHz, 8-frequency division)	
	Min.	Max.
$(2^{12} \pm 2^9)/\text{SCLK}$	Approx. 0.287s	Approx. 0.369s
$(2^{15} \pm 2^{12})/\text{SCLK}$	Approx. 2.294s	Approx. 2.949s
$(2^{16} \pm 2^{13})/\text{SCLK}$	Approx. 4.588s	Approx. 5.898s
$(2^{17} \pm 2^{14})/\text{SCLK}$	Approx. 9.175s	Approx. 11.796s

Notes:

- ❑ When the time-base timer output (carry signal) is used as a count clock to the watchdog timer, clearing the time-base timer may extend the time for a watchdog reset to occur.
- ❑ When the sub clock is used as the machine clock, be sure to set the watchdog timer clock source select bit (WDSCS) in the watch timer control register (WTC) to "0" to select the watch timer output.

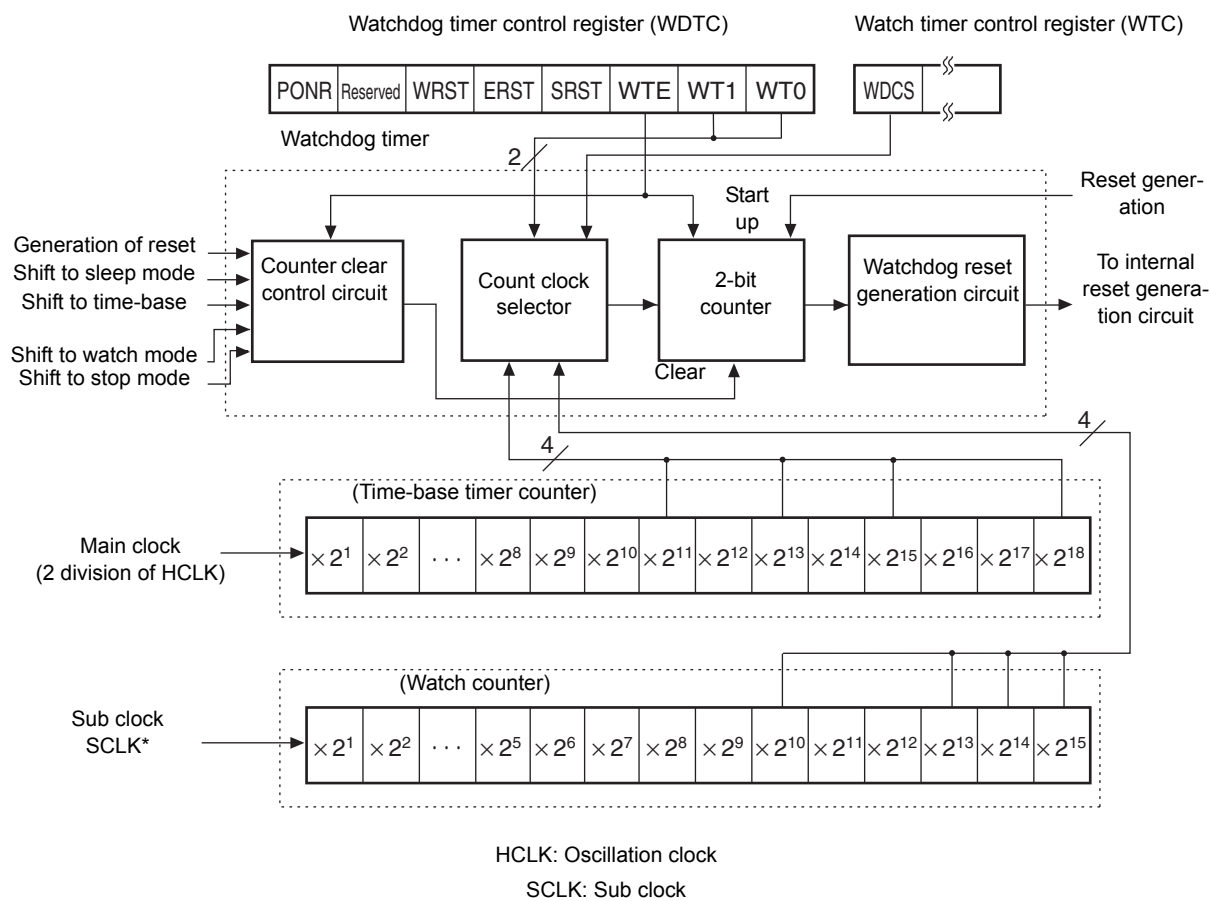
12.2 Configuration of Watchdog Timer

The watchdog timer consists of the following blocks:

- Count clock selector
- Watchdog timer counter (2-bit counter)
- Watchdog reset generator
- Counter clear control circuit
- Watchdog timer control register (WDTC)

■ Block Diagram of Watchdog Timer

Figure 12-1. Block Diagram of Watchdog Timer



*:SCLK is 8 division or 4 division of the CR oscillation clock. The division ratio is set by the SCDS bit of the PLL/sub clock control register (PSCCR). (See "5. Clocks".)

□ Count clock selector

The count clock selector selects a count clock input to the watchdog timer from the time-base timer or watch timer. Each timer output has four time intervals that can be set.

□ Watchdog timer counter (2-bit counter)

The watchdog timer counter is a 2-bit up counter that uses the time-base timer output or watch timer output as a count clock. The clock source output destination is set by the watchdog clock select bit in the watch timer control register (WTC: WDCS).

□ Watchdog reset generator

The watchdog reset generator generates a reset signal when the watchdog timer overflows (carrying).

□ Counter clear circuit

The counter clear circuit clears the watchdog timer counter.

□ Watchdog timer control register (WDTC)

The watchdog timer control register starts and clears the watchdog timer, sets the interval time, and holds reset sources.

12.3 Watchdog Timer Registers

This section explains the registers used for setting the watchdog timer.

■ List of Registers and Initial Values of Watchdog Timer

Figure 12-2. List of Registers and Initial Values of Watchdog Timer

		bit	7	6	5	4	3	2	1	0
Watchdog timer control register (WDTC)	Address		×	1	×	×	×	1	1	1
	×	: Undefined								

12.3.1 Watchdog Timer Control Register (WDTC)

The watchdog timer control register starts and clears the watchdog timer, sets the interval time, and holds reset sources.

■ Watchdog Timer Control Register (WDTC)

Figure 12-3. Watchdog Timer Control Register (WDTC)

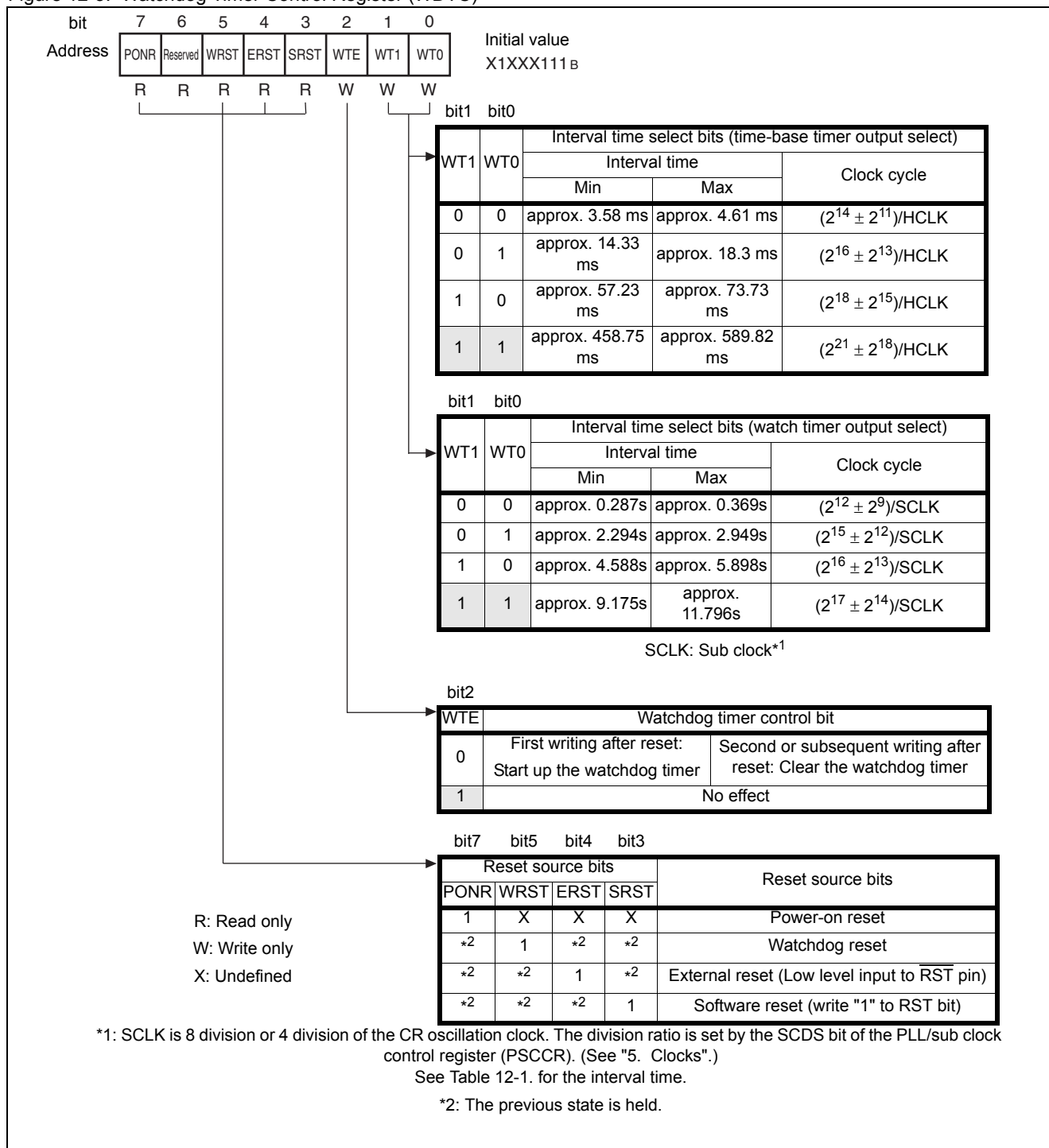


Table 12-2. Functions of the Watchdog Timer Control Register (WDTC)

Bit name		Function
bit7, bit5 to bit3	PONR, WRST, ERST, SRST: Reset source bits	<p>These bits indicate reset sources.</p> <p>When a reset occurs, the bit corresponding to the reset source is set to "1". After a reset, the reset source can be checked by reading the watchdog timer control register (WDTC).</p> <p>These bits are cleared after the watchdog timer control register (WDTC) is read.</p> <p>Note:</p> <p>No bit value other than the PONR bit after power-on reset is assured. If the PONR bit is set at read, other bit values should be ignored.</p>
bit6	Reserved bit	<p>Read: The value is undefined.</p> <p>Write: No effect</p>
bit2	WTE: Watchdog timer control bit	<p>This bit starts or clears the watchdog timer.</p> <p>When set to "0" (first time after reset): The watchdog timer is started.</p> <p>When set to "0" (second or subsequent after reset): The watchdog timer is cleared.</p>
bit1, bit0	WT1, WT0: Interval time select bits	<p>These bits set the interval time of the watchdog timer.</p> <p>The time interval when the watch timer is used as the clock source to the watchdog timer (watchdog clock select bit WDSCS=0) is different from when the main clock mode or the PLL clock mode is selected as the clock mode and the WDSCS bit in the watch timer control register (WTC) is set to "1" as shown in Figure 12-3. according to the settings of the WTC register.</p> <p>In the sub clock mode, be sure to set the watchdog clock select bit (WDSCS) in the watch timer control register (WTC) to "0" and select the output of the watch timer.</p> <p>Data after the watchdog timer is started is valid only.</p> <p>Write data after the watchdog timer is started is ignored.</p> <p>These are write-only bits.</p>

12.4 Explanation of Operations of Watchdog Timer Functions

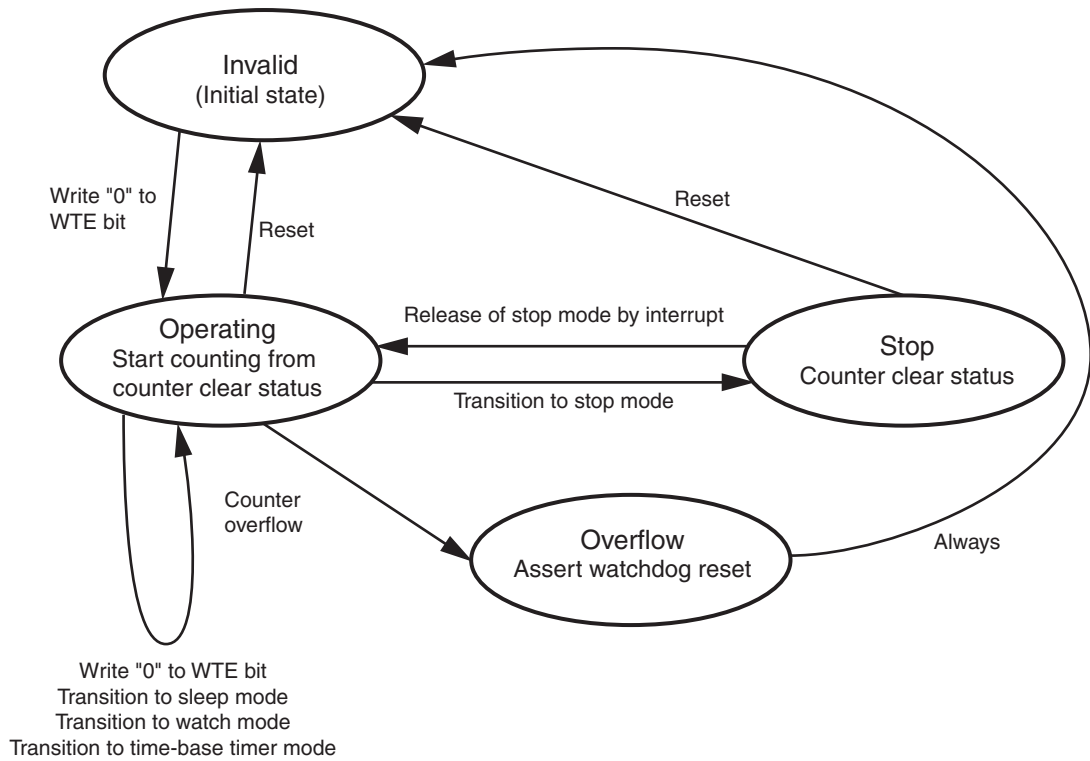
After starting, when the watchdog timer reaches the set interval time without the counter being cleared, a watchdog reset occurs.

■ State Transition Diagram of Watchdog Timer

There are the following four states in the watchdog timer.

- Invalid: It does not operate.
- Operating: The count is started from the counter clear status.
- Stop: The counter clear status is continued.
- Overflow: Watchdog reset is generated.

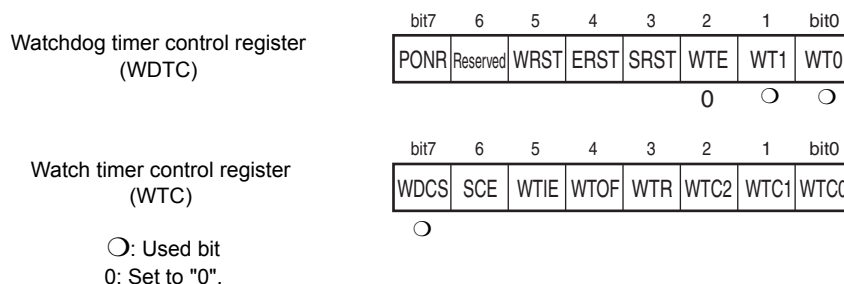
Figure 12-4. State Transition Diagram of Watchdog Timer



■ Operations of Watchdog Timer

The operation of the watchdog timer requires the settings shown in [Figure 12-5](#).

Figure 12-5. Setting of Watchdog Timer



□ Selecting clock input source

The time-base timer or watch timer can be selected as the clock input source of the count clock to the watchdog timer. When the watchdog clock select bit (WTC: WDCS) is set to "1", the time-base timer is selected. When the bit is set to "0", the watch timer is selected. After reset, the bit returns to "1" (time-base timer).

During operation in the sub clock mode, set the WDCS bit to "0" to select the watch timer.

Note:

When the watch timer is selected as the clock input source of the watchdog timer in the single clock system, the watchdog timer cannot be used.

□ Setting interval time

Set the interval time select bits (WDTC: WT1, WT0) to select the interval time for the watchdog timer.

Set the interval time concurrently when starting the watchdog timer. Writing to the bit is ignored after the watchdog timer is started.

□ Activating watchdog timer

When "0" is written to the watchdog timer control bit (WDTC: WTE) after a reset, the watchdog timer is started and starts incrementing.

□ Clearing watchdog timer

When "0" is written once again to the watchdog timer control bit (WDTC: WTE) within the interval time after starting the watchdog timer, the watchdog timer is cleared. If the watchdog timer is not cleared within the interval time, it overflows and the CPU is reset.

A reset, or transitions to the standby modes (sleep mode, stop mode, watch mode, time-base timer mode) clear the watchdog timer.

The watchdog timer remains starting though the watchdog timer counter is cleared in operating the time-base timer, operating the watch mode, and sleep mode state.

Figure 12-6. shows relationship between clear timing and interval time of watchdog timer. The interval time varies with the timing of clearing the watchdog timer.

□ Checking reset sources

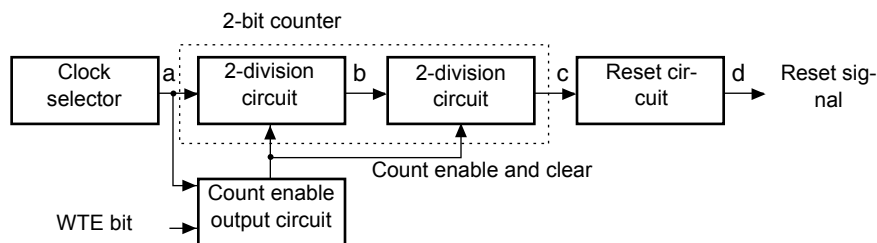
The reset source bits in the watchdog timer control register (WDTC: PONR, WRST, ERST, SRST) can be read after reset to check the reset sources.

Reference:

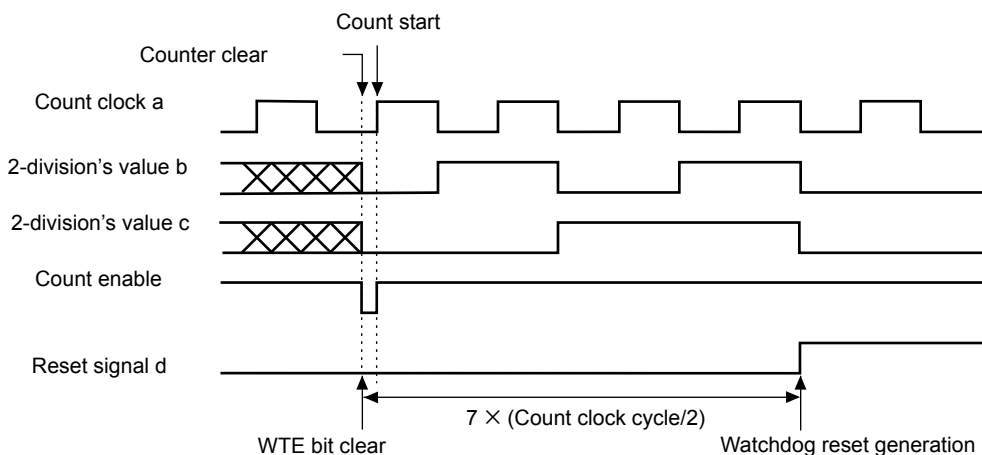
For details on the reset source bit, see "7. Rests".

Figure 12-6. Relationship between Clear Timing and Interval Time of Watchdog Timer

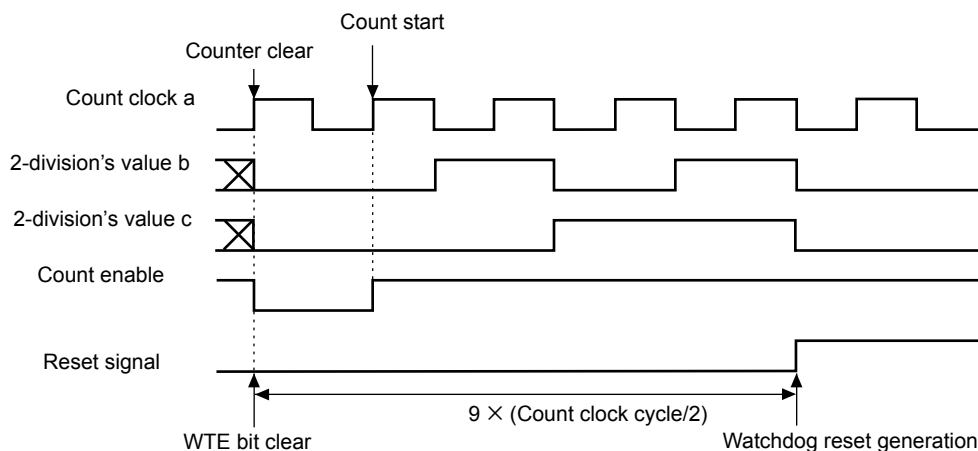
Watchdog timer block diagram



Minimum interval time (When clear WTE bit immediately before rising of count clock.)



Maximum interval time (When clear WTE bit immediately after rising of count clock.)



12.5 Notes on Using Watchdog Timer

Take the following notes when using the watchdog timer.

■ Notes on Using Watchdog Timer

- Stopping watchdog timer

The watchdog timer stops at shifting to the stop mode.

- Interval time
 - The interval time uses the 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 long. Note that the time-base timer is cleared when "0" is written to the time-base timer counter clear bit (TBR) in the time-base timer control register (TBTC) and when the clock mode changes from the main clock to PLL clock, from the sub clock to main clock, or from the sub clock to PLL clock.
 - Set the interval time concurrently when starting the watchdog timer. Setting the time interval except starting the watchdog timer is ignored.
- Notes on creating program

When clearing the watchdog timer repeatedly in the main loop, set a shorter processing time for the main loop, including interrupt processing, than the interval time of watchdog timer.

Since the watchdog timer is still working during the sleep mode, time-base timer mode and watch mode, it is necessary to take into consideration every operating time for each mode including the interval time using the user programs.

- Notes in sub clock mode

In the sub clock mode, be sure to set the watchdog clock select bit (WDCS) in the watch timer control register (WTC) to "0" and select the output of the watch timer.

- Watchdog timer operation in sleep mode, time-base timer mode, and watch mode

The watchdog timer is cleared after changing to the sleep mode, the time-base timer mode, or the watch mode, and the count is started again (see Table 12-3.).

- Watchdog timer operation in stop mode

After changing to the stop mode, the watchdog timer is cleared, and stops. When the stop mode is released, the count is started again (see Table 12-3.).

- Watchdog timer operation in resetting

All reset sources make the watchdog timer invalid. After releasing reset, the watchdog timer is invalid (see Table 12-3.).

Table 12-3. Watchdog Timer Clear Condition

Operation mode	Reset	WDTC register WTE=0	Stop mode	Sleep mode	Time-base timer mode	Watch mode
Clear	At transition	At writing	At transition	At transition	At transition	At transition
Watchdog timer state in mode	Invalid	-	Stop (Clear state is held.)	Operating (The count is started immediately after clearing.)	Operating (The count is started immediately after clearing.)	Operating (The count is started immediately after clearing.)
Watchdog reset in mode	Not generated	-	Not generated	Generated	Generated	Generated

Table 12-3. Watchdog Timer Clear Condition

Operation mode	Reset	WDTC register WTE=0	Stop mode	Sleep mode	Time-base timer mode	Watch mode
Watchdog timer state after releasing/returning the mode	Invalid	Operating	Operating (The count is restarted immediately after clearing.)	Operating (The count is continued.)	Operating (The count is continued.)	Operating (The count is continued.)

12.6 Program Example of Watchdog Timer

Program example of watchdog timer is given below:

■ Program Example of Watchdog Timer

- Processing specification
 - The watchdog timer is cleared each time in the loop of the main program.
 - The main program must be executed once within the minimum interval time of the watchdog timer.
- Coding example

```

WDTC    EQU    0000A8H                ;Watchdog timer control register
WTE     EQU    WDTC:2                ;Watchdog control bit
;-----Main program-----
CODE          CSEG
START:                                ;Stack pointer (SP), already
                                ;initialized
        MOV     I:WDTC,#00000011B    ;Start up of watchdog timer
                                ;Select interval time  $2^{21}+2^{18}$ 
                                ;cycle
LOOP:
        CLRB    I:WTE                ;Clear watchdog timer
        :
        User processing
        :
        BRA     LOOP
;-----Vector setting-----
VECT      CSEG    ABS=0FFH
        ORG     00FFDCH                ;Reset vector setting
        DSL     START
        DB      00H                    ;Setting to single chip mode
VECT      ENDS
        END     START

```


13. 16-Bit I/O Timer



This chapter explains the function and operation of the 16-bit I/O timer.

[13.1 Overview of 16-bit I/O Timer](#)

[13.2 Block Diagram of 16-bit I/O Timer](#)

[13.3 Configuration of 16-bit I/O Timer](#)

[13.4 Interrupts of 16-bit I/O Timer](#)

[13.5 Explanation of Operation of 16-bit Free-run Timer](#)

[13.6 Explanation of Operation of Input Capture](#)

[13.7 Notes on Using 16-bit I/O Timer](#)

[13.8 Program Example of 16-bit I/O Timer](#)

13.1 Overview of 16-bit I/O Timer

The 16-bit I/O timer consists of one 16-bit free-run timer and 4 input capture. The timer can be performed the measurement of input pulse and external clock cycle based on the 16-bit free-run timer.

■ Module Configuration of 16-bit I/O Timer

The 16-bit I/O timer consists of the following modules:

- 16-bit free-run timer ´ 1 unit
16-bit free-run timer 0 (ch.0)
- Input capture ´ 4 units
Input capture unit 0: capture 16-bit free-run timer 0
 - Input capture 0 (ch.0)
 - Input capture 1 (ch.1)
 - Input capture 2 (ch.2)
 - Input capture 3 (ch.3)

■ Functions of 16-bit I/O Timer

- Functions of 16-bit free-run timer

The 16-bit free-run timer consists of a 16-bit up counter, a prescaler, and a control register.

The count value of the 16-bit free-run timer can be used as the base time for the input capture.

- One of eight types of the count clock cycle can be set.
- An overflow in the counter generates an interrupt request.
- The counter of the 16-bit free-run timer is cleared to "0000_H" by reset or timer clear (TCCSL0:CLR=1).
 - Functions of input capture

The input capture consists of four 16-bit capture registers and control registers corresponding to the external input pin, and the edge detection circuit.

When the trigger edge is inputted to the external input pin, the counter value of the 16-bit free-run timer is retained and the interrupt request is generated at the same time.

- The capture interrupt can be generated independently by each channel.
- The EI²OS can be started.
- Trigger edge can be selected from rising edge, falling edge, or both edges.
- Because each channel operates independently, up to 4 input measurement is performed.
- When the input signal is set to the LIN-UART, the baud rate measurement at LIN slave operation is executed.

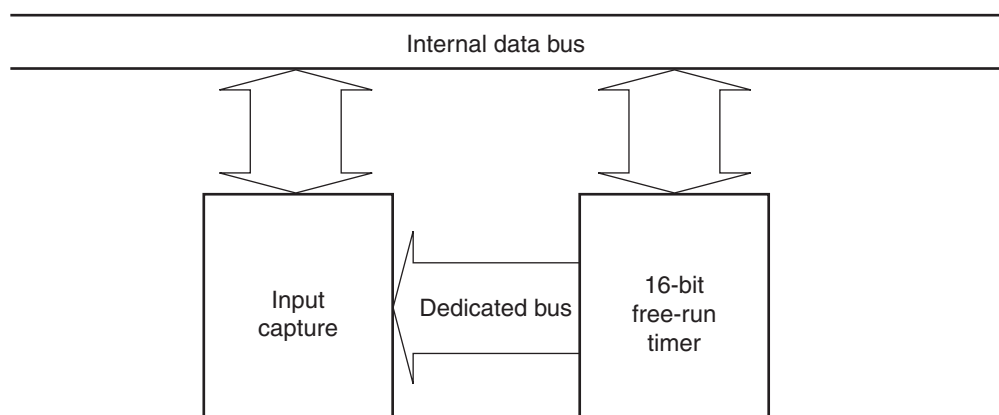
13.2 Block Diagram of 16-bit I/O Timer

The 16-bit I/O timer consists of the following modules:

- 16-bit free-run timer
- Input capture

■ Block Diagram of 16-bit I/O Timer

Figure 13-1. Block Diagram of 16-bit I/O Timer



- 16-bit free-run timer

The count value of the 16-bit free-run timer can be used as the base time for the input capture.

- Input capture

When the trigger edge is inputted to the external input pin, or when the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted, the counter value of the 16-bit free-run timer is retained and the interrupt request is generated at the same time.

■ Details of Pins and Interrupt Number

Table 13-1. shows details on the pins and interrupt used by the 16-bit I/O timer.

Table 13-1. Details of Pins and Interrupt Number

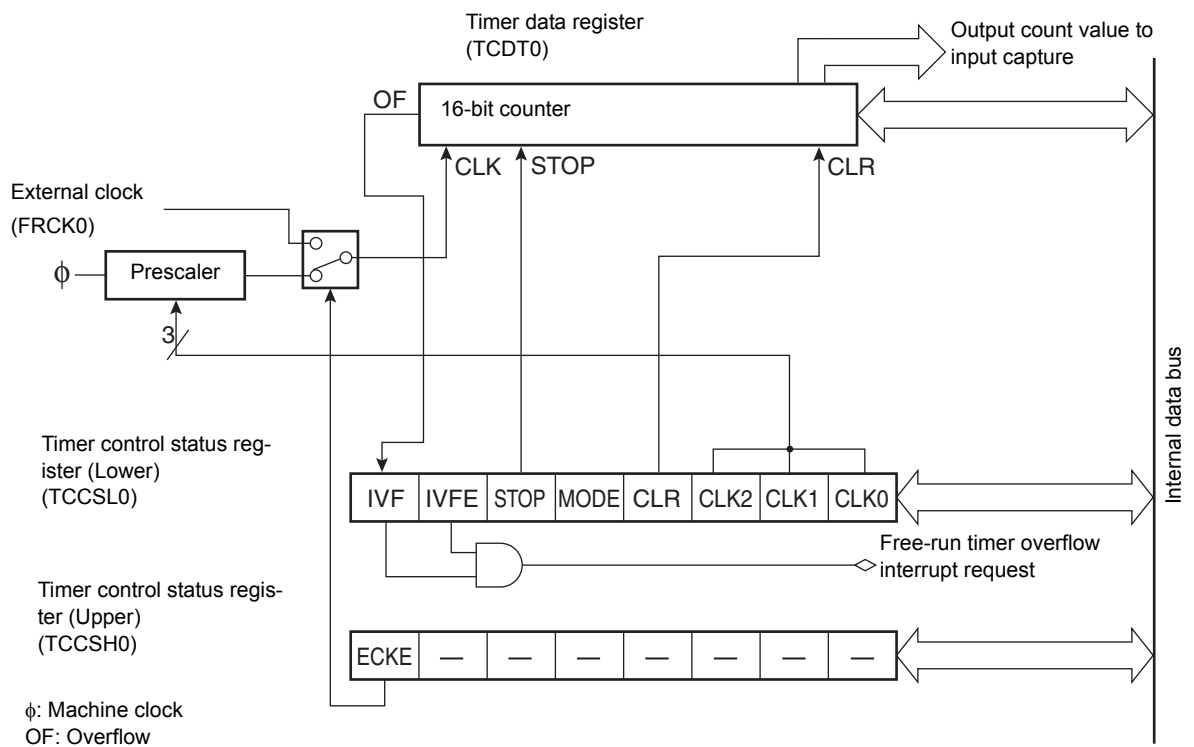
Channel	Special pin	Pin name	Interrupt No.	For EI ² OS
Input capture ch.0 (using 16-bit free-run timer ch.0)	IN0	P24/IN0	#33 (21 _H)	○
Input capture ch.1 (using 16-bit free-run timer ch.0)	IN1	P25/IN1		
Input capture ch.2 (using 16-bit free-run timer ch.0)	IN2	P26/IN2		
Input capture ch.3 (using 16-bit free-run timer ch.0)	IN3	P27/IN3		
16-bit free-run timer ch.0 (overflow interrupt)	FRCK0	P44/FRCK0	#30 (1E _H)	×

13.2.1 Block Diagram of 16-bit Free-run Timer

The MB90990 series contains 1 channel of the 16-bit free-run timer, and it consists of the following block.

■ Block Diagram of 16-bit Free-run Timer

Figure 13-2. Block Diagram of 16-bit Free-run Timer



- Prescaler

The prescaler divides the frequency of the machine clock to supply a count clock to the 16-bit counter. Any of eight count clock cycles can be selected by setting the timer control status register (TCCSL0: CLK2 to CLK0).

- Timer data register (TCDT0)

The timer data register can read the counter value of the 16-bit free-run timer. During stopping of the 16-bit free-run timer, the counter value can be set by writing the counter value to the TCDT0.

- Timer control status register (TCCSH0, TCCSL0)

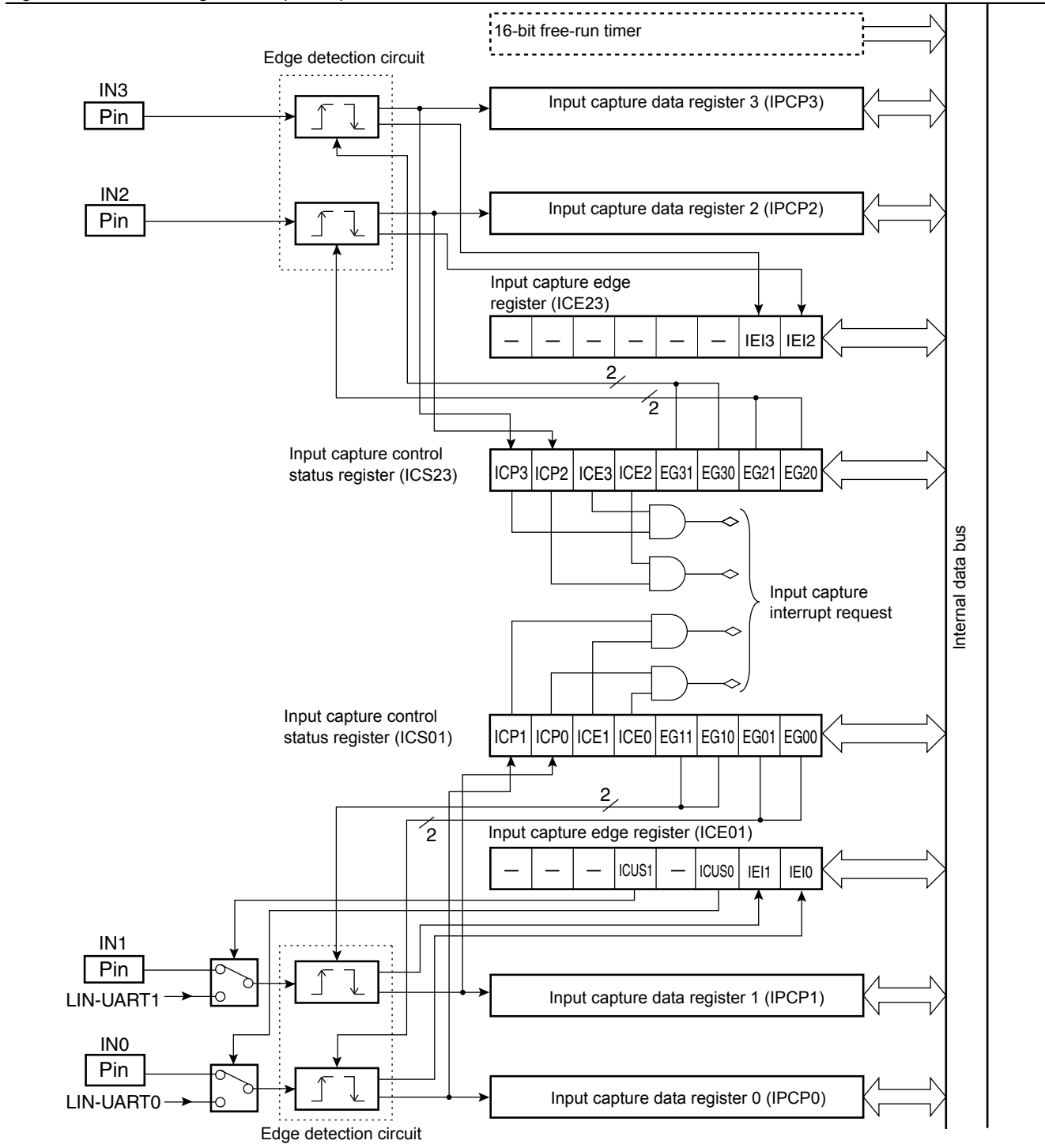
The timer control status register (upper and lower) selects the count clock and the condition for clearing the counter, clears the counter, enables the count operation and interrupt request, checks the overflow generation flag.

13.2.2 Block Diagram of Input Capture

The input capture consists of the following blocks:

■ Block Diagram of Input Capture

Figure 13-3. Block Diagram of Input Capture Unit 0



- ❑ Input capture data registers 0 to 3 (IPCP0 to IPCP3)
- ❑ Input capture data register retains the counter value of the 16-bit free-run timer fetched by the capture operation.
- ❑ Input capture data registers 0 to 3 keep the counter value of the 16-bit free-run timer 0.
- ❑ Input capture control status registers 01, 23 (ICS01, ICS23)
- ❑ Input capture control status register selects the trigger edge, enables the capture operation and capture interrupt request, and checks the valid edge detection flag for each input capture.
- ❑ Input capture control status register has 2 registers, and the input capture operation of the corresponding channel is controlled as shown in [Table 13-2](#).
- ❑ Input capture edge registers 01, 23 (ICE01, ICE23)
- ❑ Input capture edge register indicates the edge polarity detected by each input capture. Also, it selects the input signal (external pin INx/LIN-UART). When input is set to the LIN-UART, the baud rate measurement at the LIN slave operation can be performed (See Section "20.7.3 Operation with LIN Function (Operation Mode 3)").
- ❑ Input capture edge register has 2 registers, and the input capture operation of the corresponding channel is controlled as shown in [Table 13-2](#).

Table 13-2. Relationship between the Register and Pin of Input Capture

	Input capture control status register	Input capture edge register	Input capture data register	Input pin	Input from LIN-UART
Input capture unit 0	ICS01	ICE01	IPCP0	IN0	UART0
			IPCP1	IN1	UART1
	ICS23	ICE23	IPCP2	IN2	-
			IPCP3	IN3	-

- ❑ Edge detection circuit

The edge detection circuit detects the edge of the signal input to the external input pin. The detected edge can be selected from among the rising edge, falling edge, both edges, and no detection (capture stop).

13.3 Configuration of 16-bit I/O Timer

This section explains the pins, registers, and interrupt factors of the 16-bit I/O timer.

■ Pins of 16-bit I/O Timer

The pins of the 16-bit I/O timer serve as general-purpose I/O ports. [Table 13-3](#) shows the pin functions and the pin settings required to use the 16-bit I/O timer.

Table 13-3. Pins of 16-bit I/O Timer

Channel	Pin Name	Pin Function	Setting to use the pin
16-bit free-run timer 0	P44/ FRCK0	General-purpose I/O port, external clock input	Set as input port in port direction register (DDR).
Input capture 0	P24/IN0	General-purpose I/O port, capture input	Set as input port in port direction register (DDR).
Input capture 1	P25/IN1		Set as input port in port direction register (DDR).
Input capture 2	P26/IN2		Set as input port in port direction register (DDR).
Input capture 3	P27/IN3		Set as input port in port direction register (DDR).

■ Generation of Interrupt Request from 16-bit I/O Timer

The 16-bit I/O timer can generate an interrupt request as a result of the following factors:

- Timer counter overflow interrupt

If the overflow interrupt request is set to enable (TCCSL0: IVFE=1), the interrupt request is occurred by the following factor:

- 16-bit free-run timer overflow
- Input capture interrupt

If the input capture interrupt request is set to enable (ICS: ICE=1), if the trigger edge is detected by the input capture pin, or if the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted, the interrupt request is generated.

13.3.1 Timer Control Status Register (Upper) (TCCSH0)

Timer control status register (upper) selects the count clock and the conditions for clearing the counter, clears the counter, enables the count operation and interrupt, and checks the interrupt request flag.

■ Timer Control Status Register (Upper) (TCCSH0)

Figure 13-4. Timer Control Status Register (Upper) (TCCSH0)

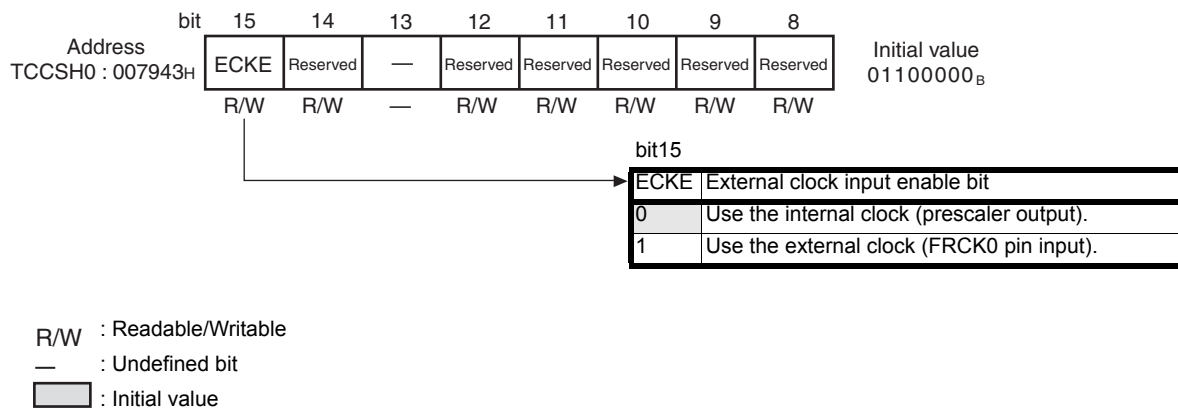


Table 13-4. Function of Timer Control Status Register (Upper) (TCCSH0)

Bit name		Function
bit15	ECKE : External clock input enable bit	This bit selects the count clock of the 16-bit free-run timer. When set to "1" : Use the clock inputted from the external pin FRCK0. When set to "0" : Use the internal clock (clock outputted from the prescaler). Note: Set the ECKE bit during stopping of the free-run timer (TCCSL : STOP=1). When inputting the clock from FRCK, set FRCK0 (DDR4 : bit4=0).
bit14	Reserved bit	This bit is reserved. Always write "1" to this bit.
bit13	Undefined bit	Read : The value is undefined Write : No effect
bit12 to bit8	Reserved bits	These bits are reserved. Always write "0" to these bits.

13.3.2 Timer Control Status Register (Lower) (TCCSL0)

The timer control status register (lower) selects the count clock and conditions for clearing the counter, clears the counter, enables the count operation or interrupt, and checks the interrupt request flag.

■ Timer Control Status Register (Lower) (TCCSL0)

Figure 13-5. Timer Control Status Register (Lower) (TCCSL0)

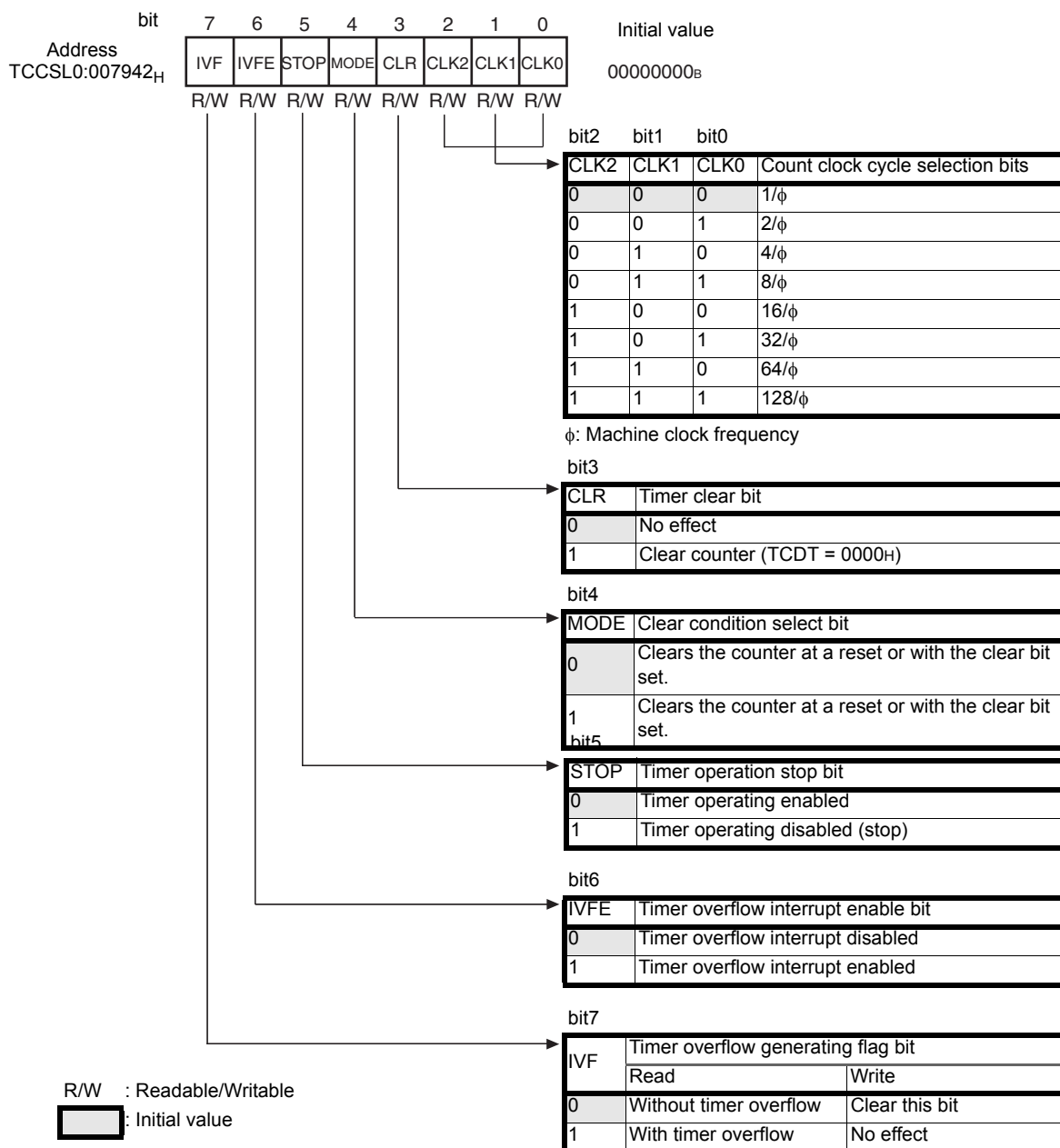


Table 13-5. Functions of Timer Control Status Register (Lower) (TCCSL0)

Bit name		Function
bit7	IVF: Timer overflow generation flag bit	<p>This bit indicates the timer overflow. [Condition set to "1"] Condition is set when the following is used. When 16-bit free-run timer overflows [When set to "1"] When the timer overflow interrupt request is set to enable (TCCSL0:IVFE=1) if the IVF bit is set to "1", the interrupt request is generated. When set to "0": The bit is cleared. When set to "1": No effect. Read by read-modify-write (RMW) instructions: "1" is always read.</p>
bit6	IVFE: Timer overflow interrupt enable bit	<p>This bit enables or disables the interrupt request when the IVF bit is set to "1". When set to "1":When the IVF bit is set to "1", the interrupt request is generated. When set to "0":The generation of the interrupt request is disabled.</p>
bit5	STOP: Timer operation stop bit	<p>This bit enables or disables (stops) the operation of the 16-bit free-run timer. When set to "0":Enable the timer operation and count up with count clock set by the CLK2 to CLK0. When set to "1":Stops timer operation.</p>
bit4	MODE: Clear condition select bit	<p>This bit selects the condition for clearing the counter value of the 16-bit free-run timer (TCDT register). Setting the bit to "0":Clears the TCDT counter value in either of the following conditions: A reset occurs. The timer clear bit is set to "1" (TCCSL:CLR=1). Setting the bit to "1":Clears the TCDT counter value in any of the following conditions: A reset occurs. The timer clear bit is set to "1" (TCCSL:CLR=1).</p>
bit3	CLR: Timer clear bit	<p>This bit clears the counter (TCDT) of the 16-bit free-run timer. The counter is cleared in synchronization with the change point of the counter. When set to "1":Clears timer data register (TCDT) to "0000_H". When set to "0":No effect. Read:"0" is always read. Note:When clearing during stopping of the 16-bit free-run timer (TCCSL0:STOP=1), write "0000_H" to the TCDT directly.</p>
bit2 to bit0	CLK2, CLK1, CLK0: Count clock cycle selection bits	<p>These bits set the count clock cycle of the 16-bit free-run timer. Note:Set the count clock cycle during stopping of the input capture operation (ICSnm:EGn1, EGn0=00_B or ICSnm:EGm1, EGm0=00_B).</p>

n=0, 2 m=n+1

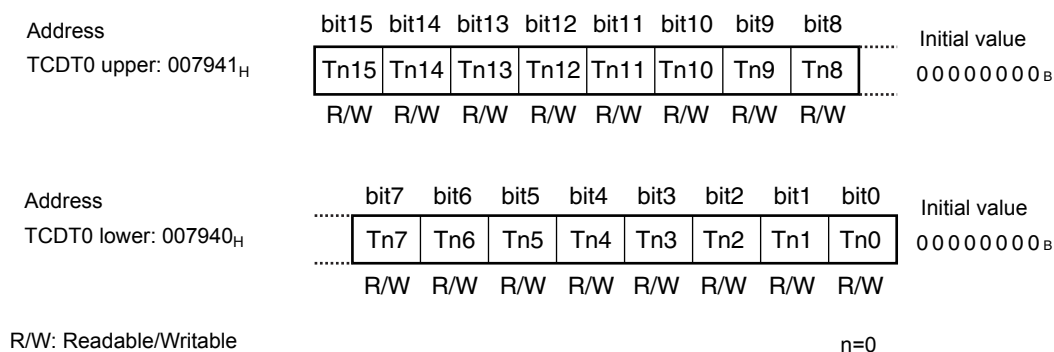
13.3.3 Timer Data Register (TCDT0)

The timer data register is a 16-bit up counter.

- The counter value of the 16-bit free-run timer is read.
- The counter value can be set during stopping of the 16-bit free-run timer.

■ Timer Data Register (TCDT0)

Figure 13-6. Timer Data Register (TCDT0)



The TCDT0 register can read the counter value of the 16-bit free-run timer.

[Condition for clear the counter value]

The counter value is cleared to "0000_H" by the following conditions.

- Overflow
- Setting of "1" to the timer clear bit of the timer control status register (TCCSL0:CLR=1)
- Setting of "0000_H" to the timer data register during stopping of 16-bit free-run timer
- Reset

[Setting of counter value]

Write the counter value to the timer data register (TCDT0) and set the timer during stopping the timer operation (TCCSL0:STOP=1).

Note:

Always use a word instruction (MOVW) to read/write the timer data register.

13.3.4 Input Capture Control Status Registers (ICS)

The function of the input capture control status register is shown below.

The correspondence between ICS01, ICS23 and input pin is as follows.

- ICS01: IN0, IN1 input capture ch.0, ch.1
- ICS23: IN2, IN3 input capture ch.2, ch.3

■ Input Capture Control Status Registers (ICS)

Figure 13-7. Input Capture Control Status Registers (ICS)

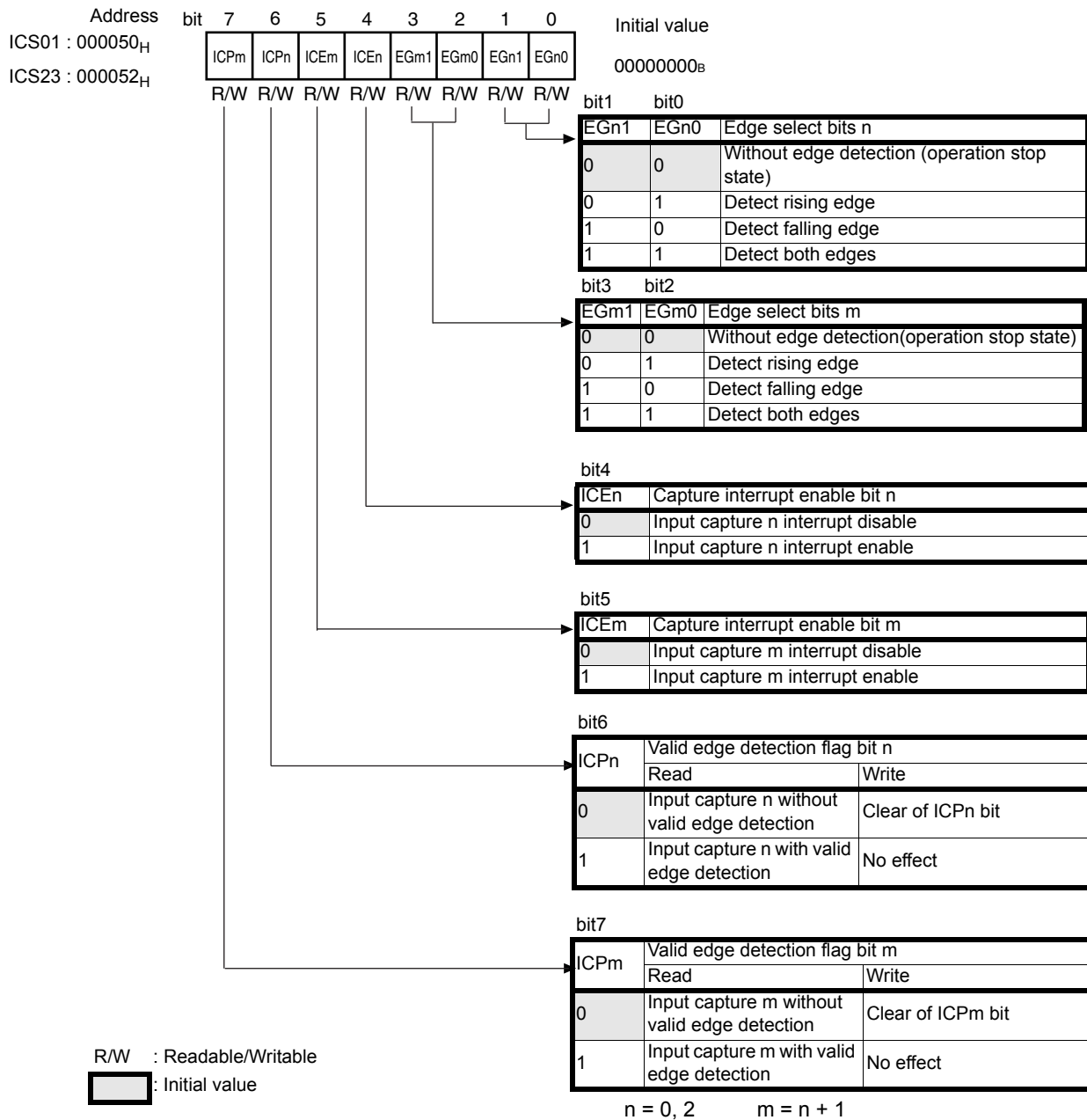


Table 13-6. Functions of Input Capture Control Status Register (ICS)

Bit name		Function
bit7	ICPm: Valid edge detection flag bit m	<p>This bit is set to "1" when the valid edge is detected by the INm pin.</p> <p>When the interrupt request of the input capture m is set to enable (ICSnm:ICEm=1), if the ICPm bit is set, the interrupt request is generated.</p> <p>When set to "0": The bit is cleared.</p> <p>When set to "1": No effect.</p> <p>"1" is read when a read-modify-write (RMW) instruction is used. If setting "1" to this bit and writing "0" to it occur simultaneously, writing "0" has priority.</p>
bit6	ICPn: Valid edge detection flag bit n	<p>This bit is set to "1" when the valid edge is detected by the INn pin.</p> <p>When the interrupt request of the input capture n is set to enable (ICSnm:ICEn=1), if the ICPn bit is set, the interrupt request is generated.</p> <p>When set to "0": The bit is cleared.</p> <p>When set to "1": No effect.</p> <p>"1" is read when a read-modify-write (RMW) instruction is used. If setting "1" to this bit and writing "0" to it occur simultaneously, writing "0" has priority.</p>
bit5	ICEm: Capture interrupt enable bit m	<p>This bit enables or disables the interrupt request of the input capture m.</p> <p>When set to "1":When the valid edge detection flag bit m is set to "1" (ICSnm: ICPm=1), the interrupt request is generated.</p>
bit4	ICEn: Capture interrupt enable bit n	<p>This bit enables or disables the interrupt request of the input capture n.</p> <p>When set to "1":When the valid edge detection flag bit n is set to "1" (ICSnm: ICPn=1), the interrupt request is generated.</p>
bit3, bit2	EGm1, EGm0: Edge select bits m	<p>For the input capture register m, the trigger edge of the capture operation is set.</p> <p>Setting of the trigger edge is used to specify enable and stop of the operation.</p> <p>When set to "00_B":The operation of input capture is disabled and no edge is detected.</p>
bit1, bit0	EGn1, EGn0: Edge select bits n	<p>For the input capture register n, the trigger edge of the capture operation is set.</p> <p>Setting of the trigger edge is used to specify enable and stop of the operation.</p> <p>When set to "00_B":The operation of input capture is disabled and no edge is detected.</p>

 $n = 0, 2 \quad m = n + 1$
Note:

When using the input capture, set the port direction register (DDRx) of the general-purpose I/O port to be shared, to an input port.

13.3.5 Input Capture Register (IPCP)

Input capture register stores the counter value fetched from 16-bit free-run timer by the capture operation. The IPCP register is the 16-bit read-only register and has the input capture registers 0 to 3 (IPCP0 to IPCP3).

■ Input Capture Register (IPCP)

Figure 13-8. Input Capture Register (IPCP)

		Initial value	
		Flash memory product	Evaluation product
Address	bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8		
IPCP0 (upper): 007921 _H	CP15 CP14 CP13 CP12 CP11 CP10 CP09 CP08	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0		
IPCP0 (lower): 007920 _H	CP07 CP06 CP05 CP04 CP03 CP02 CP01 CP00	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8		
IPCP1 (upper): 007923 _H	CP15 CP14 CP13 CP12 CP11 CP10 CP09 CP08	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0		
IPCP1 (lower): 007922 _H	CP07 CP06 CP05 CP04 CP03 CP02 CP01 CP00	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8		
IPCP2 (upper): 007925 _H	CP15 CP14 CP13 CP12 CP11 CP10 CP09 CP08	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0		
IPCP2 (lower): 007924 _H	CP07 CP06 CP05 CP04 CP03 CP02 CP01 CP00	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8		
IPCP3 (upper): 007927 _H	CP15 CP14 CP13 CP12 CP11 CP10 CP09 CP08	00000000 _B	XXXXXXXX _B
	R R R R R R R R		
Address	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0		
IPCP3 (lower): 007926 _H	CP07 CP06 CP05 CP04 CP03 CP02 CP01 CP00	00000000 _B	XXXXXXXX _B
	R R R R R R R R		

R: Read only

When the trigger edge of the capture operation (ICS_nm: set by EG_n1, EG_n0 or EG_m1, EG_m0) is detected by the IN₀ to IN₃ pins, the counter value of the 16-bit free-run timer is stored in the input capture registers 0 to 3 corresponding to each pin.

However, the input capture registers 0 and 1 can be selected a signal from the LIN-UART as the input signal (ICE: selected by IEI bit). See Section "13.3.6 Input Capture Edge Register (ICE)" for details.

The input capture register can be read, but cannot be written.

$n = 0, 2 \quad m = n + 1$

Note:

Always use a word instruction (MOVW) to read the input capture register.

13.3.6 Input Capture Edge Register (ICE)

The input capture edge register has a function to indicate the selected edge direction and to select whether the input signal is inputted from either external pin or LIN-UART.

By cooperating with the LIN-UART, the baud rate measurement at the LIN slave operation can be performed.

The correspondence between ICE01, ICE23 / channel name and input pin (UART) name is shown as follows.

ICE01: input capture ch.0, ch.1	IN0(/UART0)	IN1(/UART1)
ICE23: input capture ch.2, ch.3	IN2	IN3

■ Input Capture Edge Register (ICE)

Figure 13-9. Input Capture Edge Register (ICE)

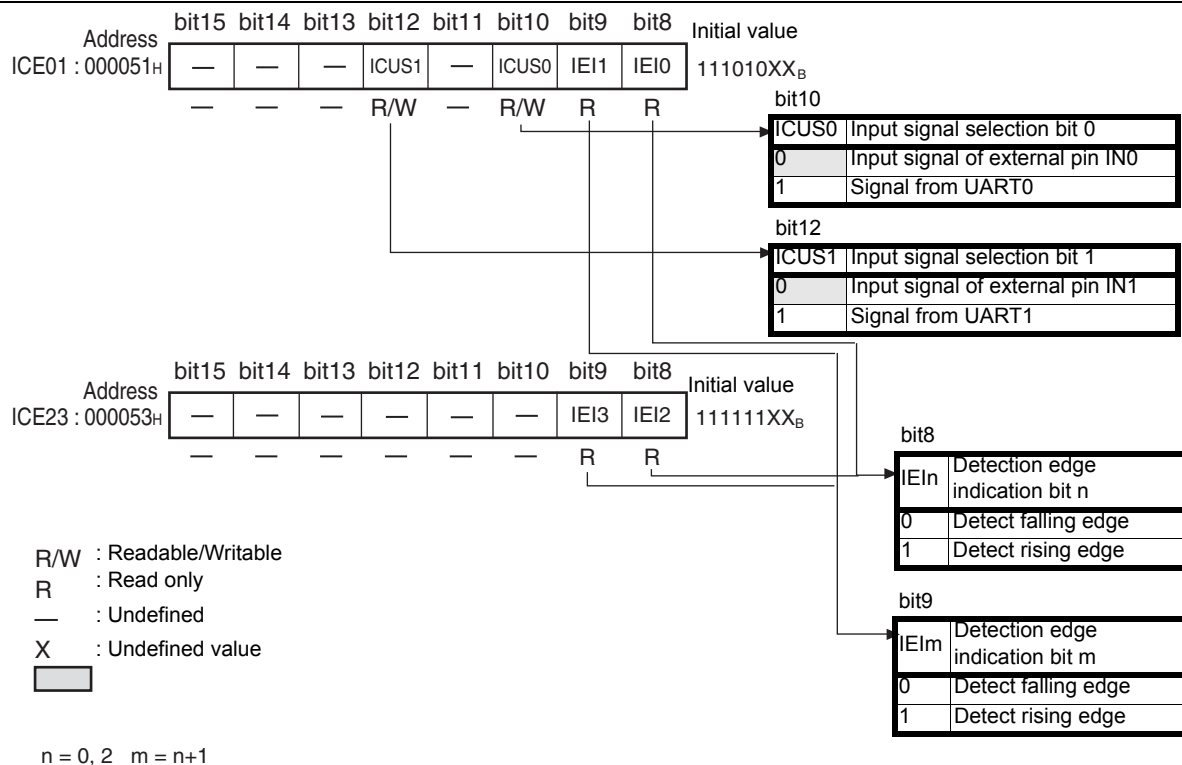


Table 13-7. Functions of Input Capture Edge Register 01 (ICE01)

Bit name		Function
bit15 to bit13	Undefined bits	Read: "1" is read. Write: No effect.
bit12	ICUS1: Input signal selection bit 1	This bit selects the input signal used as the trigger of the input capture 1. When set to "0": Select the external pin IN1. When set to "1": Select the LIN-UART1.
bit11	Undefined bit	Read: "1" is read. Write: No effect.
bit10	ICUS0: Input signal selection bit 0	This bit selects the input signal used as the trigger of the input capture 0. When set to "0": Select the external pin IN0. When set to "1": Select the LIN-UART0.
bit9	IEI1: Detection edge indication bit 1	This bit indicated the edge detected by the input capture 1 (rising/falling). This bit is read only. "0": Indicate that falling edge is detected. "1": Indicate that rising edge is detected. Note: This bit value is disabled when the capture operation is stopped (ICS01: EG11, EG10=00 _B).
bit8	IEI0: Detection edge indication bit 0	This bit indicated the edge detected by the input capture 0 (rising/falling). This bit is read only. "0": Indicate that falling edge is detected. "1": Indicate that rising edge is detected. Note: This bit value is disabled when the capture operation is stopped (ICS01: EG01, EG00=00 _B).

Table 13-8. Functions of Input Capture Edge Register 23 (ICE23)

Bit name		Function
bit15 to bit10	Undefined bits	Read: "1" is read. Write: No effect.
bit9	IEI3: Detection edge indication bit 3	This bit indicates the edges detected by the input capture 3 (rising/falling). This bit is read only. "0": Indicate that falling edge is detected. "1": Indicate that rising edge is detected. Note: This bit value is disabled when the capture operation is stopped (ICSnm: EGm1, EGm0=00 _B). (n=2, m=n+1)
bit8	IEI2 : Detection edge indication bit 2	This bit indicates the edges detected by the input capture 2 (rising/falling). This bit is read only. "0": Indicate that falling edge is detected. "1": Indicate that rising edge is detected. Note: This bit value is disabled when the capture operation is stopped (ICS23: EG21, EG20=00 _B).

Note:

In the input capture 0 and 1, if the input signal is selected to the LIN-UART (ICE01:ICUS), the input capture is used to calculate the baud rate when the LIN-UART operates the LIN slave. In this case, it must be set to the input capture interrupt enable (ICS01:ICE0=1 or ICE1=1) and to the detection of both edges (ICS01:EG01, EG00=11_B or EG11, EG10=11_B). See Section "20.7.3 Operation with LIN Function (Operation Mode 3)" for details of the baud rate calculation.

13.4 Interrupts of 16-bit I/O Timer

The interrupt factors of the 16-bit I/O timer has overflow of the counter value in the 16-bit free-run timer, trigger edge input to the input capture input pin, and trigger edge input for the LIN slave baud rate measurement from the LIN-UART.

The EI²OS can be started by the interrupt of the input capture.

■ Interrupts of 16-bit I/O Timer

Table 13-9. shows interrupt control bits and interrupt factors of 16-bit I/O timer.

Table 13-9. Interrupts of 16-bit I/O Timer

	Timer counter overflow interrupt	Input capture interrupt
Interrupt request flag	TCCSL: IVF	ICSnm: ICPn, ICPm
Interrupt request output enable bit	TCCSL: IVFE	ICSnm: ICEn, ICEm
Interrupt factor	Counter overflow of 16-bit free-run timer	Valid edge input to the input capture input pin and trigger edge input for the LIN slave baud rate measurement from the LIN-UART

n = 0, 2
m = n+1

□ Timer counter overflow interrupt

When the timer overflow interrupt request flag is set:

The timer overflow generation flag of the timer control status register is set in the following cases (TCCSL:IVF=1).

- When overflow ("FFFF_H" → "0000_H") occurs at counting up of the 16-bit free-run timer.

When the timer overflow interrupt request occurs:

When the timer overflow interrupt request is set to enable (TCCSL:IVFE=1) if the timer overflow generation flag is set to "1" (TCCSL:IVF=1), the interrupt request is generated.

□ Input capture Interrupt

When the valid edge set by the input capture pin (ICS:EG) is detected, or when the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted (valid edge must be set to both edges), the interrupt is shown below.

- The counter value of the detected 16-bit free-run timer is stored to the input capture register.
- The valid edge detection flag of the input capture control status register is set to "1" (ICS: ICP=1).
- When the output of the input capture interrupt request is set to enable (ICS: ICE=1), the interrupt request is generated.

■ 16-bit I/O Timer Interrupt and EI²OS

Reference:

For details of the interrupt number, interrupt control register, and interrupt vector address, see "Chapter 3 Interrupts".

■ Correspondence to EI²OS Function

The input capture corresponds to the EI²OS function. However, to use the EI²OS function, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

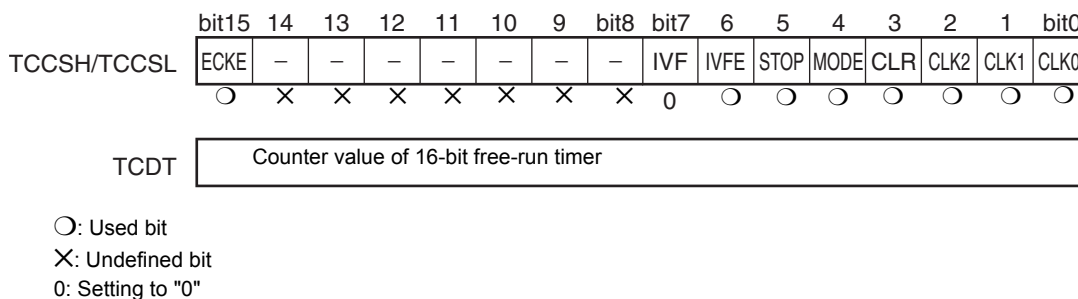
13.5 Explanation of Operation of 16-bit Free-run Timer

After a reset, the 16-bit free-run timer starts incrementing from counter value "0000_H". The counter value of the 16-bit free-run timer is the base time of the input capture.

■ Explanation of Operation of 16-bit Free-run Timer

Operation of the 16-bit free-run timer requires the setting shown in [Figure 13-10](#).

Figure 13-10. Setting of 16-bit Free-run Timer



[Setting of counter value in 16-bit free-run timer]

- Because the timer operation is enabled (TCCSL:STOP=0) after reset, the 16-bit free-run timer starts incrementing from the counter value "0000_H".

- When setting the counter value of the 16-bit free-run timer, disable the operation of the 16-bit free-run timer (TCCSL:STOP=1), set the value that starts counting to the timer data register, enable the timer operation (TCCSL:STOP=0).

[Generation of timer overflow and interrupt request]

- When overflow ("FFFF_H" @ "0000_H") occurs in the 16-bit free-run timer, the timer overflow generation flag is set to "1" (TCCSL:IVF) and starts incrementing from "0000_H".
- When the timer overflow interrupt request is enabled (TCCSL:IVFE=1), the interrupt request is occurred.

[Clear factor of counter value and clear timing]

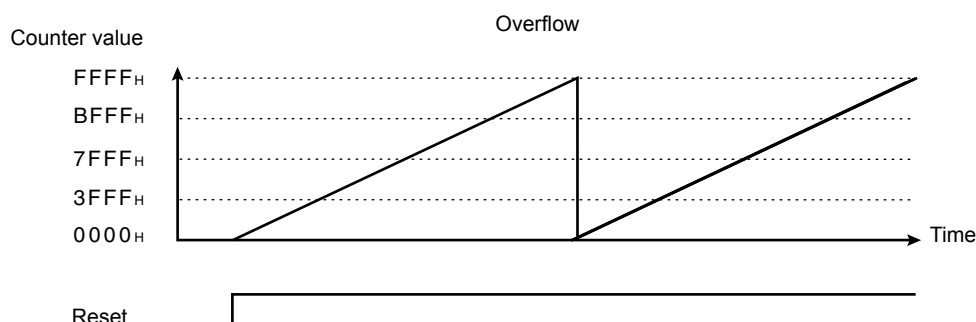
Table 13-10. shows the clear factor and clear timing of the 16-bit free-run timer.

Table 13-10. Clear Factor of Counter Value and Clear Timing

Clear factor	Clear timing
Write "1" to timer clear bit of timer control status register (TCCSL: CLR)	Synchronize with generation of factor
Write "0000 _H " to timer data register during stopping	Synchronize with generation of factor
Reset	Synchronize with generation of factor
Timer overflow	Synchronize with count timing

Figure 13-11. shows counter clearing at an overflow.

Figure 13-11. Counter Clearing at an Overflow



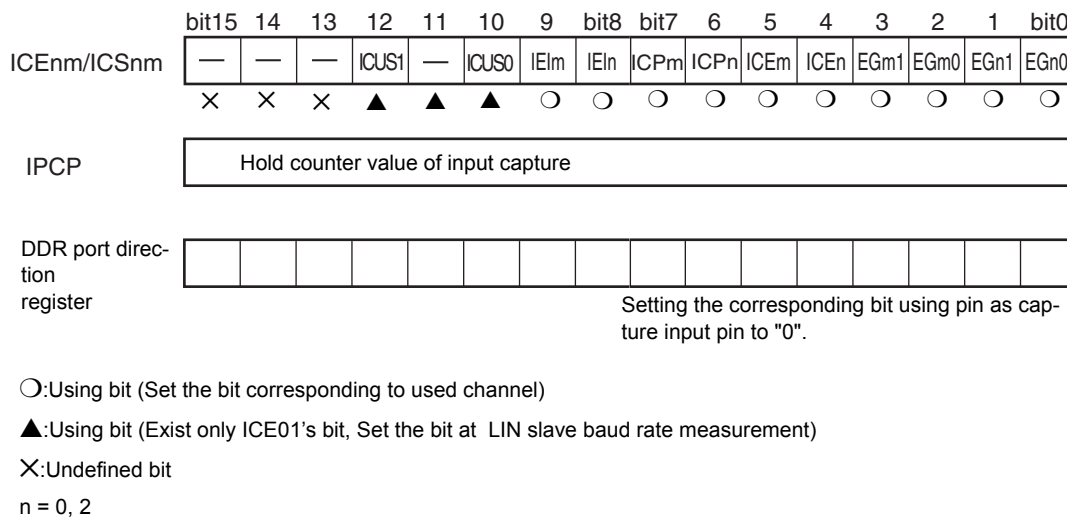
13.6 Explanation of Operation of Input Capture

The input capture stores the counter value of the 16-bit free-run timer to the input capture register at the timing that is detected the input signal of the valid edge from the external input pin or that the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted, the interrupt request is generated.

■ Explanation of Operation of Input Capture

Operation of the input capture requires the setting shown in Figure 13-12.

Figure 13-12. Setting of Input Capture

**[Input capture operation]**

The following operation is executed when the set valid edge (ICS:EG) is detected in the input capture pin, or when the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted.

- The counter value of the 16-bit free-run timer to time detected is stored in the input capture register.
- The detected edge direction is stored to the detection edge indication bit. (rising:IEI=1, falling:IEI=0)
- The valid edge detection flag of the input capture control status register is set to "1". (ICS:ICP=1)
- When the input capture interrupt request is enabled (ICS:ICE=1), the interrupt request is generated.
- To measure the baud rate at the LIN slave operation, it is necessary to set the input signal to the LIN-UART (ICS:ICUS), enable the input capture interrupt request (ICS:ICE=1), and set the valid edge to both edges (ICS:EG1, EG0=11_B). See Section "20.7.3 Operation with LIN Function (Operation Mode 3)" for the calculation of the baud rate.

Figure 13-13. shows the timing of fetching a data for the input capture. Figure 13-14. shows the operation when valid edge is set to the rising edge/falling edge. Figure 13-15. shows the operation when valid edge is set to both edges.

Figure 13-13. Timing of Fetching Data for Input Capture

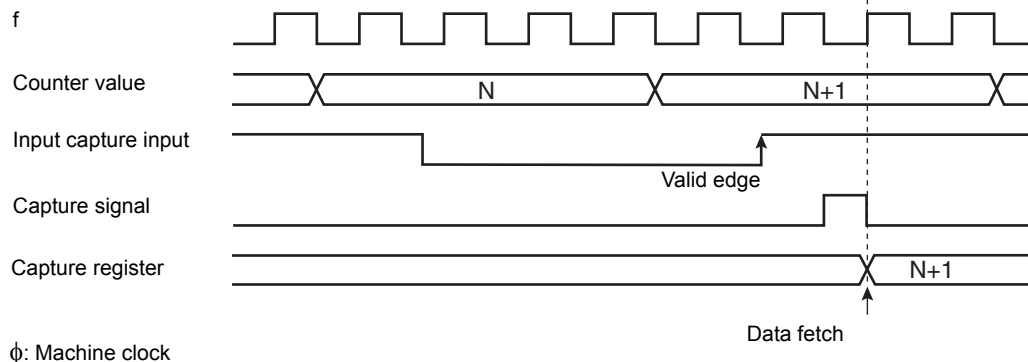


Figure 13-14. Operation of Input Capture (Rising Edge/Falling Edge)

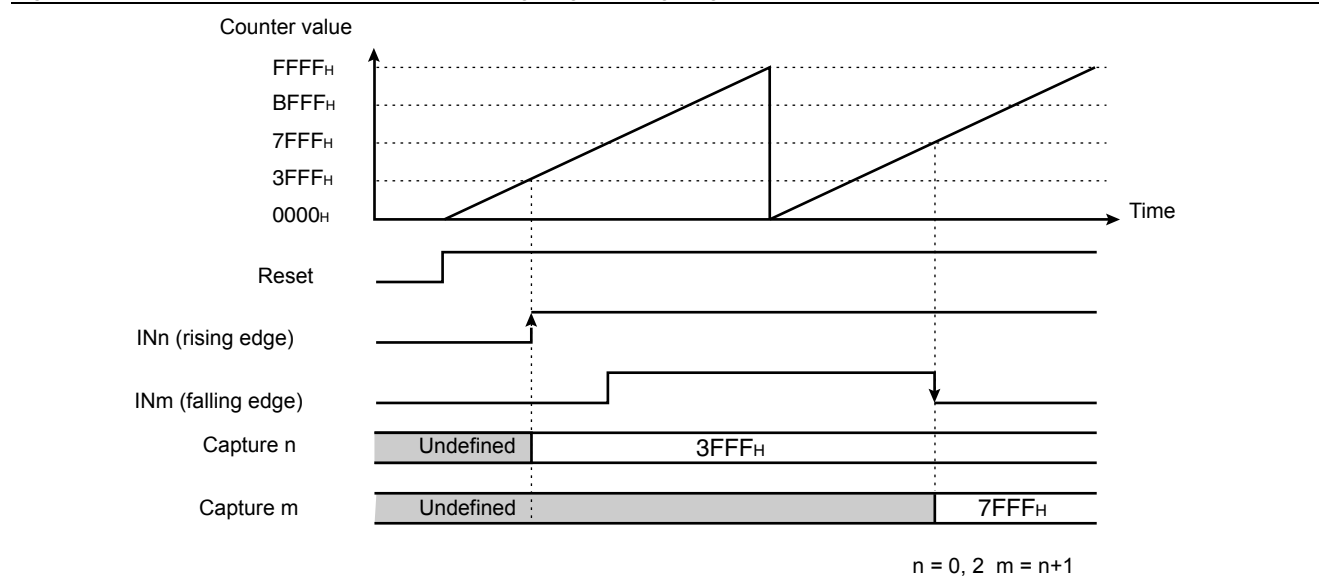
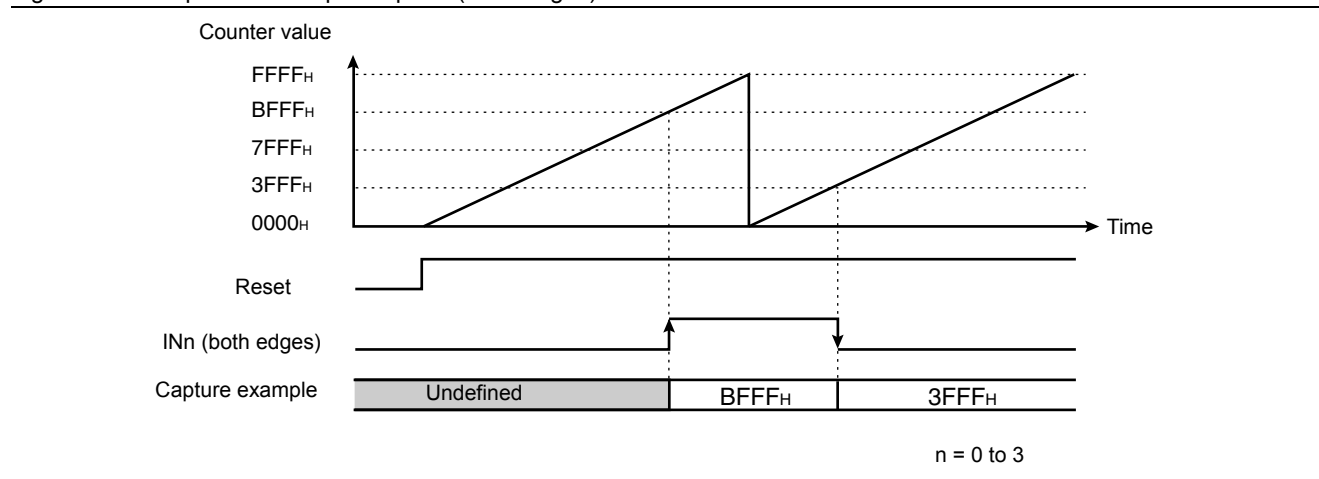


Figure 13-15. Operation of Input Capture (Both Edges)



13.7 Notes on Using 16-bit I/O Timer

This section explains the Notes on using the 16-bit I/O timer.

■ Note on Using 16-bit I/O Timer

- Notes on setting 16-bit free-run timer
- Do not change the count clock select bits (TCCSL: CLK2, CLK1, CLK0) during the operation in the 16-bit free-run timer (TCCSL: STOP = 0).
- The counter value of the 16-bit free-run timer is initialized to "0000_H" by reset.
- Stop the 16-bit free-run timer (TCCSL:STOP=1), then write the counter value to the timer data register (TCDT) directly.
- Always use a word instruction to write the timer data register (TCDT).
 - Operation delay by synchronization operation

The input capture generates delay of operation time because it synchronizes with the operation clock. After the input capture detects the trigger signal from the pin, it synchronizes with the machine clock and performs the capture operation.

13.8 Program Example of 16-bit I/O Timer

This section gives a program example of the 16-bit I/O timer.

■ Program Example of 16-bit I/O Timer

□ Processing specification

- The cycle of a signal input to the IN0 pin is measured.
- The 16-bit free-run timer 0 and input capture 0 are used.
- The rising edge is selected as the trigger to be detected.
- The machine clock (ϕ) is 24 MHz and the count clock of the free-run timer is $4/\phi$ (0.17 μ s).
- The timer overflow interrupt and capture interrupt of input capture 0 are used.
- The overflow interrupt of the 16-bit free-run timer is counted beforehand and used for the cycle calculation.
- The cycle can be determined from the following equation:

$$\begin{aligned} \text{Cycle} &= (\text{overflow count} \times "10000_H" + \text{nth IPCP0 value} - (\text{n-1})\text{th IPCP0 value}) \times \text{count clock cycle} \\ &= (\text{overflow count} \times 10000_H + \text{nth IPCP0 value} - (\text{n-1})\text{th IPCP0 value}) \times 0.17 \mu\text{s} \end{aligned}$$

□ Coding example

```

ICR09 EQU 0000B9H      ;Interrupt control register
ICR11 EQU 0000BBH      ;Interrupt control register
DDR2  EQU 000012H      ;Port 2 direction register
TCCSL EQU 007942H      ;Timer control status register
TCDT  EQU 007940H      ;Timer data register
ICS01 EQU 000050H      ;Input capture control status register
IPCP0 EQU 007920H      ;Input capture register 0
IVF0  EQU TCCSL:7      ;Timer overflow generation flag bit
ICP0  EQU ICS01:6      ;Valid edge detection flag bit
DATA  DSEG ABS=00H
      ORG 0100H
OV_CNT RW 1H
DATA  ENDS              ;Overflow count counter
;
;-----Main program-----
CODE          CSEG
START:
;
;Stack pointer (SP),
;already initialized
AND  CCR,#0BFH          ;Interrupt disable
MOV  I:ICR09,#00H       ;Interrupt level 0 (strongest)
MOV  I:ICR11,#00H       ;Interrupt level 0 (strongest)
MOV  I:DDR2,#00000000B  ;Port 2 direction setting
MOV  I:TCCSL,#01001010B ;Count enable, Counter clear,
                        ;Overflow, Interrupt enable,
                        ;Count clock 4/φ selection, Counter clear

```

```

      MOV    I:ICS01,#00010001B ;IN0 pin selection, External trigger,
                                ;IPCP0 rising edge
                                ;Without IPCP1 edge detection
                                ;Clear each valid edge detection flag
                                ;Input capture interrupt request enable

      MOV    ILM,#07H           ;Set ILM in PS to level 7
      OR     CCR,#40H           ;Interrupt enable

LOOP:
      :
      User processing
      :
      BRA    LOOP

;-----Interrupt program-----
WARIO:
      CLR    I:ICP0             ;Clear valid edge detection flag
      :                         ;Save OV-CNT and input capture value
      User processing
      :
      MOV    A,0                ;Clear overflow count counter
      MOV    OV_CNT,A           ;for next cycle measurement
      RETI                      ;Recover from interrupt

WARI1:
      CLR    I:IVF0             ;Clear timer overflow generation flag
      INC    OV_CNT             ;Increment overflow counter
      :
      User processing
      :
      RETI                      ;Recover from interrupt

CODE    ENDS

;-----Vector setting-----
VECT    CSEG    ABS=0FFH
      ORG    00FF78H            ;Setting vector to interrupt number #33 (21H)
                                ; (Input capture)

      DSL    WARIO
      ORG    00FF84H            ;Setting vector to interrupt number #30 (1EH)
                                ; (Overflow)

      DSL    WARI1
      ORG    00FFDCH            ;Reset vector setting
      DSL    START
      DB     00H                ;Setting to single-chip mode

VECT    ENDS
      END    START
  
```

14. 16-Bit Reload Timer



This chapter describes the functions and operation of the 16-bit reload timer.

- [14.1 Overview of the 16-bit Reload Timer](#)
- [14.2 Block Diagram of 16-bit Reload Timer](#)
- [14.3 Configuration of 16-bit Reload Timer](#)
- [14.4 Interrupts of 16-bit Reload Timer](#)
- [14.5 Explanation of Operation of 16-bit Reload Timer](#)
- [14.6 Notes on Using 16-bit Reload Timer](#)
- [14.7 Sample Program of 16-bit Reload Timer](#)

14.1 Overview of the 16-bit Reload Timer

The 16-bit reload timer has the following functions:

- The count clock can be selected from three internal clocks and external event clocks.
- A software trigger or external trigger can be selected as the start trigger.
- If the 16-bit timer register (TMR) underflows, an interrupt can be generated to the CPU. The 16-bit reload timer can be used as an interval timer by using an interrupt.
- If the TMR underflows, either the one-shot mode for stopping the TMR count operation, or the reload mode for reloading the value of the 16-bit reload register (TMRLR) to the TMR to continue the TMR count operation can be selected.
- The 16-bit reload timer corresponds to the EI²OS (correspond to 2 channels).
- The MB90990 series has two channels of 16-bit reload timers.

■ Operation Modes of 16-bit Reload Timer

Table 14-1. indicates the operation modes of the 16-bit reload timer.

Table 14-1. Operation Modes of 16-bit Reload Timer

Count clock	Start trigger	Operation performed upon underflow
Internal clock mode	Software trigger External trigger	One-shot mode Reload mode
Event count mode	Software trigger	One-shot mode Reload mode

■ Internal Clock Mode

- When the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) are set to "00_B", "01_B" or "10_B", the 16-bit reload timer is set in the internal clock mode.
- In the internal clock mode, the 16-bit reload timer decrements in synchronization with the internal clock.

- ❑ The count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) can be used to select three count clock cycles.
- ❑ The start trigger sets the edge detection for a software trigger or an external trigger.

■ Event Count Mode

- ❑ When the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) are set to "11_B", the 16-bit reload timer is set to the event count mode.
- ❑ In the event count mode, the 16-bit reload timer decrements in synchronization with the edge detection of the external event clock input to the TIN pin.
- ❑ A software trigger is selected as the start trigger.
- ❑ The 16-bit reload timer can be used as an interval timer by using a fixed cycle of the external clock.

■ Operation at Underflow

When the start trigger is inputted, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register, starting decrementing in synchronization with the count clock. When the 16-bit timer register (TMR) is decremented from "0000_H" to "FFFF_H", an underflow occurs.

- ❑ When an underflow occurs with an underflow interrupt enabled (TMCSR:INTE = 1), an underflow interrupt is generated.
- ❑ The 16-bit reload timer operation when an underflow occurs is set by the reload select bit in the timer control status register (TMCSR:RELD).

[One-shot mode (TMCSR: RELD=0)]

When an underflow occurs, the TMR count operation is stopped. When the next start trigger is inputted, the value set in the TMRLR is reloaded in the TMR, starting the TMR count operation.

- In the one-shot mode, during the TMR count operation, a High-level or Low-level rectangular wave is outputted from the TOT pin.
- The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of the rectangular wave.

[Reload mode (TMCSR: RELD=1)]

When an underflow occurs, the value set in the TMRLR is reloaded to the TMR, continuing the TMR count operation.

- In the reload mode, a toggle wave inverting the output level of the TOT pin is outputted each time an underflow occurs during the TMR count operation.
- The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of a toggle wave at starting the reload timer.
- The 16-bit reload timer can be used as an interval timer by using an underflow interrupt.

Table 14-2. Interval Time for the 16-bit Reload Timer

Count clock	Count clock period	Example of interval time *
Internal clock mode	$2^1T(0.083 \mu s) *$	0.083 μs to 5.46 ms
	$2^3T(0.33 \mu s) *$	0.33 μs to 21.8 ms
	$2^5T(1.3 \mu s) *$	1.3 μs to 87.4 ms
Event count mode	2^3T or more	0.33 μs or more

T: Machine cycle

*: The values in example of interval time and the parenthesized values are provided when the machine clock operates at 24 MHz.

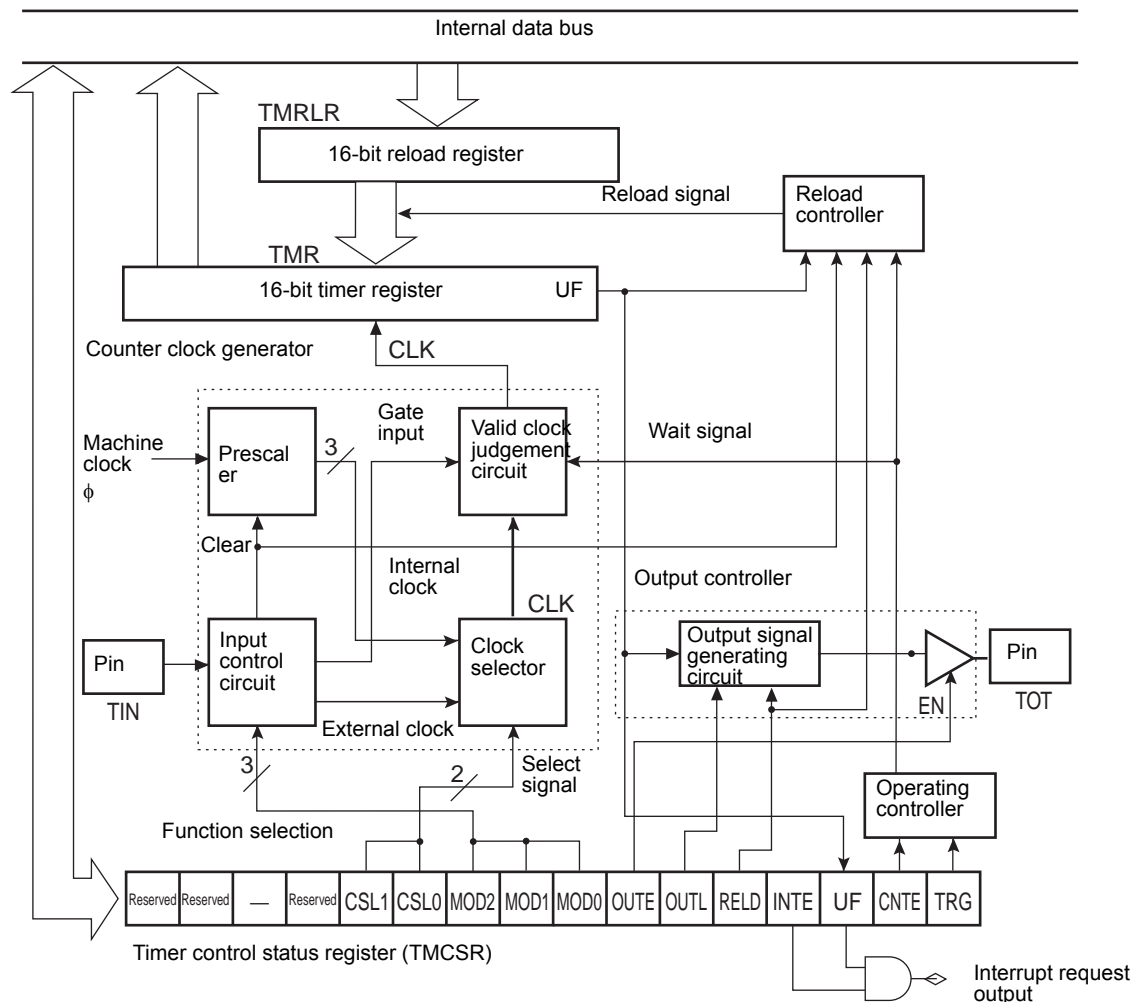
14.2 Block Diagram of 16-bit Reload Timer

The 16-bit reload timers 2 and 3 are composed of the following seven blocks:

- Count clock generator
- Reload controller
- Output controller
- Operation controller
- 16-bit timer register (TMR)
- 16-bit reload register (TMRLR)
- Timer control status register (TMCSR)

■ Block Diagram of 16-bit Reload Timer

Figure 14-1. Block Diagram of 16-bit Reload Timer



- ❑ Details of pins in block diagram

There are two channels for 16-bit reload timer.

The actual pin names, outputs to resources, and interrupt request numbers for each channel are as follows:

Table 14-3. Pin Names, Outputs to Resources, and Interrupt Request Numbers of 16-bit Reload Timer

	Reload timer 2	Reload timer 3
TIN pin	P82	P53
TOT pin	P83	P54
Output to resources	-	-
Interrupt request number	#19(13 _H)	#20(14 _H)

- ❑ Count clock generator

The count clock generator generates a count clock supplied to the 16-bit timer register (TMR) on the basis of the machine clock or external event clock.

- ❑ Reload controller

When the 16-bit reload timer starts operation or the TMR underflows, the reload controller reloads the value set in the 16-bit reload register (TMRLR) to the TMR.

- ❑ Output controller

The output controller inverts the output of the TOT pin at underflow and enables or disables the output of TOT pin.

- ❑ Operation controller

The operation controller starts or stops the 16-bit reload timer.

- ❑ 16-bit timer register (TMR)

The 16-bit timer register (TMR) is a 16-bit down counter. At read, the value being counted is read.

- ❑ 16-bit reload register (TMRLR)

The 16-bit reload register (TMRLR) sets the interval time of the 16-bit reload timer. When the 16-bit reload timer starts operation or the 16-bit timer register (TMR) underflows, the value set in the TMRLR is reloaded to the TMR.

- ❑ Timer control status register (TMCSR)

The timer control status register (TMCSR) selects the operation mode, sets the operation conditions, selects the start trigger, performs a start using the software trigger, selects the reload operation mode, enables or disables an interrupt request, sets the output level of the TOT pin, and sets the TOT output pin of the 16-bit reload timer.

14.3 Configuration of 16-bit Reload Timer

This section explains the pins, registers, and interrupt source of the 16-bit reload timer.

■ Pins of 16-bit Reload Timer

The pins of the 16-bit reload timer serve as general-purpose I/O ports. Table 14-4. shows the pin functions and the pin settings required to use the 16-bit reload timer.

Table 14-4. Pins of 16-bit Reload Timer

Pin name	Pin function	Pin Setting Required for Use in 16-bit Reload Timer
P82 / SIN0 / INT14R / TIN2	General-purpose I/O port/ UART input 0/ External interrupt 14/ 16-bit reload timer input 2	Port direction register: Setting for the input port (DDR8:D82=0) Serial control register: Setting for the reception disable (SCR0:RXE=0) Disable the external interrupt (external interrupt enable register ENIR1: EN14 = 0)
P83 / SOT0 / TOT2	General-purpose I/O port/ UART output 0/ 16-bit reload timer output 2	Serial control register: Setting for the transmission disable (SCR0:TXE=0) Timer control status register: Enable the timer output (TMCSR2: OUTE=1)
P53 / AN11 / TIN3	General-purpose I/O port/ A/D converter analog input 11/ 16-bit reload timer input 3	Port direction register: Setting for the input port (DDR5:D53=0) Analog input enable register: Setting for the prohibition (ADER5:ADE11=0)
P54 / AN12 / TOT3/ INT8	General-purpose I/O port/ A/D converter analog input 12/ 16-bit reload timer output 3/ External interrupt 8	Analog input enable register: Setting for the prohibition (ADER5:ADE12=0) Timer control status register: Enable the timer output (TMCSR3: OUTE=1) Disable the external interrupt:external interrupt enable register (ENIR1:EN8=0)

■ List of 16-bit Reload Timer Registers and Initial Value

- 16-bit reload timer 2 register

Figure 14-2. List of 16-bit Reload Timer 2 Register and Initial Value

	bit	15	14	13	12	11	10	9	8
Timer Control Status Register Upper (TMCSR2)		1	1	1	1	0	0	0	0
	bit	7	6	5	4	3	2	1	0
Timer Control Status Register Lower (TMCSR2)		0	0	0	0	0	0	0	0
	bit	15	14	13	12	11	10	9	8
16-bit Timer Register Upper (TMR2)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Timer Register Lower (TMR2)		X	X	X	X	X	X	X	X
	bit	15	14	13	12	11	10	9	8
16-bit Reload Register Upper (TMRLR2)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Reload Register Lower (TMRLR2)		X	X	X	X	X	X	X	X
X : Undefined									

□ 16-bit reload timer 3 register

Figure 14-3. List of 16-bit Reload Timer 3 Register and Initial Value

	bit	15	14	13	12	11	10	9	8
Timer Control Status Register Upper (TMCSR3)		1	1	1	1	0	0	0	0
	bit	7	6	5	4	3	2	1	0
Timer Control Status Register Lower (TMCSR3)		0	0	0	0	0	0	0	0
	bit	15	14	13	12	11	10	9	8
16-bit Timer Register Upper (TMR3)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Timer Register Lower (TMR3)		X	X	X	X	X	X	X	X
	bit	15	14	13	12	11	10	9	8
16-bit Reload Register Upper (TMRLR3)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Reload Register Lower (TMRLR3)		X	X	X	X	X	X	X	X
X : Undefined									

■ Generation of Interrupt Request from 16-bit Reload Timer

When the 16-bit reload timer is started and the count value of the 16-bit timer register is decremented from "0000_H" to "FFFF_H", an underflow occurs. When an underflow occurs, the UF bit in the timer control status register is set to "1" (TMCSR:UF). If an underflow interrupt is enabled (TMCSR:INTE = 1), an interrupt request is generated.

14.3.1 Timer Control Status Registers (Upper) (TMCSR:H)

The timer control status registers (Upper) (TMCSR:H) set the operation mode and count clock.

This section also explains the bit 7 in the timer control status registers (Lower) (TMCSR:L).

■ Timer Control Status Registers (Upper) (TMCSR:H)

Figure 14-4. Timer Control Status Registers (Upper) (TMCSR:H)

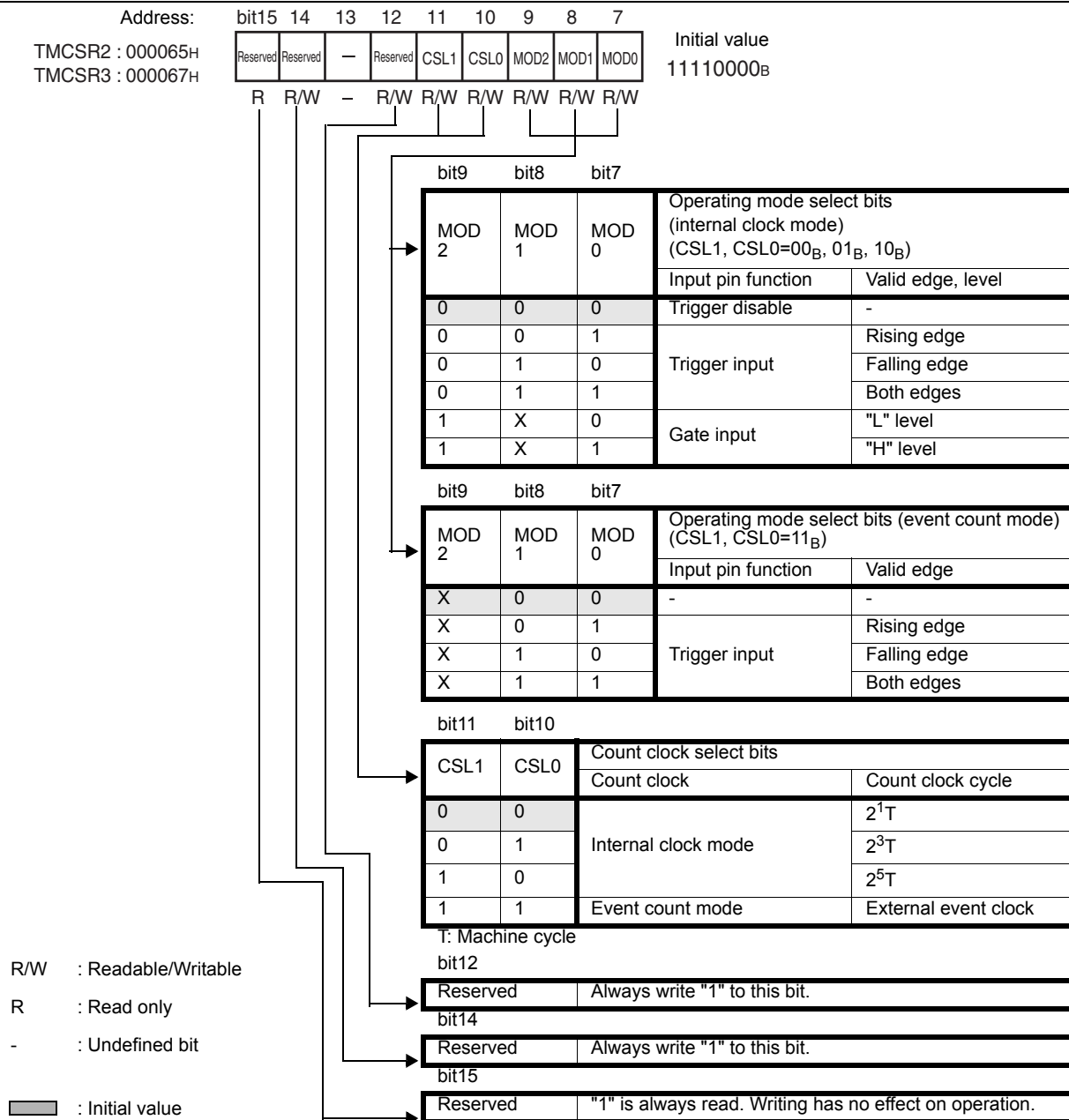


Table 14-5. Functions of Timer Control Status Registers (Upper) (TMCSR: H)

Bit name		Function
bit15	Reserved bit	Read: "1" is read. Write: No effect
bit14	Reserved bit	Always write "1" to this bit.
bit13	Undefined bit	Read: "1" is read. Write: No effect
bit12	Reserved bit	Always write "1" to this bit.
bit11, bit10	CSL1, CSL0: Count clock select bits	These bits select the count clock of the 16-bit reload timer. When set to anything other than "11_B" : These bits are counted by internal clock (internal clock mode). When set to "11_B" : The edge of the external event clock is counted (event count mode).
bit9 to bit7	MOD2, MOD1, MOD0: Operating mode select bits	These bits set the operation conditions of the 16-bit reload timer. [Internal clock mode] The MOD2 bit is used to select the function of the input pin. When MOD2 bit set to "0": The input pin functions as a trigger input. The MOD1 and MOD0 bits are used to select the edge to be detected. When the edge is detected, the value set in the 16-bit reload register (TMRLR) is reloaded in the 16-bit timer register (TMR), starting the count operation of the TMR. When MOD2 bit set to "1": The input pin functions as a gate input. The MOD1 bit is not used. The MOD0 bit is used to select the signal level (High or Low) to be detected. The count operation of the 16-bit timer register (TMR) is performed only when the signal level is inputted. [Event count mode] The MOD2 bit is not used. An external event clock is inputted from the input pin. The MOD1 and MOD0 bits are used to select the edge to be detected.

14.3.2 Timer Control Status Registers (Lower) (TMCSR: L)

The timer control status registers (Lower) (TMCSR:L) enable or disable the timer operation, check the generation of a software trigger or an underflow, enable or disable an underflow interrupt, select the reload mode, and set the output of the TOT pin.

■ Timer Control Status Registers (Lower) (TMCSR: L)

Figure 14-5. Timer Control Status Registers (Lower) (TMCSR: L)

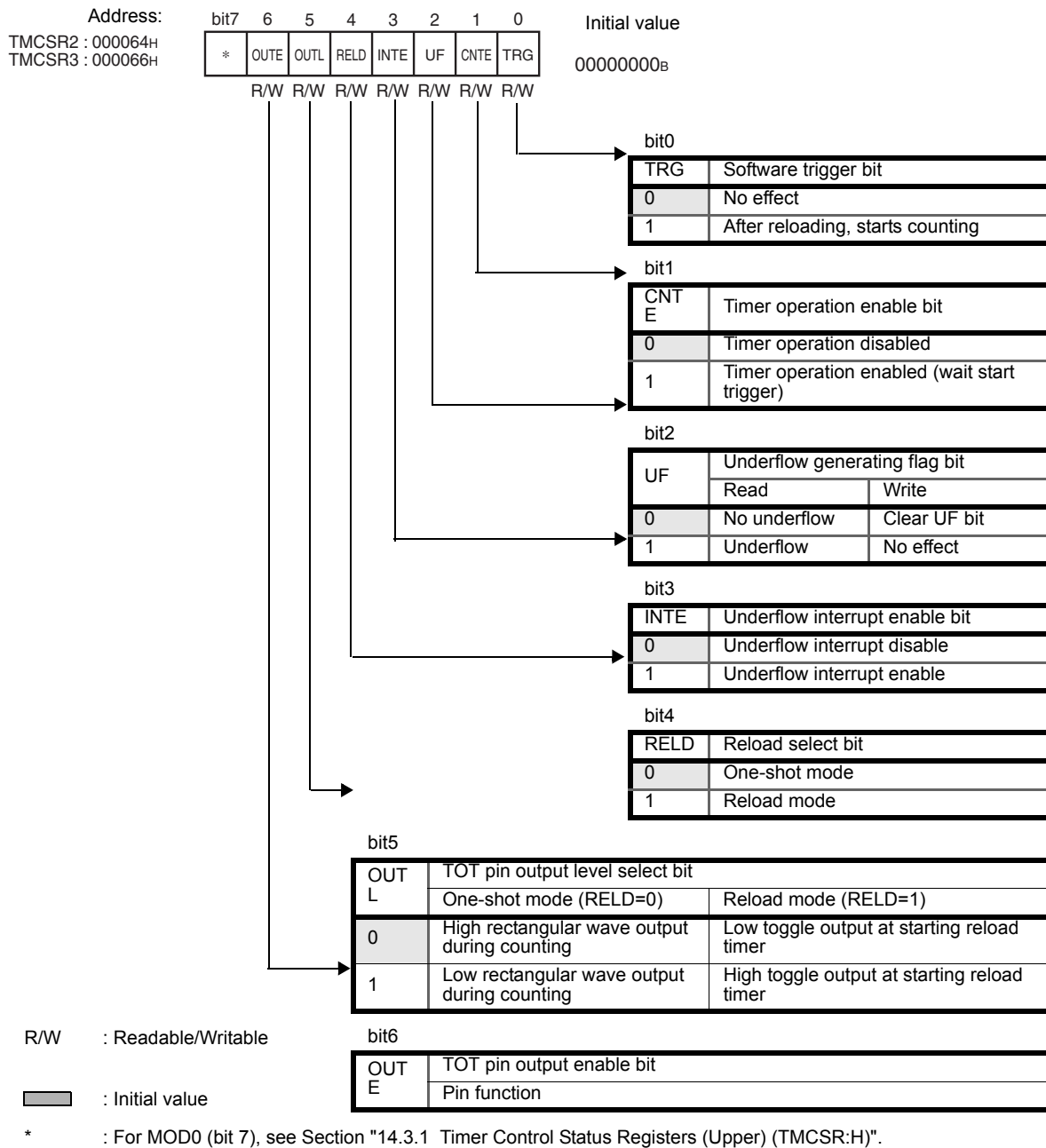


Table 14-6. Functions of Timer Control Status Registers (Lower) (TMCSR: L)

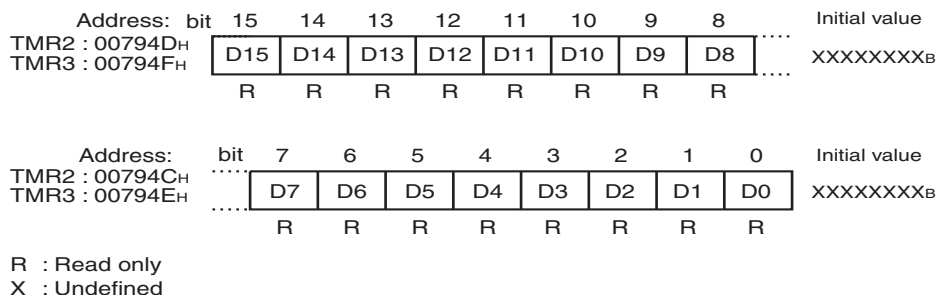
Bit Name		Function
bit6	OUTE: TOT pin output enable bit	This bit sets the function of the TOT pin of the 16-bit reload timer. When set to "0" : Functions as general-purpose I/O port When set to "1" : Functions as TOT pin of 16-bit reload timer
bit5	OUTL: TOT pin output level select bit	This bit sets the output level of the output pin of the 16-bit reload timer. The pin level is always inverted by changing the OUTL bit regardless of the CNTE bit. <One-shot mode (RELD = 0)> When set to "0" : Outputs High-level rectangular wave during TMR count operation When set to "1" : Outputs Low-level rectangular wave during TMR count operation <Reload mode (RELD = 1)> When set to "0" : Outputs Low-level toggle wave when 16-bit reload timer started When set to "1" : Outputs High-level toggle wave when 16-bit reload timer started
bit4	RELD: Reload select bit	This bit sets the reload operation at underflow. When set to "1" : At underflow, reloads value set in TMRLR to TMR, continuing count operation (reload mode) When set to "0" : At underflow, stops count operation (one-shot mode)
bit3	INTE: Underflow interrupt enable bit	This bit enables or disables an underflow interrupt. When an underflow occurs (TMCSR:UF = 1) with an underflow interrupt enabled (TMCSR:INTE = 1), an interrupt request is generated.
bit2	UF: Underflow generating flag bit	This bit indicates that the TMR underflows. When set to "0" : Clears this bit When set to "1" : No effect Read by read-modify-write (RMW) instructions : "1" is always read.
bit1	CNTE: Timer operation enable bit	This bit enables or disables the operation of the 16-bit reload timer. When set to "1" : 16-bit reload timer enters start trigger wait state. When the start trigger is inputted, the count operation of the TMR is restarted. When set to "0" : Stops count operation
bit0	TRG: Software trigger bit	This bit starts the 16-bit reload timer by software. The software trigger function works only when the timer operation is enabled (CNTE = 1). When set to "0" : Disabled. The state remains unchanged. When set to "1" : Reloads value set in 16-bit reload register (TMRLR) to 16-bit timer register (TMR), starting TMR count operation Read : "0" is always read.

14.3.3 16-bit Timer Registers (TMR)

The 16-bit timer registers are 16-bit down counters. At read, the value being counted is read.

■ 16-bit Timer Registers (TMR)

Figure 14-6. 16-bit Timer Registers (TMR)



When the timer operation is enabled (TMCSR:CNTEN = 1) and the start trigger is inputted, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR), starting the TMR count operation.

When the timer operation is disabled (TMCSR:CNTEN = 0), the TMR value is retained.

When the TMR value is counted down from "0000_H" to "FFFF_H" during the TMR count operation, an underflow occurs.

[Reload mode]

When the TMR underflows, the value set in the TMRLR is reloaded to the TMR, restarting the TMR count operation.

[One-shot mode]

When the TMR underflows, the TMR count operation is stopped, entering the start trigger input wait state. The TMR value is retained to "FFFF_H".

Notes:

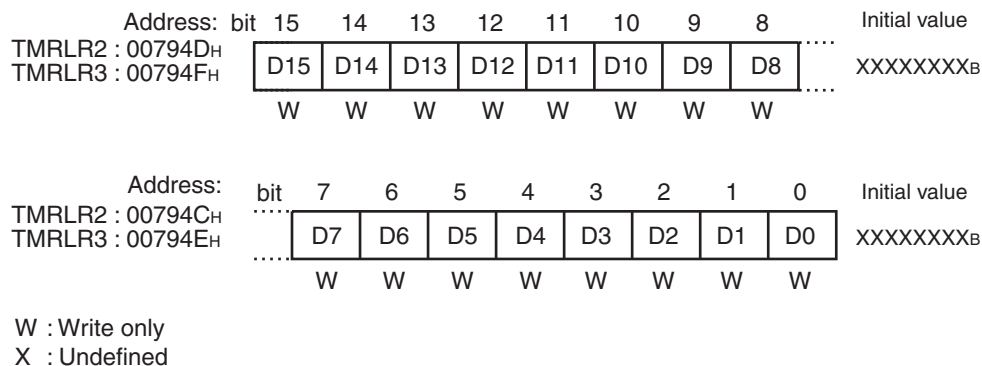
- The TMR can be read during the TMR count operation. However, always use the word instruction (MOVW).
- The TMR and the TMRLR are assigned to the same address. At write, the set value can be written to the TMRLR without affecting the TMR. At read, the TMR value being counted can be read.

14.3.4 16-bit Reload Registers (TMRLR)

The 16-bit reload registers set the value to be reloaded to the 16-bit timer register (TMR). When the start trigger is inputted, the value set in the 16-bit reload registers is reloaded to the TMR, starting the TMR count operation.

■ 16-bit Reload Registers (TMRLR)

Figure 14-7. 16-bit Reload Registers (TMRLR)



Set the 16-bit reload registers after disabling the timer operation (TMCSR:CNTE = 0). After completing setting of the 16-bit reload registers, enable the timer operation (TMCSR:CNTE = 1).

When the start trigger is inputted, the value set in the TMRLR is reloaded to the TMR, starting the TMR count operation.

Notes:

- Perform a write to the TMR after disabling the operation of the 16-bit reload timer (TMCSR:CNTE = 0). Always use the word instruction (MOVW).
- The TMRLR and the TMR are assigned to the same address. At write, the set value can be written to the TMRLR without affecting the TMR. At read, the TMR value being counted is read.
- Instructions, such as the INC/DEC instruction, which provide the read-modify-write (RMW) operation cannot be used.

14.4 Interrupts of 16-bit Reload Timer

The 16-bit reload timer generates an interrupt request when the 16-bit timer register (TMR) underflows.

■ Interrupts of 16-bit Reload Timer

When the value of the TMR is decremented from "0000_H" to "FFFF_H" during the TMR count operation, an underflow occurs. When an underflow occurs, the underflow generating flag bit in the timer control status register (TMCSR:UF) is set to "1". When an underflow interrupt is enabled (TMCSR:INTE = 1), an interrupt request is generated.

If setting "1" to the UF bit and writing "0" to it occur simultaneously, writing "0" has priority.

Table 14-7. Interrupt Control Bits and Interrupt Factors of 16-bit Reload Timer

	16-bit Reload Timer 2	16-bit Reload Timer 3
Interrupt request flag bit	TMCSR2: UF	TMCSR3: UF
Interrupt request enable bit	TMCSR2: INTE	TMCSR3: INTE
Interrupt factor	Underflow in TMR2	Underflow in TMR3

■ Correspondence between 16-bit Reload Timer Interrupt and EI²OS

For details of the interrupt number, interrupt control register, and interrupt vector address, see "3. Interrupts".

■ EI²OS Function of 16-bit Reload Timer

The 16-bit reload timers 2 and 3 correspond to the EI²OS function. An underflow in the TMR starts the EI²OS. However, the EI²OS is available only when other resources sharing the interrupt control register (ICR) do not use interrupts. The 16-bit reload timers 2 and 3 share the ICR04. When using the EI²OS in the 16-bit reload timers 2 and 3, it is necessary to disable the interrupt of the 16-bit reload timer sharing the interrupt control register.

14.5 Explanation of Operation of 16-bit Reload Timer

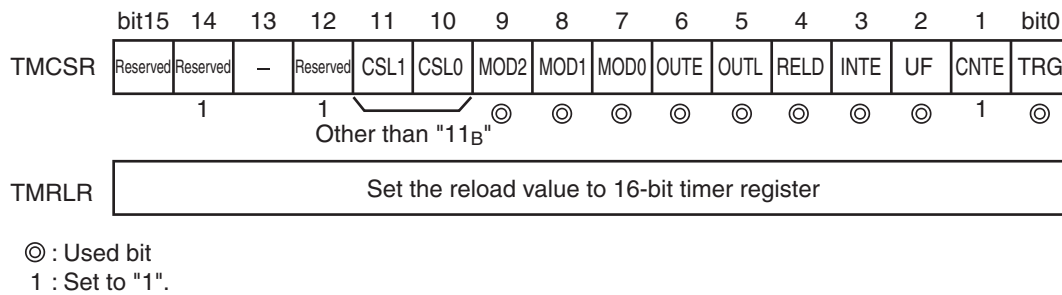
This section explains the setting of the 16-bit reload timer and the operation state of the counter.

■ Setting of 16-bit Reload Timer

- Setting of internal clock mode

Counting the internal clock requires the setting shown in [Figure 14-8](#).

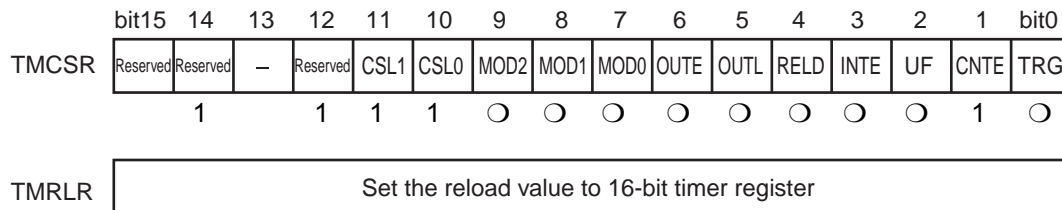
Figure 14-8. Setting of Internal Clock Mode



□ Setting of event count mode

Inputting an external event to operate the 16-bit reload timer requires the setting shown in [Figure 14-9](#).

Figure 14-9. Setting of Event Count Mode



Set the bit of DDR (port direction register) corresponding to the pin to be used as TIN pin to "0".

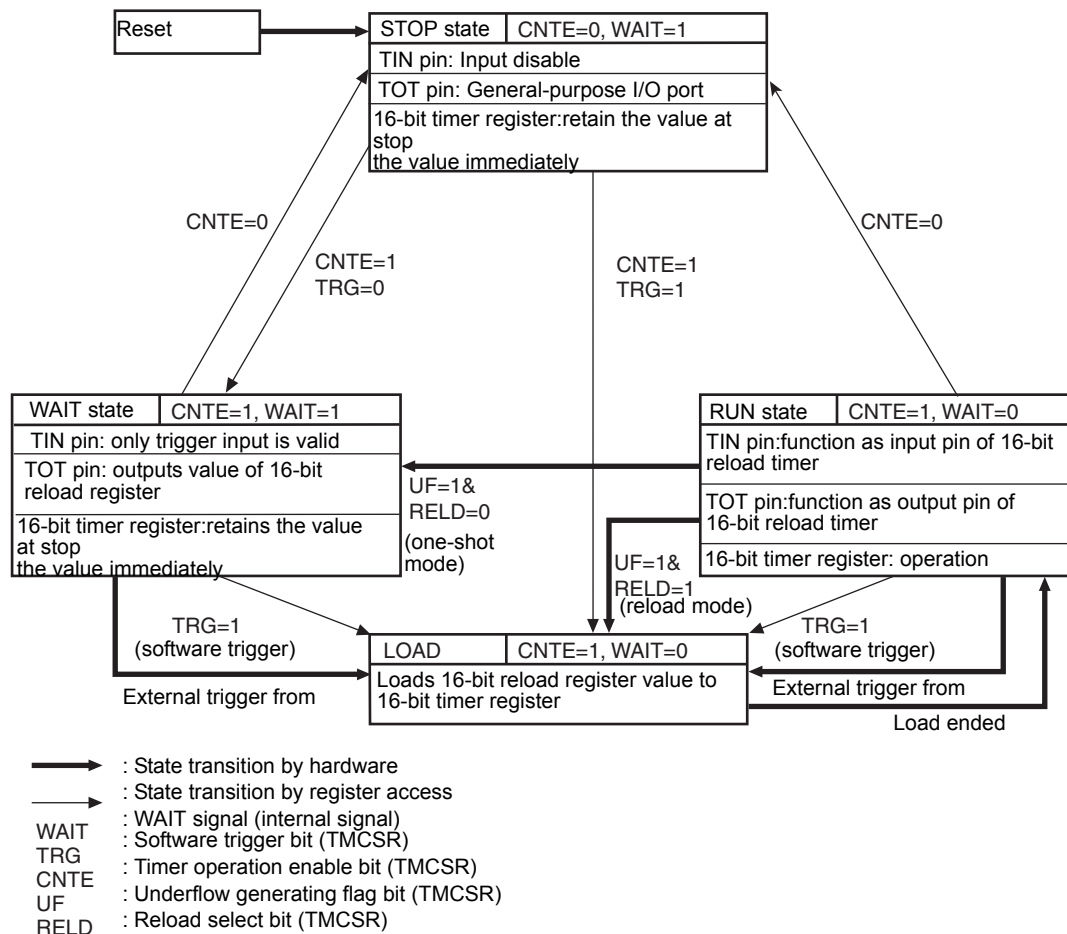
○: Used bit
 1 : Set to "1".

■ Operating State of 16-bit Timer Register

The operating state of the 16-bit timer register is determined by the timer operation enable bit in the timer control status register (TMCSR:CNTE) and the WAIT signal. The operating states include the stop state (STOP state), start trigger input wait state (WAIT state), and RUN state (RUN state).

Figure 14-10. shows the operating state transition diagram for the 16-bit timer registers.

Figure 14-10. Operating State Transition Diagram



14.5.1 Operation in Internal Clock Mode

In the internal clock mode, three operation modes can be selected by setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0). When the operation mode and reload mode are set, a rectangular wave or a toggle wave is outputted from the TOT pin.

■ Setting of Internal Clock Mode

- By setting the count clock select bits (CSL1, CSL0) in the timer control status register to "00_B", "01_B" and "10_B", the 16-bit reload timer is set to the internal clock mode.
- In the internal clock mode, the 16-bit timer register (TMR) decrements in synchronization with the internal clock.
- In the internal clock mode, three count clock cycles can be selected by setting the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0).

[Setting a reload value to TMR]

After the 16-bit reload timer is started, the value set in the TMRLR is reloaded to the TMR.

1. Disables the timer operation (TMCSR:CNTE = 0).
2. Sets a reload value to the TMR in the TMRLR.
3. Enables the timer operation (TMCSR:CNTE = 1).

Note:

It takes 1 T (T: machine cycle time), to reload the value set in the TMRLR to the TMR after the start trigger is inputted.

■ Operation as 16-bit Timer Register Underflows

When the value of the 16-bit timer register (TMR) is decremented from "0000_H" to "FFFF_H" during the TMR count operation, an underflow occurs.

- When an underflow occurs, the underflow generating flag bit in the timer control status register (TMCSR:UF) is set to "1".
- When the underflow interrupt enable bit in the timer control status register (TMCSR:INTE) is set to "1", an underflow interrupt is generated.
- The reload operation when an underflow occurs is set by the reload select bit in the timer control status register (TMCSR:RELD).

[One-shot mode (TMCSR:RELD = 0)]

When an underflow occurs, the count operation of the TMR is stopped, entering the start trigger input wait state. When the next start trigger is inputted, the TMR count operation is restarted.

In the one-shot mode, a rectangular wave is outputted from the TOT pin during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of a rectangular wave.

[Reload mode (TMCSR:RELD = 1)]

When an underflow occurs, the value set in the 16-bit reload register (TMRLR) is reloaded to the TMR, continuing the TMR count operation.

In the reload mode, a toggle wave inverting the output level of the TOT pin is outputted each time an underflow occurs during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of a toggle wave as the 16-bit reload timer is started.

■ Operation in Internal Clock Mode

In the internal clock mode, the operation mode select bits in the timer control status register (TMCSR:MOD2 to MOD0) can be used to select the operation mode. Disable the timer operation by setting the timer operation enable bit in the timer control status register (TMCSR:CNTE) to "0".

[Software trigger mode (MOD2 to MOD0=000_B)]

If the software trigger mode is set, start the 16-bit reload timer by setting the software trigger bit in the timer control status register (TMCSR:TRG) to "1". When the 16-bit reload timer is started, the value set in the TMRLR is reloaded to the TMR, starting the TMR count operation.

Note:

When both the timer operation enable bit in the timer control status register (TMCSR:CNTE) and the software trigger bit in the timer control status register (TMCSR:TRG) are set to "1", the 16-bit reload timer and the count operation of the TMR are started simultaneously.

However, timer activation at the gate input operation is enabled by the software trigger.

[External trigger mode (MOD2 to MOD0=001_B, 010_B, 011_B)]

When the external trigger mode is set, the 16-bit reload timer is started by inputting the external valid edge to the TIN pin. When the 16-bit reload timer is started, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR), starting the TMR count operation.

By setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0), the detected edge can be selected from the rising edge, falling edge, and both edges.

Note:

Refer to "Data sheet" for standard of the trigger pulse width input to the TIN pin and pulse width of a gate input.

Figure 14-13. Count Operation in External Trigger Mode (One-shot Mode)

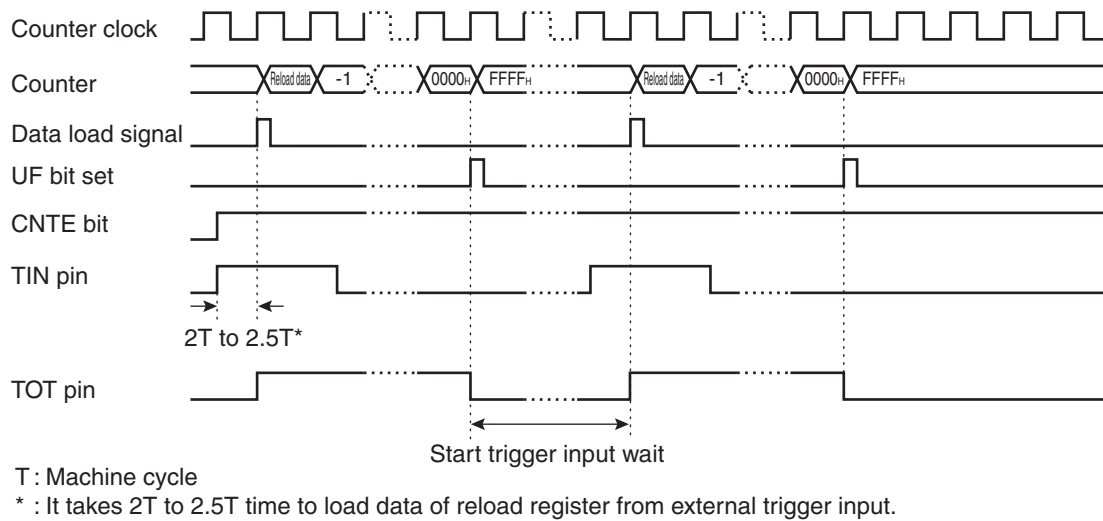
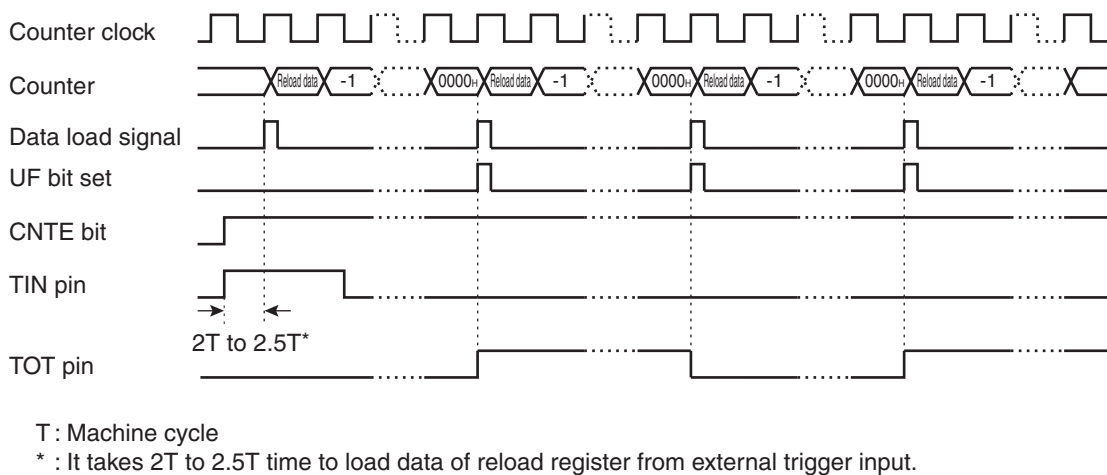


Figure 14-14. Count Operation in External Trigger Mode (Reload Mode)



[External gate input mode (MOD2 to MOD0=1x0_B, 1x1_B)]

When the external gate input mode is set, start the 16-bit reload timer by setting the software trigger bit in the timer control status register (TMCSR:TRG) to "1". When the 16-bit reload timer is started, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR).

- After the 16-bit reload timer is started, the count operation of the TMR is performed while the set gate input level is inputted to the TIN pin.
- The gate input level (High or Low) can be selected by setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0).

Figure 14-15. Count Operation in External Gate Input Mode (One-shot Mode)

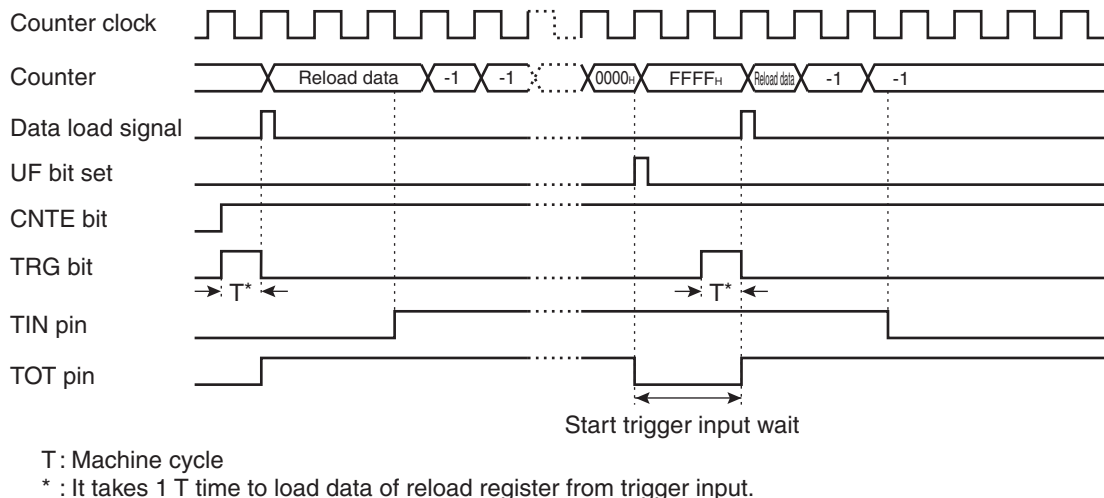
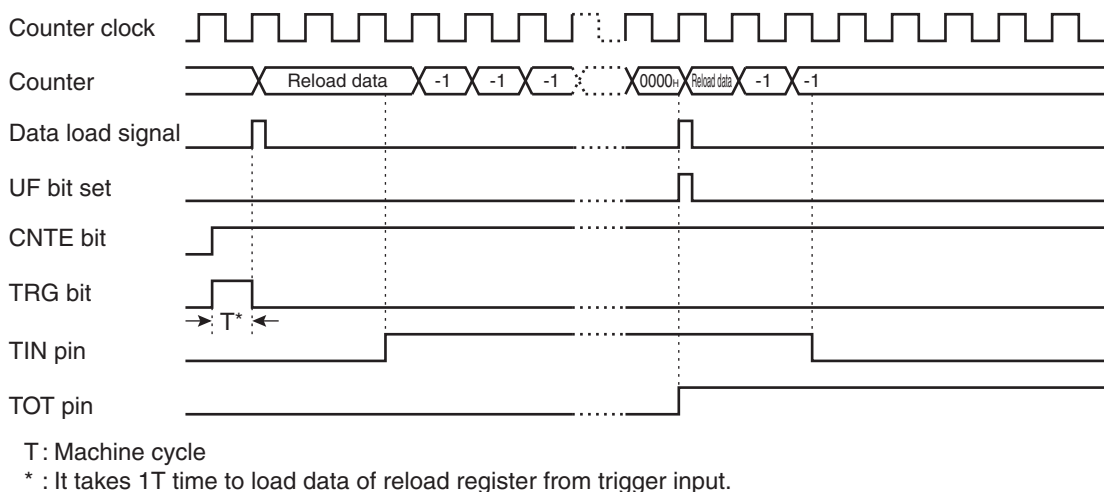


Figure 14-16. Count Operation in External Gate Input Mode (Reload Mode)



14.5.2 Operation in Event Count Mode

In the event count mode, after the 16-bit reload timer is started, the edge of the signal input to the TIN pin is detected to perform the count operation of the 16-bit timer register (TMR). When the operation mode and the reload mode are set, a rectangular wave or a toggle wave is outputted from the TOT pin.

■ Setting of Event Count Mode

- The 16-bit reload timer is placed in the event count mode by setting the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) to "11_B".
- In the event count mode, the TMR decrements in synchronization with the edge detection of the external event clock input to the TIN pin.

[Setting initial value of counter]

After the 16-bit reload timer is started, the value set in the TMRLR is reloaded to the TMR.

1. Disables the operation of the 16-bit reload timer (TMCSR:CNTEN = 0).
2. Sets a reload value to the TMR in the TMRLR.
3. Enables the operation of the 16-bit reload timer (TMCSR:CNTEN = 1).

Note:

It takes 1 T (T: machine cycle) to load the value set in the TMRLR to the TMR after the start trigger is inputted.

■ Operation as 16-bit Timer Register Underflows

When the value of the 16-bit timer register (TMR) is decremented from "0000_H" to "FFFF_H" during the TMR count operation, an underflow occurs.

- When an underflow occurs, the underflow generating flag bit in the timer control status register (TMCSR:UF) is set to "1".
- When the underflow interrupt enable bit in the timer control status register (TMCSR:INTE) is set to "1", an underflow interrupt is generated.
- The reload operation when an underflow occurs is set by the reload select bit in the timer control status register (TMCSR:RELD).

[One-shot mode (TMCSR: RELD=0)]

When an underflow occurs, the TMR count operation is stopped, entering the start trigger input wait state. When the next start trigger is inputted, the TMR count operation is restarted.

In the one-shot mode, a rectangular wave is outputted from the TOT pin during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of the rectangular wave.

[Reload mode (TMCSR: RELD=1)]

When an underflow occurs, the value set in the TMRLR is reloaded to the TMR, continuing the TMR count operation.

In the reload mode, a toggle wave inverting the output level of the TOT pin is outputted each time an underflow occurs during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of the toggle wave when the 16-bit reload timer is started.

■ Operation in Event Count Mode

The operation of the 16-bit reload timer is enabled by setting the timer operation enable bit in the timer control status register (TMCSR:CNTE) to "1". When the software trigger bit in the timer control status register (TMCSR:TRG) is set to "1", the 16-bit reload timer is started. When the 16-bit reload timer is started, the value set in the 16-bit reload register (TMRLR) is loaded to the 16-bit timer register (TMR), starting the TMR count operation. After the 16-bit reload timer is started, the edge of the external event clock input to the TIN pin is detected to perform the TMR count operation.

Timer activation in event mode is enabled by the software trigger.

By setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0), the detected edge can be selected from the rising edge, falling edge, and both edges.

Note:

For "H" width and "L" width of the clock input to the TIN pin, conform to standard of "Data sheet".

Figure 14-17. Count Operation in Event Count Mode (One-shot Mode)

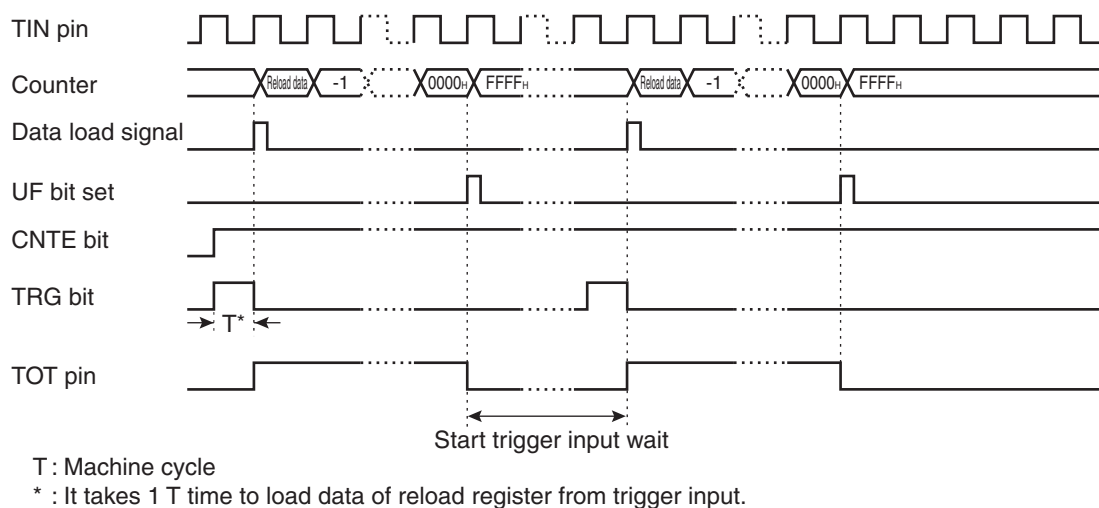
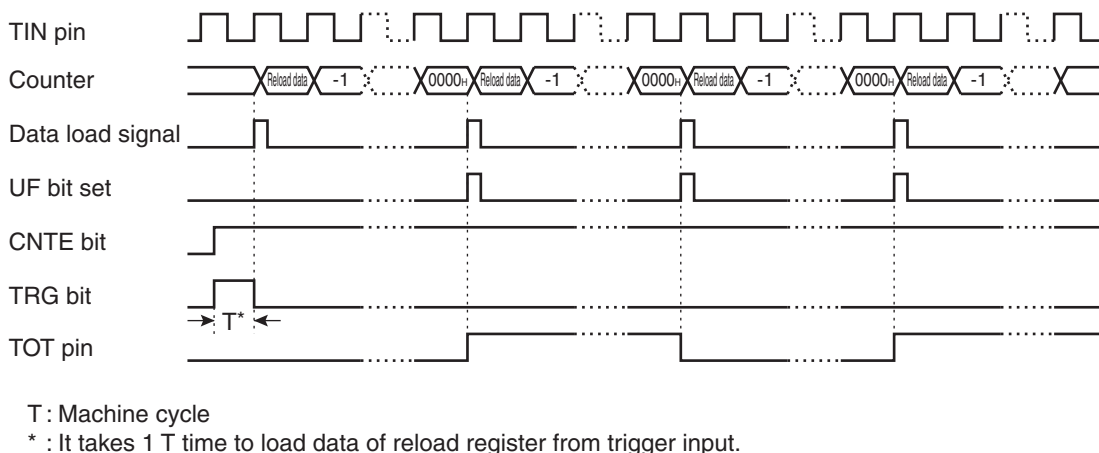


Figure 14-18. Count Operation in Event Count Mode (Reload Mode)



14.6 Notes on Using 16-bit Reload Timer

This section explains the notes when using the 16-bit reload timer.

■ Notes on Using 16-bit Reload Timer

□ Notes on setting by program

Set the 16-bit reload register (TMRLR) after disabling the timer operation (TMCSR:CNTE = 0).

The 16-bit timer register (TMR) can be read during the TMR count operation. However, always use the word instruction.

Change the CSL1 and CSL0 bits in the TMCSR after disabling the timer operation (TMCSR:CNTE = 0).

Notes on interrupt

When the UF bit in the TMCSR is set to "1" and the underflow interrupt output is enabled (TMCSR:INTE = 1), it is impossible to return from interrupt processing. Always clear the UF bit. However, when the EI²OS is used, the UF bit is cleared automatically.

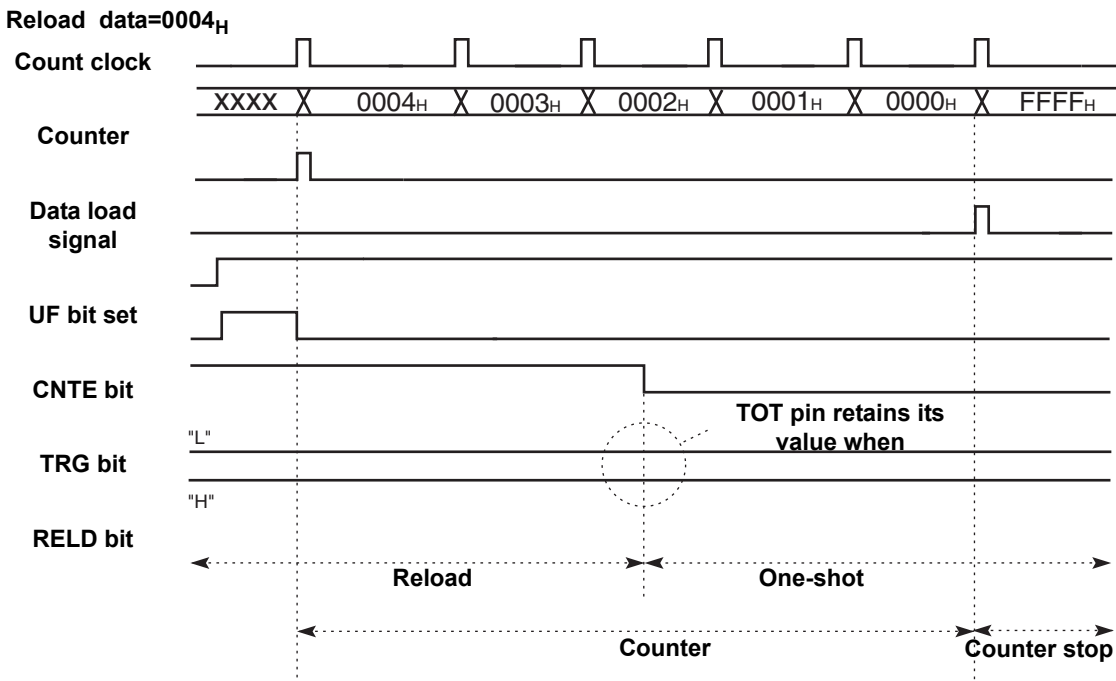
When using the EI²OS in the 16-bit reload timer, it is necessary to disable the interrupt of the 16-bit reload timer that shares the interrupt control register (ICR).

□ Timer operation when changing RELD bit during counter operation

When changing the RELD bit during counter operation, the operation is continued and the TOT pin output value is retained until next underflow occurs.

An example of operation when changing the RELD bit during counter is shown below. When the bit is "1", it indicates the reload mode. When the bit is "0", it indicates the one-shot mode.

Figure 14-19. Changing RELD bit from "1" to "0" in OUTL= 0 and TOT= L during Counter Operation



220

(1) Changing RELD bit from "1" to "0" (reload mode to one-shot mode)

The counter is stopped at FFFF_H for the counter value and at value which counter is stopped (initial value) for TOT pin (Example : Figure 14.6-4).

(2) Changing RELD bit from "0" to "1" (one-shot mode to reload mode)

The counter is stopped at the reload data value for the counter value and at value which counter is stopped (initial value) for TOT pin (Example : Figure 14.6-5).

Figure 14-22. Changing RELD bit from "1" to "0" Simultaneously with Underflow

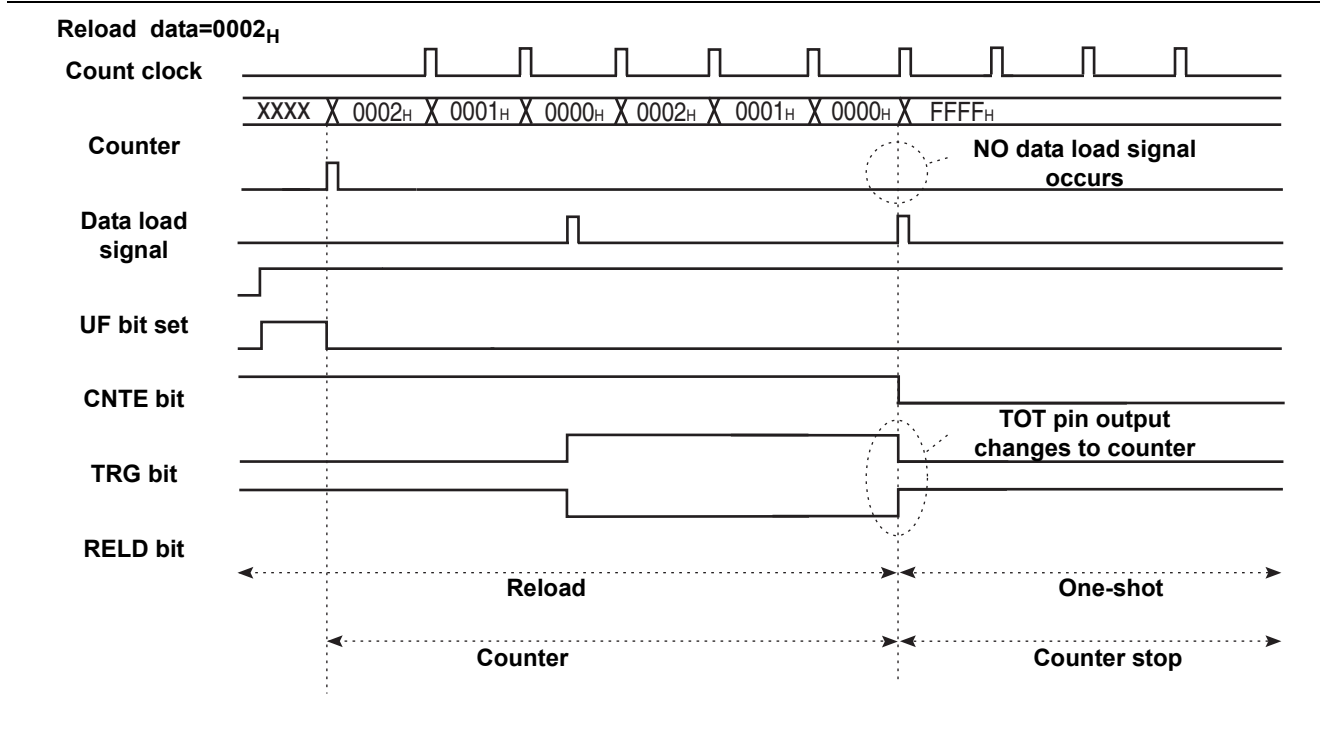
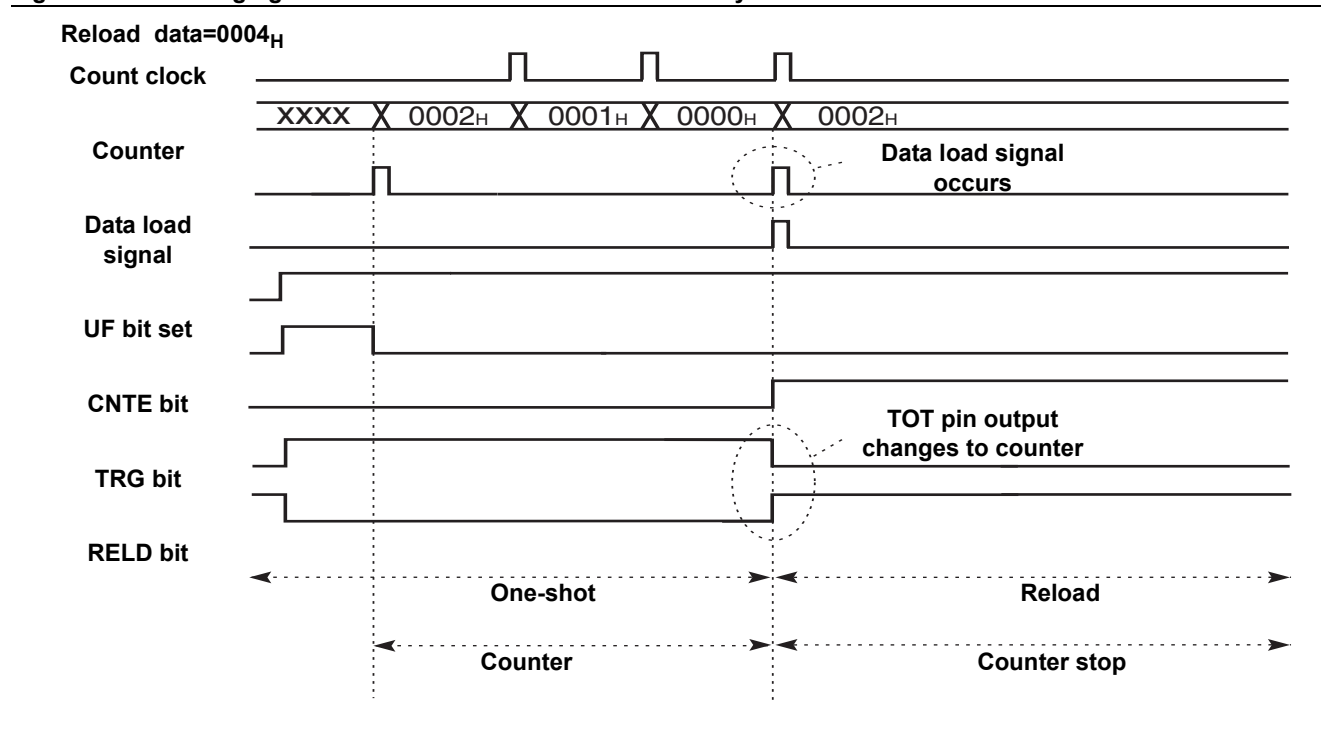


Figure 14-23. Changing RELD bit from "0" to "1" Simultaneously with Underflow



14.7 Sample Program of 16-bit Reload Timer

This section gives a program example of the 16-bit reload timer operated in the internal clock mode and the event count mode:

■ Program Example in Internal Clock Mode

□ Processing specification

The 24 ms interval timer interrupt is generated by the 16-bit reload timer 2.

The repeated interrupts are generated in the reload mode.

The timer is started using the software trigger instead of the external trigger input.

EI²OS is not used.

The machine clock is 24 MHz; the count clock is 1.33 μs.

□ Coding example

```
ICR04 EQU 0000B4H           ;Interrupt control register for 16-bit
                               ;reload timer
TMCSR2 EQU 000064H           ;Timer control status register
TMR2 EQU 00794CH             ;16-bit timer register
TMRLR2 EQU 00794CH           ;16-bit reload register
UF2 EQU TMCSR2:2             ;Interrupt request flag bit
CNTE2 EQU TMCSR2:1           ;Counter operation enable bit
TRG2 EQU TMCSR2:0            ;Software trigger bit
;-----Main program-----
CODE CSEG
;                               ;Stack pointer (SP), already initialized
AND CCR,#0BFH                ;Interrupts disabled
```

```

MOV  I:ICR04,#00H      ;Interrupt level 0 (highest)
CLRB I:CNT2           ;Counter suspended
MOVW I:TMRLR2,#4650H   ;Set data for 24 ms timer
MOVW I:TMCSR2,#0000100000011011B
                                ;Operation of interval timer,
                                clock = 1.33 μs
                                ;External trigger disabled, external
                                output disabled
                                ;Reload mode selected, interrupt enabled
                                ;Interrupt flag cleared, count started
MOV  ILM,#07H          ;ILM in PS set to level 7
OR   CCR,#40H          ;Interrupts enabled
LOOP:
    :
    Processing by user
    :
    BRA  LOOP
;-----Interrupt program-----
WARI:
    CLR  I:UF2          ;Interrupt request flag cleared
    :
    :
    Processing by user
    :
    :
    RETI                ;Return from interrupt

CODE  ENDS
;-----Vector setting-----
VECT  CSEG ABS=0FFH
      ORG  00FFB0H      ;Set vector to interrupt #19(13H)
      DSL  WARI
      ORG  00FFDCH      ;Reset vector set
      DSL  START
      DB   00H          ;Set to single-chip mode
VECT  ENDS
      END  START

```

■ Program Example in Event Count Mode

□ Processing specification

An interrupt is generated when rising edges of the pulse input to the external event input pin are counted 10000 times by the 16-bit reload timer 2.

Operation is performed in the one-shot mode.

The rising edge is selected for the external trigger input.

EI²OS is not used.

□ Coding example

```

ICR04 EQU 0000B4H      ;Interrupt control register for 16-bit
                        ;reload timer
TMCSR2 EQU 000064H      ;Timer control status register
TMR2   EQU 00794CH      ;16-bit timer register
TMRLR2 EQU 00794CH      ;16-bit reload register
DDR8   EQU 000018H      ;Port data register
UF2    EQU TMCSR2:2      ;Interrupt request flag bit
CNT2   EQU TMCSR2:1      ;Counter operation enable bit
TRG2   EQU TMCSR2:0      ;Software trigger bit
;-----Main program-----

```

```

CODE    CSEG
;      :                               ;Stack pointer (SP), already initialized
;      :                               ;Used at software starting mode(ADCS1: STS1, STS0 = 00B)

        AND    CCR,#0BFH              ;Interrupts disabled
        MOV    I:ICR04,#00H           ;Interrupt level 0 (highest)
        MOV    I:DDR8,00H             ;Sets P82/TIN2 pin to input
        CLRB   I:CNTE0                ;Counter suspended
        MOVW   I:TMRLR2,#2710H;Reload value set to 10000 times
        MOVW   I:TMCSR2,#0000110001001011B
                                   ;Counter operation, rising edge,
                                   ;and external output disabled
                                   ;One-shot mode selected, interrupt enabled
                                   ;Interrupt flag cleared, count started
        MOV    ILM,#07H               ;Set ILM in PS to level 7
        OR     CCR,#40H               ;Interrupts enabled
LOOP:
        :
        Processing by user
        :
        BRA    LOOP
;-----Interrupt program-----
WARI:
        CLR    I:UF2                  ;Interrupt request flag cleared
        :
        :
        Processing by user
        :
        :
        RETI                          ;Return from interrupt

CODE    ENDS
;-----Vector setting-----
VECT    CSEG    ABS=0FFH
        ORG    00FFB0H                ;Set vector to interrupt #19 (13H)
        DSL    WARI
        ORG    00FFDCH                ;Reset vector set
        DSL    START
        DB     00H                    ;Set to single-chip mode
VECT    ENDS
        END    START
  
```


15. Watch Timer



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

15.1 Overview of Watch Timer

15.2 Block Diagram of Watch Timer

15.3 Configuration of Watch Timer

15.4 Watch Timer Interrupt

15.5 Explanation of Operation of Watch Timer

15.6 Program Example of Watch Timer

15.1 Overview of Watch Timer

The watch timer is a 15-bit free-run counter that increments in synchronization with the sub clock.

- Eight interval times can be selected and an interrupt request can be generated for each interval time.
- An operation clock can be supplied to the oscillation stabilization wait time timer of the sub clock and the watchdog timer.
- The sub clock is always used as a count clock regardless of the settings of the clock select register (CKSCR).

■ Interval Timer Function

- When the watch timer reaches the interval time set by the interval time select bits (WTC:WTC2 to WTC0), the bit corresponding to the interval time of the watch timer counter overflows (carries) and the overflow flag bit is set (WTC:WTOF = 1).
- When the overflow flag bit is set (WTC:WTOF = 1) with interrupt enabled when an overflow occurs (WTC:WTIE = 1), an interrupt request is generated.

- The interval time of the watch timer can be selected from eight types shown in [Table 15-1](#).

Table 15-1. Interval Times of Watch Timer

Sub clock Cycle	Interval Time
1/SCLK(80 μs)	$2^8/\text{SCLK}(20.48\text{ms})$
	$2^9/\text{SCLK}(40.96\text{ms})$
	$2^{10}/\text{SCLK}(81.92\text{ms})$
	$2^{11}/\text{SCLK}(163.84\text{ms})$
	$2^{12}/\text{SCLK}(327.68\text{ms})$
	$2^{13}/\text{SCLK}(655.36\text{ms})$
	$2^{14}/\text{SCLK}(1310.72\text{ms})$
	$2^{15}/\text{SCLK}(2621.44\text{ms})$

SCLK: Sub clock frequency

The parenthesized values are provided when the sub clock operates at 12.5 kHz.

■ Cycle of Clock Supply

The watch timer supplies an operation clock to the oscillation stabilization wait time timer of the sub clock and the watchdog timer. [Table 15-2](#) shows the cycles of clocks supplied from the watch timer.

Table 15-2. Cycle of Clock Supplied from Watch Timer

Where to Supply Clock	Clock Cycle
Oscillation stabilization wait time of sub clock	$2^{14}/\text{SCLK}(1.31\text{s})$
Watchdog timer	$2^{10}/\text{SCLK}(82\text{ms})$
	$2^{13}/\text{SCLK}(655\text{ms})$
	$2^{14}/\text{SCLK}(1.31\text{s})$
	$2^{15}/\text{SCLK}(2.62\text{s})$

SCLK: Sub clock frequency

The parenthesized values are provided when the sub clock operates at 12.5 kHz.

Note:

The frequency of the sub clock (SCLK) is a value for 8 division/4 division of the CR oscillation clock. The division ratio is set by the SCDS bit of the PLL/sub clock control register (PSCCR).

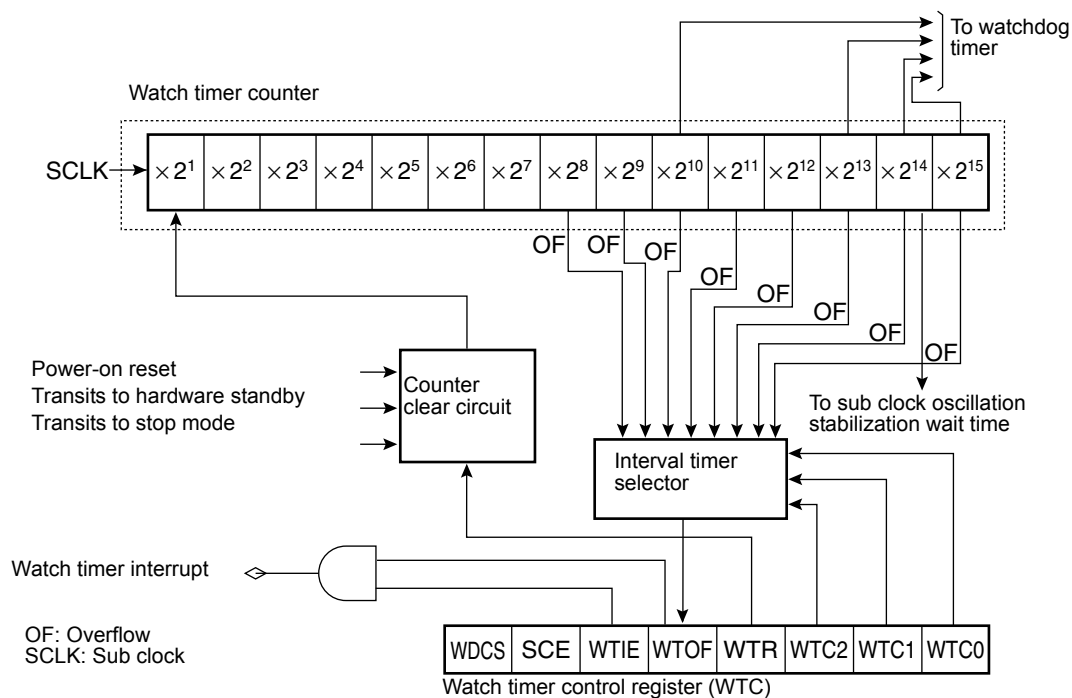
15.2 Block Diagram of Watch Timer

The watch timer consists of the following blocks:

- Watch timer counter
- Counter clear circuit
- Interval timer selector
- Watch timer control register (WTC)

■ Block Diagram of Watch Timer

Figure 15-1. Block Diagram of Watch Timer



The actual interrupt request number of the watch timer is as follows:

Interrupt request number: #27(1B_H)

- Watch timer counter

The watch timer counter is a 15-bit up counter that uses the sub clock (SCLK) as a count clock.

- Counter clear circuit

The counter clear circuit clears the watch timer counter.

□ Interval timer selector

The interval timer selector sets the overflow flag bit when the watch timer counter reaches the interval time set in the watch timer control register (WTC).

□ Watch timer control register (WTC)

The watch timer control register (WTC) selects the interval time, clears the watch timer counter, enables or disables an interrupt, checks the overflow (carry) state, and clears the overflow flag bit.

15.3 Configuration of Watch Timer

This section explains the registers and interrupt factors of the watch timer.

■ List of Registers and Initial Values of Watch Timer

Figure 15-2. List of Registers and Initial Values of Watch Timer

Watch timer control register (WTC)		bit	7	6	5	4	3	2	1	0
Address: 0000AA _H			1	×	0	0	1	0	0	0
×: Undefined										

■ Generation of Interrupt Request from Watch Timer

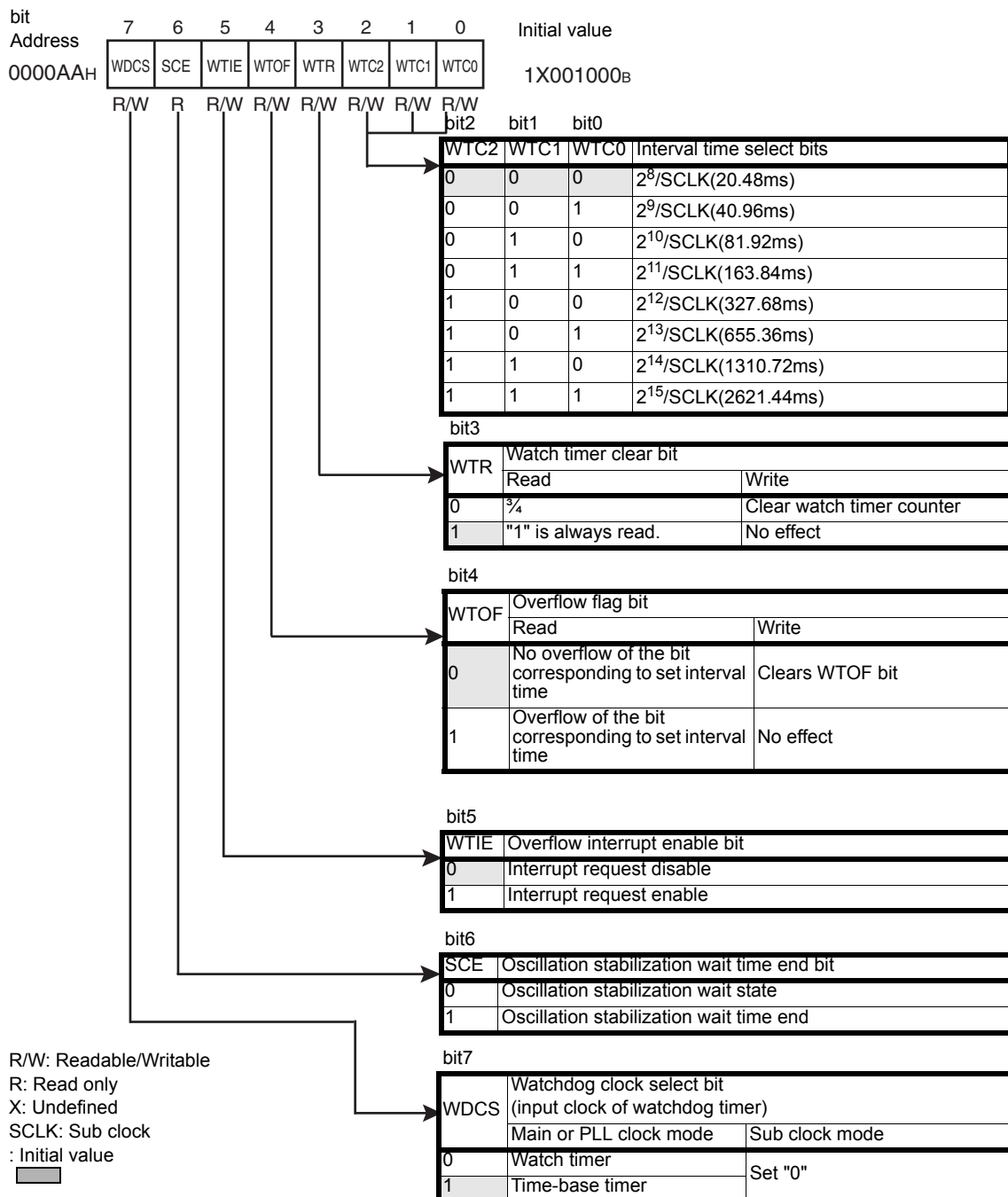
- When the interval time set by the interval time select bits (WTC:WTC2 to WTC0) is reached, the overflow flag bit (WTC:WTOF) is set to "1".
- When the overflow flag bit is set (WTC:WTOF = 1) with interrupt enabled when the watch timer counter overflows (carries) (WTC:WTIE = 1), an interrupt request is generated.

15.3.1 Watch Timer Control Register (WTC)

This section explains the functions of the watch timer control register (WTC).

■ Watch Timer Control Register (WTC)

Figure 15-3. Watch Timer Control Register (WTC)



The parenthesized values are provided when sub clock operates at 12.5 kHz.

Table 15-3. Functions of Watch Timer Control Register (WTC)

Bit Name		Function
bit7	WDCS: Watchdog clock select bit	<p>This bit selects the operation clock of the watchdog timer. <Main clock mode or PLL clock mode> When set to "0": Selects output of watch timer as operation clock of watchdog timer. When set to "1": Selects output of time-base timer as operation clock of watchdog timer. <Sub clock mode> Always set this bit to "0" to select the output of the watch timer. Note: The watch timer and the time-base timer operate asynchronously. When the WDCS bit is changed from "0" to "1", the watchdog timer may run fast. The watchdog timer must be cleared before and after changing the WDCS bit.</p>
bit6	SCE: Oscillation stabilization wait time end bit	<p>This bit indicates that the oscillation stabilization wait time of the sub clock ends. When cleared to "0": Sub clock in oscillation stabilization wait state When set to "1": Sub clock oscillation stabilization wait time ends The oscillation stabilization wait time of the sub clock is fixed at $2^{14}/\text{SCLK}$ (SCLK: sub clock frequency).</p>
bit5	WTIE: Overflow interrupt enable bit	<p>This bit enables or disables generation of an interrupt request when the watch timer counter overflows (carries). When set to "0": Interrupt request not generated even at overflow (WTOF=1) When set to "1": Interrupt request generated at overflow (WTOF = 1)</p>
bit4	WTOF: Overflow flag bit	<p>This bit is set to "1" when the counter value of the watch timer reaches the value set by the interval time select bits. When an overflow (carry) occurs (WTOF = 1) with interrupt request enabled (WTIE = 1), an interrupt request is generated. When set to "0": Clears this bit When set to "1": No effect The overflow flag bit is set to "1" when the bit of the watch timer counter corresponding to the interval time set by the interval time select bits (WTC2 to WTC0) overflows (carries). The read value by a read-modify-write (RMW) instruction is always "1".</p>
bit3	WTR: Watch timer clear bit	<p>This bit clears the watch timer counter. When set to "0": Clears watch timer counter to "0000_H". When set to "1": No effect Read: "1" is always read.</p>
bit2 to bit0	WTC2, WTC1, WTC0: Interval time select bits	<p>These bits set the interval time of the watch timer. When the interval time set by the WTC2 to WTC0 bits is reached, the corresponding bit of the watch timer counter overflows (carries) and the overflow flag bit is set (WTC:WTOF = 1). To set the WTC2 to WTC0 bits, set the WTOF bit to "0" at the same time.</p>

15.4 Watch Timer Interrupt

When the interval time is reached with the watch timer interrupt enabled, the overflow flag bit is set to "1" and an interrupt request is generated.

■ Watch Timer Interrupt

Table 15-4. shows the interrupt control bits and interrupt factors of the watch timer.

Table 15-4. Interrupt Control Bits of Watch Timer

	Watch Timer
Interrupt factor	Interval time of watch timer counter
Interrupt request flag bit	WTC: WTOF(overflow flag bit)
Interrupt factor enable bit	WTC: WTIE

- ❑ When the value set by the interval time select bits (WTC2 to WTC0) in the watch timer control register (WTC) is reached, the overflow flag bit in the WTC register is set to "1" (WTC:WTOF = 1).
- ❑ When the overflow flag bit is set (WTC:WTOF = 1) with the watch timer interrupt enabled (WTC:WTIE = 1), an interrupt request is generated.
- ❑ At interrupt processing, set the WTOF bit to "0" and cancel the interrupt request.

■ Watch Timer Interrupt and EI²OS Transfer Function

- ❑ The watch timer does not correspond to the EI²OS function.
- ❑ For details of the interrupt number, interrupt control register, and interrupt vector address, see "3. Interrupts".

15.5 Explanation of Operation of Watch Timer

The watch timer operates as an interval timer or an oscillation stabilization wait time timer of sub clock. It also supplies an operation clock to the watchdog timer.

■ Watch Timer Counter

The watch timer counter continues incrementing in synchronization with the sub clock (SCLK) while the sub clock (SCLK) is operating.

- ❑ Clearing watch timer counter

The watch timer counter is cleared to "0000_H" when:

A power-on reset occurs.

The mode transits to the stop mode.

The watch timer clear bit (WTR) in the watch timer control register (WTC) is set to "0".

Note:

When the watch timer counter is cleared, the interrupts of the watchdog timer and interval timer that use the output of the watch timer counter are affected.

To clear the watch timer by writing "0" to the watch timer clear bit (WTR) in the watch timer control register (WTC), set the overflow interrupt enable bit (WTIE) in the WTC register 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 overflow flag bit (WTOF) in the WTC register.

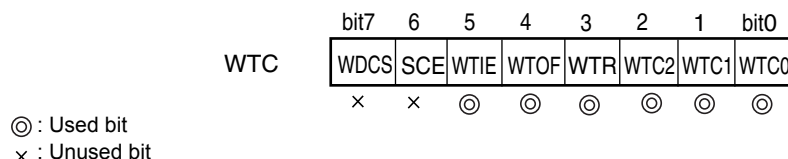
■ Interval Timer Function

The watch timer can be used as an interval timer by generating an interrupt at each interval time.

- Settings when using watch timer as interval timer

Operating the watch timer as an interval timer requires the settings shown in Figure 15-4. .

Figure 15-4. Setting of Watch Timer



When the value set by the interval time select bits (WTC1, WTC0) in the watch timer control register (WTC) is reached, the overflow flag bit in the WTC register is set to "1" (WTC:WTOF = 1).

When the overflow flag bit is set (WTC:WTOF = 1) with the overflow interrupt of the watch timer counter enabled (WTC:WTIE = 1), an interrupt request is generated.

The overflow flag bit (WTC:WTOF) is set when the interval time is reached at the starting point of the timing at which the watch timer is finally cleared.

- Clearing overflow flag bit (WTC:WTOF)

When the mode is switched to the stop mode, the watch timer is used as an oscillation stabilization wait time timer of sub clock. The WTOF bit is cleared concurrently with mode switching.

■ Setting Operation Clock of Watchdog Timer

The watchdog clock select bit (WDSCS) in the watch timer control register (WTC) can be used to set the clock input source of the watchdog timer.

When using the sub clock as the machine clock, always set the WDSCS bit to "0" and select the output of the watch timer.

■ Oscillation Stabilization Wait Time Timer of Sub clock

When the watch timer returns from the power-on reset and the stop mode, it functions as an oscillation stabilization wait time timer of sub clock.

The sub clock oscillation stabilization wait time is fixed at $2^{14} / \text{SCLK}$ (SCLK: sub clock frequency).

15.6 Program Example of Watch Timer

This section gives a program example of the watch timer.

■ Program Example of Watch Timer

- Processing specifications

An interval interrupt at $2^{13} / \text{SCLK}$ (SCLK: sub clock) is generated repeatedly. The interval time is approximately 655.36 ms (when sub clock operates at 12.5 kHz).

- Coding example

```
ICR08 EQU 0000B8H ;Interrupt control register
WTC EQU 0000AAH ;Watch timer control register
WTOF EQU WTC:4 ;Overflow flag bit
;
;-----Main program-----
CODE CSEG
START:
; ;Stack pointer (SP) already
; ;initialized
AND CCR,#0BFH ;Interrupt disabled
```



```

MOV    I:ICR07,#00H      ;Interrupt level 0 (highest)
MOV    I:WTC,#10100101B ;Interrupt enabled
                        ;Overflow flag cleared
                        ;Watch timer counter cleared,
                        ;213/SCLK(approx. 655.36 ms)
MOV    ILM,#07H          ;Set ILM in PS to level 7
OR     CCR,#40H          ;Interrupt enabled

LOOP:
    •
    Processing by user
    •
    BRA    LOOP
;-----Interrupt program-----
WARI:
    CLRB   I:WTOF          ;Overflow flag cleared
    •
    Processing by user
    •
    RETI                    ;Return from interrupt processing
CODE    ENDS
;-----Vector setting-----
VECT    CSEG    ABS=0FFH
        ORG     00FF90H    ;Vector setting to interrupt number #27 (1BH)
        DSL     WARI
        ORG     00FFDCH    ;Reset vector set
        DSL     START
        DB      00H        ;Set to single-chip mode
VECT    ENDS
        END     START

```


16. 8-/16-Bit PPG Timer



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

[16.1 Overview of 8-/16-bit PPG Timer](#)

[16.2 Block Diagram of 8-/16-bit PPG Timer](#)

[16.3 Configuration of 8-/16-bit PPG Timer](#)

[16.4 Interrupts of 8-/16-bit PPG Timer](#)

[16.5 Explanation of Operation of 8-/16-bit PPG Timer](#)

[16.6 Notes on Using 8-/16-bit PPG Timer](#)

16.1 Overview of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer is a reload timer module with two channels (PPGC and PPGD) that outputs a pulse in any cycle and at any duty ratio. A combination of two channels provides:

- 8-bit PPG output 2-channel independent operation mode
- 16-bit PPG output operation mode
- 8+8-bit PPG output operation mode

The MB90990 series has three 8-/16-bit PPG timers. This section explains the functions of PPGC/PPGD. PPGA/PPGB, PPGE/PPGF has the same functions as PPGC/PPGD.

■ Functions of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer consists of four 8-bit reload registers (PRLHC, PRLLC, PRLHD, and PRLLD) and two PPG down counters (PCNTC and PCNTD).

- ❑ Individual setting of "H" and "L" widths in output pulse enables an output pulse of any cycle and duty ratio.
- ❑ The count clock can be selected from six internal clocks.
- ❑ The 8-/16-bit PPG timer can be used as an interval timer by generating an interrupt request at each interval time.
- ❑ An external circuit enables the 8-/16-bit PPG timer to be used as a D/A converter.

■ Operation Modes of 8-/16-bit PPG Timer

□ 8-bit PPG output 2-channel independent operation mode

The 8-bit PPG output 2-channel independent operation mode causes the 2-channel modules (PPGC and PPGD) to operate as each independent 8-bit PPG timer.

Table 16-1. shows the interval times in the 8-bit PPG output 2-channel independent operation mode.

Table 16-1. Interval Times in 8-bit PPG Output 2-channel Independent Operation Mode

Count Clock Cycle	PPGC, PPGD	
	Interval Time	Output Pulse Time
$1/\phi$ (31.25 ns)	$1/\phi$ to $2^8/\phi$	$2/\phi$ to $2^9/\phi$
$2/\phi$ (62.5 ns)	$2/\phi$ to $2^9/\phi$	$2^2/\phi$ to $2^{10}/\phi$
$2^2/\phi$ (125 ns)	$2^2/\phi$ to $2^{10}/\phi$	$2^3/\phi$ to $2^{11}/\phi$
$2^3/\phi$ (250 ns)	$2^3/\phi$ to $2^{11}/\phi$	$2^4/\phi$ to $2^{12}/\phi$
$2^4/\phi$ (500 ns)	$2^4/\phi$ to $2^{12}/\phi$	$2^5/\phi$ to $2^{13}/\phi$
$2^9/\text{HCLK}$ (128 μs)	$2^9/\text{HCLK}$ to $2^{17}/\text{HCLK}$	$2^{10}/\text{HCLK}$ to $2^{18}/\text{HCLK}$

HCLK: Oscillation clock

ϕ : Machine clock frequency

The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock frequency operates at 32 MHz.

□ 16-bit PPG output operation mode

The 16-bit PPG output operation mode concatenates the 2-channel modules (PPGC and PPGD) to operate as a 16-bit 1-channel PPG timer. The combination of PPGC + PPGD and PPGE + PPGF is used.

Table 16-2. shows the interval times in this mode.

Table 16-2. Interval Times in 16-bit PPG Output Operation Mode

Count clock cycle	Interval time	Output pulse time
$1/\phi$ (31.25 ns)	$1/\phi$ to $2^{16}/\phi$	$2/\phi$ to $2^{17}/\phi$
$2/\phi$ (62.5 ns)	$2/\phi$ to $2^{17}/\phi$	$2^2/\phi$ to $2^{18}/\phi$
$2^2/\phi$ (125 ns)	$2^2/\phi$ to $2^{18}/\phi$	$2^3/\phi$ to $2^{19}/\phi$
$2^3/\phi$ (250 ns)	$2^3/\phi$ to $2^{19}/\phi$	$2^4/\phi$ to $2^{20}/\phi$
$2^4/\phi$ (500 ns)	$2^4/\phi$ to $2^{20}/\phi$	$2^5/\phi$ to $2^{21}/\phi$
$2^9/\text{HCLK}$ (128 μs)	$2^9/\text{HCLK}$ to $2^{25}/\text{HCLK}$	$2^{10}/\text{HCLK}$ to $2^{26}/\text{HCLK}$

HCLK: Oscillation clock

ϕ : Machine clock frequency

The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock frequency operates at 32 MHz.

□ 8+8-bit PPG output operation mode

The 8 + 8-bit PPG output operation mode causes the PPGC of the 2-channel modules to operate as an 8-bit prescaler and the underflow output of the PPGC to operate as the count clock of the PPGD.

Table 16-3. shows the interval times in this mode.

Table 16-3. Interval Times in 8+8-bit PPG Output Operation Mode

Count Clock Cycle	PPGC		PPGD	
	Interval Time	Output Pulse Time	Interval Time	Output Pulse Time
$1/\phi$ (31.25 ns)	$1/\phi$ to $2^8/\phi$	$2/\phi$ to $2^9/\phi$	$1/\phi$ to $2^{16}/\phi$	$2/\phi$ to $2^{17}/\phi$
$2/\phi$ (62.5 ns)	$2/\phi$ to $2^9/\phi$	$2^2/\phi$ to $2^{10}/\phi$	$2/\phi$ to $2^{17}/\phi$	$2^2/\phi$ to $2^{18}/\phi$
$2^2/\phi$ (125 ns)	$2^2/\phi$ to $2^{10}/\phi$	$2^3/\phi$ to $2^{11}/\phi$	$2^2/\phi$ to $2^{18}/\phi$	$2^3/\phi$ to $2^{19}/\phi$
$2^3/\phi$ (250 ns)	$2^3/\phi$ to $2^{11}/\phi$	$2^4/\phi$ to $2^{12}/\phi$	$2^3/\phi$ to $2^{19}/\phi$	$2^4/\phi$ to $2^{20}/\phi$
$2^4/\phi$ (500 ns)	$2^4/\phi$ to $2^{12}/\phi$	$2^5/\phi$ to $2^{13}/\phi$	$2^4/\phi$ to $2^{20}/\phi$	$2^5/\phi$ to $2^{21}/\phi$
$2^9/\text{HCLK}$ (128 μs)	$2^9/\text{HCLK}$ to $2^{17}/\text{HCLK}$	$2^{10}/\text{HCLK}$ to $2^{18}/\text{HCLK}$	$2^9/\text{HCLK}$ to $2^{25}/\text{HCLK}$	$2^{10}/\text{HCLK}$ to $2^{26}/\text{HCLK}$

HCLK: Oscillation clock

ϕ : Machine clock frequency

The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock frequency operates at 32 MHz.

16.2 Block Diagram of 8-/16-bit PPG Timer

The MB90990 series contains three 8-/16-bit PPG timers (each with 2 channels).

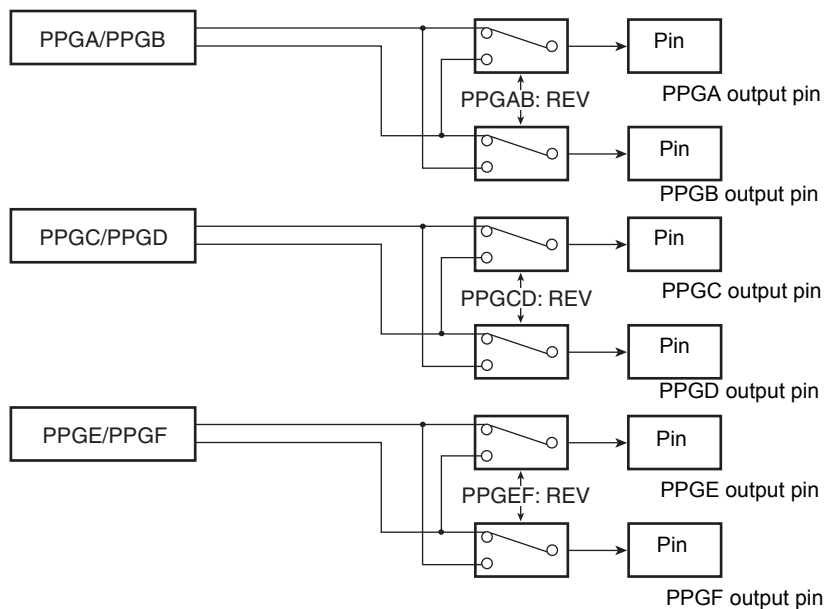
One 8-/16-bit PPG timer consists of 8-bit PPG timers with two channels.

This section shows the block diagrams for the 8-/16-bit PPG timer C and 8-/16-bit PPG timer D. The PPGA, PPGE has the same function as the PPGC, and PPGB, PPGF has the same function as PPGD.

■ Channels and PPG Pins of PPG Timers

Figure 16-1. shows the relationship between the channels and the PPG pins of the 8-/16-bit PPG timers in the MB90990 series.

Figure 16-1. Channels and PPG Pins of PPG Timers

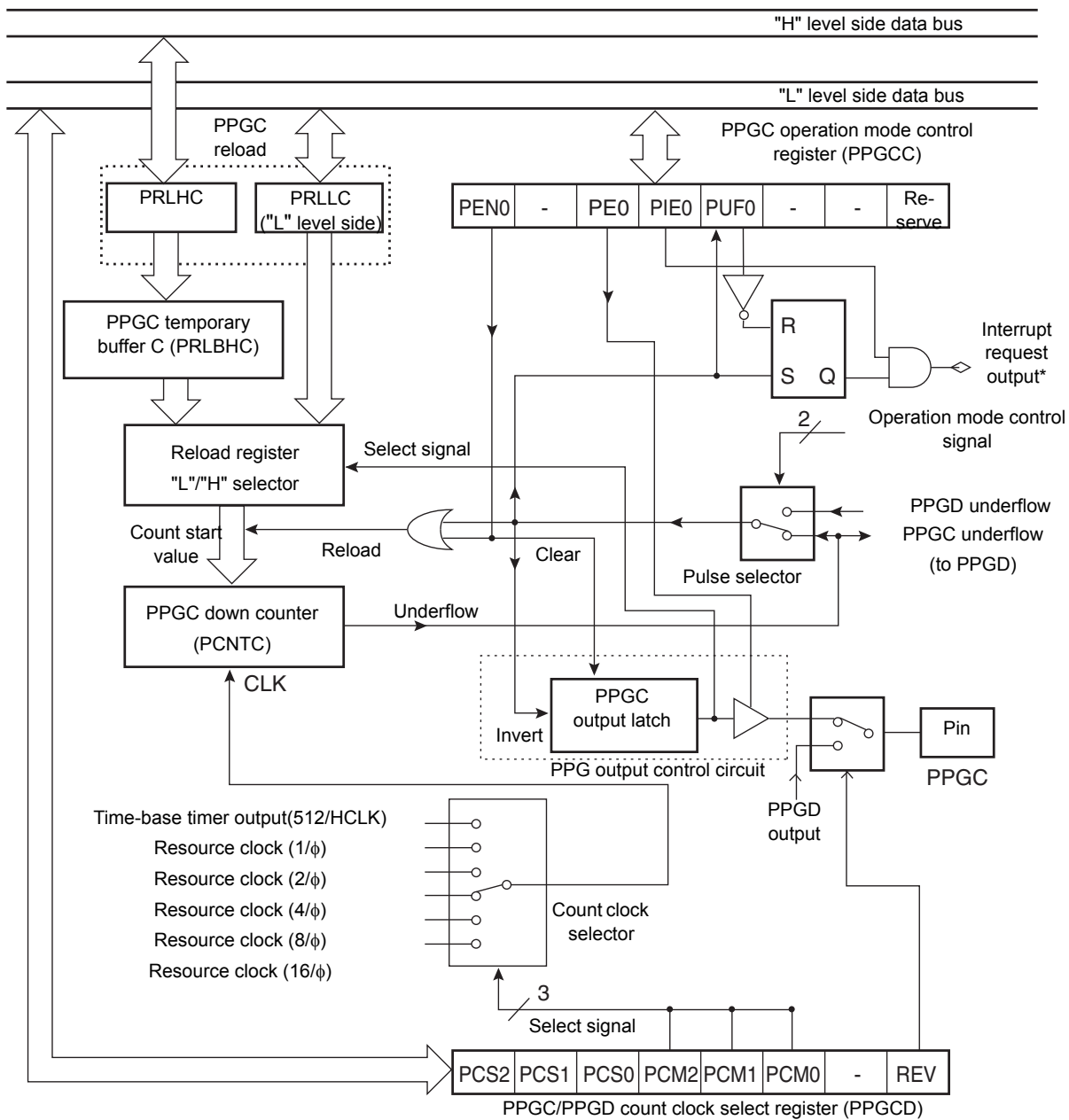


16.2.1 Block Diagram for 8-/16-bit PPG Timer C

The 8-/16-bit PPG timer C consists of the following blocks.

■ Block Diagram of 8-/16-bit PPG Timer C

Figure 16-2. Block Diagram of 8-/16-bit PPG Timer C



-: Undefined

Reserved: Reserved bit

HCLK: Oscillation clock frequency

φ: Machine clock frequency

*: The interrupt output of 8-/16-bit PPG timer C is combined to one interrupt by OR circuit with the interrupt request output of PPG timer D.

- Details of pins in block diagram

Table 16-4. lists the actual pin names and interrupt request numbers of the 8-/16-bit PPG timer.

Table 16-4. Pins and Interrupt Request Numbers in Block Diagram

Channel	Output Pin		Interrupt Request Number
	PPG:REV=0	PPG:REV=1	
PPGA	P65 / PPGA	P21 / PPGB	#24 (18 _H)
PPGB	P21 / PPGB	P65 / PPGA	
PPGC	P66 / PPGC	P22 / PPGD	#23 (17 _H)
PPGD	P22 / PPGD	P66 / PPGC	
PPGE	P67 / PPGE	P23 / PPGF	#24 (18 _H)
PPGF	P23 / PPGF	P67 / PPGE	

- PPG operation mode control register C (PPGCC)

This register enables or disables operation of the 8-/16-bit PPG timer, pin output and an underflow interrupt. It also indicates the occurrence of an underflow.

- PPGC/PPGD count clock select register (PPGCD)

This register sets the count clock of the 8-/16-bit PPG timer and switching the output pin between PPGC and PPGD.

- PPGC reload registers (PRLHC, PRLLC)

These registers set the "H" width or "L" width of the output pulse. The values set in these registers are reloaded to the PPGC down counter (PCNTC) when the 8-/16-bit PPG timer is started.

- PPGC down counter (PCNTC)

This counter is an 8-bit down counter that alternately reloads the values set in the PPGC reload registers (PRLHC and PRLLC) to decrement. When an underflow occurs, the pin output is inverted. The 2-channel PPG down counters (PPGC and PPGD) can also be concatenated for use as a single-channel 16-bit PPG down counter.

- PPGC temporary buffer (PRLBHC)

This buffer prevents deviation of the output pulse width caused at writing to the PPG reload registers (PRLHC and PRLLC). This buffer stores the PRLHC value temporarily and enables it in synchronization with the timing of writing to the PRLLC.

- Reload register L/H selector

This selector detects the current pin output level to select which register value, "L" reload register (PRLLC) or "H" reload register (PRLHC), should be reloaded to the PPGC down counter.

- Count clock selector

This selector selects the count clock to be inputted to the PPGC down counter from five frequency-divided clocks of the machine clock or the frequency-divided clocks of the time-base timer.

- PPG output control circuit

This circuit inverts the pin output level and the output when an underflow occurs.

- Details of pins in block diagram

Table 16-5. lists the actual pin names and interrupt request numbers of the 8-/16-bit PPG timer.

Table 16-5. Pins and Interrupt Request Numbers in Block Diagram

Channel	Output Pin		Interrupt Request Number
	PPG:REV=0	PPG:REV=1	
PPGA	P65 / PPGA	P21 / PPGB	#24 (18 _H)
PPGB	P21 / PPGB	P65 / PPGA	
PPGC	P66 / PPGC	P22 / PPGD	#23 (17 _H)
PPGD	P22 / PPGD	P66 / PPGC	
PPGE	P67 / PPGE	P23 / PPGF	#24 (18 _H)
PPGF	P23 / PPGF	P67 / PPGE	

- PPG operation mode control register D (PPGCD)

This register sets the operation mode of the 8-/16-bit PPG timer, enables or disables the operation of the 8-/16-bit PPG timer D, the pin output and an underflow interrupt, and also indicates the generation of an underflow.

- PPGC/PPGD count clock select register (PPGCD)

This register sets the count clock of the 8-/16-bit PPG timer.

- PPGD reload registers (PRLHD, PRLLD)

These registers set the High width or Low width of the output pulse. The values set in these registers are reloaded to the PPGD down counter (PCNTD) when the 8-/16-bit PPG timer D is started.

- PPGD down counter (PCNTD)

This counter is an 8-bit down counter that alternately reloads the values set in the PPGD reload registers (PRLHD and PRLLD) to decrement. When an underflow occurs, the pin output is inverted. The 2-channel PPG down counters (PPGC and PPGD) can also be connected for use as a single-channel 16-bit PPG down counter.

- PPGD temporary buffer (PRLBHD)

This buffer prevents deviation of the output pulse width caused at writing to the PPG reload registers (PRLHD and PRLLD). This buffer stores the PRLHD value temporarily and enables it in synchronization with the timing of writing to the PRLLD.

- Reload register "L"/"H" selector

This selector detects the current pin output level to select which register value, Low reload register (PRLLD) or High reload register (PRLHD), should be reloaded to the PPGD down counter.

- Count clock selector

This selector selects the count clock to be inputted to the PPGD down counter from five frequency-divided clocks of the machine clock or the frequency-divided clocks of the time-base timer.

- PPG output control circuit

This circuit inverts the pin output level and the output when an underflow occurs.

16.3 Configuration of 8-/16-bit PPG Timer

This section explains the pins, registers and interrupt factors of the 8-/16-bit PPG timer.

■ Pins of 8-/16-bit PPG Timer

The pins of the 8-/16-bit PPG timer serve as general-purpose I/O ports. [Table 16-6](#) indicates the pin functions and pin settings required to use the 8-/16-bit PPG timer.

Table 16-6. Pins of 8-/16-bit PPG Timer

Channel	Pin Name	Pin Function	Pin Setting Required for Use of 8-/16-bit PPG Timer
PPGA	P65 / AN5 / PPGA	General-purpose I/O port/ A/D converter analog input 5/ PPG output A	Analog input enable register: setting to disable(ADER6:ADE5=0) PPG operating mode control register: pin output enable (PPGCA:PE0=1)
PPGB	P21 / PPGB	General-purpose I/O port/ PPG output B	PPG operating mode control register : pin output enable (PPGCB:PE1=1)
PPGC	P66 / AN6 / PPGC	General-purpose I/O port/ A/D converter analog input 6/ PPG output C	Analog input enable register: setting to disable(ADER6:ADE6=0) PPG operating mode control register: pin output enable (PPGCC:PE0=1)
PPGD	P22 / PPGD	General-purpose I/O port/ PPG output D	PPG operating mode control register : pin output enable (PPGCD:PE1=1)
PPGE	P67 / AN7 / PPGE	General-purpose I/O port/ A/D converter analog input 7/ PPG output E	Analog input enable register: setting to disable (ADER6:ADE7=0) PPG operating mode control register: pin output enable (PPGCE:PE0=1)
PPGF	P23 / PPGF	General-purpose I/O port/ PPG output F	PPG operating mode control register : pin output enable (PPGCF:PE1=1)

■ List of Registers and Initial Values of 8-/16-bit PPG Timer

Figure 16-4. List of Registers and Initial Values of 8-/16-bit PPG Timer

PPGm operation mode control register: H (PPGCm)	bit	15	14	13	12	11	10	9	8
		0	x	0	0	0	0	0	1
PPGn operation mode control register: L (PPGCn)	bit	7	6	5	4	3	2	1	0
		0	x	0	0	0	x	x	1
PPGn/m count clock select register (PPGnm)	bit	7	6	5	4	3	2	1	0
		0	0	0	0	0	0	x	0
PPGn reload register: H(PRLHn)	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
PPGn reload register: L(PRLLn)	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
PPGm reload register: H(PRLHm)	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
PPGm reload register: L(PRLm)	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x

x: Undefined

n = A, C, E

m = B, D, F

■ Generation of Interrupt Request from 8-/16-bit PPG Timer

In the 8-/16-bit PPG timer, the underflow generation flag bits in the PPG operation mode control registers (PPGCn:PUFn, PPGCm:PUFm) are set to "1" when an underflow occurs. If the underflow interrupts of channels causing an underflow are enabled (PPGCn: PIE0=1, PPGCm: PIE1=1), an underflow interrupt request is generated to the interrupt controller.

16.3.1 PPGC Operation Mode Control Register (PPGCC)

The PPGC operation mode control register provides the following settings for the operation of 8-/16-bit PPG timer C:

- Enabling or disabling operation of 8-/16-bit PPG timer C
- Switching between pin functions (enabling or disabling pulse output)
- Enabling or disabling underflow interrupt
- Setting underflow interrupt request flag

This section explains the PPGCC function only. The PPGCA, PPGCE has the same function as the PPGCC, and the 8-/16-bit PPG timer A, C and E are set.

■ PPGC Operation Mode Control Register (PPGCC)

Figure 16-5. PPGC Operation Mode Control Register (PPGCC)

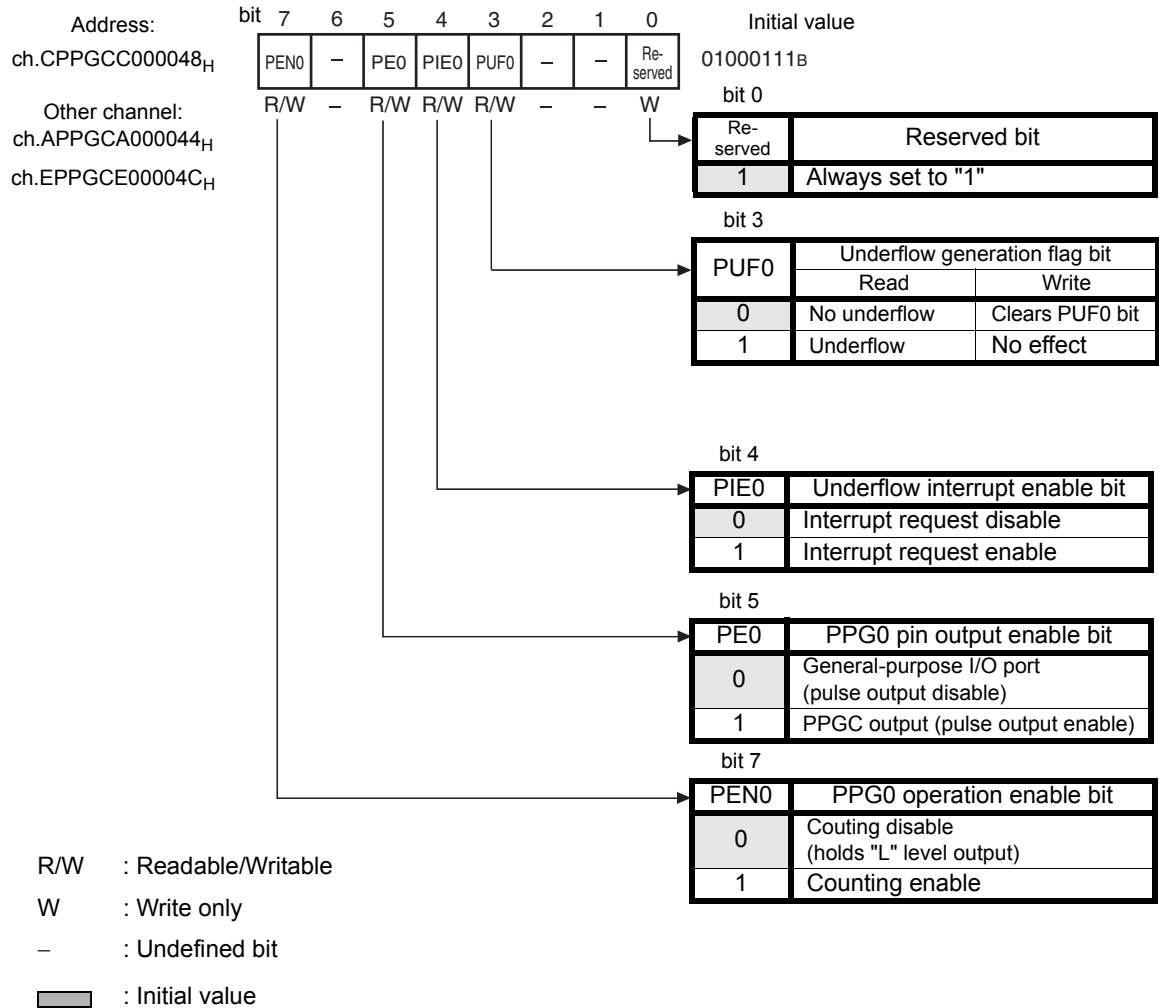


Table 16-7. Functions of PPGC Operation Mode Control Register (PPGCC)

Bit Name		Function
bit7	PEN0: PPG0 operation enable bit	This bit enables or disables the count operation of the 8-/16-bit PPG timer C. When set to "0" : Count operation disabled When set to "1" : Count operation enabled When the count operation is disabled (PEN0=0), and the pulse output is enabled (PE0=1), the output is held at a Low level.
bit6	Undefined bit	Read : The value is undefined. Write : No effect
bit5	PE0: PPG0 pin output enable bit	This bit switches between PPGC pin functions and enables or disables the pulse output. When set to "0" : PPGC pin functions as general-purpose I/O port. The pulse output is disabled. When set to "1" : PPGC pin functions as PPGC output pin. The pulse output is enabled.
bit4	PIE0: Underflow interrupt enable bit	This bit enables or disables an interrupt. When set to "0" : No interrupt request generated even at underflow (PUF0 = 1). When set to "1" : Interrupt request generated at underflow (PUF0 = 1)
bit3	PUF0: Underflow generation flag bit	8-bit PPG output 2-channel independent operation mode, 8+8-bit PPG output operation mode: When the value of the PPGC down counter is decremented from "00 _H " to "FF _H ", an underflow occurs (PUF0 = 1). 16-bit PPG output operation mode: When the values of the PPGC and PPGD down counters are decremented from "0000 _H " to "FFFF _H ", an underflow occurs (PUF0 = 1). When an underflow occurs (PUF0 = 1) with an underflow interrupt enabled (PIE0 = 1), an interrupt request is generated. When set to "0" : Clears this bit. When set to "1" : No effect Read by read-modify-write (RMW) instructions: "1" is read.
bit2, bit1	Undefined bits	Write : No effect Read : The value is undefined.
bit0	Reserved: Reserved bit	Always set this bit to "1".

16.3.2 PPGD Operation Mode Control Register (PPGCD)

The PPGD operation mode control register provides the following settings about operation of 8-/16-bit PPG timer D:

- Enabling or disabling operation of 8-/16-bit PPG timer D
- Switching between pin functions (enabling or disabling pulse output)
- Enabling or disabling underflow interrupt
- Setting underflow interrupt request flag

- Setting the operation mode of the 8-/16-bit PPG timer D and C

This section explains the PPGCD function only. The PPGCF has the same function as the PPGCB, PPGCD, and the 8-/16-bit PPG timer B, D and F are set.

■ PPGD Operation Mode Control Register (PPGCD)

Figure 16-6. PPGD Operation Mode Control Register (PPGCD)

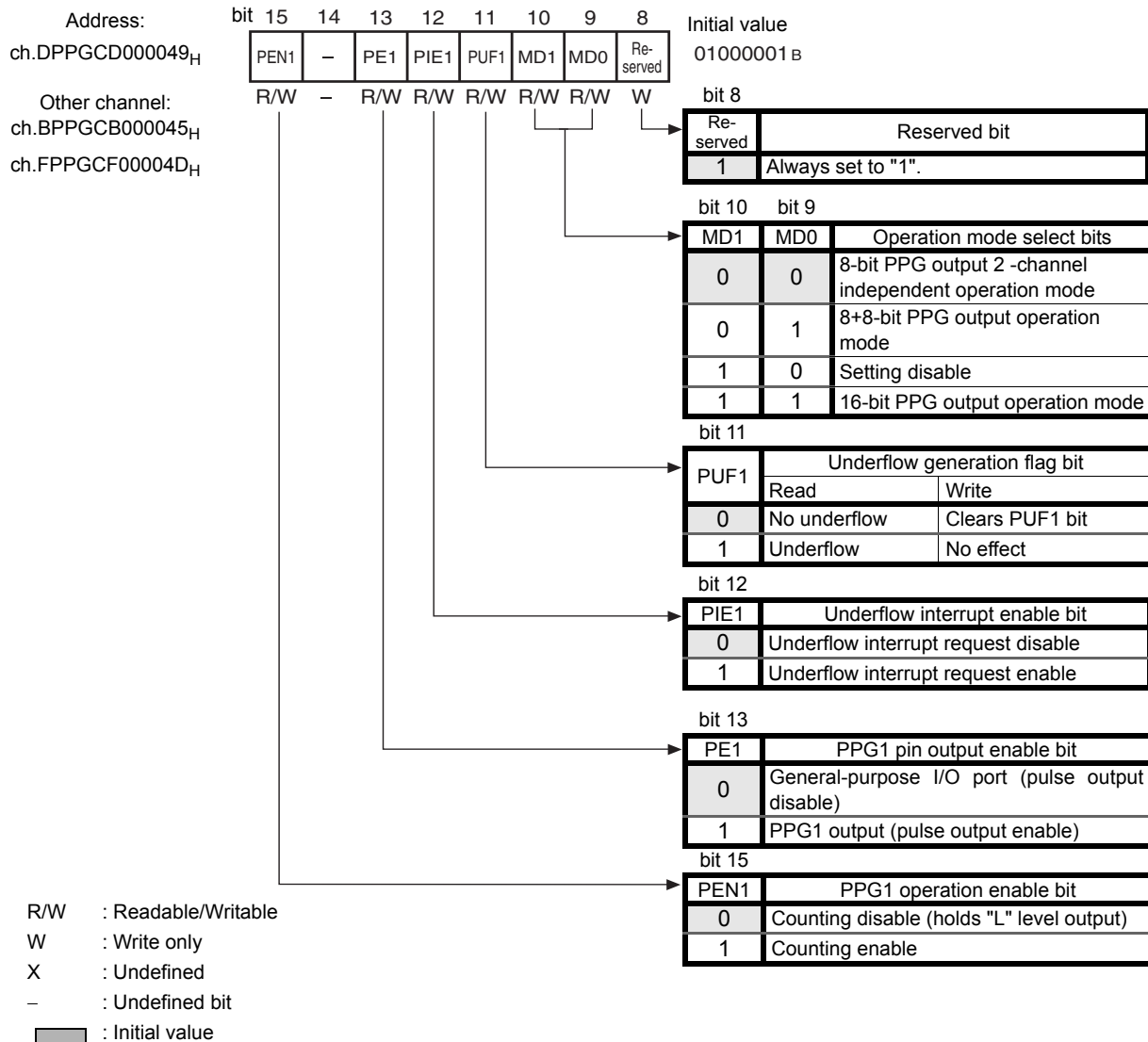


Table 16-8. Functions of PPGD Operation Mode Control Register (PPGCD)

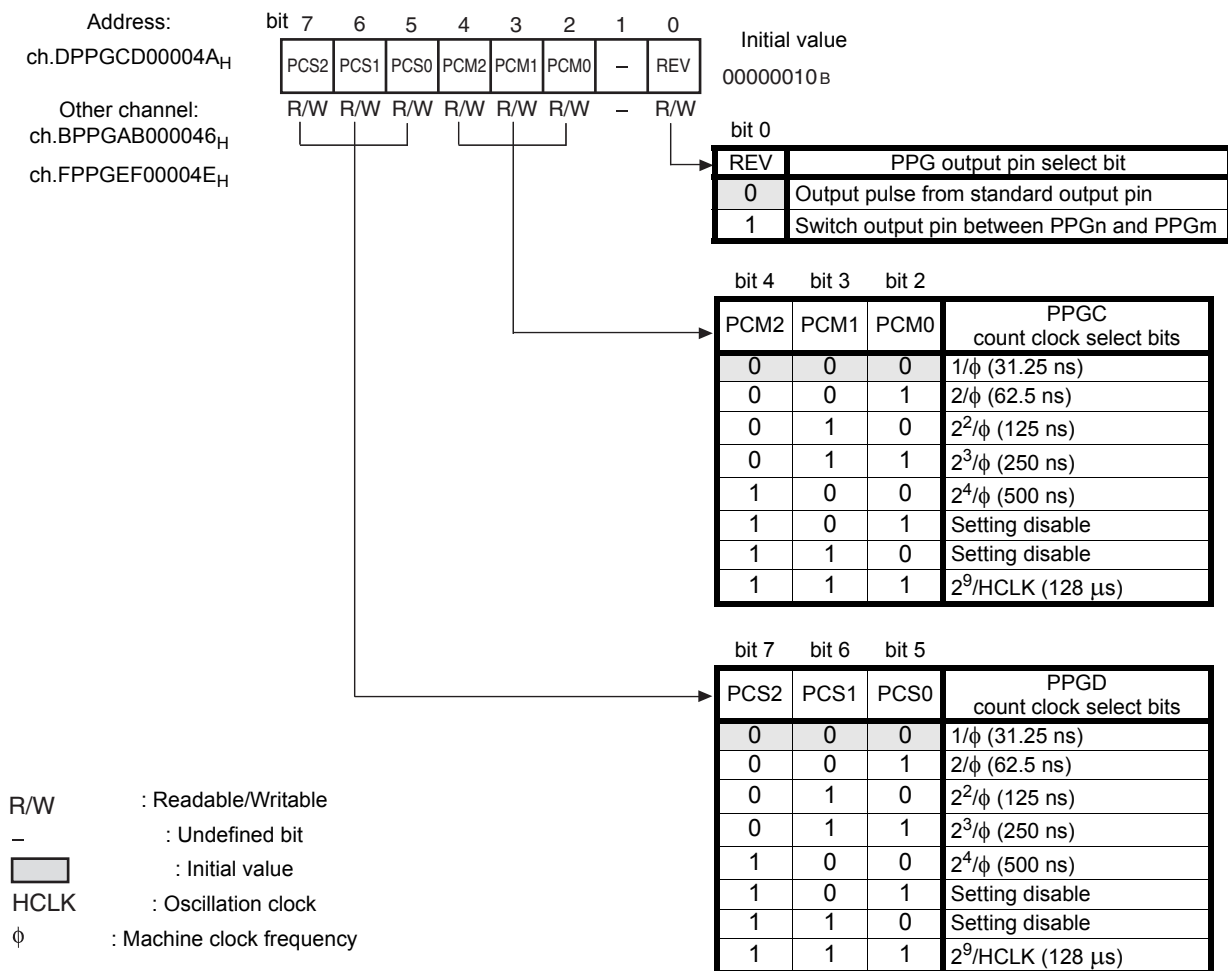
Bit name		Function
bit15	PEN1: PPG1 operation enable bit	<p>This bit enables or disables the count operation of the 8-/16-bit PPG timer D.</p> <p>When set to "0": Count operation disabled</p> <p>When set to "1": Count operation enabled</p> <p>When the count operation is disabled (PEN1 = 0), and the pulse output is enabled (PE1=1), the output is held at a Low level.</p>
bit14	Undefined bit	<p>Read: The value is undefined.</p> <p>Write: No effect</p>
bit13	PE1: PPG1 Pin output enable bit	<p>This bit switches between PPGD pin functions and enables or disables the pulse output.</p> <p>When set to "0": PPGD pin functions as general-purpose I/O port. The pulse output is disabled.</p> <p>When set to "1": PPGD pin functions as PPGD output pin. The pulse output is enabled.</p>
bit12	PIE1: Underflow interrupt enable bit	<p>This bit enables or disables an interrupt.</p> <p>When set to "0": No interrupt request is generated even at underflow (PUF1 = 1)</p> <p>When set to "1": Interrupt request is generated at underflow (PUF1 = 1)</p>
bit11	PUF1: Underflow generation flag bit	<p>8-bit PPG output 2-channel independent operation mode, 8+8-bit PPG output operation mode:</p> <p>When the value of the PPGD down counter is decremented from "00_H" to "FF_H", an underflow occurs (PUF1 = 1).</p> <p>16-bit PPG output operation mode:</p> <p>When the values of the PPGC and PPGD down counters are decremented from "0000_H" to "FFFF_H", an underflow occurs (PUF1 = 1).</p> <p>When an underflow occurs (PUF1 = 1) with an underflow interrupt request enabled (PIE1=1), an interrupt request is generated.</p> <p>When set to "0": Clears counter</p> <p>When set to "1": No effect</p> <p>Read by read-modify-write (RMW) instructions: "1" is read.</p>
bit10, bit9	MD1, MD0: Operation mode select bits	<p>These bits set the operation mode of the 8-/16-bit PPG timer.</p> <p>[Any mode other than 8-bit PPG output 2-channel independent operation mode]</p> <p>Use a word instruction to set the PPG operation enable bits (PEN0 and PEN1) at one time. Do not set operation of only one of the two channels (PEN1 = 0/PEN0 = 1 or PEN1 = 1/PEN0 = 0).</p> <p>Note: Do not set the MD1 and MD0 bits to "10_B".</p>
bit8	Reserved: Reserved bit	Always set this bit to "1".

16.3.3 PPGC/PPGD Count Clock Select Register (PPGCD)

The PPGC/PPGD count clock select register selects the count clock of the 8-/16-bit PPG timers C and D and the output pin. This section explains the PPGCD function only. The PPGAB, PPGEF has the same function as the PPGCD, and the 8-/16-bit PPG timers A and B, C and D, E and F are set.

■ PPGC/PPGD Count Clock Select Register (PPGCD)

Figure 16-7. PPGC/PPGD Count Clock Select Register (PPGCD)



The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock frequency operates at 32 MHz.

n = A, C, E

m = n+1

Table 16-9. Functions of PPGC/PPGD Count Clock Select Register (PPGCD)

Bit Name		Function
bit7 to bit5	PCS2 to PCS0: PPGD count clock select bits	These bits set the count clock of the 8-/16-bit PPG timer D. The count clock can be selected from five frequency-divided clocks of the machine clock and the frequency-divided clocks of the time-base timer. The settings of the PPGD count clock select bits (PCS2 to PCS0) are enabled only in the 8-bit PPG output 2-channel independent mode (PPGCD: MD1, MD0=00 _B).
bit4 to bit2	PCM2 to PCM0: PPGC count clock select bits	These bits set the count clock of the 8-/16-bit PPG timer C. The count clock can be selected from five frequency-divided clocks of the machine clock and the frequency-divided clocks of the time-base timer.
bit1	Undefined bit	Read: The value is undefined. Write: No effect
bit0	REV: PPG output pin select bit	This bit switches the output pin in the 8-/16-bit PPG timers C and D. When set to "0": Output from the standard output pin. PPGC → PPGC output pin PPGD → PPGD output pin When set to "1": Switch the output pin. PPGC → PPGD output pin PPGD → PPGC output pin

16.3.4 PPG Reload Registers (PRLLC/PRLHC, PRLLD/PRLHD)

The value (reload value) from which the PPG down counter starts counting is set in the PPG reload registers, which are an 8-bit register at Low level and an 8-bit register at High level.

This section explains the function of PRLLC/PRLHC and PRLLD/PRLHD only. The PRLLA/PRLHA, PRLLB/PRLHB, PRLLE/PRLHE, PRLLF/PRLHF have the same function as the PRLLC/PRLHC, and the 8-/16-bit PPG timers A, B, E, F are set.

■ PPG Reload Registers (PRLLC/PRLHC, PRLLD/PRLHD)

Figure 16-8. PPG Reload Registers (PRLLC/PRLHC, PRLLD/PRLHD)

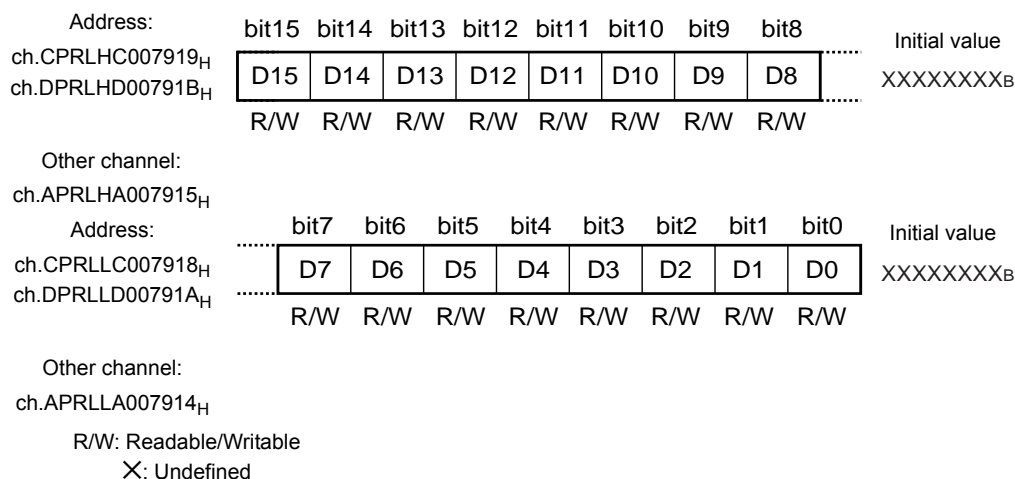


Table 16-10. indicates the functions of the PPG reload registers.

Table 16-10. Functions of PPG Reload Registers

Function	8-/16-bit PPG Timer C	8-/16-bit PPG Timer D
Retains reload value on Low-level side	PRLLC	PRLLD
Retains reload value on High-level side	PRLHC	PRLHD

Notes:

- In the 16-bit PPG output operation mode (PPGCD: MD1, MD0=11_B), use a long-word instruction to set the PPG reload registers or the word instruction to set the PPGC and PPGD in this order.
- In the 8 + 8-bit PPG output operation mode (PPGCD: MD1, MD0=01_B), set the same value in both the "L" level and "H" level PPG reload registers (PRLLC/PRLHC) of the 8-/16-bit PPG timer C. Setting a different value in the "L" level and "H" level PPG reload registers may cause the 8-/16-bit PPG timer D to have different PPG output waveforms at each clock cycle.

16.4 Interrupts of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer can generate an interrupt request when the PPG down counter underflows. It also does not correspond to the EI²OS.

■ **Interrupts of 8-/16-bit PPG Timer**

Table 16-11. shows the interrupt control bits and interrupt factor of the 8-/16-bit PPG timer.

Table 16-11. Interrupt Control Bits of 8-/16-bit PPG Timer

	PPGn	PPGm
Interrupt request flag bit	PPGCn: PUF0	PPGCm: PUF1
Interrupt request enable bit	PPGCn: PIE0	PPGCm: PIE1
Interrupt factor	Underflow in PPGn down counter	Underflow in PPGm down counter

n = A, C, E
m = n + 1

[8-bit PPG output 2-channel independent operation mode or 8 + 8-bit PPG output operation mode]

- In the 8-bit PPG output 2-channel independent operation mode or the 8 + 8-bit PPG output operation mode, the PPGn and PPGm can generate an interrupt independently.
- When the value of the PPGn or PPGm down counter is decremented from "00_H" to "FF_H", an underflow occurs. When an underflow occurs, the underflow generation flag bit in the channel causing an underflow is set (PPGCn: PUF0=1 or PPGCm: PUF1=1).
- If an interrupt request from the channel that causes an underflow is enabled (PPGCn: PIE0=1 or PPGCm: PIE1=1), an interrupt request is generated.

[16-bit PPG output operation mode]

- In the 16-bit PPG output operation mode, when the values of the PPGn and PPGm down counters are decremented from "0000_H" to "FFFF_H", an underflow occurs. When an underflow occurs, the underflow generation flag bits in the two channels are set at one time (PPGCn: PUF0=1 and PPGCm: PUF1=1).
- When an underflow occurs with either of the two channel of the interrupt requests enabled (PPGCn:PIE0=0, PPGCm:PIE1=1 or PPGCn:PIE0=1, PPGCm:PIE1=0), an interrupt request is generated.

- To prevent duplication of interrupt requests, disable either of the two channel of the underflow interrupt enable bits in advance (PPGCn: PIE0=0, PPGCm: PIE1=1 or PPGCn: PIE0=1, PPGCm: PIE1=0).
- When the two channels of the underflow generation flag bits are set (PPGCn: PUF0=1 and PPGCm: PUF1=1), clear the two channels at the same time.

For details of the interrupt number, interrupt control register, and interrupt vector address, see "3. Interrupts".

16.5 Explanation of Operation of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer outputs a pulse width at any frequency and at any duty ratio continuously.

■ Operation of 8-/16-bit PPG Timer

□ Output operation of 8-/16-bit PPG timer

The 8-/16-bit PPG timer has two (Low-level and High-level) 8-bit reload registers (PRLLn/PRLHn and PRLm/PRLHm) for per channel.

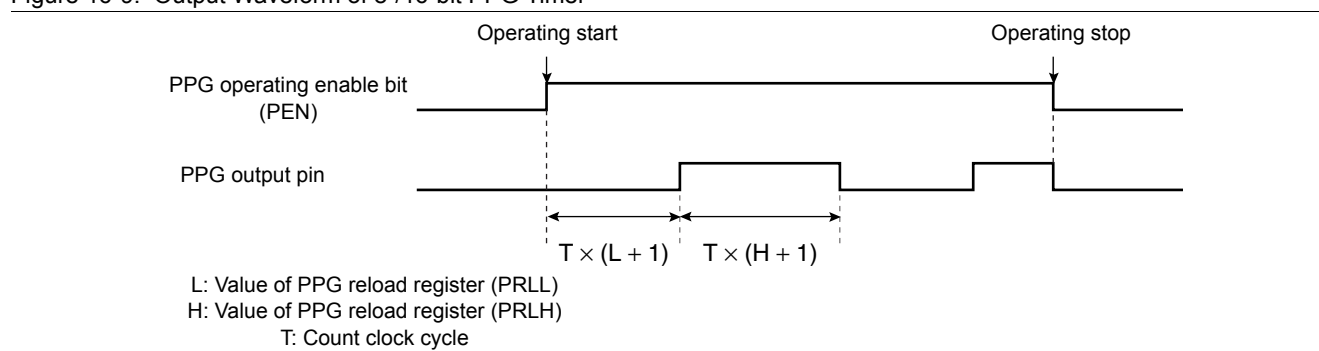
The values set in the 8-bit reload registers (PRLLn/PRLHn and PRLm/PRLHm) are reloaded alternately to the PPG down counters (PCNTn and PCNTm).

After reloading the values in the PPG down counters, decrementing is performed in synchronization with the count clocks set by the PPG count clock select bits (PPGnm: PCM2 to PCM0 and PCS2 to PCS0).

If the values set in the reload registers are reloaded to the PPG down counters when an underflow occurs, the pin output is inverted.

Figure 16-9. shows the output waveform of the 8-/16-bit PPG timer.

Figure 16-9. Output Waveform of 8-/16-bit PPG Timer



□ Operation modes of 8-/16-bit PPG timer

As long as the operation of the 8-/16-bit PPG timer is enabled (PPGCn: PEN0=1, PPGCm: PEN1=1), a pulse waveform is outputted continuously from the PPG output pin. A pulse waveform of any frequency and duty ratio can be set.

The pulse output of the 8-/16-bit PPG timer is not stopped until operation of the 8-/16-bit PPG timer is stopped (PPGCn: PEN0=0, PPGCm: PEN1=0).

- 8-bit PPG output 2-channel independent operation mode
- 16-bit PPG output operation mode
- 8 + 8-bit PPG output operation mode

Note:

$n = A, C, Em = n + 1$

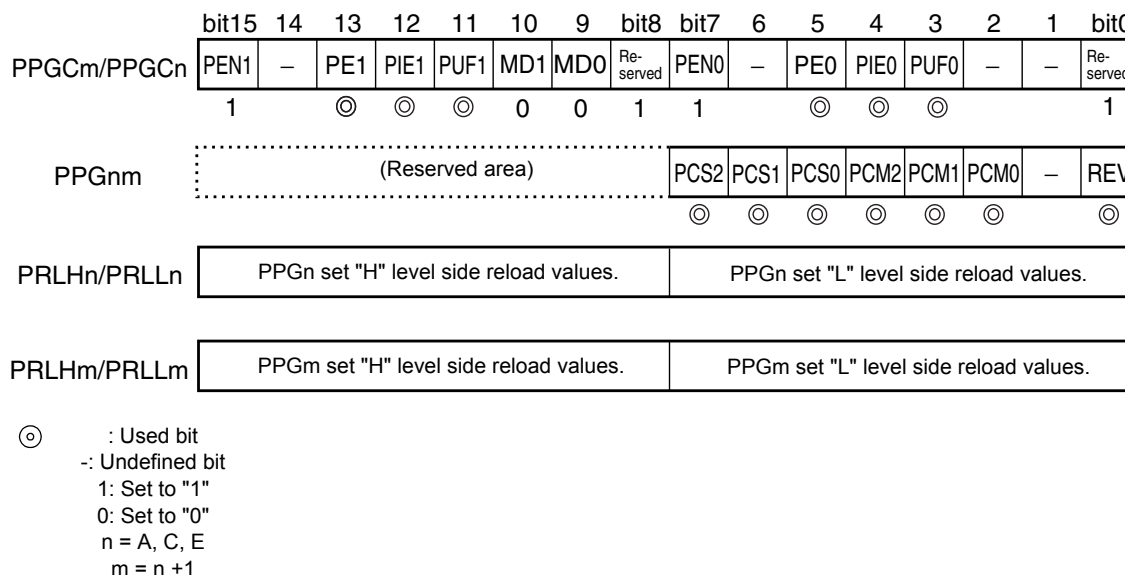
16.5.1 8-bit PPG Output 2-channel Independent Operation Mode

In the 8-bit PPG output 2-channel independent operation mode, the 8-/16-bit PPG timer is set as an 8-bit PPG timer with two independent channels. PPG output operation and interrupt request generation can be performed independently for each channel.

■ Setting for 8-bit PPG Output 2-channel Independent Operation Mode

Operating the 8-/16-bit PPG timer in the 8-bit PPG output 2-channel independent operation mode requires the setting shown in Figure 16-10.

Figure 16-10. Setting for 8-bit PPG Output 2-channel Independent Operation Mode



Note:

Use the word instruction to set both "H" level and "L" level PPG reload registers (PRLLn/PRLHn and PRLm/PRLHm) at the same time.

□ Operation in 8-bit PPG output 2-channel independent operation mode

The 8-bit PPG timer with two channels performs an independent PPG operation.

When the pin output is enabled (PPGCn: PE0=1, PPGCm: PE1=1), if the PPG output pin selection is set to standard (PPGnm: REV=0), the PPGn pulse wave is outputted from the PPGn pin and the PPGm pulse wave is outputted from the PPGm pin. When the PPG output pin is switched (PPGnm: REV=1), the PPGm pulse wave is outputted from the PPGn pin and the PPGn pulse wave is outputted from the PPGm pin.

When the reload value is set in the PPG reload registers (PRLLn/PRLHn, PRLm/PRLHm) to enable the operation of the PPG timer (PPGCn: PENC=1, PPGCm: PEND=1), the PPG down counter of the enabled channel starts counting.

To stop the count operation of the PPG down counter, disable the operation of the PPG timer of the channel to be stopped (PPGCn: PENC=0, PPGCm: PEND=0). The count operation of the PPG down counter is stopped and the output of the PPG output pin is held at a "L" level.

When the PPG down counter of each channel underflows, the reload values set in the PPG reload registers (PRLLn/PRLHn, PRLm/PRLHm) are reloaded to the PPG down counter that underflows.

When an underflow occurs, the underflow generation flag bit in the channel that causes an underflow is set (PPGCn: PUF0=1, PPGCm: PUF1=1). If an interrupt request is enabled at the channel that causes an underflow (PPGCn: PIE0=1, PPGCm: PIE1=1), the interrupt request is generated.

□ Output waveform in 8-bit PPG output 2-channel independent operation mode

The "H" and "L" pulse widths to be outputted are determined by adding "1" to the value in the PPG reload register of each channel and multiplying it by the count clock cycle. For example, if the value in the PPG reload register is "00_H", the pulse width has one count clock cycle, and if the value is "FF_H", the pulse width has 256 count clock cycles.

The equations for calculating the pulse width are shown below:

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

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

P_L : "L" width of output pulse

P_H : "H" width of output pulse

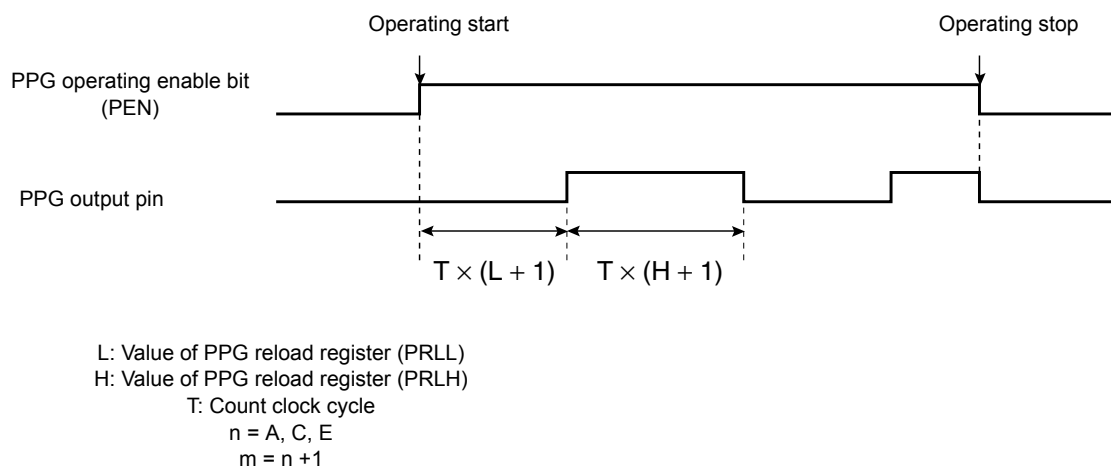
L: Values of 8 bits in PPG reload register (PRLLn or PRLm)

H: Values of 8 bits in PPG reload register (PRLHn or PRLHm)

T: Count clock cycle

Figure 16-11. shows the output waveform in the 8-bit PPG output 2-channel independent operation mode.

Figure 16-11. Output Waveform in 8-bit PPG Output 2-channel Independent Operation Mode



16.5.2 16-bit PPG Output Operation Mode

In the 16-bit PPG output operation mode, the 8-/16-bit PPG timer is set as a 16-bit PPG timer with one channel.

■ Setting for 16-bit PPG Output Operation Mode

Operating the 8-/16-bit PPG timer in the 16-bit PPG output operation mode requires the setting shown in [Figure 16-12](#).

Figure 16-12. Setting for 16-bit PPG Output Operation Mode

	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
PPGCm/PPGCn	PEN1	–	PE1	PIE1	PUF1	MD1	MD0	Re-served	PEN0	–	PE0	PIE0	PUF0	–	–	Re-served
	1		⊙	⊙	⊙	1	1	1	1		⊙	⊙	⊙			1
PPGnm	(Reserved area)								PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	–	REV
									x	x	x	⊙	⊙	⊙		x
PRLHn/PRLLn	PPGn set "H" level side reload values of lower 8								PPGn set "L" level side reload values of lower 8							
PRLHm/PRLm	PPGm set "H" level side reload values of upper 8								PPGm set "L" level side reload values of upper 8							

⊙ : Used bit
 –: Undefined bit
 X: Unused bit
 1: Set to "1"
 n = A, C, E
 m = n + 1

Note:

Use a long-word instruction to set the values in the PPG reload registers or a word instruction to set the PPGn and PPGm (PRLLn --> PRLm or PRLHn --> PRLHm) in this order.

□ Operation in 16-bit PPG output operation mode

When either PPGn pin output or PPGm pin output is enabled (PPGCn:PE0=1, PPGCm: PE1=1), the same pulse wave is outputted from both the PPGn and PPGm pins.

When the reload value is set in the PPG reload registers (PRLLn/PRLHn, PRLlm/PRLHm) to enable operation of the PPG timer (PPGCn:PENC=1 and PPGCm: PEND=1), the PPG down counters start counting as 16-bit down counters (PCNTn + PCNTm).

To stop the count operation of the PPG down counters, disable the operation of the PPG timers of both channels (PPGCn: PENC=0 and PPGCm: PEND=0). The count operation of the PPG down counters is stopped and the output of the PPG output pin is held at a Low level.

If the PPGm down counter underflows, the reload values set in the PPGn and PPGm reload registers (PRLLn/PRLHn, PRLlm/PRLHm) are reloaded simultaneously to the PPG down counters (PCNTn + PCNTm).

When an underflow occurs, the underflow generation flag bits in both channels are set simultaneously (PPGCn:PUF0=1, PPGCm:PUF1=1). If an interrupt request is enabled at either channel (PPGCn: PIE0=1, PPGCm: PIE1=1), an interrupt request is generated.

Notes:

- In the 16-bit PPG output operation mode, the underflow generation flag bits in the two channels are set simultaneously when an underflow occurs (PPGCn: PUF0=1 and PPGCm: PUF1=1). To prevent duplication of interrupt requests, disable either of the underflow interrupt enable bits in the two channels (PPGCn:PIE0=0, PPGCm:PIE1=1 or PPGCn:PIE0=1, PPGCm:PIE1=0).
- If the underflow generation flag bits in the two channels are set (PPGCn: PUF0=0 and PPGCm: PUF1=0), clear the two channels at the same time.
- n = A, C, E
m = n + 1

□ Output waveform in 16-bit PPG output operation mode

The High and Low pulse widths to be outputted are determined by adding "1" to the value in the PPG reload register of each channel and multiplying it by the count clock cycle. For example, if the value in the PPG reload register is "0000_H", the pulse width has one count clock cycle, and if the value is "FFFF_H", the pulse width has 65536 count clock cycles.

The equations for calculating the pulse width are shown below:

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

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

P_L: "L" width of output pulse

P_H: "H" width of output pulse

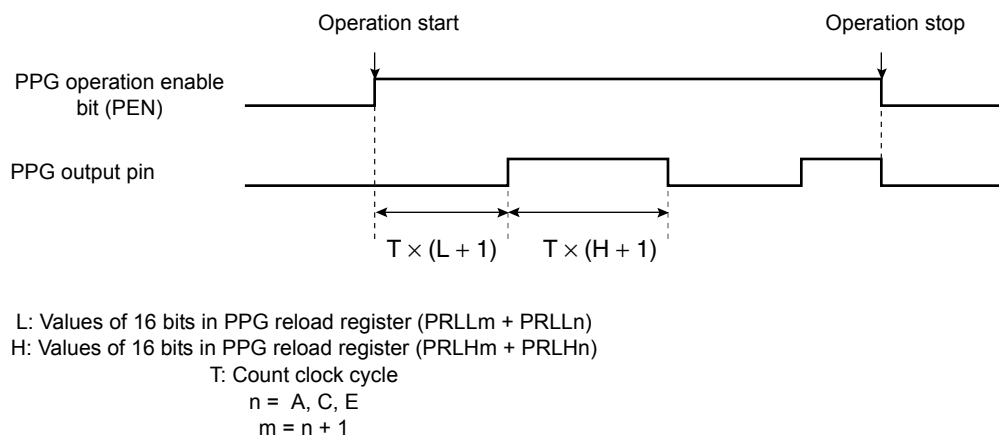
L: Values of 16 bits in PPG reload register (PRLLn+PRLlm)

H: Values of 16 bits in PPG reload register (PRLHn+PRLHm)

T: Count clock cycle

Figure 16-13. shows the output waveform in the 16-bit PPG output operation mode.

Figure 16-13. Output Waveform in 16-bit PPG Output Operation Mode



16.5.3 8+8-bit PPG Output Operation Mode

In the 8 + 8-bit PPG output operation mode, the 8-/16-bit PPG timer is set as an 8-bit PPG timer. The PPGC operates as an 8-bit prescaler and the PPGD operates using the PPG output of the PPGC as a clock source.

■ Setting for 8+8-bit PPG Output Operation Mode

Operating the 8-/16-bit PPG timer in the 8+8-bit PPG output operation mode requires the setting shown in [Figure 16-14](#).

Figure 16-14. Setting for 8+8-bit PPG Output Operation Mode

	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
PPGCm/PPGCn	PEN1	—	PE1	PIE1	PUF1	MD1	MD0	Re-served	PEN0	—	PE0	PIE0	PUF0	—	—	Re-served
	1		⊙	⊙	⊙	0	1	1	1		⊙	⊙	⊙			1
PPGnm	(Reserved area)								PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	—	REV
									x	x	x	⊙	⊙	⊙		
PRLHn/PRLLn	PPGn set High level side reload values.								PPGn set Low level side reload values.							
PRLHm/PRLm	PPGm set High level side reload values.								PPGm set Low level side reload values.							

⊙ : Used bit
 —: Undefined bit
 X: Unused bit
 1: Set to "1"
 0: Set to "0"
 n = A, C, E
 m = n + 1

Note:

Use the word instruction to set both "H" level and "L" level PPG reload registers (PRLn/PRLHn, PRLm/PRLHm) at the same time.

❑ Operation in 8+8-bit PPG output operation mode

The PPGn operates as the prescaler of the PPGm timer and the PPGm operates using the PPGn output as a count clock.

When the pin output is enabled (PPGCn: PE0=1, PPGCm: PE1=1) if PPG output pin selection is set to standard (PPGnm:REV=0), PPGn pulse wave is outputted from the PPGn pin and the PPGm pulse wave is outputted from the PPGm pin. When the PPG output pin is switched (PPGnm:REV=1), the output pins PPGn and PPGm are switched.

When the reload value is set in the PPG reload registers (PRLLn/PRLHn, PRLlm/PRLHm) to enable operation of the PPG timer (PPGCn: PEN0=1 and PPGCm: PEN1=1), the PPG down counter starts counting.

To stop the count operation of the PPG down counters, disable the operation of the PPG timers of both channels (PPGCn: PEN0=0 and PPGCm: PEN1=0). The count operation of the PPG down counters is stopped and the output of the PPG output pin is held at a Low level.

If the PPG down counter of each channel underflows, the reload values set in the PPG reload registers (PRLLn/PRLHn, PRLlm/PRLHm) are reloaded to the PPG down counter that underflows.

When an underflow occurs, the underflow generation flag bit in the channel that causes an underflow (PPGCn: PUF0=1, PPGCm: PUF1=1) is set. If an interrupt request is enabled at the channel that causes an underflow (PPGCn: PIE0=1, PPGCm: PIE1=1), an interrupt request is generated.

Notes:

- Do not operate PPGm (PPGCm: PEN1 = 1) when PPGn is stopped (PPGCn: PEN0 = 0).
- It is recommended to set the same value in both Low-level and High-level PPG reload registers (PRLLn/PRLHn, PRLlm/PRLHm).
- n = A, C, E
m = n + 1

The "H" and "L" pulse widths to be outputted are determined by adding "1" to the value in the PPG reload register of each channel and multiplying it by the count clock cycle.

The equations for calculating the pulse width are shown below:

$$P_L = T \times (L_n + 1) \times (L_m + 1)$$

$$P_H = T \times (H_n + 1) \times (H_m + 1)$$

P_L : "L" width of output pulse of PPGm pin

P_H : "H" width of output pulse of PPGm pin

L_n : Values of 8 bits in PPG reload register (PRLLn)

H_n : Values of 8 bits in PPG reload register (PRLHn)

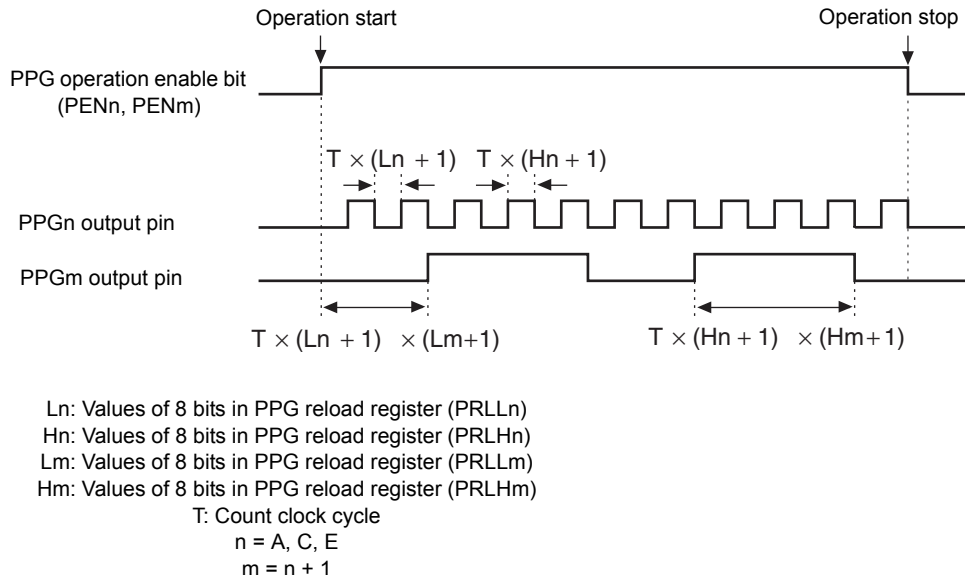
L_m : Values of 8 bits in PPG reload register (PRLlm)

H_m : Values of 8 bits in PPG reload register (PRLHm)

T: Count clock cycle

Figure 16-15. shows the output waveform in the 8+8-bit PPG output operation mode.

Figure 16-15. Output Waveform in 8+8-bit PPG Output Operation Mode



16.6 Notes on Using 8-/16-bit PPG Timer

This section explains the notes on using the 8-/16-bit PPG timer.

■ Notes on Using 8-/16-bit PPG Timer

- Effect on 8-/16-bit PPG timer when using time-base timer output

If the output signal of the time-base timer is used as the input signal for the count clock of the 8-/16-bit PPG timer (PPGnm: PCM2 to PCM0=111_B, PCS2 to PCS0=111_B), deviation may occur in the first count cycle in which the PPG timer is started by trigger input or in the count cycle immediately after the PPG timer is stopped.

When the time-base timer counter is cleared (TBTC: TBR=0) during the count operation of the PPG down counter, deviation may occur in the count cycle.

Setting of PPG reload registers when using 8-bit PPG timer

The "L" level and "H" level pulse widths are determined at the timing of reloading the values in the "L" level PPG reload registers (PRLLn, PRLm) to the PPG down counter.

If the 8-bit PPG timer is used in the 8-bit PPG output 2-channel independent operation mode or the 8 + 8-bit PPG output operation mode, use a word instruction to set both "H" level and "L" level PPG reload registers (PRLLn/PRLHn, PRLm/PRLHm) at the same time.

Using a byte instruction may cause an unexpected pulse to be generated.

[Example of rewriting PPG reload registers using byte instruction]

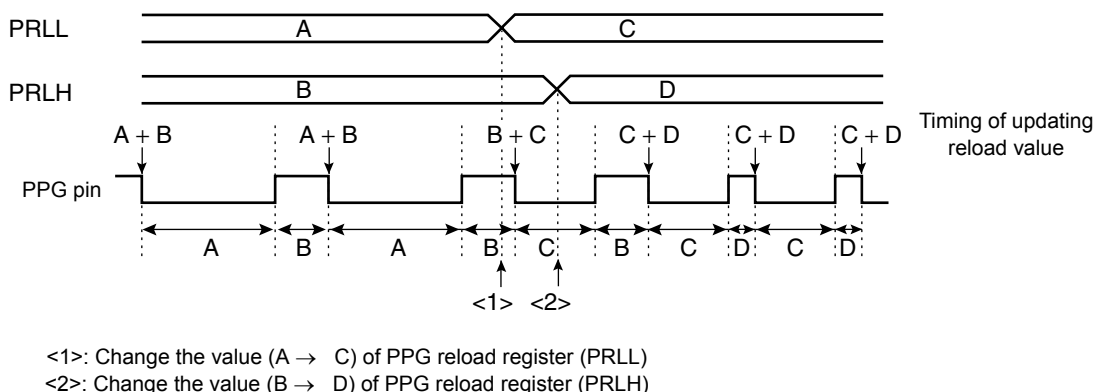
Immediately before the signal level of the PPG pin switches from "H" to "L", if the value in the "H" level PPG reload register (PRLH) is rewritten after the value in the "L" level PPG reload register (PRL) is rewritten using the byte instruction, a "L" level pulse width is generated after rewriting and a "H" level pulse width is generated before rewriting.

Figure 16-16. shows the waveform as the values in the PPG reload registers are rewritten using the byte instruction.

Note:

n = A, C, E
 m = B, D, F

Figure 16-16. Waveform When Values in PPG Reload Registers Rewritten Using Byte Instruction



□ Setting of PPG reload registers when using 16-bit PPG timer

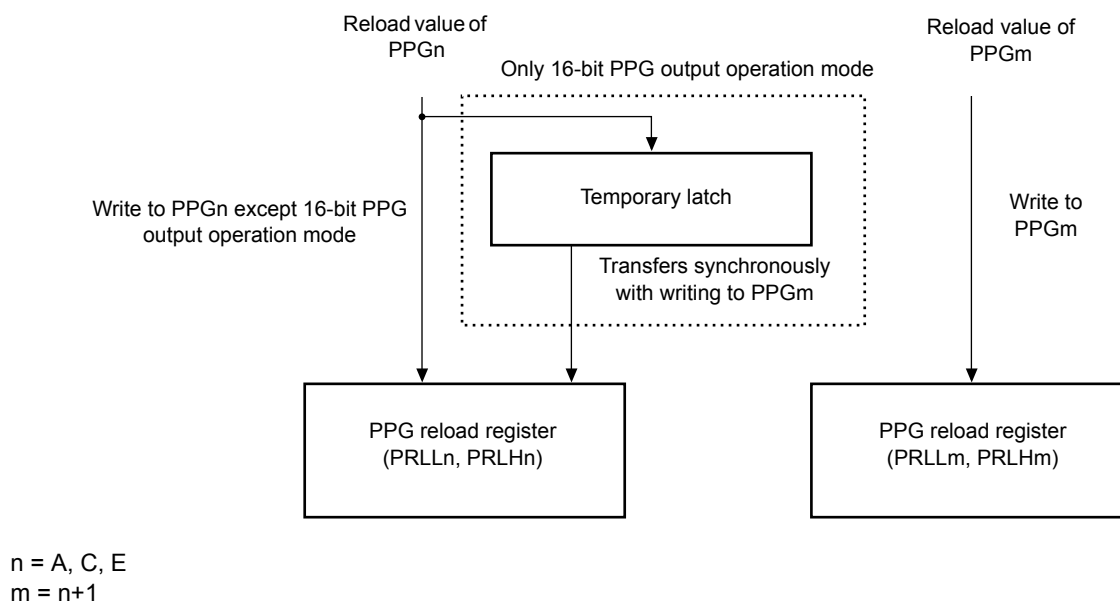
Use a long-word instruction to set the PPG reload registers (PRLn/PRLHn, PRLm/PRLHm) or a word instruction to set the PPGn and PPGm (PRLn / PRLHn → PRLm / PRLHm) in this order.

[Reload timing in 16-bit PPG output operation mode]

In the 16-bit PPG output operation mode, the reload values written to the PPGn reload registers (PRLn, PRLHn) are written temporarily to the temporary latch, written to the PPGm reload registers (PRLm/PRLHm), and then transferred to the PPGn reload registers (PRLn/PRLHn). Therefore, when setting the reload value in the PPGm reload registers (PRLm/PRLHm), it is necessary to set the reload value in the PPGn reload registers (PRLn/PRLHn) simultaneously or set the reload value in the PPGn reload registers (PRLn/PRLHn) before setting it in the PPGm reload registers (PRLm/PRLHm).

Figure 16-17. shows the reload timing in the 16-bit PPG output operation mode.

Figure 16-17. Reload Timing in 16-bit PPG Output Operation Mode



17. DTP/External Interrupt



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

[17.1 Overview of DTP/External Interrupt](#)

[17.2 Block Diagram of DTP/External Interrupt](#)

[17.3 Configuration of DTP/External Interrupt](#)

[17.4 Explanation of Operation of DTP/External Interrupt](#)

[17.5 Notes on Using DTP/External Interrupt](#)

[17.6 Program Example of DTP/External Interrupt Circuit](#)

17.1 Overview of DTP/External Interrupt

The DTP/external interrupt sends interrupt requests from external peripheral devices or data transfer requests to the CPU to generate an external interrupt request, or starts the EI²OS.

■ DTP/External Interrupt Function

The DTP/external interrupt follows the same procedure as resource interrupts to send interrupt requests from external peripheral devices to the CPU to generate an external interrupt request, or starts the EI²OS.

If the EI²OS is disabled in the interrupt control register (ICR: ISE=0), the external interrupt function is enabled, branching to interrupt processing.

If the EI²OS is enabled, the DTP function is enabled and automatic data transfer is performed, branching to interrupt processing after the completion of data transfer for the specified number of times.

Table 17-1. shows an overview of the DTP/external interrupt.

Table 17-1. Overview of DTP/External Interrupt

	External Interrupt	DTP Function
Input pin	8 pins: INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R, INT15R	
Interrupt factor	The interrupt factor is set in unit of pins using the detection level setting registers (ELVR1).	
	Input of "H" level, "L" level, rising edge, or falling edge	Input of "H" level or "L" level
Interrupt number	#26(1A _H), #28(1C _H)	
Interrupt control	The interrupt request output is enabled/disabled using the DTP/external interrupt enable register (ENIR1).	
Interrupt flag	The interrupt factor is held using the DTP/external interrupt factor register (EIRR1).	
Processing selection	The EI ² OS is disabled. (ICR: ISE=0)	The EI ² OS is enabled. (ICR: ISE=1)

Table 17-1. Overview of DTP/External Interrupt

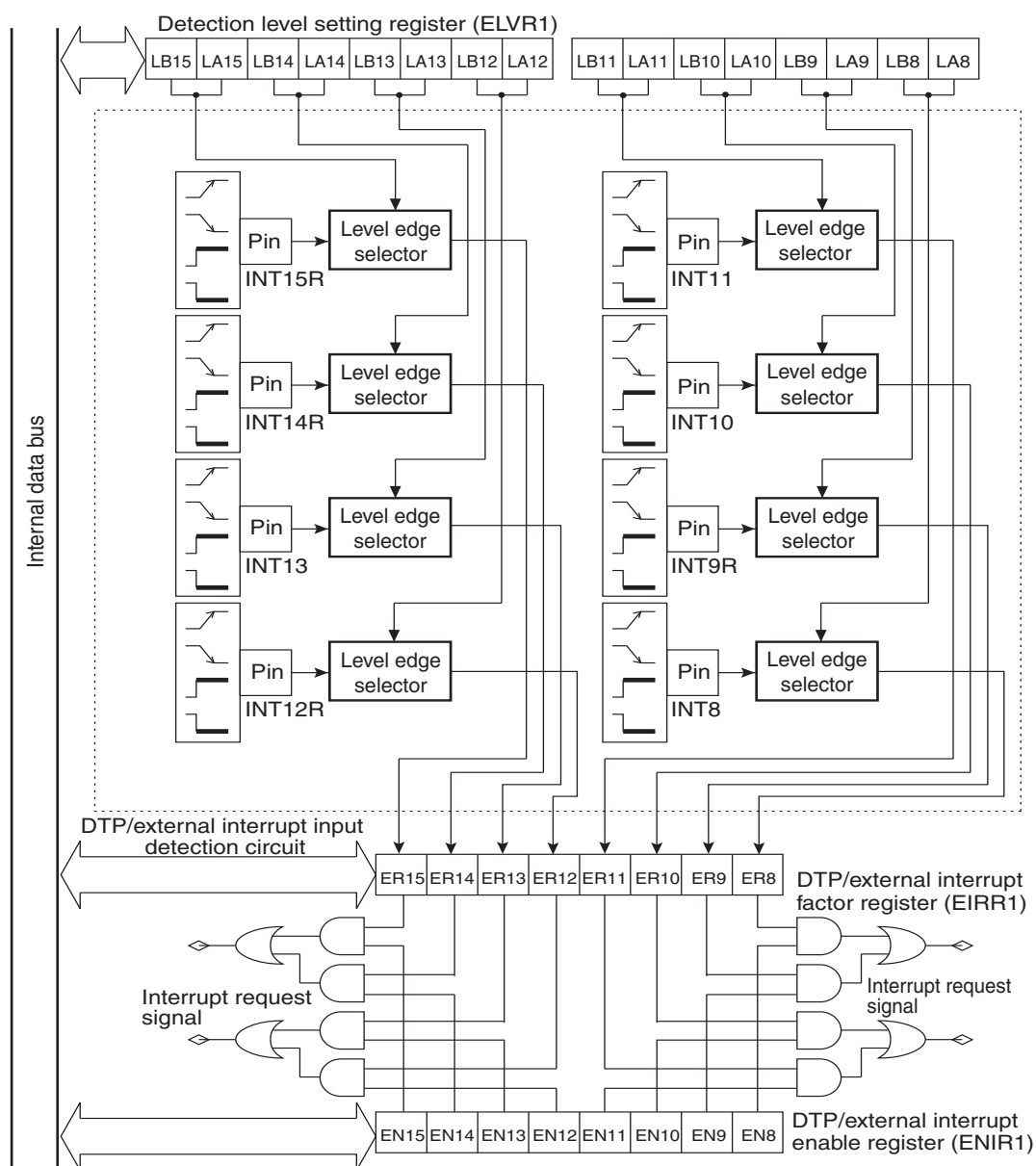
	External Interrupt	DTP Function
Processing contents	A branch is caused to the external interrupt processing.	EI ² OS performs automatic data transfer and completes the specified number of time for data transfers, causing a branch to the interrupt processing.

17.2 Block Diagram of DTP/External Interrupt

The block diagram of the DTP/external interrupt is shown below.

■ Block Diagram of DTP/External Interrupt

Figure 17-1. Block Diagram of DTP/External Interrupt



- DTP/external interrupt input detection circuit

This circuit detects interrupt requests or data transfer requests generated from external peripheral devices.

The interrupt request flag bit corresponding to the pin whose level or edge set by the detection level setting register (ELVR) is detected is set to "1" (EIRR1:ER).

- Detection level setting register (ELVR1)

This register sets the level or edge of input signals from external peripheral devices that cause DTP/external interrupt factors.

- DTP/external interrupt source register (EIRR1)

This register holds DTP/external interrupt sources.

If an enable signal is inputted to the DTP/external interrupt pin, the corresponding DTP/external interrupt request flag bit is set to "1".

- DTP/external interrupt enable register (ENIR1)

This register enables or disables DTP/external interrupt requests from external peripheral devices.

■ Details of Pins and Interrupt Numbers

Table 17-2. shows the pins and interrupt numbers used in the DTP/external interrupt.

Table 17-2. Pins and Interrupt Numbers Used by DTP/External Interrupt

Pin	Channel	Interrupt number
P54	INT8	#26(1A _H)
P42	INT9R	
P55	INT10	
P56	INT11	
P80	INT12R	#28(1C _H)
P57	INT13	
P82	INT14R	
P84	INT15R	

INT9R, INT12R, INT14R, and INT15R are enabled by setting the corresponding bit of the external interrupt source select register (EISSR) to "1".

17.3 Configuration of DTP/External Interrupt

This section lists and details the pins, interrupt factors, and registers in the DTP/external interrupt.

■ Pins of DTP/External Interrupt

The pins used by the DTP/external interrupt serve as general-purpose I/O ports.

Table 17-3. lists the pin functions and the pin setting required for use in the DTP/external interrupt.

Table 17-3. Pins of DTP/External Interrupt

Pin Name	Pin Function	Pin Settings Required for Use in DTP/ External Interrupt
P54/AN12/TOT3/ INT8	General-purpose I/O ports, analog input, event pin for reload timer, DTP external interrupt inputs	Set external interrupt source select register (EISSR) to "0". Set as input ports in port direction register (DDR5).
P55/AN13/INT10	General-purpose I/O ports, analog input, DTP external interrupt inputs	
P56/AN14/INT11/ DA0		
P57/AN15/INT13		
P42/ INT9R/ RX1	General-purpose I/O ports, DTP external interrupt inputs, CAN1 input Rx1	Set external interrupt source select register (EISSR) to "1". Set as input ports in port direction register (DDR4).
P80/ INT12R/ ADTG	General-purpose I/O ports, DTP external interrupt inputs, A/D converter trigger input ADTG	Set external interrupt source select register (EISSR) to "1". Set as input ports in port direction register (DDR8).
P82/ INT14R/ SIN0/ TIN2	General-purpose I/O ports, DTP external interrupt inputs, UART0 input SIN0, reload timer 2 trigger input TIN2	
P84/ INT15R/ SCK0	General-purpose I/O ports, DTP external interrupt inputs, UART0 clock I/O SCK0	

■ List of Registers and Initial Values in DTP/External Interrupt

Figure 17-2. List of Registers and Initial Values in DTP/External Interrupt

ENIR1	bit	7	6	5	4	3	2	1	0	Initial value
Address: 0000CA _H		EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	00000000 _B
EIRR1	bit	15	14	13	12	11	10	9	8	Initial value
Address: 0000CB _H		ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8	XXXXXXXX _B
ELVR1	bit	7	6	5	4	3	2	1	0	Initial value
Address: 0000CC _H		LB11	LA11	LB10	LA10	LB9	LA9	LB8	LA8	00000000 _B
ELVR1	bit	15	14	13	12	11	10	9	8	Initial value
Address: 0000CD _H		LB15	LA15	LB14	LA14	LB13	LA13	LB12	LA12	00000000 _B
EISSR	bit	7	6	5	4	3	2	1	0	Initial value
Address: 0000CE _H		INT15R	INT14R	INT13R	INT12R	INT11R	INT10R	INT9R	INT8R	00000000 _B

17.3.1 DTP/External Interrupt Source Register 1 (EIRR1)

The DTP/external interrupt source register holds DTP/external interrupt factors.

When a valid signal is inputted to the DTP/external interrupt pin, the corresponding DTP/external interrupt request flag bit is set to "1".

The EIRR1 register is corresponding to INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R, and INT15R.

■ DTP/External Interrupt Source Register 1 (EIRR1)

Figure 17-3. DTP/External Interrupt Source Register 1 (EIRR1)

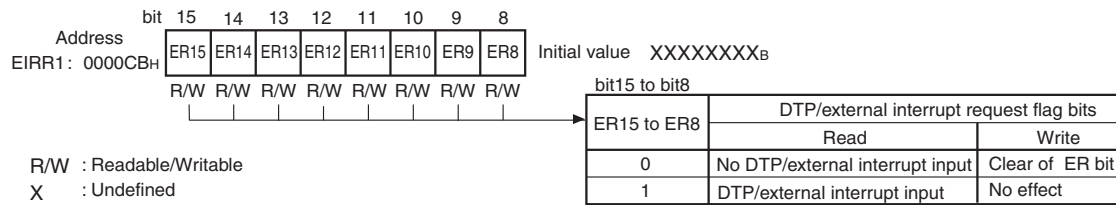


Table 17-4. Function of DTP/External Interrupt Factor Register 1 (EIRR1)

Bit Name		Function
bit15 to bit8	ER15 to ER8: DTP/External interrupt request flag bits	<p>These bits are set to "1" when the edges or level signals set by the detection condition select bits in the detection level setting register (ELVR1:LB, LA) are inputted to the DTP/external interrupt pins.</p> <p>When set to "1":</p> <p>When the DTP/external interrupt request enable bit (ENIR1:EN) is set to "1", an interrupt request is generated to the corresponding DTP/external interrupt channel.</p> <p>When set to "0":Cleared</p> <p>When set to "1":No effect</p> <p>If setting "1" to these bits and writing "0" to them occur simultaneously, writing "0" has priority.</p> <p>Notes:</p> <p>Reading by read-modify-write (RMW) instructions always returns "1".</p> <p>If more than one DTP/external interrupt request is enabled (ENIR1:EN = 1), clear only the bit in the channel that accepts an interrupt (EIRR1:ER = 0). No other bits must be cleared unconditionally.</p> <p>The value of DTP/external interrupt request flag bit (EIRR:ER) is valid only when DTP/external interrupt request enable bit (ENIR: EN) is set to "1".</p> <p>When DTP/external interrupt is disabled (ENIR: EN=0), DTP/external interrupt request flag bit may be set without relation that DTP/external interrupt factor exists.</p> <p>Clear the corresponding DTP/external interrupt request flag bit (EIRR:ER) immediately before enabling the DTP/external interrupt (ENIR: EN=1).</p> <p>Reference:</p> <p>When the EI²OS is started, the interrupt request flag bit is automatically cleared after the completion of data transfer (EIRR1:ER = 0).</p>

17.3.2 DTP/External Interrupt Enable Register 1 (ENIR1)

The DTP/external interrupt enable register (ENIR1) enables/disables the DTP/external interrupt request in the external peripheral devices.

ENIR1 is corresponding to INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R and INT15R.

■ DTP/External Interrupt Enable Register 1 (ENIR1)

Figure 17-4. DTP/External Interrupt Enable Register 1 (ENIR1)

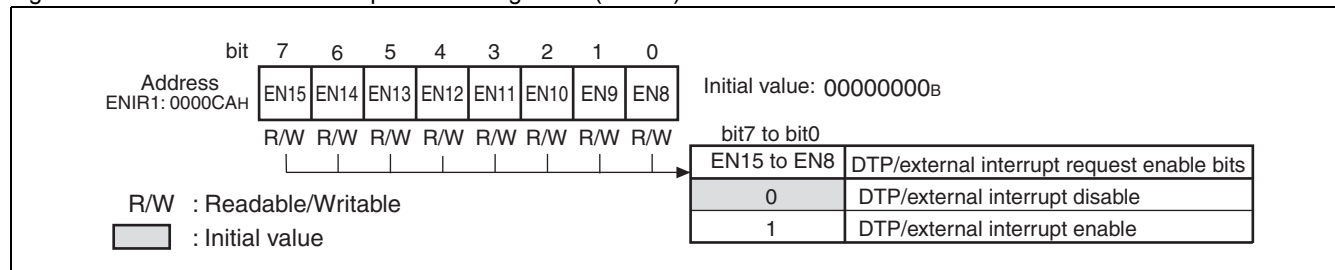


Table 17-5. Functions of DTP/External Interrupt Enable Register 1 (ENIR1)

Bit Name		Function
bit7 to bit0	EN15 to EN8: DTP/external interrupt request enable bits	<p>These bits enable or disable the DTP/external interrupt request to the DTP/external interrupt channel.</p> <p>If the DTP/external interrupt request enable bit (ENIR1:EN) and the DTP/external interrupt request flag bit (EIRR1:ER) are set to "1", the interrupt request is generated to the corresponding DTP/external interrupt pin.</p> <p>Note:</p> <p>Clear the corresponding DTP/external interrupt request flag bit (EIRR:ER) immediately before the DTP/external interrupt is enabled (ENIR: EN=1).</p> <p>Reference:</p> <p>The state of the DTP/external interrupt pin can be read directly using the port data register irrespective of the setting of the DTP/external interrupt request enable bit.</p>

Table 17-6. Correspondence between DTP/External Interrupt Pins, DTP/External Interrupt Request Flag Bits, and DTP/External Interrupt Request Enable Bits

DTP/external interrupt pins	DTP/external interrupt request flag bits	DTP/external interrupt request enable bits
INT8	ER8	EN8
INT9R	ER9	EN9
INT10	ER10	EN10
INT11	ER11	EN11
INT12R	ER12	EN12
INT13	ER13	EN13
INT14R	ER14	EN14
INT15R	ER15	EN15

17.3.3 Detection Level Setting Register 1 (ELVR1)

The detection level setting register sets the level or edge of input signals that cause the interrupt factors of the DTP/external interrupt pin.

ELVR1 is corresponding to INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R and INT15R.

■ Detection Level Setting Register 1 (ELVR1)

Figure 17-5. Detection Level Setting Register 1 (ELVR1)

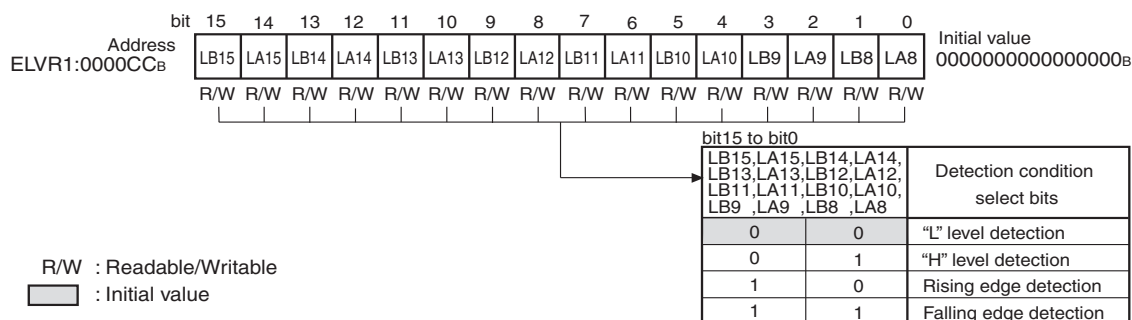


Table 17-7. Functions of Detection Level Setting Register 1 (ELVR1)

Bit Name		Function
bit15 to bit0	LB15, LA15 to LB8, LA8: Detection condition select bits	<p>These bits set the levels or edges of input signals from external peripheral devices that cause interrupt factors in the DTP/external interrupt pins.</p> <p>Two levels or two edges are selectable for external interrupts, and two levels are selectable for the EI²OS.</p> <p>Reference:</p> <p>When the set detection signal is input to the DTP/external interrupt pins, the DTP/external interrupt request flag bits are set to "1" even if DTP/external interrupt requests are disabled (ENIR1:EN = 0).</p>

Note:

If any setting of this register is changed, the interrupt source flag can be set.

Therefore, If you want to change the setting of this register, disable the interrupt (or, set the corresponding bit of ENIR1 to "0") in advance.

To enable the interrupt (or, set the corresponding bit of ENIR1 to "1") after the setting of this register is changed, be sure to clear the interrupt source flag bit (the corresponding bit of EIRR1).

Table 17-8. Correspondence between Detection Level Setting Register and Channels

DTP/External Interrupt Pins	Register Name	Bit Name
INT8	ELVR1	LB8, LA8
INT9R		LB9, LA9
INT10		LB10, LA10
INT11		LB11, LA11
INT12R		LB12, LA12
INT13		LB13, LA13
INT14R		LB14, LA14
INT15R		LB15, LA15

17.3.4 DTP/External Interrupt Source Select Register (EISSR)

The DTP/external interrupt source select register (EISSR) can change the assignment of the external interrupt pin. This allows the external interrupt. Also, the function such as CAN wakeup is implemented.

■ Selection of External Interrupt Source

The external interrupt pin of the upper 8-bit is assigned to INT13, INT11, INT10, and INT8 normally and shares the port 5 and pin.

The pin is switched by the DTP/external interrupt source select register (EISSR). In addition, because INT15R, INT14R, INT12R, and INT9R share the function such as CAN input pin, the function such as CAN wakeup can be implemented.

See [Table 17-10](#) for the pin function of INT15R, INT14R, INT12R, and INT9R.

Figure 17-6. DTP/external interrupt source select register (EISSR)

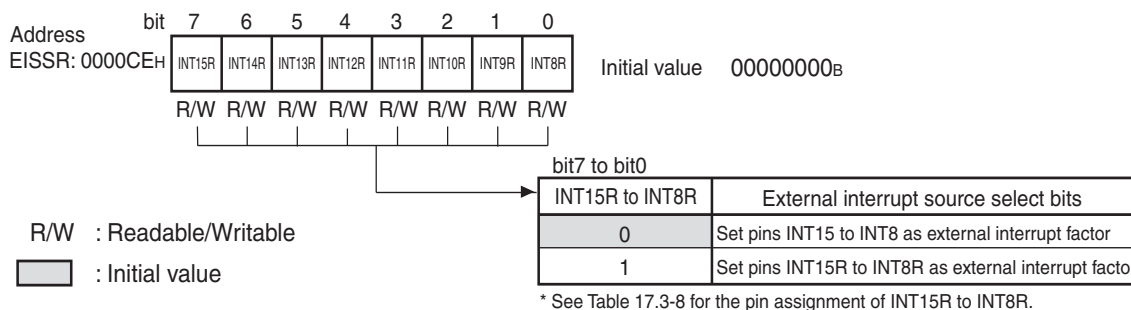


Table 17-9. Function of DTP/external interrupt source select register (EISSR)

Bit Name	Function
bit7 to bit0 INT15R to INT8R: External interrupt source select bits	<p>When these bits are set to "1", the input pin of the corresponding external interrupt factor (upper 8-bit) is assigned to the INT15R to INT8R.</p> <p>When set to "0":The external interrupt factor of the upper 8-bit is assigned to INT15 to INT8 pins.</p> <p>When set to "1":The external interrupt factor of the upper 8-bit is assigned to the INT15R to INT8R pins.</p>

Table 17-10. External Interrupt Factor Select (Upper 8-bit)

EISSR Bits	"0" (Initial Value)	"1"
INT8R	INT8: P54 (AN12/TOT3)	-
INT9R	-	INT9R: P42 (RX1)
INT10R	INT10: P55 (AN13)	-
INT11R	INT11: P56 (AN14)	-
INT12R	-	INT12R: P80 (ADTG)
INT13R	INT13: P57 (AN15)	-
INT14R	-	INT14R: P82 (SIN0/TIN2)
INT15R	-	INT15R: P84 (SCK0)

17.4 Explanation of Operation of DTP/External Interrupt

The DTP/external interrupt has an external interrupt function and a DTP function. The setting and operation of each function are explained.

■ Setting of DTP/External Interrupt

Using the DTP/external interrupt requires the setting shown in [Figure 17-7](#).

□ Setting procedure

To use the DTP/external interrupt, set each register by using the following procedure:

1. Set the input port to the general-purpose I/O port, which is shared with the pin to be used as external interrupt input.
2. Set the external interrupt source select register (EISSR) corresponding to the DTP/external interrupt channel to be used.
3. Set the interrupt request enable bit corresponding to the DTP/external interrupt channel to be used to "0" (ENIR1:EN).
4. Use the detection condition select bit corresponding to the DTP/external pin to be used to set the edge or level to be detected (ELVR1: LA, LB).
5. Set the interrupt request flag bit corresponding to the DTP/external interrupt channel to be used to "0" (EIRR1: ER).
6. Set the interrupt request enable bit corresponding to the DTP/external interrupt channel to be used to "1" (ENIR1: EN).
Note that concurrent writing with 16-bit data is available in 5 and 6.

When setting the registers for the DTP/external interrupt, the external interrupt request must be disabled in advance (ENIR1: EN = 0).

When enabling the DTP/external interrupt request (ENIR1:EN = 1), the corresponding DTP/external interrupt request flag bit must be cleared in advance (EIRR1:ER = 0). These actions prevent the mistaken interrupt request from occurring when setting the register.

□ Selecting of DTP or external interrupt function

Whether the DTP function or the external interrupt function is executed depends on the setting of the EI²OS enable bit in the corresponding interrupt control register (ICR:ISE).

If the ISE bit is set to "1", the EI²OS is enabled.

If the ISE bit and EN bit are set to "0", the EI²OS is disabled and the external interrupt function is executed.

Notes:

- All interrupt requests assigned to one interrupt control register have the same interrupt levels (IL2 to IL0).
- If two or more interrupt requests are assigned to one interrupt control register and the EI²OS is used in one of them, other interrupt requests cannot be used.
- Enabling unequipped pins causes a false operation. First set the EISSR and then set each of the registers when DTP/ external interrupt is used.

■ DTP/External Interrupt Operation

The control bits and the interrupt sources for the DTP/external interrupt are shown in [Table 17-11](#).

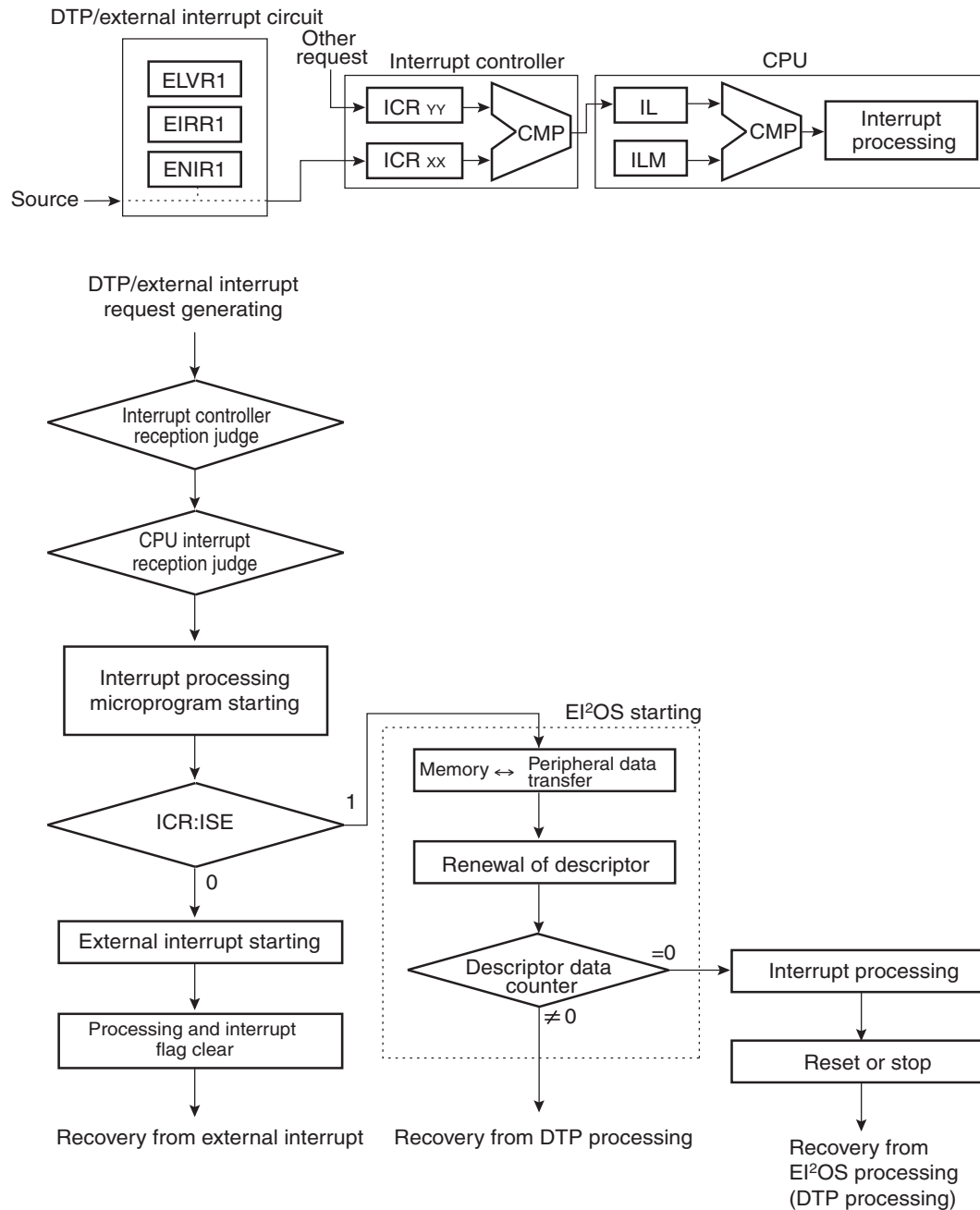
Table 17-11. Control Bits and Interrupt Sources for DTP/External Interrupt

	DTP/External interrupt
Interrupt request flag bit	EIRR1: ER15 to ER8
Interrupt request enable bit	ENIR1: EN15 to EN8
Interrupt source	Input of valid edge or level to INT13, INT11, INT10, INT8, INT9R, INT12R, INT14R, INT15R pins

If the interrupt request signal from the DTP/external interrupt is output to the interrupt controller and the EI²OS enable bit in the interrupt control register (ICR:ISE) is set to "0", the interrupt processing is executed. This bit is set to "1", the EI²OS is executed.

Figure 17-8. shows the operation of the DTP/external interrupt.

Figure 17-8. Operation of DTP/External Interrupt



17.4.1 External Interrupt Function

The DTP/external interrupt has an external interrupt function for generating an interrupt request by detecting the signal (edge or level) in the DTP/external interrupt pin.

■ External Interrupt Function

- When the signal (edge or level) set in the detection level setting register is detected in the DTP/external interrupt pin, the interrupt request flag bit in the DTP/external interrupt factor register (EIRR1:ER) is set to "1".

- ❑ If the interrupt request enable bit in the DTP/external interrupt enable register is enabled (ENIR1:EN = 1) with the interrupt request flag bit set to "1", the interrupt request generation is posted to the interrupt controller.
- ❑ If an interrupt request is preferred to other interrupt request by the interrupt controller, the interrupt request is generated.
- ❑ If the level of an interrupt request (ICR:IL) is higher than that of the interrupt level mask bit in the condition code register (CCR:ILM) and the interrupt enable bit is enabled (PS:CCR:I = 1), the CPU performs interrupt processing after completion of the current instruction execution and branches to interrupt processing.
- ❑ At interrupt processing, set the corresponding DTP/external interrupt request flag bit to "0" and clear the DTP/external interrupt request.

Notes:

- When the DTP/external interrupt start factor is generated, the DTP/external interrupt request flag bit (EIRR1:ER) is set to "1", regardless of the setting of the DTP/external interrupt request enable bit (ENIR1:EN).
- When the interrupt processing is started, clear the DTP/external interrupt request flag bit that caused the start factor. Control cannot be returned from the interrupt while the DTP/external interrupt request flag bit is set to "1". When clearing, do not clear any flag bit other than the accepted DTP/external interrupt factor.

17.4.2 DTP Function

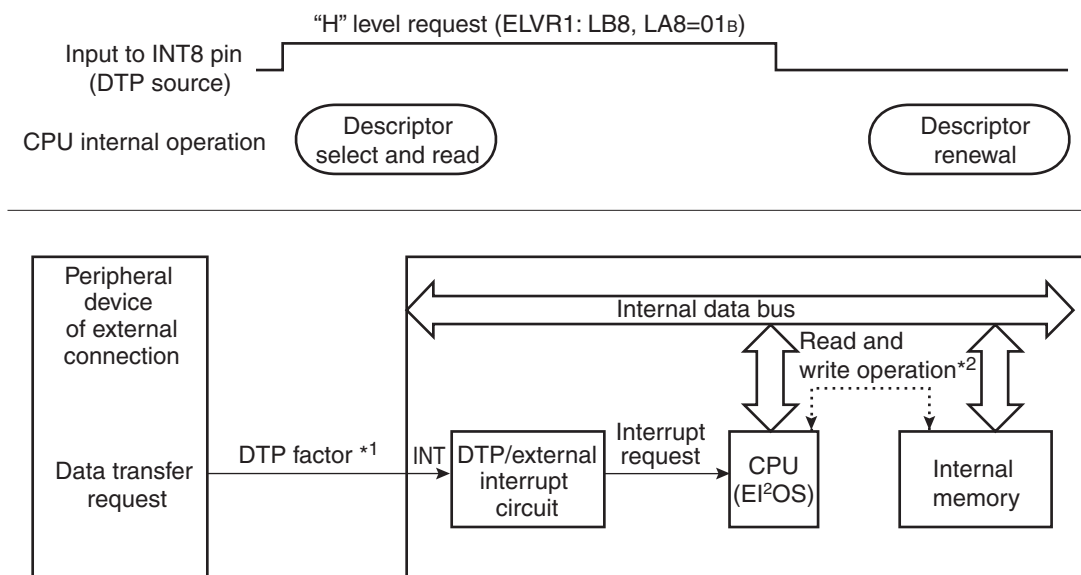
The DTP/external interrupt has the DTP function that detects the signal of the external peripheral device from the DTP/external interrupt pin to start the EI²OS.

■ DTP Function

The DTP function detects the signal level set by the detection level setting register of the DTP/external interrupt function to start the EI²OS.

- ❑ When the EI²OS operation is already enabled (ICR:ISE = 1) at the point when the interrupt request is accepted by the CPU, the DTP function starts the EI²OS and starts data transfer.
- ❑ When transfer of one data item is completed, the descriptor is updated and the DTP/external interrupt request flag bit is cleared to prepare for the next request from the DTP/external interrupt pin.
- ❑ When the EI²OS completes transfer of all the data, control branches to the interrupt processing.

Figure 17-9. Example of Interface with External Peripheral Device
(When Using EI²OS in Single-chip Mode)



*1: This must be canceled within three machine clocks after the start of data transfer.

*2: When EI²OS is "peripheral function" → "internal memory transfer".

17.5 Notes on Using DTP/External Interrupt

This section explains the notes on using the DTP/external interrupt.

■ Notes on Using DTP/External Interrupt

- Condition of external-connected peripheral device when DTP function is used

When using the DTP function, the peripheral device must automatically clear a data transfer request when data transfer is performed.

Inactivate the transfer request signal within three machine clocks after starting data transfer. If the transfer request signal remains active, the DTP/external interrupt regards the transfer request signal as a generation of next transfer request.

External interrupt input polarity

When the edge detection is set in the detection level setting register, the minimum pulse width is required to detect the edge described in the data sheet. Refer to the data sheet.

When a level causing an interrupt source is input with level detection set in the detection level setting register, the interrupt request flag bit in the DTP/external interrupt source register (EIRR1:ER) is set to "1" and the source is held as shown in [Figure 17-10](#).

With the source is held in the interrupt request flag bit (EIRR1:ER), the request to the interrupt controller remains active if the interrupt request is enabled (ENIR1:EN=1) even after the DTP/external interrupt source is canceled. To cancel the request to the interrupt controller, clear the interrupt request flag bit (EIRR1:ER) as shown in [Figure 17-11](#).

Figure 17-10. Clearing Factor Hold Circuit when Level Set

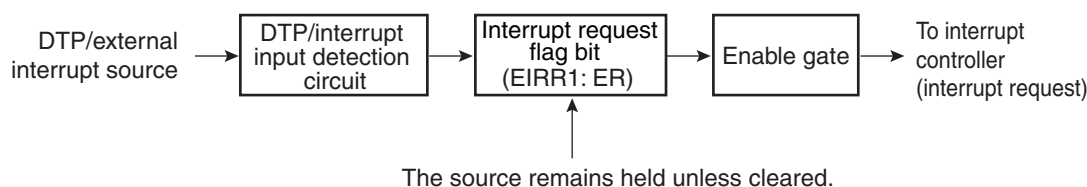
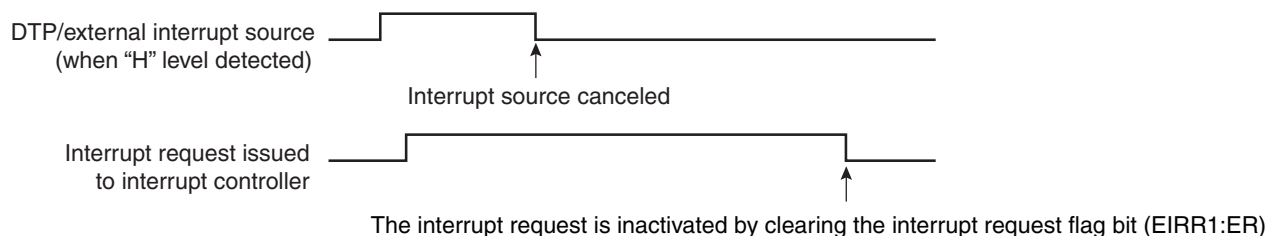


Figure 17-11. DTP/External Interrupt Source and Interrupt Request Generated when Interrupt Request Enabled



□ Notes on interrupts

When the DTP/external interrupt is used as the external interrupt function, no return from interrupt processing can be made with the DTP/external interrupt request flag bit set to "1" (EIRR1:ER) and the DTP/external interrupt request set to "enabled" (ENIR1:EN = 1). Always set the DTP/external interrupt request flag bit to 0 (EIRR1:ER) at interrupt processing.

When the level detection is set in the detection level setting register and the level that becomes the interrupt factor remains input, the DTP/external interrupt request flag bit is reset immediately even when cleared (EIRR1:ER = 0). Disable the DTP/external interrupt request output as needed (ENIR1:EN = 0), or cancel the interrupt factor itself.

17.6 Program Example of DTP/External Interrupt Circuit

This section gives a program example of the DTP/external interrupt function.

■ Program Example of DTP/External Interrupt Function

□ Processing specification

An external interrupt is generated by detecting the rising edge of the pulse input to the INT8 pin.

□ Coding example

```

ICR07 EQU 0000B7H      ;Interrupt control register ICR07
DDR5 EQU 000015H       ;Port 5 direction register
ENIR1 EQU 0000CAH      ;DTP/external interrupt enable
                        register 1
EIRR1 EQU 0000CBH      ;DTP/external interrupt factor
                        register 1
ELVR1L EQU 0000CCH     ;Detection level setting register 1:"L"
ELVR1H EQU 0000CDH     ;Detection level setting register 1:"H"
ADER5 EQU 00000BH      ;Port5 analog input enable register
ER8 EQU EIRR1:0         ;INT8 Interrupt request flag bit
EN8 EQU ENIR1:0         ;INT8 Interrupt request enable bit
;-----Main program-----
CODE CSEG
START:                  ;Stack pointer (SP) already
                        initialized
MOV I:ADER5,#00000000B ;Set analog input of port5 to disable
MOV I:DDR5,#00000000B ;Set DDR5 to input port
AND CCR,#0BFH          ;Interrupts disabled
MOV I:ICR07,#00H       ;Interrupt level 0 (highest)
CLRB I:EN8             ;INT8 disabled using ENIR1
MOV I:ELVR1L,#00000010B;Rising edge selected for INT8
CLRB I:ER0             ;INT8 interrupt request flag cleared
                        ;using EIRR1
SETB I:EN8             ;INT8 interrupt request enabled using
                        ENIR1
MOV ILM,#07H          ;Set ILM in PS to level 7

```

```

        OR     CCR,#40H           ;Interrupts enabled
LOOP:
        •
        Processing by user
        •
        BRA    LOOP
;-----Interrupt program-----
WARI:
        CLRB   I:ER8             ;Interrupt request flag cleared
        •
        Processing by user
        •
        RETI                    ;Return from interrupt processing
CODE    ENDS
;-----Vector setting-----
VECT    CSEG   ABS=0FFH
        ORG    00FF94H           ;Set vector to interrupt number
                                   #26 (1AH)
        DSL    WARI
        ORG    00FFDCH           ;Reset vector set
        DSL    START
        DB     00H               ;Set to single-chip mode
VECT    ENDS
        END     START

```

■ Program Example of DTP Function

□ Processing specification

ch.0 of the EI²OS is started by detecting the "H" level of the signal input to the INT8 pin.

RAM data is outputted to port 5 by performing DTP processing (EI²OS).

□ Coding example

```

ICR07 EQU 0000B7H           ;DTP/external interrupt control
                               register
DDR6 EQU 000016H           ;Port 6 direction register
DDR5 EQU 000015H           ;Port 5 direction register
ENIR1 EQU 0000CAH           ;DTP/external interrupt enable
                               register 1
EIRR1 EQU 0000CBH           ;DTP/external interrupt factor
                               register 1
ELVR1L EQU 0000CCH          ;Detection level setting register 1:"L"
ELVR1H EQU 0000CDH          ;Detection level setting register 1:"H"
ADER5 EQU 00000BH           ;Port5 analog input enable register
ADER6 EQU 00000CH           ;Port6 analog input enable register
ER1 EQU EIRR:0              ;INT8 interrupt request flag bit
EN1 EQU ENIR:0              ;INT8 interrupt request enable bit
;
BAPL EQU 000100H            ;Buffer address pointer lower
BAPM EQU 000101H            ;Buffer address pointer middle
BAPH EQU 000102H            ;Buffer address pointer higher
ISCS EQU 000103H            ;EI2OS status register
IOAL EQU 000104H            ;I/O address register lower
IOAH EQU 000105H            ;I/O address register higher
DCTL EQU 000106H            ;Data counter lower
DCTH EQU 000107H            ;Data counter higher
;
;-----Main program-----
CODE    CSEG

```

```

START:                                ;Stack pointer (SP) already
                                      ;initialized
MOV   I:ADER5,#00000000B ;Set analog input of port5 to disable
MOV   I:ADER6,#00000000B ;Set analog input of port6 to disable
MOV   I:DDR6,#11111111B  ;Set DDR6 to output port
MOV   I:DDR5,#00000000B  ;Set DDR5 to input port
AND   CCR,#0BFH          ;Interrupts disabled
MOV   I:ICR07,#08H       ;Interrupt level 0 (highest) EI2OS
                                      ;ch.0
;Data bank register (DTB) = 00H
MOV   BAPL,#00H          ;Address for storing output data set
MOV   BAPM,#06H          ;(600H to 60AH used)
MOV   BAPH,#00H
MOV   ISCS,#12H          ;Byte transfer, buffer address +1,
                                      ;I/O address fixed,
                                      ;transfer from memory to I/O
MOV   IOAL,#00H          ;Set port 0 as transfer destination
MOV   IOAH,#00H          ;address pointer
MOV   DCTL,#0AH          ;Set transfer count to 10
MOV   DCTH,#00H
;
CLR   I:EN8              ;INT8 disabled using ENIR1
MOV   I:ELVR1L,#00000001B;H level detection set for INT8
CLR   I:ER8              ;INT8 interrupt request flag cleared
                                      ;using EIRR1
SETB  I:EN8              ;INT8 interrupt request enabled using
                                      ;ENIR1
MOV   ILM,#07H           ;Set ILM in PS to level 7
OR    CCR,#40H           ;Interrupts enabled
LOOP:
    •
    Processing by user
    •
    BRA  LOOP

```

```

;-----Interrupt program-----
WARI:
    CLRB    I:ER8                ;INT8 interrupt request flag cleared
    •
    Processing by user
    •
    RETI                    ;Return from interrupt processing
CODE    ENDS
;-----Vector setting-----
VECT    CSEG    ABS=0FFH
        ORG     00FF94H          ;Set vector to interrupt number
                                   #26 (1AH)
        DSL     WARI
        ORG     00FFDCH          ;Reset vector set
        DSL     START
        DB      00H              ;Set to single-chip mode
VECT    ENDS
END      START

```


18. 8-/10-Bit A/D Converter



This chapter describes the functions and operations of the 8-/10-bit A/D converter.

18.1 Overview of 8-/10-bit A/D Converter

18.1 Block Diagram of 8-/10-bit A/D Converter

18.2 Configuration of 8-/10-bit A/D Converter

18.3 Interrupts of 8-/10-bit A/D Converter

18.4 Explanation of 8-/10-bit A/D Converter Operations

18.5 Notes on Using 8-/10-bit A/D Converter

18.1 Overview of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter converts analog input voltages to 8-bit or 10-bit digital values in the RC successive approximation conversion method.

- Up to 16 channels can be selected as analog input pins for input signals.
- The software trigger or external trigger can be selected as a start trigger.

■ Functions of 8-/10-bit A/D Converter

This converter converts an analog voltage (input voltage) which is input to the analog input pin to an 8-bit or 10-bit digital value (A/D conversion).

The 8-/10-bit A/D converter has the following functions:

- ❑ The minimum A/D conversion time is 1.78 μs per channel, including the sampling time.*
- ❑ The minimum sampling time is 0.75 μs per channel.*
- ❑ The RC successive approximation conversion method with a sample and hold circuit is used as the conversion method.
- ❑ 8-bit or 10-bit resolution can be specified.
- ❑ Up to 16 channels can be used as analog input pins.
- ❑ Interrupt requests can be generated by storing A/D conversion results in the A/D data register.
- ❑ When an interrupt request is generated, EI²OS can be started.
- ❑ The software or external trigger (falling edge) can be selected as a start trigger.

*: When the frequency of the machine clock is 32 MHz and AV_{CC} is 4.5 V or higher

■ Conversion Modes of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter has the following conversion modes:

Table 18-1. Conversion Modes of 8-/10-bit A/D Converter

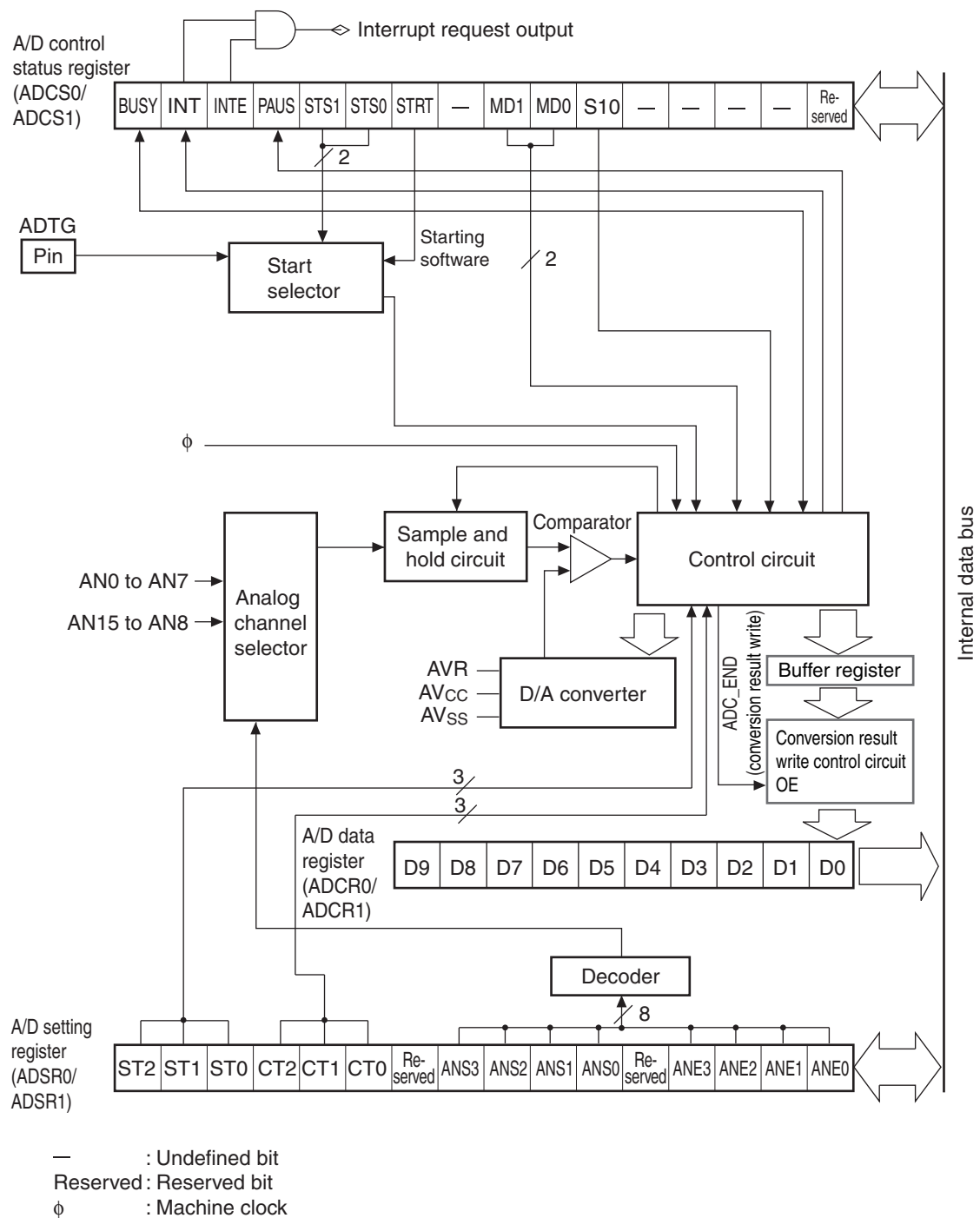
Conversion mode	Description
Single conversion mode	A/D conversion is performed from the start channel to the end channel in succession. Once the A/D conversion is completed for the end channel, the A/D conversion process is stopped.
Continuous conversion mode	A/D conversion is performed from the start channel to the end channel in succession. When the A/D conversion is completed for the end channel, the conversion sequence returns to the start channel to continue the A/D conversion operation.
Stop conversion mode	A/D conversion is performed while stopping after each channel is processed. Once the conversion is completed for the end channel, the conversion sequence returns to the start channel to repeat the same procedure.

18.1 Block Diagram of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter consists of the following blocks.

■ Block Diagram of 8-/10-bit A/D Converter

Figure 18-1. Block Diagram of 8-/10-bit A/D Converter



□ Detailed descriptions of pins and other items in the block diagram

Table 18-2. lists the actual pin names and interrupt request numbers of the 8-/10-bit A/D converter.

Table 18-2. Pins and Interrupt Request Numbers in Block Diagram

Pin name / interrupt request number in block diagram		Actual pin name / interrupt request number
ADTG	Trigger input pin	P80/ADTG/INT12R
AN0 to AN7	Analog input pins ch.0 to ch.7	P60/AN0 to P64/AN4 P65/AN5/PPGA(B) P66/AN6/PPGC(D) P67/AN7/PPGE(F)
AN8 to AN15	Analog input pins ch.8 to ch.15	P50/AN8/SIN2 P51/AN9/SOT2 P52/AN10/SCK2 P53/AN11/TIN3 P54/AN12/TOT3/INT8 P55/AN13/INT10 P56/AN14/INT11/DA0 P57/AN15/INT13
AVR	Vref+ Input pin	AVR
AV _{CC}	V _{CC} Input pin	AV _{CC}
AV _{SS}	V _{SS} Input pin	AV _{SS}
Interrupt request output	Interrupt request output	#29 (1D _H)

❑ A/D control status register (ADCS)

This register starts A/D conversion by software, selects the start trigger for the A/D conversion, selects the conversion mode, enables or disables interrupt requests, checks and clears the interrupt request flag, pauses A/D conversion operation, checks the status during the conversion, and selects the resolution type.

❑ Buffer register

This register temporarily stores the A/D conversion result.

❑ A/D data register (ADCR)

The A/D conversion result is temporarily stored in the buffer register. The result is stored in this register when the result is determined after the A/D conversion. This register can be used to read the result.

❑ A/D setting register (ADSR)

This register sets up the start channel and end channel for A/D conversion as well as sets the compare time and sampling time for A/D conversion.

❑ Start selector

This selects the type of triggers used to start A/D conversion. The internal timer output or external pin input can be specified as the start trigger.

❑ Decoder

The decoder selects an analog input pin to be used for A/D conversion by setting the A/D conversion start channel select bits (ADSR: ANS3 to ANS0) and A/D conversion end channel select bits (ADSR: ANE3 to ANE0).

❑ Analog channel selector

This selector selects a pin to be used for A/D conversion from the group of analog input pins (16 channels) after receiving a signal from the decoder.

□ Sample and hold circuit

This circuit holds the input voltage selected by the analog channel selector. As the circuit maintains the input voltage generated immediately after start of A/D conversion, the conversion can be performed without being affected by input voltage changes during the A/D conversion process.

□ D/A converter

This converter generates a reference voltage which is compared with the input voltage held in the sample and hold circuit.

□ Comparator

The comparator compares the input voltage held in the sample and hold circuit with the output voltage of the D/A converter to determine which is higher/lower.

□ Control circuit

This circuit determines an A/D conversion value after receiving a signal containing voltage comparison results from the comparator. Once the conversion results are confirmed, the result is stored in the A/D data register via the conversion result write control circuit.

When the output of interrupt requests is enabled, an interrupt is generated.

□ Conversion result write control circuit

When the conversion result is determined after A/D conversion, the result temporarily stored in the buffer register is transferred to the A/D data register.

18.2 Configuration of 8-/10-bit A/D Converter

The pins, registers and interrupt sources of the A/D converter are shown below.

■ Pins of 8-/10-bit A/D Converter

The pins of the 8-/10-bit A/D converter are also used as general-purpose I/O ports. [Table 18-3](#) lists the functions of each pin and settings for when using the 8-/10-bit A/D converter.

Table 18-3. Pins of 8-/10-bit A/D Converter

Function name	Pin name	Pin function	Setting for when using 8-/10-bit A/D converter
Trigger input	P80 / ADTG/INT12R	General-purpose I/O port / External trigger input / External interrupt input	Set as input port by port direction register DDR8
ch.0	P60 / AN0	General-purpose I/O ports / Analog inputs	Enable analog signal input (Corresponding bits of ADER6 : ADE7 to ADE0 set to "1")
ch.1	P61 / AN1		
ch.2	P62 / AN2		
ch.3	P63 / AN3		
ch.4	P64 / AN4		
ch.5	P65 / AN5/PPGA(B)	General-purpose I/O ports / Analog inputs / PPG outputs	
ch.6	P66 / AN6/PPGC(D)		
ch.7	P67 / AN7/PPGE(F)		

Table 18-3. Pins of 8-/10-bit A/D Converter

Function name	Pin name	Pin function	Setting for when using 8-/10-bit A/D converter
ch.8	P50 / AN8/SIN2	General-purpose I/O ports / Analog inputs / UART2 input/output	Enable analog signal input (Corresponding bits of ADER5 : ADE15 to ADE8 set to "1")
ch.9	P51 / AN9/SOT2		
ch.10	P52 / AN10/SCK2		
ch.11	P53 / AN11/TIN3	General-purpose I/O ports / Analog inputs / Event inputs of reload timer	
ch.12	P54 / AN12/TOT3 / INT8	General-purpose I/O ports / Analog inputs / Outputs of reload timer / External interrupt input	
ch.13	P55 / AN13/INT10	General-purpose I/O ports / Analog inputs / External interrupt input	
ch.14	P56 / AN14/INT11		
ch.15	P57 / AN15/INT13		

■ List of Registers and Initial Values of 8-/10-bit A/D Converter

Figure 18-2. List of Registers and Initial Values of 8-/10-bit A/D Converter

A/D control status register 1 (upper bits) ADCS1

bit	15	14	13	12	11	10	9	8	Initial value
Address: 000069 _H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	-	00000001 _B
	R/W	R/W	R/W	R/W	R/W	R/W	W	-	

A/D control status register 0 (lower bits) ADCS0

bit	7	6	5	4	3	2	1	0	Initial value
Address: 000068 _H	MD1	MD0	S10	-	-	-	-	Re-served	00011110 _B
	R/W	R/W	R/W	-	-	-	-	R/W	

Data register 1 (upper bits) ADCR1

bit	15	14	13	12	11	10	9	8	Initial value
Address: 00006B _H	-	-	-	-	-	-	D9	D8	11111100 _B
	-	-	-	-	-	-	R	R	

Data register 0 (lower bits) ADCR0

bit	7	6	5	4	3	2	1	0	Initial value
Address: 00006A _H	D7	D6	D5	D4	D3	D2	D1	D0	00000000 _B
	R	R	R	R	R	R	R	R	

A/D setting register 1 (upper bits) ADSR1

bit	15	14	13	12	11	10	9	8	Initial value
Address: 00006D _H	ST2	ST1	ST0	CT2	CT1	CT0	Re-served	ANS3	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

A/D setting register 0 (lower bits) ADSR0

bit	7	6	5	4	3	2	1	0	Initial value
Address: 00006C _H	ANS2	ANS1	ANS0	Re-served	ANE3	ANE2	ANE1	ANE0	00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Readable/Writable

R : Read only

W : Write only

- : Undefined bit

18.2.1 A/D Control Status Register 1 (ADCS1)

A/D control status register 1 (ADCS1) enables the following settings:

- Starting A/D conversion by software
- Selecting the start trigger for A/D conversion
- Enabling or disabling interrupt requests by storing A/D conversion results into the A/D data register
- Checking and clearing the interrupt request flag by storing A/D conversion results into the A/D data register
- Pausing A/D conversion operation and checking the status during the conversion

■ A/D Control Status Register 1 (ADCS1)

Figure 18-3. A/D Control Status Register 1 (ADCS1)

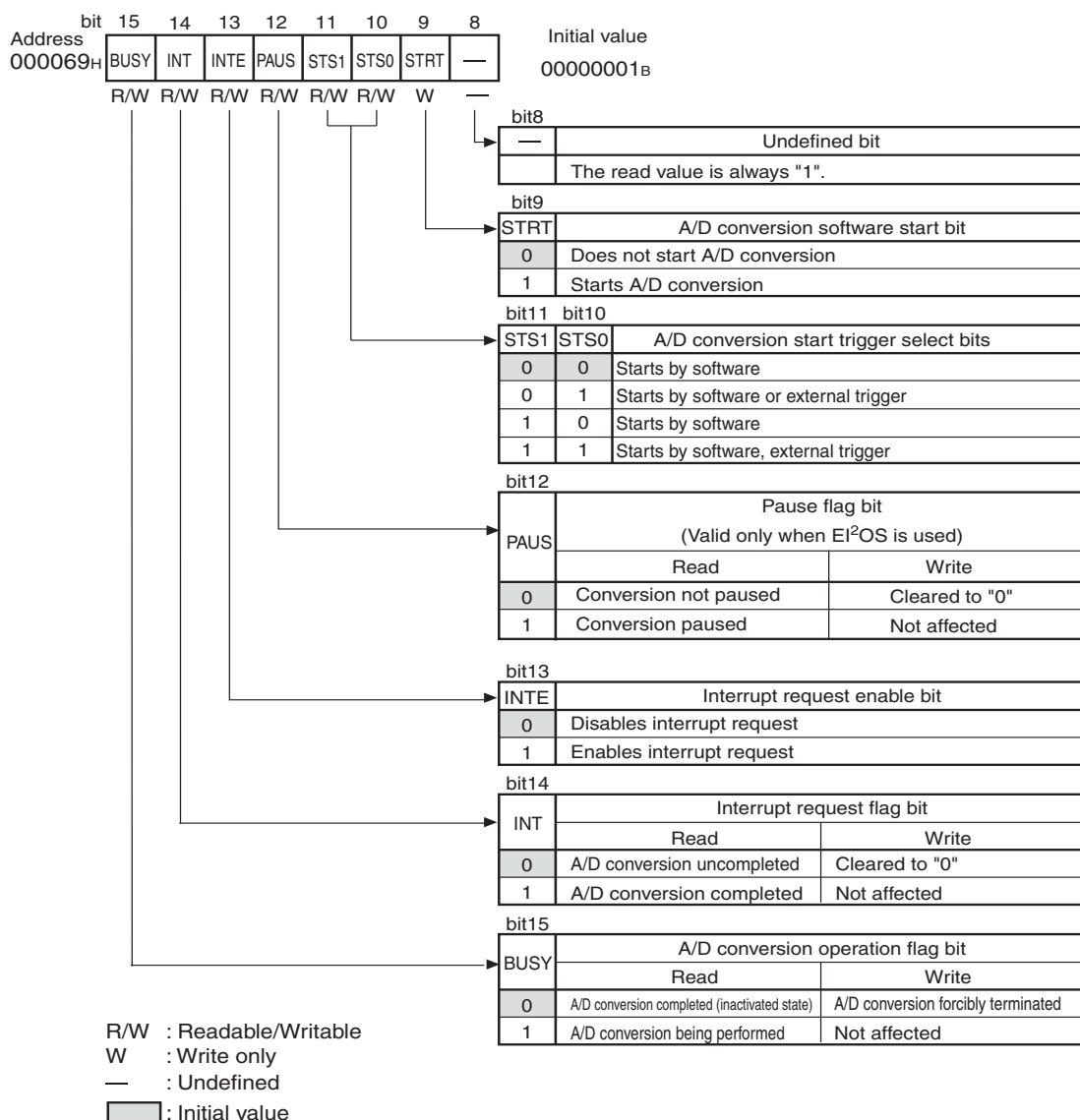


Table 18-4. Functions of A/D Control Status Register 1 (ADCS1) (Sheet 1 of 3)

Bit name	Function
bit 15 BUSY: A/D conversion operation flag bit	<p>This bit forcibly terminates the 8-/10-bit A/D converter. When read, it indicates whether the 8-/10-bit A/D converter is running or stopped.</p> <p>When set to "0": Forcibly terminates the 8-/10-bit A/D converter.</p> <p>When set to "1": Does not affect the operation</p> <p>When read: "1" is read if the 8-/10-bit A/D converter is running, while "0" is read if the converter is stopped. "1" is read at "stop state" in the stop conversion mode.</p> <p>Notes:</p> <p>"1" is read in the read-modify-write (RMW) instructions.</p> <p>In the single conversion mode, it is cleared upon the completion of A/D conversion.</p> <p>In the continuous conversion mode and the stop conversion mode, it is not cleared until stopped by writing "0".</p> <p>When the forced stop (BUSY=0) and activation (software (STRT=1)/external trigger/timer) of the A/D converter occur at the same time, the forced stop is preferred.</p>
bit 14 INT: Interrupt request flag bit	<p>This bit indicates the generation of an interrupt request.</p> <p>When A/D conversion is completed and the A/D conversion results are stored in the A/D data register (ADCR), the INT bit is set to "1".</p> <p>When interrupt requests are enabled (INTE = 1) and the interrupt request flag bit is set (INT = 1), an interrupt request is generated.</p> <p>This bit is cleared when "0" is written. Moreover, it is automatically cleared when the data transfer of the A/D conversion results is completed by EI²OS.</p> <p>If setting "1" to this bit and writing "0" to it occur simultaneously, writing "0" has priority.</p> <p>When set to "0": Cleared</p> <p>When set to "1": Not affected</p> <p>Note: "1" is read in the read-modify-write (RMW) instructions.</p>
bit 13 INTE: Interrupt request enable bit	<p>This bit enables or disables the output of interrupt requests.</p> <p>When interrupt requests are enabled (INTE = 1) and the interrupt request flag bit is set (INT = 1), an interrupt request is generated.</p> <p>Note: When transferring the A/D conversion results by EI²OS, always set the bit to "1".</p>

Table 18-4. Functions of A/D Control Status Register 1 (ADCS1) (Sheet 2 of 3)

Bit name	Function
bit 12	<p>PAUS: Pause flag bit</p> <p>The PAUS bit indicates that the A/D conversion data protection function has been activated. The PAUS bit is valid only when the output of interrupt requests is enabled (ADCS: INTE = 1).</p> <p>When the A/D conversion data protection function is activated: Set to "1"</p> <p>When written, this bit is set by "1" and it is cleared by "0".</p> <p>When the output of interrupt requests is enabled (ADCS: INTE = 1) and A/D conversion is performed, an interrupt request is generated as soon as the interrupt request flag bit (ADCS: INT) is set upon the completion of the first A/D conversion session. If the next A/D conversion session is completed without clearing the interrupt request flag bit (ADCS: INT) beforehand, the A/D conversion operation is paused to prevent the previous data from being overwritten (A/D conversion data protection function). When A/D conversion operation is paused, the PAUS bit is set to "1".</p> <p>When the interrupt request flag bit (ADCS: INT) is cleared, the 8-/10-bit A/D converter releases the pause status and restarts the A/D conversion operation.</p> <p>The interrupt request flag bit (ADCS: INT) is cleared by writing "0". Moreover, when the A/D data register is set to transfer the A/D conversion results by EI²OS, the interrupt request flag bit (ADCS: INT) is cleared by EI²OS upon the completion of transfer of the A/D conversion results.</p> <p>Notes:</p> <p>For the A/D conversion data protection function, see Section "18.4.5 A/D Conversion Data Protection Function".</p> <p>Even when the pause status is released, the PAUS bit is not cleared automatically. To clear the PAUS bit, write "0".</p>
bit 11, bit 10	<p>STS1, STS0: A/D conversion start trigger select bits</p> <p>These bits select the type of triggers used to start the 8-/10-bit A/D converter (start trigger).</p> <p>"00_B": Starting by software</p> <p>"01_B": Starting by external pin trigger / software</p> <p>"10_B": Starting by software</p> <p>"11_B": Starting by external pin trigger / software</p> <p>Notes:</p> <p>When external pin trigger is selected (01_B, 11_B):</p> <p>A/D conversion starts when the falling edge is detected at the ADTG pin.</p> <p>When more than one start triggers are selected (STS1 and STS0 are set to value other than "00_B"), the 8-/10-bit A/D converter is started by the first start trigger generated.</p> <p>To change start trigger settings, change them when the operations of the peripheral functions used to generate a start trigger are stopped (inactive trigger state).</p>

Table 18-4. Functions of A/D Control Status Register 1 (ADCS1) (Sheet 3 of 3)

Bit name		Function
bit 9	STRT: A/D conversion software start bit	<p>This bit activates the 8/10-bit A/D converter by software.</p> <p>When set to "1": Activates 8/10-bit A/D converter.</p> <p>Reactivates by the STRT bit only in the signal conversion mode 1(MD1/MD0=00).</p> <p>When set to "0": Invalid, no changes</p> <p>Notes:</p> <p>"0" is always read.</p> <p>(For the evaluation product, "0" is read in the read-modify-write (RMW) instruction and "1" is read in the read instruction except the read-modify-write (RMW) instruction.)</p> <p>When the forced stop (BUSY=0) and software activation (STRT=1) of the A/D converter are written simultaneously, the forced stop is preferred.</p>
bit 8	Undefined bit	<p>Read: "1" is always read.</p> <p>Write: Not affected.</p>

18.2.2 A/D Control Status Register 0 (ADCS0)

A/D control status register 0 enables the following settings:

- Selecting the A/D conversion mode
- Selecting the start channel and end channel of A/D conversion

■ A/D Control Status Register 0 (ADCS0)

Figure 18-4. A/D Control Status Register 0 (ADCS0)

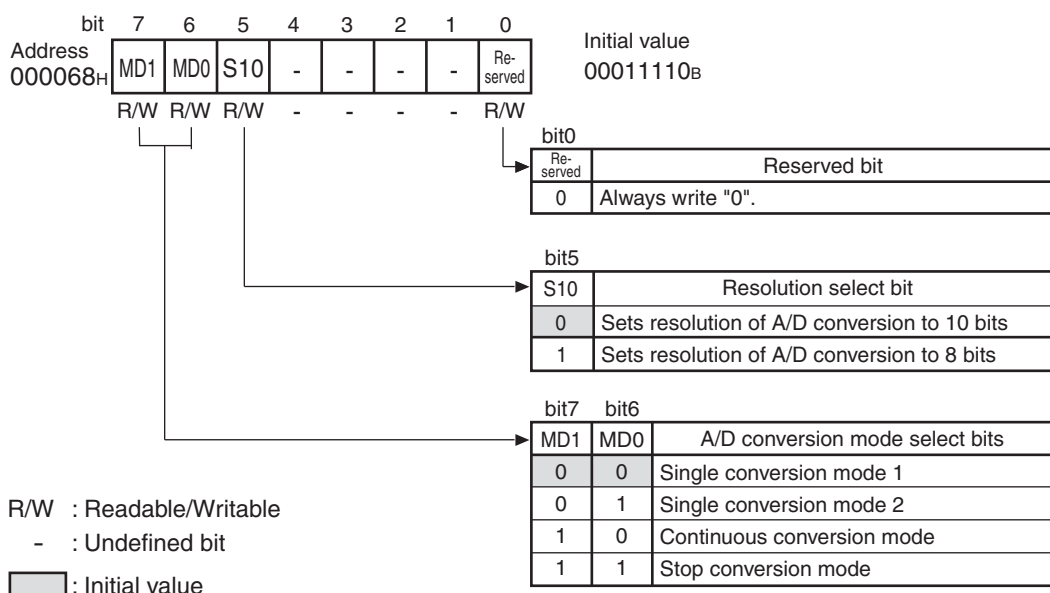


Table 18-5. Functions of A/D Control Status Register 0 (ADCS0)

Bit name		Function
bit 7, bit 6	MD1, MD0: A/D conversion mode select bits	<p>These bits set the A/D conversion mode.</p> <p>For detailed information about how to use each mode, see Section "18.4 Explanation of 8-/10-bit A/D Converter Operations".</p> <p>In single conversion modes 1 and 2: A/D conversion is performed on the analog input from the start channel (ADSR: ANS3 to ANS0) to the end channel (ADSR: ANE3 to ANE0) in succession. Once A/D conversion is completed for the end channel, the A/D conversion operation is stopped.</p> <p>For the difference between single conversion modes 1 and 2, see Section "18.4 Explanation of 8-/10-bit A/D Converter Operations".</p> <p>In single conversion mode 1, reactivation during the A/D conversion is allowed. However, reactivation is ignored when A/D conversion data protection function is activated and A/D is in the pause state.</p> <p>In continuous conversion mode: A/D conversion is performed on the analog input from the start channel (ADSR: ANS3 to ANS0) to the end channel (ADSR: ANE3 to ANE0) in succession. When the A/D conversion is completed for the end channel, the conversion sequence returns to the analog input of the start channel to continue the A/D conversion.</p> <p>In stop conversion mode: A/D conversion starts from the start channel (ADSR: ANS3 to ANS0). When the A/D conversion is completed for one channel, the A/D conversion operation is stopped. If a start trigger is entered while the conversion operation is still stopped, A/D conversion will be performed for the next channel.</p> <p>When A/D conversion is completed for the end channel, the A/D conversion operation is stopped. If a start trigger is entered while the conversion operation is still stopped, the conversion sequence will return to the analog input of the start channel to continue the A/D conversion.</p> <p>Note:When changing the conversion mode, make sure that A/D conversion has not already started (stop state). In single conversion mode 1, stop operation during A/D conversion and result when A/D conversion is completed simultaneously with A/D reactivation.</p>
bit 5	S10: Resolution select bit	<p>This bit sets the resolution of A/D conversion.</p> <p>When set to "0": Set resolution of A/D conversion to A/D conversion data bits D9 to D0 (10 bits)</p> <p>When set to "1": Set resolution of A/D conversion to A/D conversion data bits D7 to D0 (8 bits)</p> <p>Note:When changing S10 bit, make sure that A/D conversion has not already started (stop state). If S10 bit is changed after the A/D conversion has already started, the conversion results stored in the A/D conversion data bits (D9 to D0) will become invalid.</p>
bit4 to bit1	Undefined bits	These bits are read only. Initial value is "1".
bit0	Reserved bit	Always write "0" to this bit.

18.2.3 A/D Data Registers (ADCR0/ADCR1)

The data registers (ADCR0 and ADCR1) are used to record digital values derived from the conversion results. While ADCR0 records the lower 8 bits of the conversion results, ADCR1 records the highest 2 bits. Every time conversion is completed, these registers are rewritten, and in general, the last conversion values are stored.

■ A/D Data Registers (ADCR0/ADCR1)

Figure 18-5. A/D Data Registers (ADCR0/ADCR1)

A/D Data register - upper bits

Address	bit 15	14	13	12	11	10	9	8	Initial value
ADCR1 00006BH	-	-	-	-	-	-	D9	D8	11111100 _B
	-	-	-	-	-	-	R	R	

A/D Data register - lower bits

Address	bit 7	6	5	4	3	2	1	0	Initial value
ADCR0 00006AH	D7	D6	D5	D4	D3	D2	D1	D0	00000000 _B
	R	R	R	R	R	R	R	R	

R : Read only

- : Undefined bit

Table 18-6. Functions of A/D Data Registers (ADCR0/ADCR1)

Bit name		Function
bit 15 to bit 10	Undefined bits	When reading, "1" is always read.
bit 9 to bit 0	D9 to D0: A/D conversion data bits	<p>These bits store A/D conversion results.</p> <p>When resolution is set to 10 bits (S10 = 0): Conversion data is stored in D9 to D0 (10 bits).</p> <p>When resolution is set to 8 bits (S10 = 1): Conversion data is stored in D7 to D0 (8 bits). The read value of D9 and D8 becomes "1".</p> <p>Notes: Writing to these registers is prohibited. To read the conversion results stored in the A/D conversion data bits (D9 to D0), a word instruction (MOVW) must be used.</p>

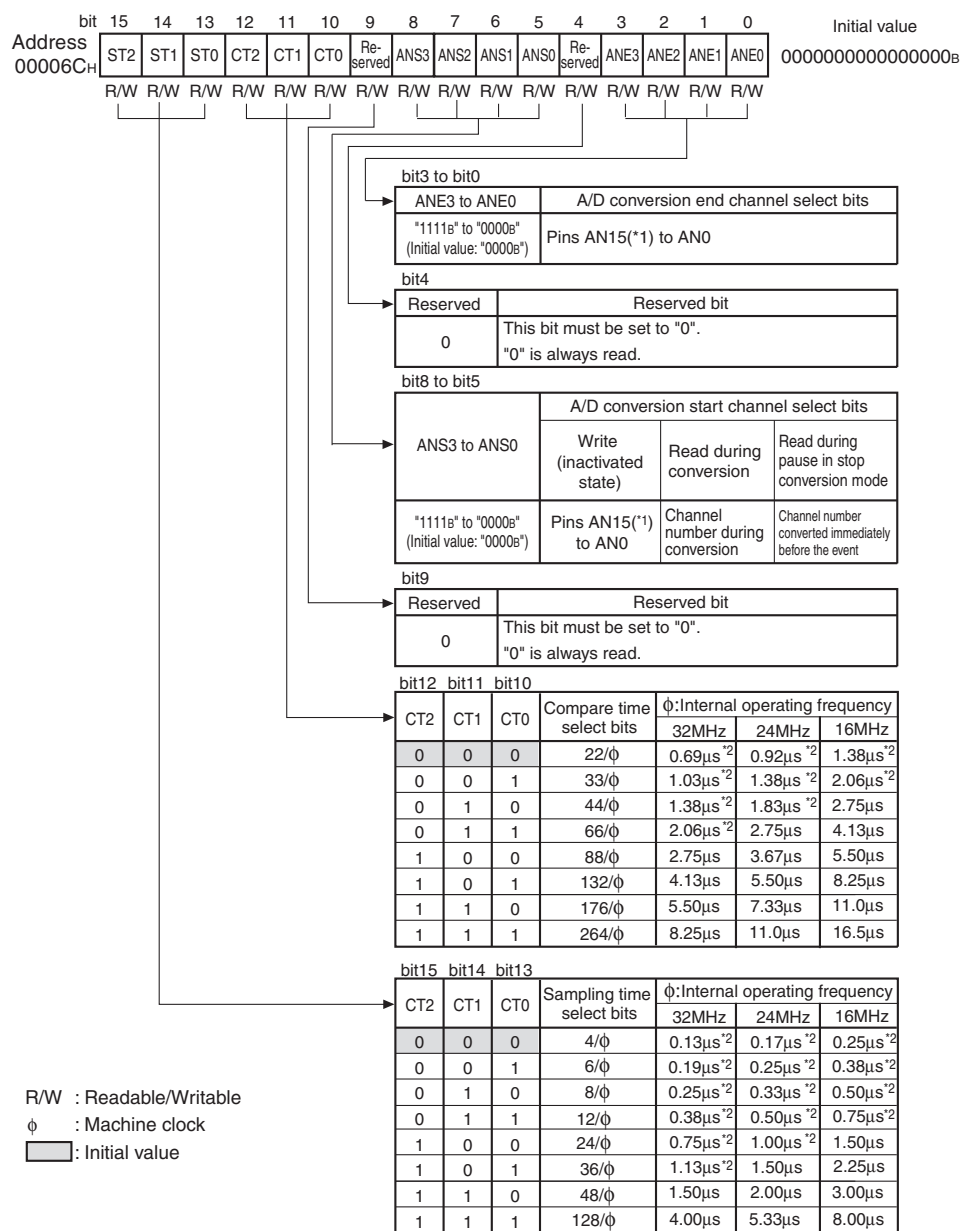
18.2.4 A/D Setting Registers (ADSR0/ADSR1)

The A/D setting registers (ADSR0/ADSR1) enable the following settings:

- Setting A/D conversion times (sampling time and compare time)
- Setting sampling channels (start channel and end channel)
- Displaying the current sampling channels

■ A/D Setting Registers (ADSR0/ADSR1)

Figure 18-6. A/D Setting Registers (ADSR0/ADSR1)



*1 : Pin AN15 to Pin AN0 can be set.

*2 : When AV_{cc} is lower than 4.5V, do not set. Conversion accuracy cannot be assured.

Table 18-7. Functions of A/D Setting Registers (ADSR0/ADSR1) (Sheet 1 of 2)

Bit name		Function
bit 15 to bit 13	ST2, ST1, ST0: Sampling time select bits	<p>These bits set the sampling time for A/D conversion.</p> <p>They set the time spent from start of A/D conversion to when the input analog voltage is sampled and then held in the sample and hold circuit.</p> <p>For information about the settings of these bits, see Table 18-8..</p>
bit 12 to bit 10	CT2, CT1, CT0: Compare time select bits	<p>These bits set the compare time for A/D conversion.</p> <p>They set the time spent from when A/D conversion is performed on the analog input to when the results are stored in the data bits (D9 to D0).</p> <p>For information about the settings of these bits, see Table 18-9..</p>
bit 9	Reserved bit	The bit must be set to "0". "0" is always read.
bit 8 to bit 5	ANS3 to ANS0: A/D conversion start channel select bits	<p>These bits set the channel from which A/D conversion is started. Reading them allows the channel number currently being converted to be identified during A/D conversion, and it allows the channel number of the last channel on which the A/D conversion was performed to be identified when the A/D conversion is completed or stopped. In addition, even when a particular value is set to these bits, the channel number of the channel on which the A/D conversion was performed previously is read rather than that set value until the A/D conversion starts. At reset, the value is initialized to "0000_B".</p> <p>Start channel < End channel:</p> <p>A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) and ends at the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).</p> <p>Start channel = End channel:</p> <p>A/D conversion is performed for the only one channel that is set by the A/D conversion start (= end) channel select bits (ANS3 to ANS0 = ANE3 to ANE0).</p> <p>Start channel > End channel:</p> <p>Do not set it.</p> <p>In continuous conversion mode and stop conversion mode:</p> <p>Once A/D conversion is completed at the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), the conversion sequence returns to the channel set by A/D conversion start channel select bits (ANS3 to ANS0).</p> <p>Read (in mode other than stop conversion mode):</p> <p>The channel number of the channel (15 to 0) for which A/D conversion is currently being performed is read.</p> <p>Read (in stop conversion mode):</p> <p>When a read is performed in the stop mode, the channel number of the last channel for which A/D conversion was performed immediately before it was stopped is read.</p> <p>Notes:</p> <p>Do not set the A/D conversion start channel bits (ANS3 to ANS0) during A/D conversion.</p> <p>Use the word access method when writing to these bits. If a byte-access or bit manipulation is performed, A/D conversion may start from an unintended channel.</p> <p>Do not use a read-modified write (RMW) instruction to set A/D conversion mode select bits (MD1, MD0) and A/D conversion end channel select bits (ANE3, ANE2, ANE1, ANE0) after setting the A/D conversion start channel select bits (ANS3, ANS2, ANS1, ANS0). The previously converted channel is read by bits ANS3, ANS2, ANS1, and ANS0 until A/D conversion operation starts.</p> <p>The bit values of ANE3, ANE2, ANE1, and ANE0 may be rewritten when a read-modified write (RMW) instruction is used to set bits MD1, MD0, ANE3, ANE2, ANE1, and ANE0 after setting the starting channel to bits of ANS3, ANS2, ANS1, and ANS0.</p>
bit 4	Reserved bit	The bit must be set to "0". "0" is always read.

Table 18-7. Functions of A/D Setting Registers (ADSR0/ADSR1) (Sheet 2 of 2)

	Bit name	Function
bit 3 to bit 0	ANE3 to ANE0: A/D conversion end channel select bits	<p>These bits set the channel at which A/D conversion ends. At reset, the value is initialized to "0000_B".</p> <p>Start channel < End channel: A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) and ends at the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).</p> <p>Start channel = End channel: A/D conversion is performed for the only one channel that is set by the A/D conversion start (= end) channel select bits (ANS3 to ANS0 = ANE3 to ANE0).</p> <p>Start channel > End channel: Do not set it.</p> <p>In continuous conversion mode and stop conversion mode: Once A/D conversion is completed at the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), the conversion sequence returns to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0).</p> <p>Notes: Do not set the A/D conversion end channel bits (ANE3 to ANE0) during A/D conversion. Do not use a read-modified write (RMW) instruction to set the sampling time select bits (ST2, ST1 and ST0), the compare time select bits (CT2, CT1 and CT0), and the A/D conversion end channel select bits (ANE3, ANE2, ANE1 and ANE0) after setting the A/D conversion start channel select bits (ANS3, ANS2, ANS1 and ANS0). The previously converted channel is read by bits ANS3, ANS2, ANS1 and ANS0 until A/D conversion operation starts. The bit values of ANS3, ANS2, ANS1 and ANS0 may be rewritten when a read-modified write instruction is used to set bits ST2, ST1, ST0, CT2, CT1, CT0, ANE3, ANE2, ANE1 and ANE0 after setting bits ANS3, ANS2, ANS1, and ANS0.</p>

■ Sampling Time Setup (Bits ST2 to ST0)

Table 18-8. Correlation between Bits ST2 to ST0 and Sampling Times

ST2	ST1	ST0	Sampling time setting	Example setting (ϕ : Internal operating frequency)		
				$\phi = 32 \text{ MHz}$	$\phi = 24 \text{ MHz}$	$\phi = 16 \text{ MHz}$
0	0	0	4 machine cycles	0.13 μs	0.17 μs	0.25 μs
0	0	1	6 machine cycles	0.19 μs	0.25 μs	0.38 μs
0	1	0	8 machine cycles	0.25 μs	0.38 μs	0.50 μs
0	1	1	12 machine cycles	0.38 μs	0.50 μs	0.75 μs
1	0	0	24 machine cycles	0.75 μs	1.00 μs	1.50 μs
1	0	1	36 machine cycles	1.13 μs	1.50 μs	2.25 μs
1	1	0	48 machine cycles	1.50 μs	2.00 μs	3.00 μs
1	1	1	128 machine cycles	4.00 μs	5.33 μs	8.00 μs

The sampling time must be set in accordance with the drive impedance (R_{ext}), which is connected to the analog input. The conversion accuracy is not guaranteed unless the following conditions are satisfied.

Refer to the data sheet for parameters.

- $R_{\text{ext}} \leq R_{\text{extmax}}$:
Set the sampling time to STmin or higher.
- $R_{\text{ext}} > R_{\text{extmax}}$: Set the sampling time to ST shown in the following expression, or higher.

$$\text{ST} = (R_{\text{in}} + R_{\text{ext}}) \times C_{\text{in}} \times 7$$

■ Compare Time Setup (Bits CT2 to CT0)

Table 18-9. Correlation between Bits CT2 to CT0 and Compare Times

CT2	CT1	CT0	Compare time setting	Example setting (ϕ : Internal operating frequency)		
				$\phi = 32 \text{ MHz}$	$\phi = 24 \text{ MHz}$	$\phi = 16 \text{ MHz}$
0	0	0	22 machine cycles	0.69 μs	0.92 μs	1.38 μs
0	0	1	33 machine cycles	1.03 μs	1.38 μs	2.06 μs
0	1	0	44 machine cycles	1.38 μs	1.83 μs	2.75 μs
0	1	1	66 machine cycles	2.06 μs	2.75 μs	4.13 μs
1	0	0	88 machine cycles	2.75 μs	3.67 μs	5.50 μs
1	0	1	132 machine cycles	4.13 μs	5.50 μs	8.25 μs
1	1	0	176 machine cycles	5.50 μs	7.33 μs	11.0 μs
1	1	1	264 machine cycles	8.25 μs	11.0 μs	16.5 μs

The compare time must be set in accordance with the analog power supply voltage (AV_{CC}). Refer to the data sheet for details.

18.2.5 Analog Input Enable Registers (ADER5, ADER6)

These registers enable or disable the analog input pins used for the 8-/10-bit A/D converter.

■ Analog Input Enable Registers (ADER5, ADER6)

Figure 18-7. Analog Input Enable Registers (ADER5, ADER6)

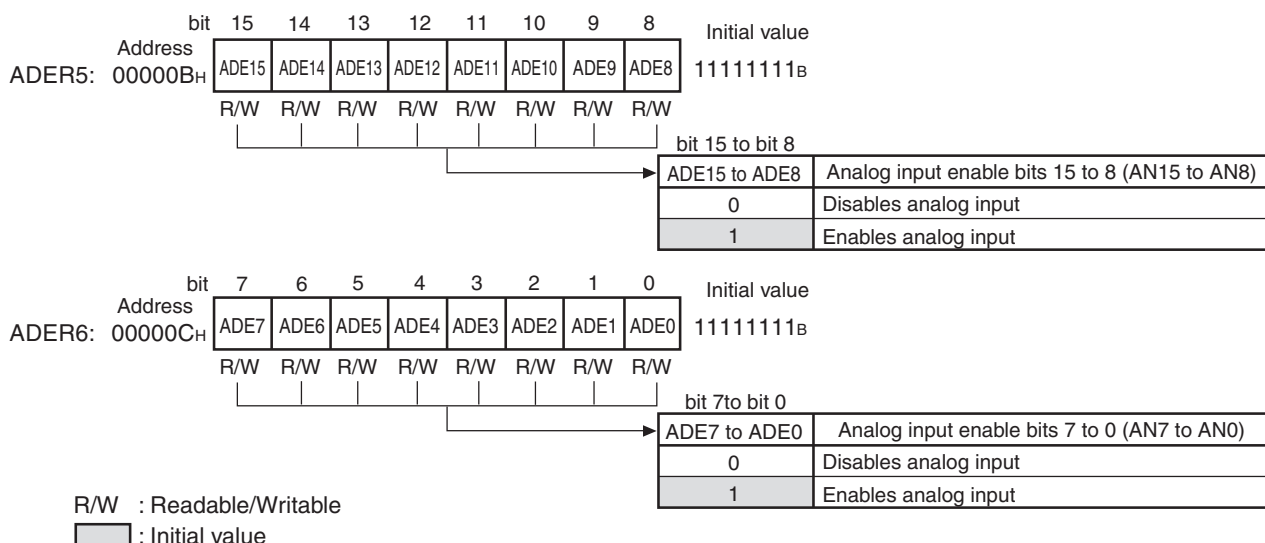


Table 18-10. Function of Port 5 Analog Input Enable Register (ADER5)

Bit name		Function
bit 15 to bit 8	ADE15 to ADE8: Analog input enable bits 15 to 8	These bits enable or disable analog input from A/D conversion analog input pins AN15 to AN8, located on port 5. When set to "0": Analog input disabled When set to "1": Analog input enabled

Table 18-11. Function of Port 6 Analog Input Enable Register (ADER6)

Bit name		Function
bit 7 to bit 0	ADE7 to ADE0: Analog input enable bits 7 to 0	These bits enable or disable analog input from A/D conversion analog input pins AN7 to AN0, located on port 6. When set to "0": Analog input disabled When set to "1": Analog input enabled

Notes:

- To use any of the registers as an analog input pin, write "1" to the bits of the analog input enable registers (ADER5, ADER6) corresponding to the pin to be used in order to set it as analog input.
- Each analog input pin is also used as a general-purpose I/O port and input/output of a peripheral function. When ADERx is set to "1", the pin is forced to become an analog input pin regardless of the settings of the port direction registers (DDR5, DDR6) and I/O settings of peripheral functions, therefore it cannot be used otherwise.

18.3 Interrupts of 8-/10-bit A/D Converter

In the 8-/10-bit A/D converter, an interrupt request is generated when the conversion results are stored in the A/D data register (ADCR) upon the completion of A/D conversion. Extended intelligent I/O service (EI²OS) can be used.

■ Interrupts of A/D Converter

When the A/D conversion results are stored in the A/D data register (ADCR) upon the completion of A/D conversion of analog input voltage, the interrupt request flag bit of the A/D control status register (ADCS: INT) is set to "1". An interrupt request is generated when the output of interrupt requests is enabled (ADCS: INTE = 1) and the interrupt request flag bit is set (ADCS: INT = 1).

■ Interrupts, EI²OS of 8-/10-bit A/D Converter

Reference:

For information about the interrupt number, interrupt control register and interrupt vector address, see "3. Interrupts".

■ EI²OS of 8-/10-bit A/D Converter

In the 8-/10-bit A/D converter, EI²OS can be used to transfer the A/D conversion results from the A/D data register (ADCR) to memory. For information about how to use the EI²OS functions, see Section "18.4.4 Conversion Operation by EI²OS Function" as well as Section "18.4.5 A/D Conversion Data Protection Function".

18.4 Explanation of 8-/10-bit A/D Converter Operations

The 8-/10-bit A/D converter performs A/D conversion operation in the following conversion modes. Each mode is set by setting the A/D conversion mode select bits of the A/D control status register (ADCS: MD1 and MD0).

- Single conversion mode
- Continuous conversion mode
- Stop conversion mode

■ Single Conversion Mode (ADCS: MD1 and MD0 = 00_B or 01_B)

- When a start trigger is entered, A/D conversion is performed on the analog input from the start channel (ADSR: ANS3 to ANS0) to the end channel (ADSR: ANE3 to ANE0) in succession.
- Once the A/D conversion is completed for the end channel, the A/D conversion operation is stopped.
- When setting to single conversion mode 1 (ADCS: MD1, MD0 = 00_B), reactivation during A/D conversion is allowed. However, reactivation is ignored when A/D conversion data protection function is triggered and A/D is in the pause state.
- When setting to single conversion mode 2 (ADCS: MD1, MD0 = 01_B), reactivation during A/D conversion is not allowed.

Notes:

- In single conversion mode 1 (ADCS: MD1 and MD0 = 00_B), the 8-/10-bit A/D converter may be restarted when a start trigger is entered while A/D conversion is being performed or paused*. Therefore, do not enter the start trigger while A/D conversion is being performed or paused.
In single conversion mode 2 (ADCS: MD1 and MD0 = 01_B), the 8-/10-bit A/D converter is not restarted even when a start trigger is entered while A/D conversion is being performed or paused*.
- When restarting the converter in single conversion mode 1 or 2, follow the procedure shown Section "18.4.1 Single Conversion Mode".

*:"Paused" represents the state where the A/D conversion protection function is temporarily stopping the conversion operation. For details, see Section "18.4.5 A/D Conversion Data Protection Function".

■ Continuous Conversion Mode (ADCS: MD1 and MD0 = 10_B)

- When a start trigger is entered, A/D conversion is performed on the analog input from the start channel (ADSR: ANS3 to ANS0) to the end channel (ADSR: ANE3 to ANE0) in succession.

- Once the A/D conversion is completed for the end channel, the conversion sequence returns to the analog input of the start channel to continue the A/D conversion.

■ Stop Conversion Mode (ADCS: MD1 and MD0 = 11_B)

- When a start trigger is entered, A/D conversion starts at the start channel (ADSR: ANS3 to ANS0). Every time the A/D conversion is completed for each channel, the A/D conversion operation is stopped. This state is called "pause state". When the start trigger is reentered during the pause state of the A/D conversion operation, A/D conversion is performed for the next channel.
- When the A/D conversion is completed for the end channel, the A/D conversion operation is stopped. When the start trigger is reentered during the pause state of the A/D conversion operation, the conversion sequence returns to the analog input of the start channel to continue the A/D conversion.

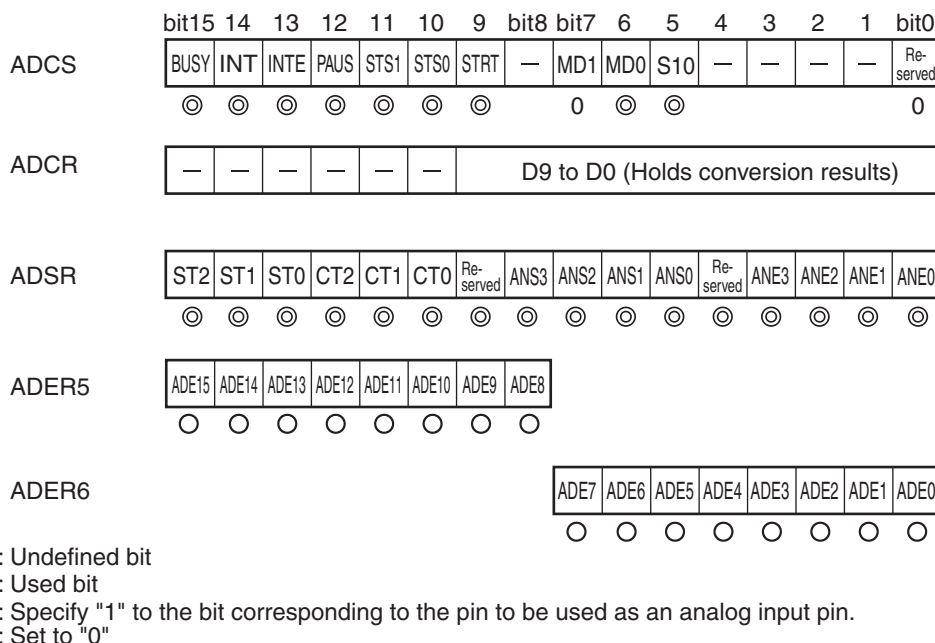
18.4.1 Single Conversion Mode

In single conversion mode, A/D conversion is performed from the start channel to the end channel in succession. Once the A/D conversion is completed for the end channel, the A/D conversion operation is stopped.

■ Single Conversion Mode Setup

To operate the 8-/10-bit A/D converter in the single conversion mode, the settings described in [Figure 18-8](#) must be selected.

Figure 18-8. Single Conversion Mode Setup



■ Operations and Applications of Single Conversion Mode

- When a start trigger is entered, A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0), and continues through the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).
- When the A/D conversion is completed for the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), the A/D conversion operation is stopped.
- To terminate the A/D conversion operation forcibly, "0" should be written to the A/D conversion operation flag bit of the A/D control status register (ADCS: BUSY).

[When the start channel and end channel are the same]

When the channel number of the start channel is set to the same number of the end channel (ADSR: ANS3 to ANS0 = ADSR: ANE3 to ANE0), A/D conversion is terminated after performing conversion for only one channel once, which is specified as the start channel (= end channel).

[Conversion sequence in single conversion mode]

Table 18-12. shows examples of the conversion sequence in single conversion mode.

Table 18-12. Conversion Sequence in Single Conversion Mode

Start channel	End channel	Conversion sequence in single conversion mode
Pin AN0 (ADSR: ANS=0000 _B)	Pin AN3 (ADSR: ANE=0011 _B)	AN0 → AN1 → AN2 → AN3 → Terminated
Pin AN3 (ADSR: ANS=0011 _B)	Pin AN3 (ADSR: ANE=0011 _B)	AN3 → Terminated

[Restart]

To restart A/D conversion while the A/D conversion is still being performed or paused, forcibly terminate that conversion session beforehand. Follow the procedure below.

- 1) Clear the A/D conversion operation flag bit (ADCS: BUSY).
- 2) Clear the interrupt request flag bit (ADCS: INT).
- 3) Set the A/D conversion software start bit (ADCS: STRT).

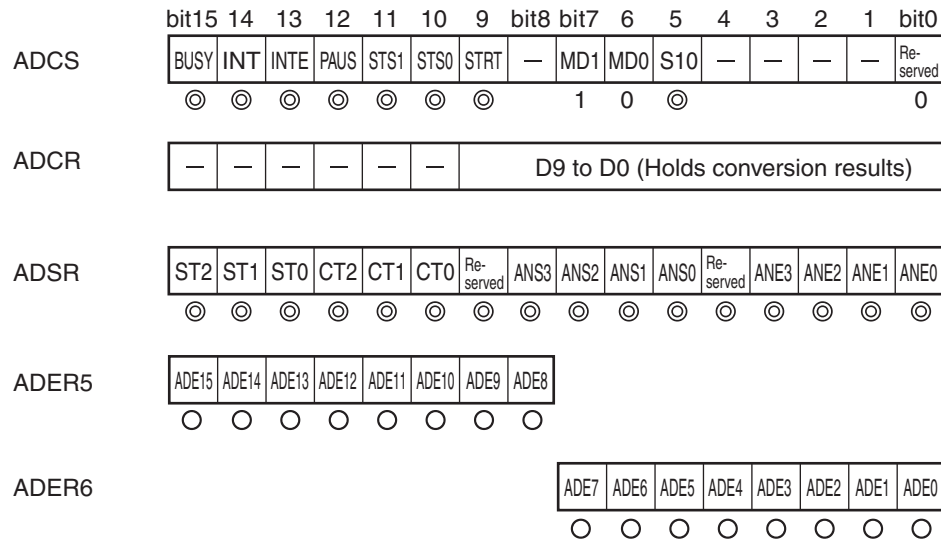
18.4.2 Continuous Conversion Mode

In the continuous conversion mode, A/D conversion is performed from the start channel to the end channel in succession. When the A/D conversion is completed for the end channel, the conversion sequence returns to the start channel to continue the A/D conversion operation.

■ Continuous Conversion Mode Setup

To operate the 8-/10-bit A/D converter in the continuous conversion mode, the settings described in [Figure 18-9](#). must be selected.

Figure 18-9. Continuous Conversion Mode Settings



- : Undefined bit
 ⊙ : Used bit
 ○ : Specify "1" to the bit corresponding to the pin to be used as an analog input pin.
 1 : Set to "1".
 0 : Set to "0".

■ Operations and Applications of Continuous Conversion Mode

- When a start trigger is entered, A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) and continues through the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).
- When the A/D conversion is completed for the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), the conversion sequence returns to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) to continue the A/D conversion.
- To terminate the A/D conversion operation forcibly, "0" should be written to the A/D conversion operation flag bit of the A/D control status register (ADCS: BUSY).

[When the start channel and the end channel are the same]

When the start channel is set to the same channel as the end channel (ADSR: ANS3 to ANS0 = ADSR: ANE3 to ANE0), A/D conversion is performed at the one channel set as the start channel (= end channel) repeatedly.

[Conversion sequence in continuous conversion mode]

Table 18-13. shows an example of the conversion sequences for the continuous conversion mode.

Table 18-13. Conversion Sequence in Continuous Conversion Mode

Start channel	End channel	Conversion sequence in continuous conversion mode
Pin AN0 (ADSR: ANS=0000 _B)	Pin AN3 (ADSR: ANE=0011 _B)	AN0 → AN1 → AN2 → AN3 → AN0 → Repeat
Pin AN3 (ADSR: ANS=0011 _B)	Pin AN3 (ADSR: ANE=0011 _B)	AN3 → AN3 → Repeat

[Restart]

To restart A/D conversion while the A/D conversion is still being performed or paused, forcibly terminate that conversion session beforehand, and then follow the procedure below.

- 1) Clear the A/D conversion operation flag bit (ADCS: BUSY).
- 2) Clear the interrupt request flag bit (ADCS: INT).
- 3) Set the A/D conversion software start bit (ADCS: STRT).

18.4.3 Stop Conversion Mode

In the stop conversion mode, A/D conversion is performed by repeatedly stopping and starting at each channel. The A/D conversion sequence returns to the start channel to continue the A/D conversion when a start trigger is entered after the A/D conversion is completed for the end channel and the A/D conversion operation is stopped.

■ Stop Conversion Mode Setup

To operate the 8-/10-bit A/D converter in the stop conversion mode, the settings described in [Figure 18-10](#) must be selected.

Figure 18-10. Stop Conversion Mode Settings

	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
ADCS	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	—	MD1	MD0	S10	—	—	—	—	Re-served
	⊙	⊙	⊙	⊙	⊙	⊙	⊙		1	1	⊙					0

ADCR	—	—	—	—	—	D9 to D0 (Holds conversion results)										
------	---	---	---	---	---	-------------------------------------	--	--	--	--	--	--	--	--	--	--

ADSR	ST2	ST1	ST0	CT2	CT1	CT0	Re-served	ANS3	ANS2	ANS1	ANS0	Re-served	ANE3	ANE2	ANE1	ANE0
	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙

ADER5	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8								
	○	○	○	○	○	○	○	○								

ADER6	<table> <tr> <td>ADE7</td> <td>ADE6</td> <td>ADE5</td> <td>ADE4</td> <td>ADE3</td> <td>ADE2</td> <td>ADE1</td> <td>ADE0</td> </tr> <tr> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> </tr> </table>																ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	○	○	○	○	○	○	○	○
ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0																									
○	○	○	○	○	○	○	○																									

— : Undefined bit

⊙ : Used bit

○ : Specify "1" to the bit corresponding to the pin to be used as an analog input pin.

1 : Set to "1".

0 : Set to "0".

■ Operations and Applications of Stop Conversion Mode

- When a start trigger is entered, A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0). Every time the A/D conversion is completed for each channel, the A/D conversion operation is stopped. When the start trigger is reentered while the A/D conversion operation is stopped, A/D conversion is performed for the next channel.
- Once the A/D conversion is completed for the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), the A/D conversion operation is stopped. When the start trigger is reentered while the A/D conversion operation is stopped, the conversion sequence returns to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) to continue the A/D conversion.

- ❑ To terminate the A/D conversion operation forcibly, "0" should be written to the A/D conversion operation flag bit of the A/D control status register (ADCS: BUSY).

[When the start channel and end channel are the same]

When the start channel is set to the same channel as the end channel (ADSR: ANS3 to ANS0 = ADSR: ANE3 to ANE0), A/D conversion is performed and then stopped repeatedly at the only one channel specified as the start channel (= end channel).

[Conversion sequence in stop conversion mode]

Table 18-14. shows an example on the conversion sequences for the stop conversion mode.

Table 18-14. Conversion Sequence in Stop Conversion Mode

Start channel	End channel	Conversion sequence in single conversion mode
Pin AN0 (ADSR: ANS=0000 _B)	Pin AN3 (ADSR: ANE=0011 _B)	AN0 → stop/start → AN1 → stop/start → AN2 → stop/start → AN3 → stop/start → AN0 → repeat
Pin AN3 (ADSR: ANS=0011 _B)	Pin AN3 (ADSR: ANE=0011 _B)	AN3 → stop/start → AN3 → stop/start → repeat

[Restart]

To restart A/D conversion while the A/D conversion is still being performed or paused, forcibly terminate that conversion session beforehand and then follow the procedure below.

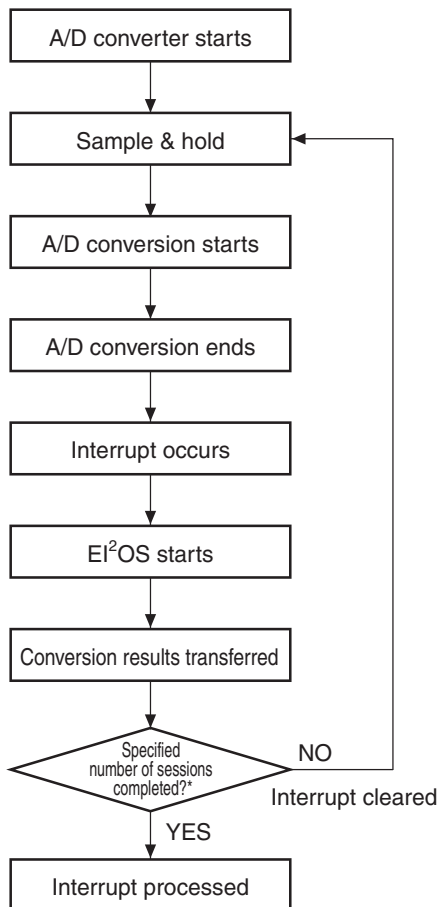
- 1) Clear the A/D conversion operation flag bit (ADCS: BUSY).
- 2) Clear the interrupt request flag bit (ADCS: INT).
- 3) Set the A/D conversion software start bit (ADCS: STRT).

18.4.4 Conversion Operation by EI²OS Function

In the 8-/10-bit A/D converter, the EI²OS function can be used to transfer the A/D conversion results to memory.

■ Conversion Operation by EI²OS Function

Figure 18-11. shows the flow of the conversion operation when the EI²OS function is used.

Figure 18-11. Conversion Operation Flow when EI²OS Function is Used

*: Determined by EI²OS settings

18.4.5 A/D Conversion Data Protection Function

The data protection function is activated when A/D conversion is performed while the output of interrupt requests is enabled.

■ Explanation of A/D Conversion Data Protection Function of 8-/10-bit A/D Converter

The A/D conversion data protection function prevents A/D conversion data from being unretrieved.

The 8-/10-bit A/D converter has one A/D data register (ADCR1/ADCR0) to store conversion data and one buffer register to store the next conversion result. Once A/D conversion is completed, the conversion data is first stored in the buffer register, and then stored into the A/D data register.

The operation of the A/D converter varies depending on whether the A/D conversion data protection function is switched ON or OFF, as described below.

- When the interrupt request enable bit (ADCS:INTE) is set to "0", the data protection function is turned off. In this case, if conversion is performed more than once continuously, the conversion result is stored in the A/D data register after each conversion (This allows the latest conversion data to be always stored.).
- When the interrupt request enable bit (ADCS:INTE) is set to "1", the data protection function is turned on. When conversion is performed more than once continuously in this state and the next conversion is completed before the result of the previous conversion is read from the A/D data register (interrupt request flag bit:ADCS:INT=1; State A), the conversion result is stored in the buffer register and the A/D conversion pauses (State B).

If the interrupt request flag bit (ADCS:INT) is cleared to "0" during State B, the data stored in the buffer register is transferred to the A/D data register (See Figure 18-12..).

Figure 18-12. Operation of A/D Conversion Data Protection Function 1(Normal Operation)

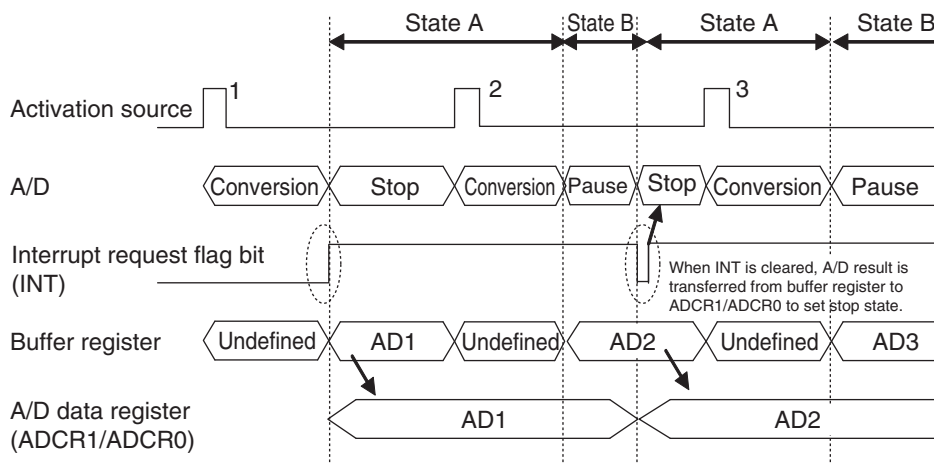
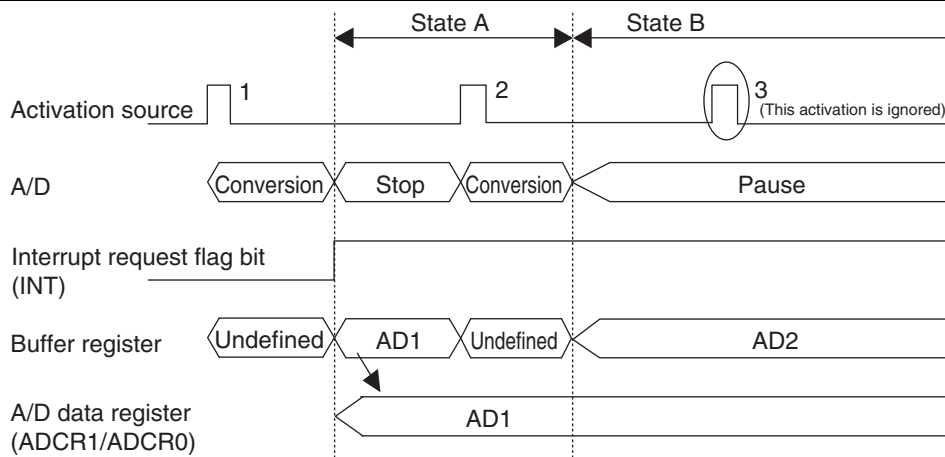


Figure 18-13. Operation of A/D Conversion Data Protection Function 2



■ How to Use A/D Conversion Data Protection

This item explains how to use the A/D conversion data protection function for when EI²OS is used as well as when it is not used.

- How to use the data protection function when EI²OS is not used
- Set the interrupt request flag bit in the A/D control status register (ADCS:INTE) to "1" to start A/D conversion.
- After completion of the A/D conversion, the result of the A/D conversion is stored in the buffer register and the succeeding content of the buffer register is transferred to the A/D data register (ADCR1/ ADCR0). At this point, the interrupt request flag bit in the A/D control status register (ADCS:INT) is set to "1".
- If the interrupt request flag bit (ADCS:INT), which was set upon the completion of the previous A/D conversion, is still set to "1" upon the completion of the next A/D conversion, the A/D conversion operation pauses in order to prevent overwriting, immediately before the content of the buffer register is transferred to the A/D data register. At this point, the pause flag bit in the A/D control status register (ADCS:PAUS) is set to "1".
- Since interrupt request in the A/D control status register is enabled (ADCS: INTE=1), an interrupt request is generated when the interrupt request flag bit is set. When the interrupt request flag bit is cleared after the A/D data register is

read in the interrupt routine, the pause state is canceled and the content of the buffer register is transferred to the A/D data register. In case of continuous A/D conversion, the A/D conversion operation resumes. At this point, however, the pause flag bit (ADCS:PAUS) is not cleared automatically. To clear the bit, write "0" to it.

Notes:

- The A/D conversion data protection function does not operate unless interrupt requests are enabled. Make sure that the interrupt request output enable bit in the A/D control status register (ADCS:INTE) is always set to "1". If interrupt requests are disabled during a pause (ADCS:INTE=0), A/D conversion starts and data in the A/D data register may be rewritten.
 - To perform continuous conversion, always program it so that the data stored in the A/D data register is read before the interrupt request flag bit (ADCS:INT) is cleared. If the interrupt request flag bit (ADCS:INT) is cleared before the data stored in the A/D data register is read while A/D conversion is paused, the initially stored conversion data is deleted and the next data is read.
- How to use the data protection function when EI²OS is used
 - Enable the use of the EI²OS function and set the interrupt request enable bit in the A/D control status register (ADCS:INT) to "1" to start A/D conversion.
 - After completion of the A/D conversion, the result of the A/D conversion is stored in the buffer register, and then the content of the buffer register is transferred to the A/D data register (ADCR). At this point, the interrupt request flag bit in the A/D control register (ADCS:INT) is set to "1" and EI²OS starts. If the next A/D conversion is completed before the result of the A/D conversion is transferred from the A/D data register to the memory, the A/D conversion operation pauses in order to prevent overwriting, immediately before the content of the buffer register is transferred to the A/D data register. At this point, the pause flag bit in the A/D control status register (ADCS:PAUS) is set to "1".
 - Once the EI²OS transfer is completed, the interrupt request flag bit (ADCS:INT) is cleared to "0", the pause state is canceled, and the content of the buffer register is transferred to the A/D data register. In case of continuous A/D conversion, the A/D conversion operation resumes. At this point, however, the pause flag bit (ADCS:PAUS) is not cleared automatically. To clear the bit, write "0" to it.

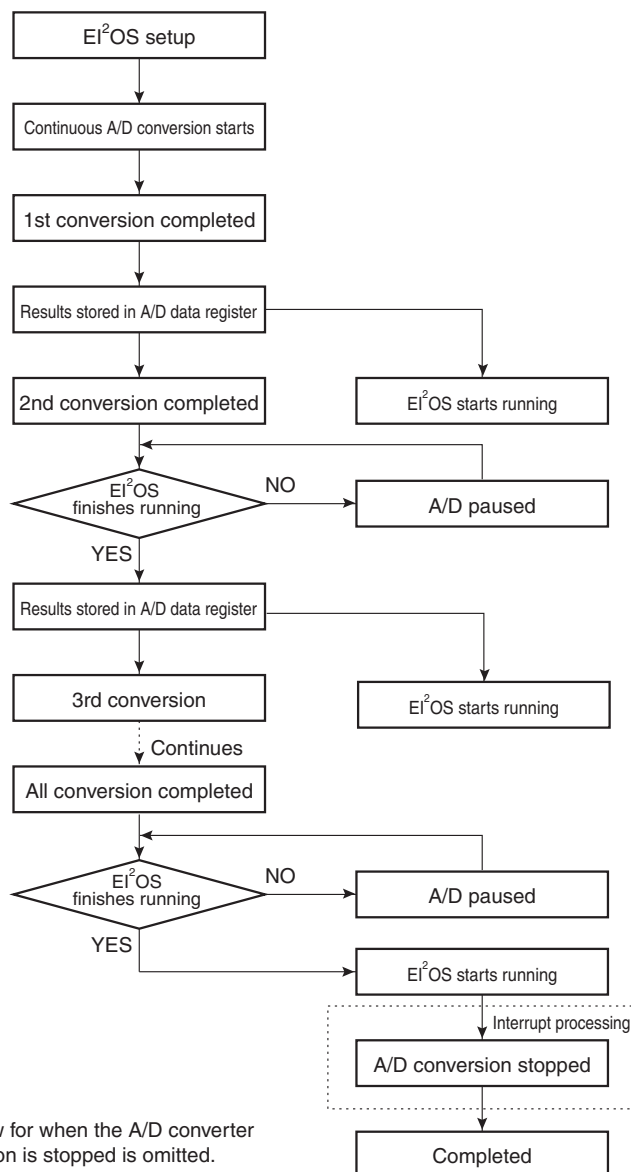
Notes:

- The A/D conversion data protection function does not operate unless interrupt requests are enabled. Make sure that the interrupt request output enable bit in the A/D control status register (ADCS:INTE) is always set to "1".
- If the result of A/D conversion is transferred to the memory by the EI²OS function, do not clear the interrupt request flag bit (ADCS:INT=0), or this may rewrite the data in the A/D data register currently being transferred.
- If the result of A/D conversion is transferred to the memory by the EI²OS function, do not disable interrupt requests. If interrupt requests are disabled (ADCS:INTE=0) during a pause, A/D conversion starts and the data in the A/D data register may be rewritten.

- Processing flow of A/D conversion data protection function when EI²OS is used

Figure 18-14. shows the processing flow of the A/D conversion data protection function when EI²OS is used.

Figure 18-14. Processing Flow of A/D Conversion Data Protection Function When EI²OS is Used



18.5 Notes on Using 8-/10-bit A/D Converter

Notes must be taken for the following points when using the 8-/10-bit A/D converter.

■ Notes on Using 8-/10-bit A/D Converter

- Analog input pins

The analog input pins are also used as general-purpose I/O ports for port 5, port 6. When using the pin as analog input pins, set up the analog input enable registers (ADER5, ADER6).

When using the pin as the analog input pins, write "1" to the bit of the analog input enable register (ADER5, ADER6) which corresponds to the pin to be used in order to enable the analog input.

If a medium-level signal is input while the pin remains set as a general-purpose I/O port, input leakage current is supplied to the gate. When using it as an analog input pin, always enable the analog input before use.

Power application sequence of the 8-/10-bit A/D converter and analog input

Always apply power to the 8-/10-bit A/D converter and analog inputs (Pins AN0 to AN15) after the digital power supply (V_{CC}).

At power-off, turn off the digital power supply after turning off the 8-/10-bit A/D converter power supply and the analog inputs.

Do not allow AVR to exceed AV_{CC} when turning on and off the power.

□ Power supply voltage of the 8-/10-bit A/D converter

Care must be taken to ensure that the power supply of the 8-/10-bit A/D converter (AV_{CC}) does not exceed the digital power supply (V_{CC}) voltage to prevent latch-up.

19. Low Voltage Detection/CPU Operating

Detection Reset



This chapter explains the function and operating the low voltage detection/CPU operating detection reset. This function can use only the product with "J" suffix of MB90990 series.

Evaluation products correspond to the cpu operation detection.

19.1 Overview of Low Voltage/CPU Operating Detection Reset Circuit

19.2 Configuration of Low Voltage/CPU Operating Detection Reset Circuit

19.3 Low Voltage/CPU Operating Detection Reset Circuit Register

19.4 Operating of Low Voltage/CPU Operating Detection Reset Circuit

19.5 Notes on Using Low Voltage/CPU Operating Detection Reset Circuit

19.6 Sample Program for Low Voltage/CPU Operating Detection Reset Circuit

19.1 Overview of Low Voltage/CPU Operating Detection Reset Circuit

The low voltage detection reset circuit monitors the power-supply voltage and has the function to detect the power-supply voltages falling lower than the detection voltage values. When the low voltage is detected, internal reset is generated.

The CPU operation detection reset circuit is a 20-bit counter that uses the oscillation clock as a count clock. If it is not cleared within a specific time after it has started, an internal reset is generated.

■ Low Voltage Detection Reset Circuit

Figure 19-1. Detection Voltage of Low Voltage/CPU Operating Detection Reset Circuit

Detection voltage	
4.2 V \pm 0.2 V	
4.1 V \pm 0.2 V	
4.0 V \pm 0.2 V	Initial value
3.9 V \pm 0.2 V	
3.8 V \pm 0.2 V	
3.7 V \pm 0.2 V	
3.6 V \pm 0.2 V	
3.2 V \pm 0.2 V	
3.0 V \pm 0.2 V	
2.8 V \pm 0.2 V	

After detecting the low voltage, low voltage detection flag (LVRC:LVRF) is set to "1", and internal reset is outputted.

After the low voltage is detected, internal reset is generated and the STOP mode is canceled, because the operation is continued even in the STOP mode.

After writing ends, low voltage reset is generated for internal RAM writing period.

The detection voltage is set by the low-voltage/CPU operation detection reset setting register (LVRS).

■ CPU Operating Detection Reset Circuit

CPU operating detection reset circuit is a counter for preventing the program out of control. After power-on reset, it starts automatically. After it starts, it is necessary to keep clearing regularly within the fixed time. Internal reset is generated when not cleared during the fixed time by an program infinite loop, etc. The width of internal reset generated by CPU operating detection circuit is five machine cycles.

Figure 19-2. Interval Time of CPU Operating Detection Reset Circuit

Interval time	
$2^{21}/F_c$ (approx.524 ms)*	
$2^{20}/F_c$ (approx.262 ms)*	
$2^{19}/F_c$ (approx.131 ms)*	
$2^{18}/F_c$ (approx.65.5 ms)*	
$2^{17}/F_c$ (approx.32.8 ms)*	Initial value

*: It is interval time at oscillation clock 4 MHz.

The interval time is set by the low-voltage/CPU operation detection reset setting register (LVRS).

In the mode that CPU stops operating, the circuit stops.

The counter clear condition of CPU operating detection reset circuit is indicated as follows.

- ☐ Writing "0" and "1" to CL bit of LVRC register in this order
- ☐ Internal reset
- ☐ Oscillation clock stop (Include sub clock mode)
- ☐ Transition to sleep mode
- ☐ Transition to time-base timer mode

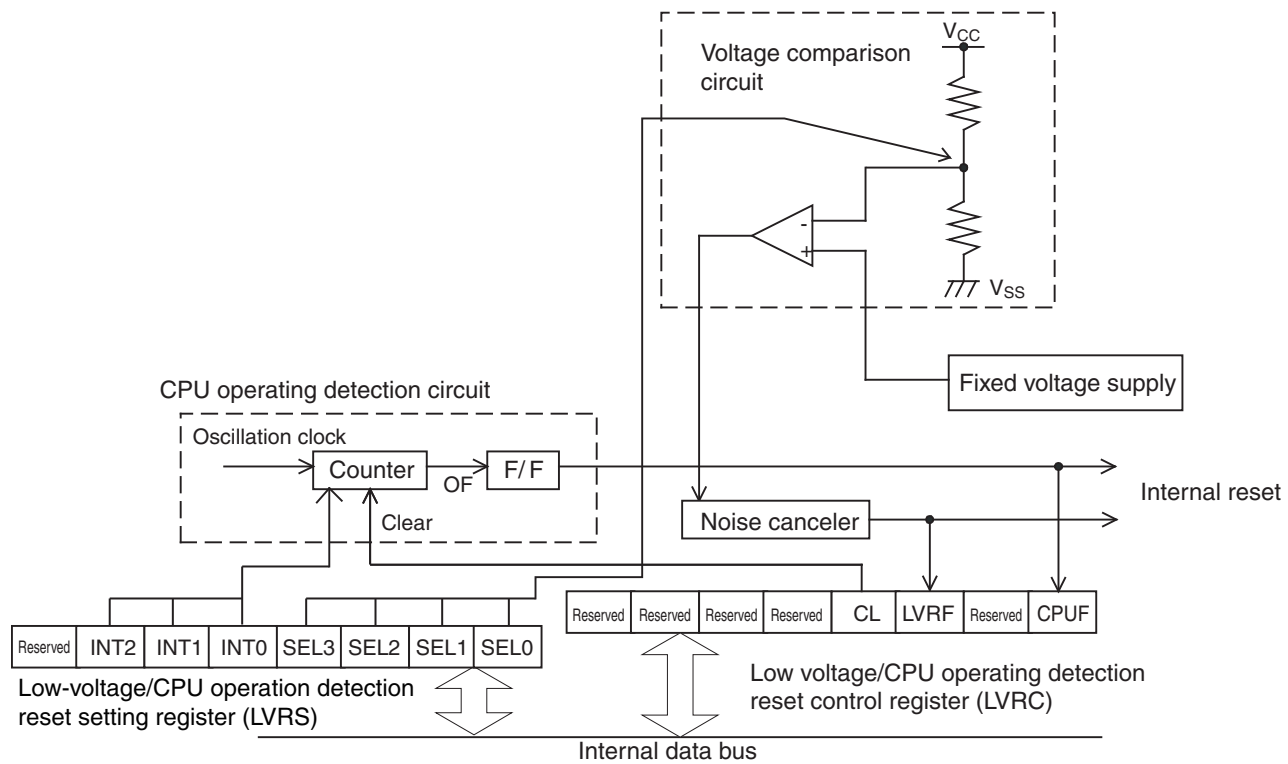
19.2 Configuration of Low Voltage/CPU Operating Detection Reset Circuit

Low voltage/CPU operating detection reset circuit has following four blocks.

- CPU operating detection circuit
- Voltage comparison circuit
- Low voltage/CPU operating detection reset control register (LVRC)
- Low-voltage/CPU operation detection reset setting register (LVRS)

■ Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit

Figure 19-3. Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit



□ CPU operating detection circuit

It is a counter for preventing the program out of control. After it starts, it is necessary to keep clearing regularly within the fixed time.

□ Voltage comparison circuit

When the detection voltage is compared with the power-supply voltage, the output is set to "H" after the low voltage detection. After the power supply is turned on, it always operates.

□ Low voltage/CPU operating detection reset control register (LVRC)

This register clears the low voltage/CPU operating detection reset flag and the counter of CPU operating detection function.

□ Low-voltage/CPU operation detection reset setting register (LVRS)

Sets the detection voltage and the interval time for low-voltage/CPU operation detection reset.

□ Reset source of low voltage/CPU operating detection reset circuit

After power-supply voltages have fallen lower than the detection voltages, internal reset is generated.

When the counter of CPU operating detection circuit is not cleared during the fixed time, internal reset is generated.

19.3 Low Voltage/CPU Operating Detection Reset Circuit Register

The low-voltage/CPU operating detection reset control register (LVRC) clears the low voltage/CPU operating detection reset flag and the counter of CPU operating detection circuit.

The low-voltage/CPU operation detection reset setting register (LVRS) sets the detection voltage for low-voltage detection and the interval time for CPU operation detection.

■ Low Voltage/CPU Operating Detection Reset Control Register (LVRC)

Figure 19-4. Low Voltage/CPU Operating Detection Reset Control Register (LVRC)

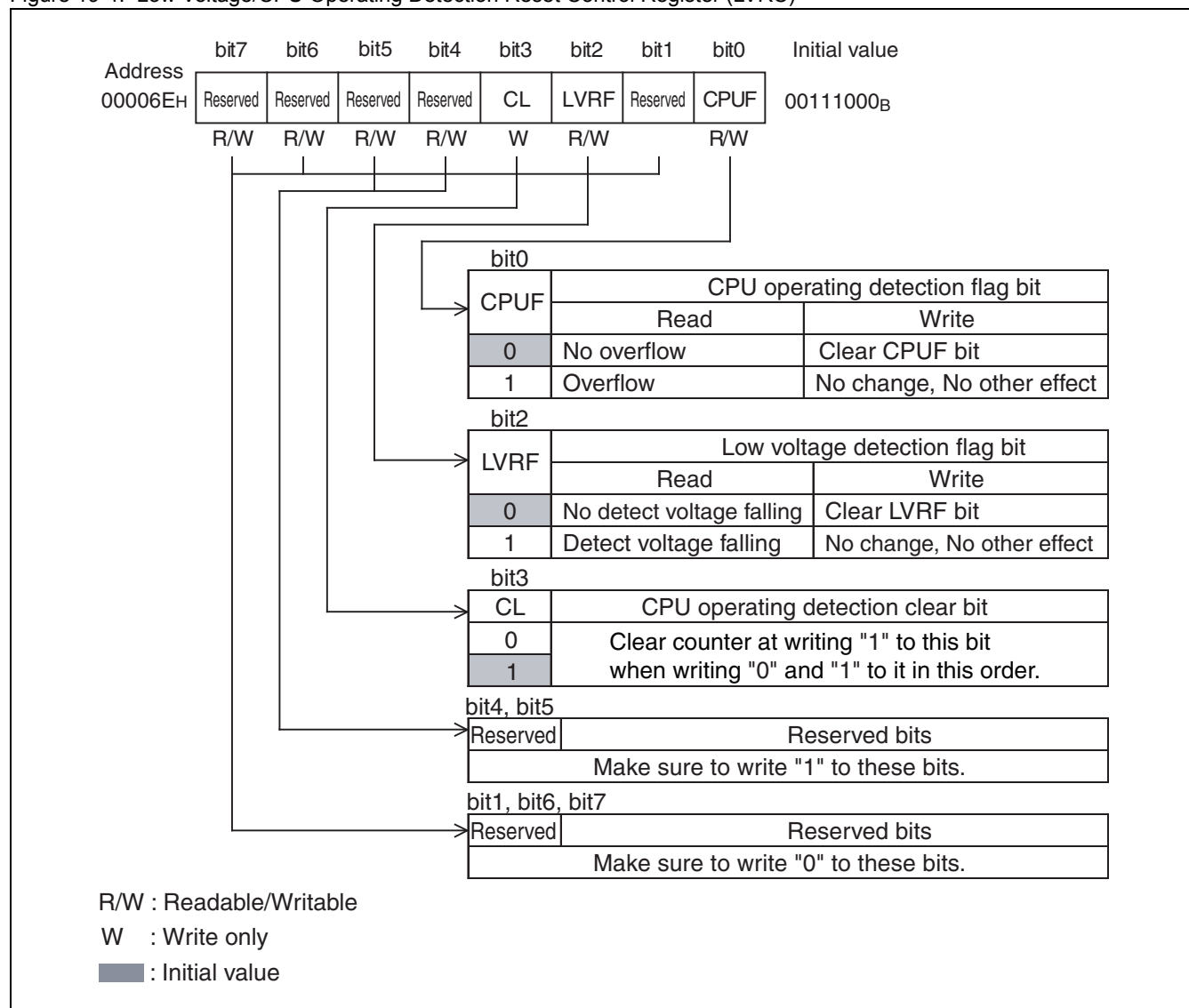


Table 19-1. Functional Description of Low Voltage/CPU Operating Detection Reset Control Register

Bit name		Function
bit7, bit6	Reserved: Reserved bits	Note: Make sure to write "0" to these bits.
bit5, bit4	Reserved: Reserved bits	Note: Make sure to write "1" to these bits.
bit3	CL: CPU operating detection clear bit	Used to clear the counter of the CPU operation detection circuit. The counter of the CPU operation detection circuit is cleared when "0" and "1" is written to the CL bit in this order.
bit2	LVRF: Low voltage detection flag bit	When falling of the power-supply voltage is detected, the LVRF bit is set to "1". This bit is cleared by "0" at write. And even if "1" is written in this bit, the LVRF bit is no effect. This bit is not initialized by internal reset, and it is initialized only by the external reset input.
bit1	Reserved: Reserved bit	Note: Make sure to write "0" to this bit.
bit0	CPUF: CPU operation detection flag bit	When the counter of CPU operating detecting function overflows, the CUPF bit is set to "1". This bit is cleared by "0" at write. And even if "1" is written in this bit, the CUPF bit has no effect. This bit is not initialized by internal reset, and it is initialized only by the external reset input.

Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)

Figure 19-5. Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)

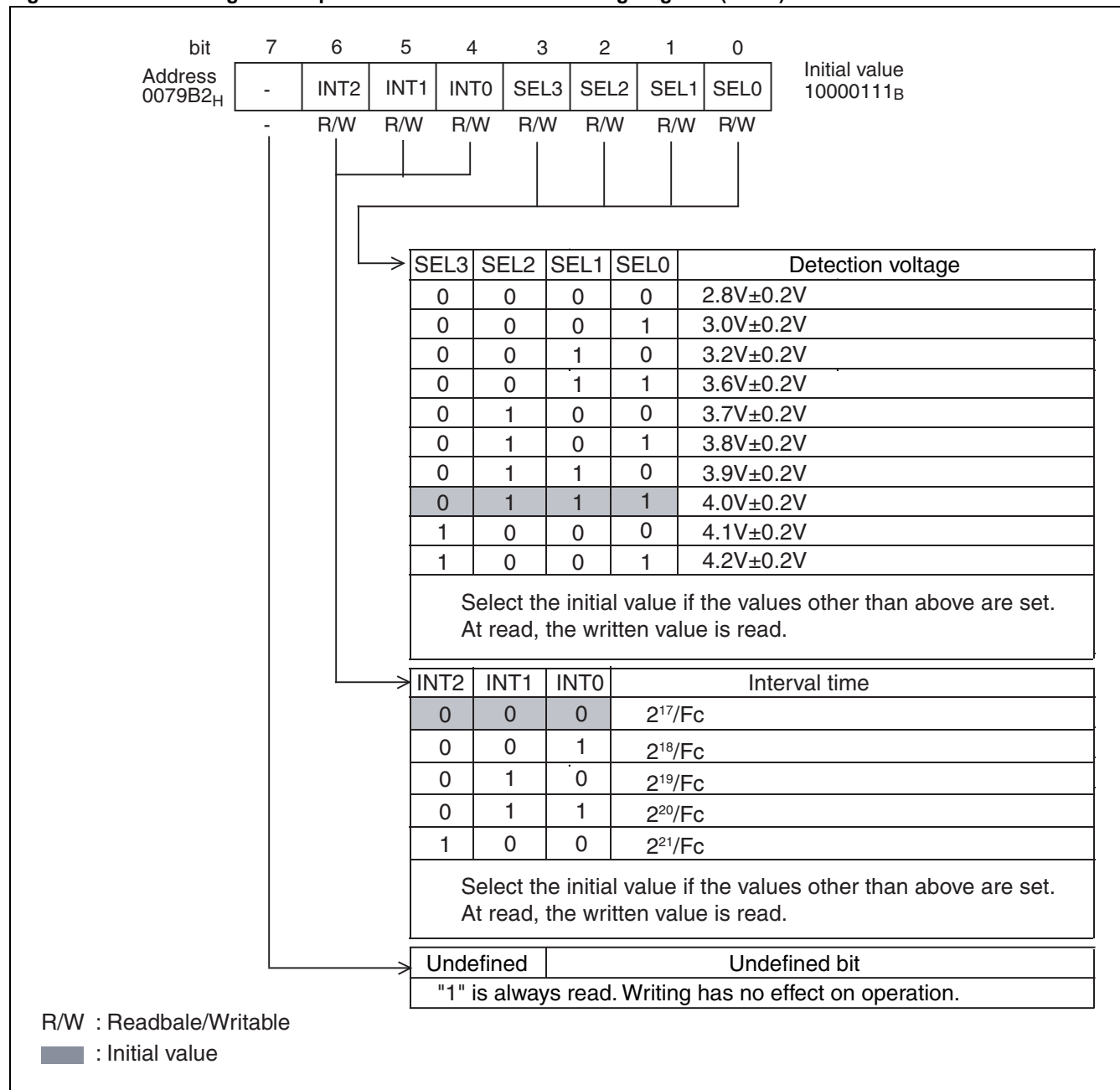
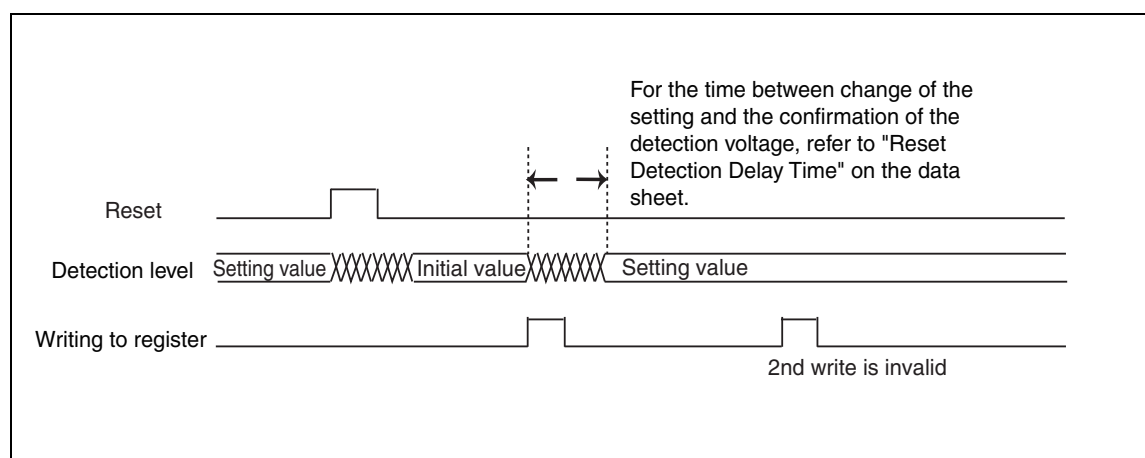


Table 19-2. Functional Description of Low-voltage/CPU Operation Detection Reset Setting Register

Bit name		Function
bit7	Undefined bit	At read : Always read "1". At write: No effect.
bit6 to bit4	INT2 to INT0: CPU operation detection interval time setting bits	Set an interval time for the CPU operation detection circuit.
bit3 to bit0	SEL3 to SEL0: Low-voltage detection voltage setting bits	Set a detection voltage for the low-voltage detection function.



The low-voltage/CPU operation detection reset setting register (LVRS) can be written only once, after the reset is canceled. The second and any succeeding write attempts are invalid. The reset sources for the register and write count are a power-on reset, external reset, low-voltage detection reset and CPU operation detection reset only. A clock supervisor reset, watchdog reset and software reset do not initialize the register or write count.

Write to the low-voltage/CPU operation detection reset setting register (LVRS) either within the initialization routine immediately after the cancellation of a reset, or immediately after clearing the CPU operation detection counter using the CL bit in the low-voltage/CPU operation detection reset control register (LVRC). If the setting is changed when the CPU operation detection counter is still counting, a CPU operation detection reset may be generated unintentionally.

19.4 Operating of Low Voltage/CPU Operating Detection Reset Circuit

The circuit monitors the power-supply voltage. When the power supply voltage is lower than the set value, internal reset is generated. In CPU operating detecting function, internal reset is generated without the counter clear at constant intervals. When internal reset is generated by the detection of the low voltage or the out of CPU control, the content of the register is not guaranteed. The program restarts from the address specified by the reset vector after the reset sequence is executed when the low voltage reset is canceled.

■ Operating of Low Voltage Detection Reset Circuit

The low voltage detection reset circuit starts the detection of the low voltage without taking the operating stability wait time after reset is canceled.

■ Operating of CPU Operating Detection Reset Circuit

The CPU operating detection reset circuit starts the detection of the CPU operation without taking the operating stability wait time after reset is canceled.

Note:

The current is consumed during the sleep or stop mode because the low voltage reset circuit always operates.

19.5 Notes on Using Low Voltage/CPU Operating Detection Reset Circuit

This section explains the note on using the low voltage/CPU operating detection reset circuit.

■ Notes on Using Low Voltage Detection Reset Circuit

- Disabled operating stop from program

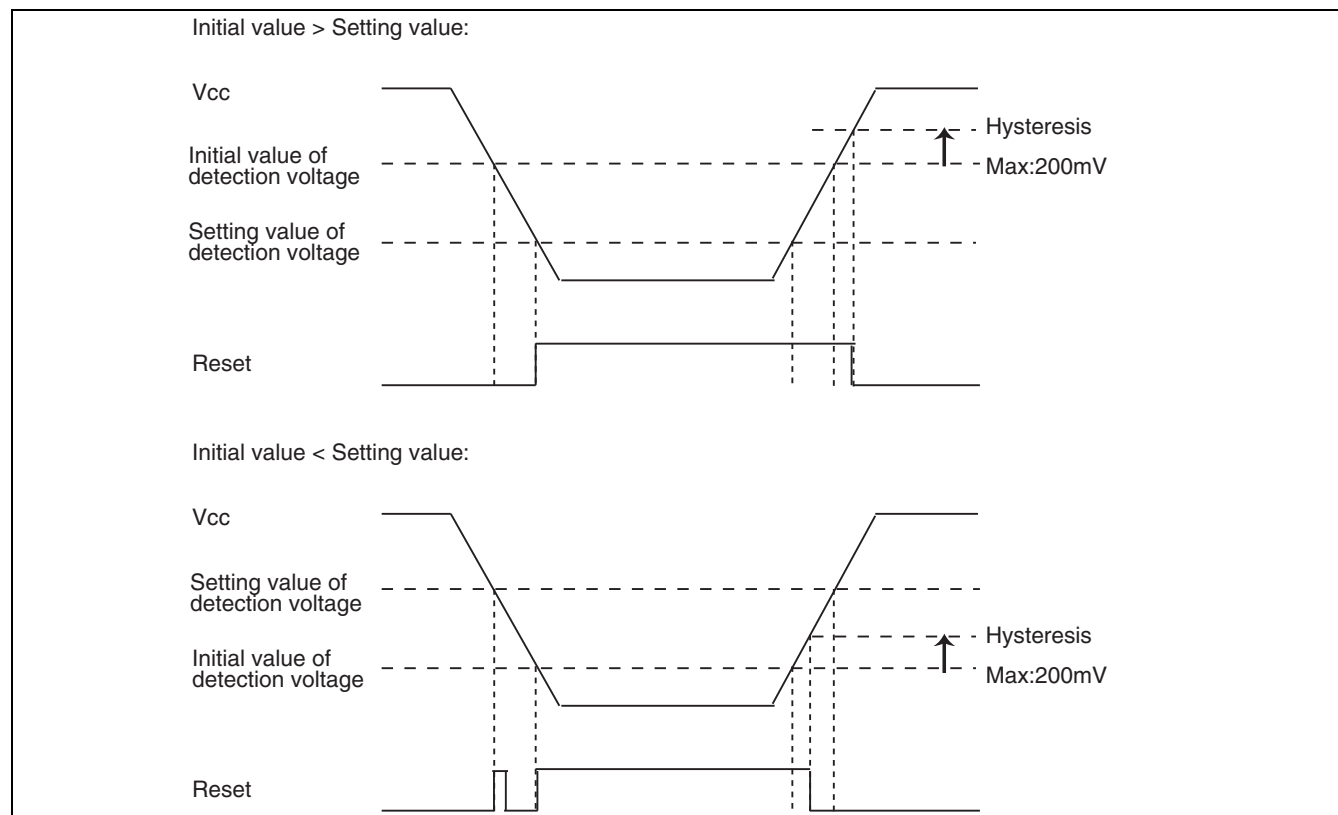
The low voltage detection reset circuit operates continuously after the power supply is turned on and the operating stabilization wait time passes. Operating cannot be stopped with software.

- Operating at STOP mode

Low voltage detection reset keeps operating at the STOP mode. Therefore, if a low voltage is detected in the STOP mode, reset is generated and the STOP mode is canceled.

- Relationship between the initial value and the setting value of the detection voltage

As the low-voltage/CPU operation detection reset setting register is initialized by a power-on reset, external reset and low-voltage detection reset, a low-voltage detection reset may be generated/canceled unintentionally, depending on the relationship between the initial value and the setting value of the detection voltage.



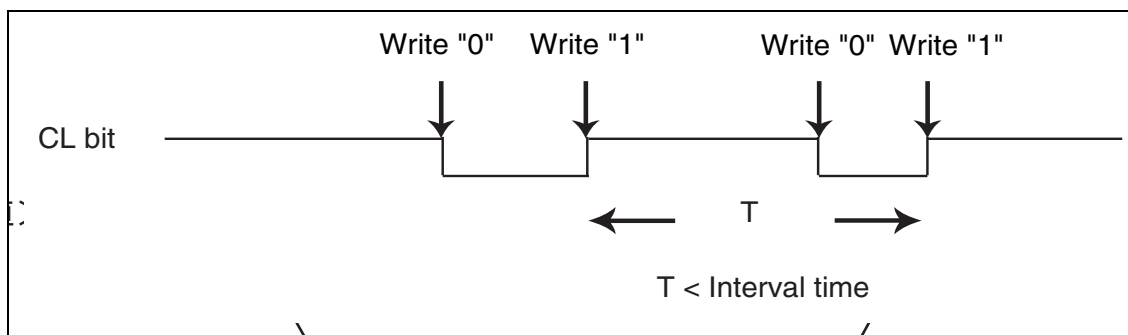
■ Notes on Using CPU Operating Detection Reset Circuit

- Disabled operating stop from program

CPU operating detection reset circuit operates continuously after turning on the power supply. Operating cannot be stopped with software.

- Resets by CPU operation detection function suppressed

The CPU operation detection function must clear the counter in constant intervals. The counter can be cleared to prevent a reset from being generated, by writing "0" then "1" to the CL bit in the LVRC register.



The counter for the CPU operation detection function is cleared when a rising edge of the CL bit is detected. Therefore, write "0" then "1" to the CL bit so that the interval between rising edges of the CL bit does not exceed the setting value of the interval time.

- Stop and clear of counter


In the mode CPU stops operating, CPU operating detecting function clears the counter and stops operating.

- Operation in sub-oscillation mode

The CPU operation detection function will stop operation in sub-oscillation mode. For this reason, also use the watchdog reset function.

- Processing in the initialization routine

It takes at least 32768 instruction cycles from the cancellation of a reset at program start to the generation of a reset by CPU operation detection. Therefore, clear the counter and set the interval time within the 32768 instruction cycles in the initialization routine.

Table 19-3.  Configuration of CPU Operation Detection Circuit in each Operation Mode

Operation mode	Condition	CPU Operation Detection	
		Counter clock	Reset
Main/PLL clock mode	The oscillation clock halt detection disabled (CSVCR:MM = 0)	Oscillation clock (HCLK)	Generated
	The oscillation clock halt detection enabled (CSVCR:MM = 1)	CR clock	Generated
CR subclock mode	-	Halt	Not generated

19.6 Sample Program for Low Voltage/CPU Operating Detection Reset Circuit

This section shows the sample program for low voltage/CPU operating detection reset circuit.

■ Sample Program for Low Voltage/CPU Operating Detection Reset Circuit

- Processing specification

The counter of CPU operating detecting function is cleared.

- Coding example

```
LVRC    EQU    006EH    ; Address of low voltage/CPU operating detection reset control register
```

```
-----Main program-----
```

```
        CSEG    ; [CODE SEGMENT]
```

```
        :
```

```
MOV     LVRC,#00110101B
```

```
        :
```

```
MOV     LVRC,#00111101B
```

```
        :
```

```
END
```


20. LIN-UART



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

20.1 Overview of LIN-UART

20.2 Configuration of LIN-UART

20.3 LIN-UART Pins

20.4 LIN-UART Registers

20.5 LIN-UART Interrupts

20.6 LIN-UART Baud Rates

20.7 Operation of LIN-UART

20.8 Notes on Using LIN-UART

20.1 Overview of LIN-UART

The LIN-UART with LIN (Local Interconnect Network) - Function is a general-purpose serial data communication interface for performing synchronous or asynchronous communication (start-stop synchronization) with external devices. LIN-UART provides bidirectional communication function (normal mode), master-slave communication function (multiprocessor mode in master/slave systems), and special features for LIN-bus systems (**LIN bus is available only for LIN-UART0,1**).

■ LIN-UART Functions

- LIN-UART functions

LIN-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](#).

Table 20-1. LIN-UART Functions (Sheet 1 of 2)

	Function
Data buffer	Full-duplicate double-buffer
Serial input	The LIN-UART performs oversampling 5 times using machine clock and determine the received value by majority decision of sampling time (only asynchronous mode).
Transfer mode	<ul style="list-style-type: none">• Clock synchronous (selecting start/stop synchronous or start/stop bit)• Clock asynchronous (start/stop bits can be used.)
Baud rate	<ul style="list-style-type: none">• Dedicated baud-rate generator (The baud rate is consisted of 15-bit reload counter.)• An external clock can be inputted and also be adjusted by reload counter.
Data length	<ul style="list-style-type: none">• 7 bits (other than synchronous or LIN mode)• 8 bits
Signal type	NRZ (Non Return to Zero)
Start bit timing	Synchronization to the falling edge of the start bit in the asynchronous mode
Detection of receive error	<ul style="list-style-type: none">• Framing error• Overrun error• Parity error (not supported for operation mode 1)

Table 20-1. LIN-UART Functions (Sheet 2 of 2)

	Function
Interrupt request	<ul style="list-style-type: none"> • Receive interrupt (receive termination, detection of receive error, LIN Synch break detection*) • Transmit interrupt (transmit data empty) • Interrupt request to ICU* (LIN Synch field detection: LSYN) • Both the transmission and reception support EI²OS * : LIN is available only for LIN-UART0,1
Master/slave type communication function (multiprocessor mode)	This function enables communication between "1" (only use master) and n (slave) (This function supports for the both of master and slave system.)
Synchronous mode	Master or slave function
Pin access	Capable of reading the state of serial I/O pin directly
LIN bus option (LIN-UART0,1 only)	<ul style="list-style-type: none"> • Master device operation • Slave device operation • LIN Synch break detection • LIN Synch break generation • Detection of start/stop edges in LIN Synch field connected to input capture 0 and 1 (UART2 does not connect to input capture internally. Therefore, detection of start/stop edges in LIN Synch field cannot be used.)
Synchronous serial clock	Synchronous serial clock can be continuously outputted to SCK pin for synchronous communication with start/stop bits.
Clock delay option	Special synchronous clock mode for delaying clock (useful to SPI)

■ LIN-UART Operation Modes

The LIN-UART operates in four different modes, which are determined by the MD0- and the MD1-bit of the LIN-UART 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 20-2. Setting of LIN-UART Serial Mode Register (SMR)

MD1	MD0	Mode	Type	LIN-UART0,1	UART2
0	0	0	Asynchronous (normal mode)	Provided	Provided
0	1	1	Asynchronous (multiprocessor mode)		
1	0	2	Synchronous (normal mode)		
1	1	3	Asynchronous (LIN mode)		Not Provided

Table 20-3. Operation Mode of LIN-UART

Operation Mode		Data Length		Synchronous/Asynchronous	Length of Stop Bit	Data Bit Format
		No Parity	With Parity			
0	Normal mode	7 or 8 bits		Asynchronous	1 bit or 2 bits	LSB first MSB first
1	Multiprocessor mode	7 or 8 bits-+1 *	-	Asynchronous		
2	Normal mode	8 bits		Synchronous	None, 1 bit, 2 bits	
3	LIN mode	8 bits	-	Asynchronous	1 bit	LSB first

- : Setting disabled

*: +1 is the address/data select bit (A/D) used for controlling communication in multiprocessor mode.

Notes:

- Mode 1 operation is supported both for master or slave operation of LIN-UART in a master-slave connection system.
- In Mode 3 the LIN-UART function is locked to the communication format 8N1-Format, LSB first.

- If the mode is changed, LIN-UART cuts off all possible transmission or reception and awaits then new action.

■ LIN-UART Interrupt and EI²OS

Table 20-4. LIN-UART Interrupt and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI ² OS
		Register name	Address	Lower	Upper	Bank	
LIN-UART0 reception	#35(23 _H)	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	*1
LIN-UART0 transmission	#36(24 _H)	ICR12	0000BC _H	FFFF6C _H	FFFF6D _H	FFFF6E _H	*2
LIN-UART1 reception	#37(25 _H)	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	*1
LIN-UART1 transmission	#38(26 _H)	ICR13	0000BD _H	FFFF64 _H	FFFF65 _H	FFFF66 _H	*2
UART2 reception	#39(27 _H)	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	*1
UART2 transmission	#40(28 _H)	ICR14	0000BE _H	FFFF5C _H	FFFF5D _H	FFFF5E _H	*2

*1: EI²OS service is usable if the other interrupt (s) which shares the ICR12, ICR13, ICR14 and same interrupt vector is (are) not enabled. Detection of receive errors is possible and stop function for EI²OS service is supported.

*2: EI²OS service is usable if the other interrupt (s) which shares the ICR12, ICR13, ICR14 and same interrupt vector is (are) not enabled.

20.2 Configuration of LIN-UART

This section provides a short overview on the building blocks of LIN-UART.

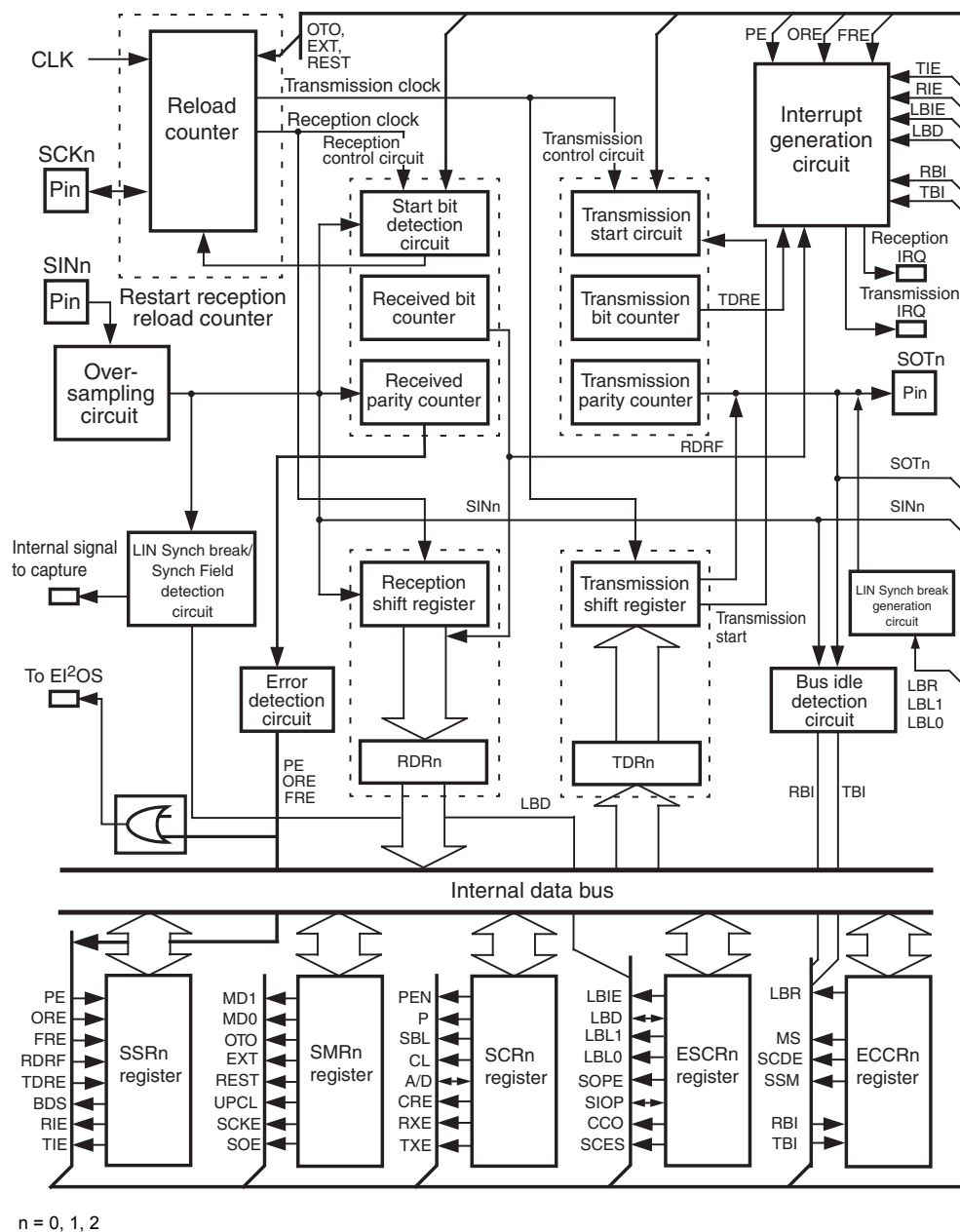
■ Configuration of LIN-UART

LIN-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 circuit
- Interrupt Generation Circuit
- LIN Synch Break/Synch Field Detection Circuit
- LIN Synch Break Generation Circuit
- Bus Idle Detection Circuit
- LIN-UART Serial Mode Register (SMR)
- Serial Control Register (SCR)
- Serial Status Register (SSR)
- Extended Communication Control Register (ECCR)
- Extended Status/Control Register (ESCR)

■ Block Diagram of LIN-UART

Figure 20-1. Block Diagram of LIN-UART



■ Explanation of the Different Blocks

□ Reload Counter

The reload counter is a 15-bit reload counter that functions as the dedicated baud rate generator. It can select external 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 BGRn1/BGRn0.

□ 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 completed, the received bit counter sets a flag in the serial status register. In this case, if the reception interrupt is enabled, the reception

interrupt request is generated. 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 received 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 completed, the reception shift register transfers receive data to the RDR register.

- Reception Data Register (RDR)

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 completed, the transmission bit counter sets a flag in the transmission data register. In this case, if the transmission interrupt is enabled, the transmission interrupt request is generated. The transmission start circuit starts transmission when data is written to TDR register. 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 outputted.

- 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 Circuit

The oversampling circuit oversamples for five times in the machine clock. The received value is determined by majority decision of sampling time. 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 interrupt enable bit is set, the interrupt will be generated immediately.

- LIN Synch Break and Synchronization Field Detection Circuit

The LIN synch break and LIN synchronization field detection circuit detects a LIN synch break if a LIN master node is sending a message header. If a LIN synch break is detected LBD flag bit is generated. The first and the fifth falling edge of the LIN synchronization field is recognized by this circuit by generating an internal signal for the Input Capture Unit to measure the actual serial clock synchronization of the transmitting master node.

- LIN Synch Break Generation Circuit

The LIN synch break generation circuit generates a LIN synch 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 software (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;

- Checking status of receive/transmit operations and errors
- Specifying LSB first or MSB first
- Receive interrupt enable/disable
- Transmit interrupt enable/disable

❑ Extended Communication Control Register (ECCR)

This register performs the following functions;

- Indicating bus idle detection
- Specifying synchronous clock
- Specifying LIN synch break generation

❑ 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 in LIN-UART synchronous clock mode
- Specifying sampling clock edge

20.3 LIN-UART Pins

This section lists and details the pins, interrupt sources, and registers of the LIN-UART.

■ LIN-UART Pins

The LIN-UART pins also serve as general-purpose ports. Table 20-5. lists the pin functions, I/O formats, and settings required to use LIN-UART.

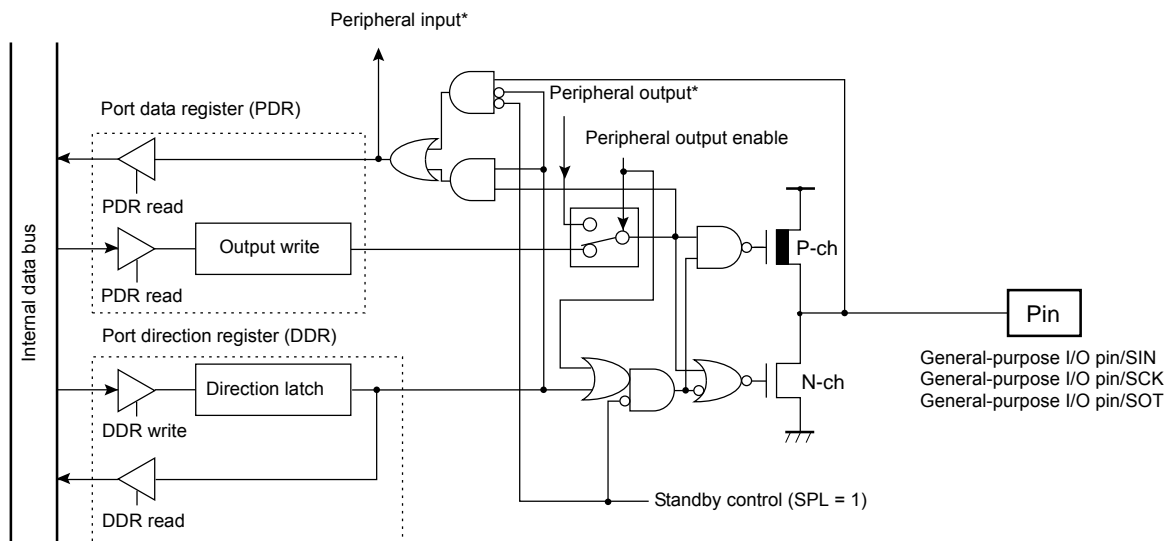
Table 20-5. LIN-UART Pin

Pin Name	Pin Function	I/O Format	Standby Control	Setting Required to Use Pin
P82/SIN0 P85/SIN1 P50/SIN2	Port I/O or serial data input	CMOS output/CMOS, Automotive input	Provided	Set as input port (DDR: corresponding bit = 0)
P83/SOT0 P86/SOT1 P51/SOT2	Port I/O or serial data output			Set to output enable mode (SMRn: SOE = 1)
P84/SCK0 P87/SCK1 P52/SCK2	Port I/O or serial clock input/output			Set as an input port when a clock is inputted (DDR: corresponding bit = 0) Set to output enable mode when a clock is outputted (SMRn: SCKE = 1)

See "3. DC Characteristics in "Electrical Characteristics" in the data sheet for the value.

■ Block Diagram of LIN-UART Pins

Figure 20-2. Block Diagram of LIN-UART Pins



Standby control: Stop mode (SPL=1), watch mode (SPL=1), time-base timer mode (SPL=1)

*: Peripheral I/O signals are inputted or outputted from pins having peripheral functions.

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. Configuration of the Serial Control Register (SCR)

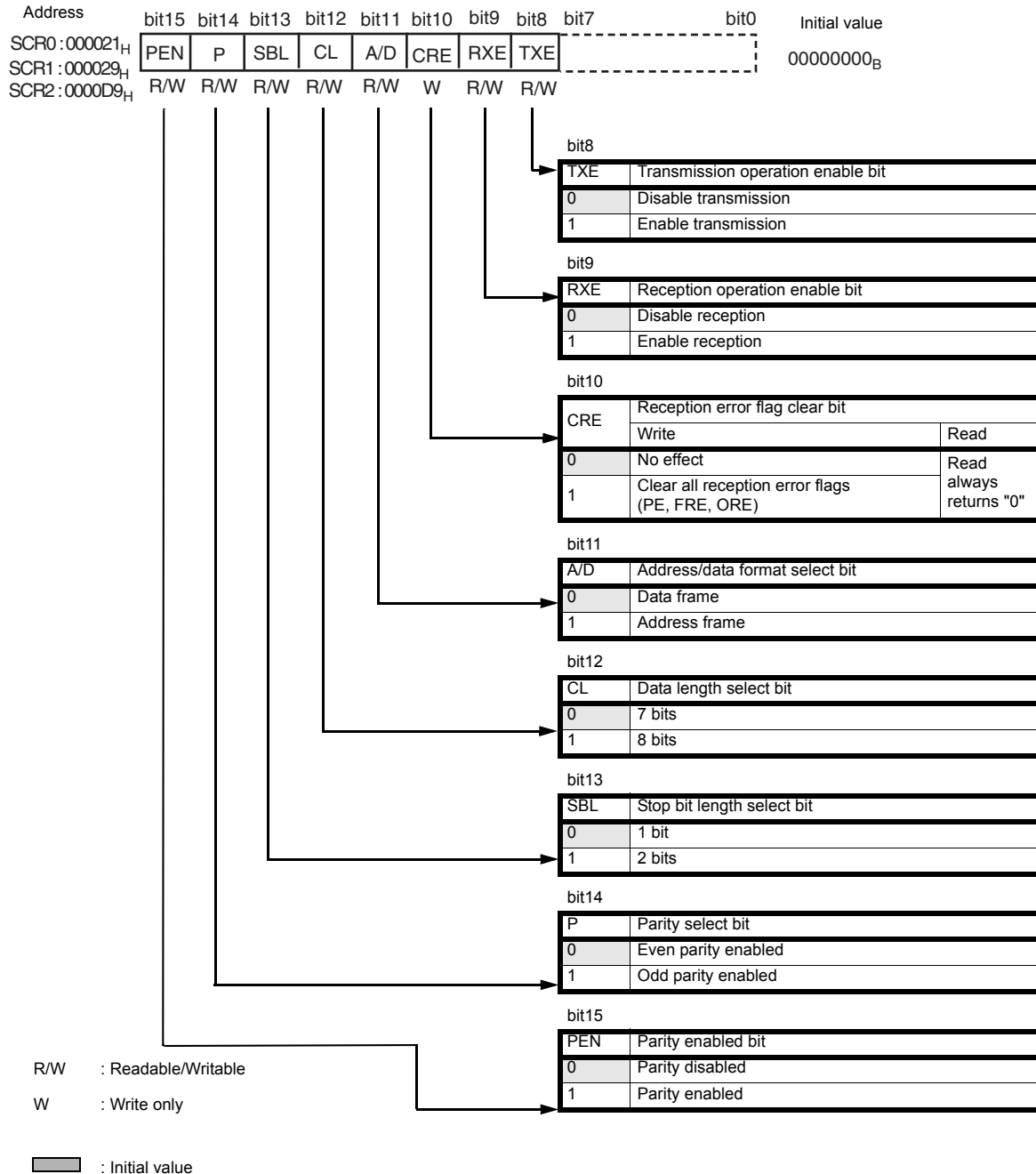


Table 20-6. Function of Each Bit in Serial Control Register (SCR)

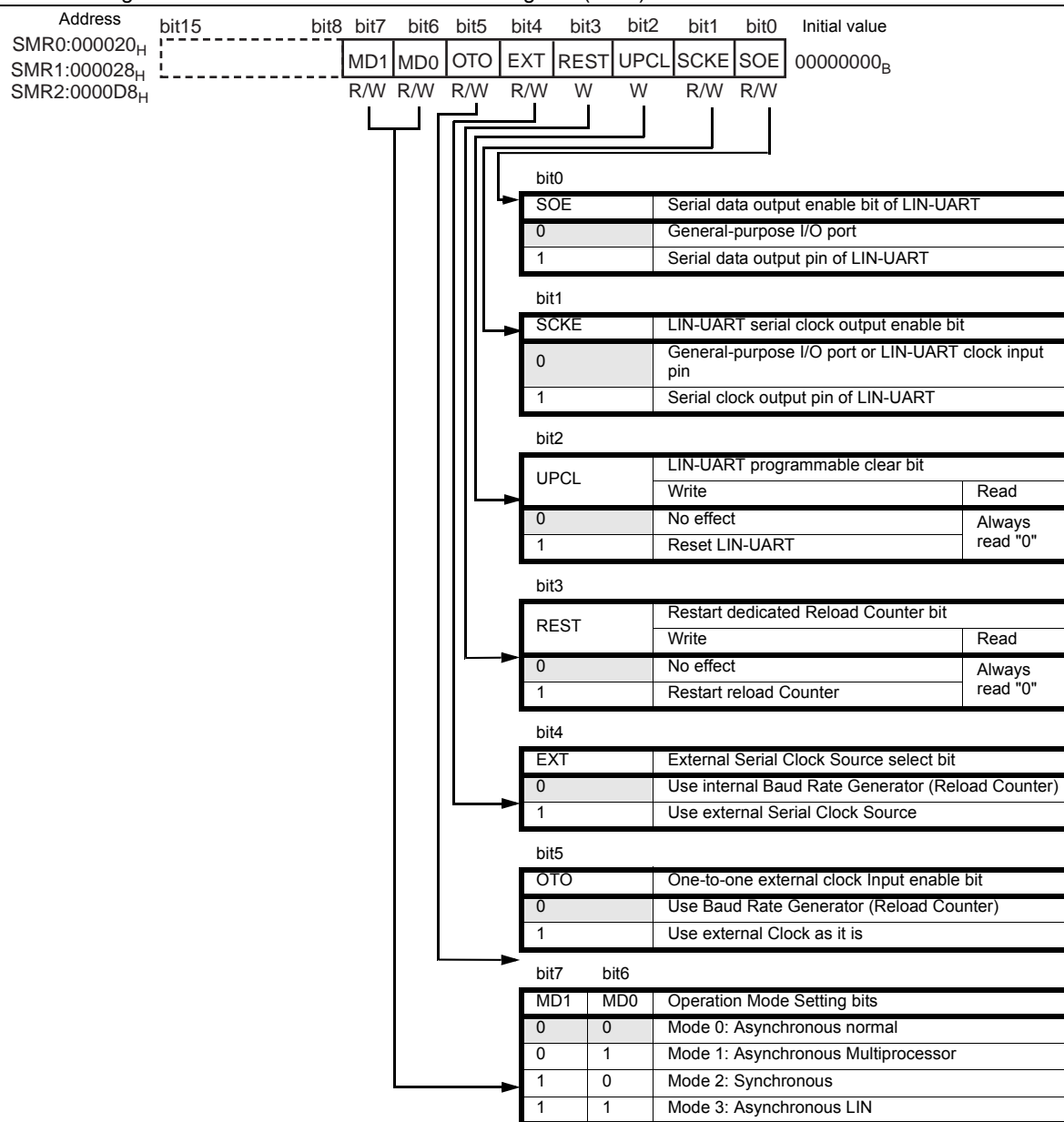
Bit Name		Function
bit15	PEN: Parity enable bit	This bit selects whether to add a parity bit during transmission or detect it during reception. Note: Parity bit is only provided in mode 0 and in mode 2 if SSM of the ECCR is selected to "1". This bit is fixed to "0" (no parity) in mode 1, 3.
bit14	P: Parity selection bit	When parity is provided (SCR: PEN=1), this bit selects even (0) or odd (1) parity.
bit13	SBL: Stop bit length selection bit	This bit sets the bit length of the stop bit (frame end mark in transmit data) in operating mode 0, 1 (asynchronous) or in operating mode 2 (synchronous) with the settings that start/stop bit is set (ECCR: SSM = 1). This bit is fixed to "0" (1 stop bit) in mode 3 (LIN). Note: At reception, only the first bit of the stop bit is always detected.
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	A/D: Address/data format selection bit	This bit specifies the frame data format to be transmitted and received in multiprocessor mode 1. Writing to this bit is provided for a master CPU, reading from it for slave CPU. "1" indicates an address data frame, "0" indicates a usual data frame. The reading value is a value of last received data format. Note: See "20.8 Notes on Using LIN-UART" to use this bit.
bit10	CRE: Reception error flag clear bit	This bit clears the FRE, ORE, and PE flags of the serial status register (SSR). Writing "1" to it clears the error flag. Writing "0" has no effect. Reading from it always returns "0".
bit9	RXE: Reception operation enable bit	This bit enables/disables LIN-UART reception. If this bit is set to "0", LIN-UART disables the reception of data frames. If this bit is set to "1", LIN-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 operation enable bit	This bit enables/disables LIN-UART transmission. If the bit is set to "0", LIN-UART disables the transmission of data frames. If the bit is set to "1", LIN-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 LIN-UART 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.

■ LIN-UART Serial Mode Register (SMR)

Figure 20-5. Configuration of the LIN-UART Serial Mode Register (SMR)



R/W : Readable/Writable

W : Write only

: Initial value

Table 20-7. Function of Each Bit in Serial Mode Register (SMR)

Bit name		Function
bit7, bit6	MD1, MD0: Operation mode setting bits	These two bits set the LIN-UART operation mode.
bit5	OTO: One-to-one external clock input enable bit	This bit sets an external clock directly to the LIN-UART serial clock by writing "1". This function is used for operating mode 2 (synchronous) slave mode operation (ECCR:MS=1). When EXT=0, this bit is fixed to "0".
bit4	EXT: External serial clock source selection bit	This bit selects the clock input. When "0" is set to this bit, it selects the clock of internal baud rate generator (reload counter). When "1" is set to it, it selects the external serial clock source.
bit3	REST: Restart of dedicated reload counter bit	If "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: LIN-UART programmable clear bit (LIN-UART software reset)	Writing "1" to this bit resets LIN-UART immediately (LIN-UART software reset). The register settings are preserved. Possible reception or transmission will cut off. All of the transmission/reception interrupt sources (TDRE, RDRF, LBD, PE, ORE, FRE) are cleared. Reset the LIN-UART after disabling the interrupt and transmission. Also, when the reception data register is cleared (RDR = 00 _H), the reload counter restarts. Writing "0" to this bit has no effect. Reading from it always returns "0". Note:Execute LIN-UART software reset (UPCL=1) when the TXE bit of the serial control register (SCR) is "0".
bit1	SCKE: LIN-UART serial clock output enable bit	This bit controls the serial clock I/O ports. When this bit is "0", SCKn pin operates 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). When ECCR:MS=1, the SCKE bit is fixed to "0". Note:When using SCKn pin as serial clock input (SCKE=0) pin, set the corresponding DDR bit of general-purpose I/O port as input port. Also, select external clock (EXT = 1) using the 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: LIN-UART 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 port. When this bit is "1", SOTn pin operates as serial data output pins (SOTn). Reference:When the output of serial data is enabled (SOE=1), SOTn pin functions as serial data output pin (SOTn) regardless of the status of general-purpose I/O ports.

20.4.3 Serial Status Register (SSR)

This register checks the transmission and reception status and error status, and enables and disables the interrupts.

■ Serial Status Register (SSR)

Figure 20-6. Configuration of the Serial Status Register (SSR)

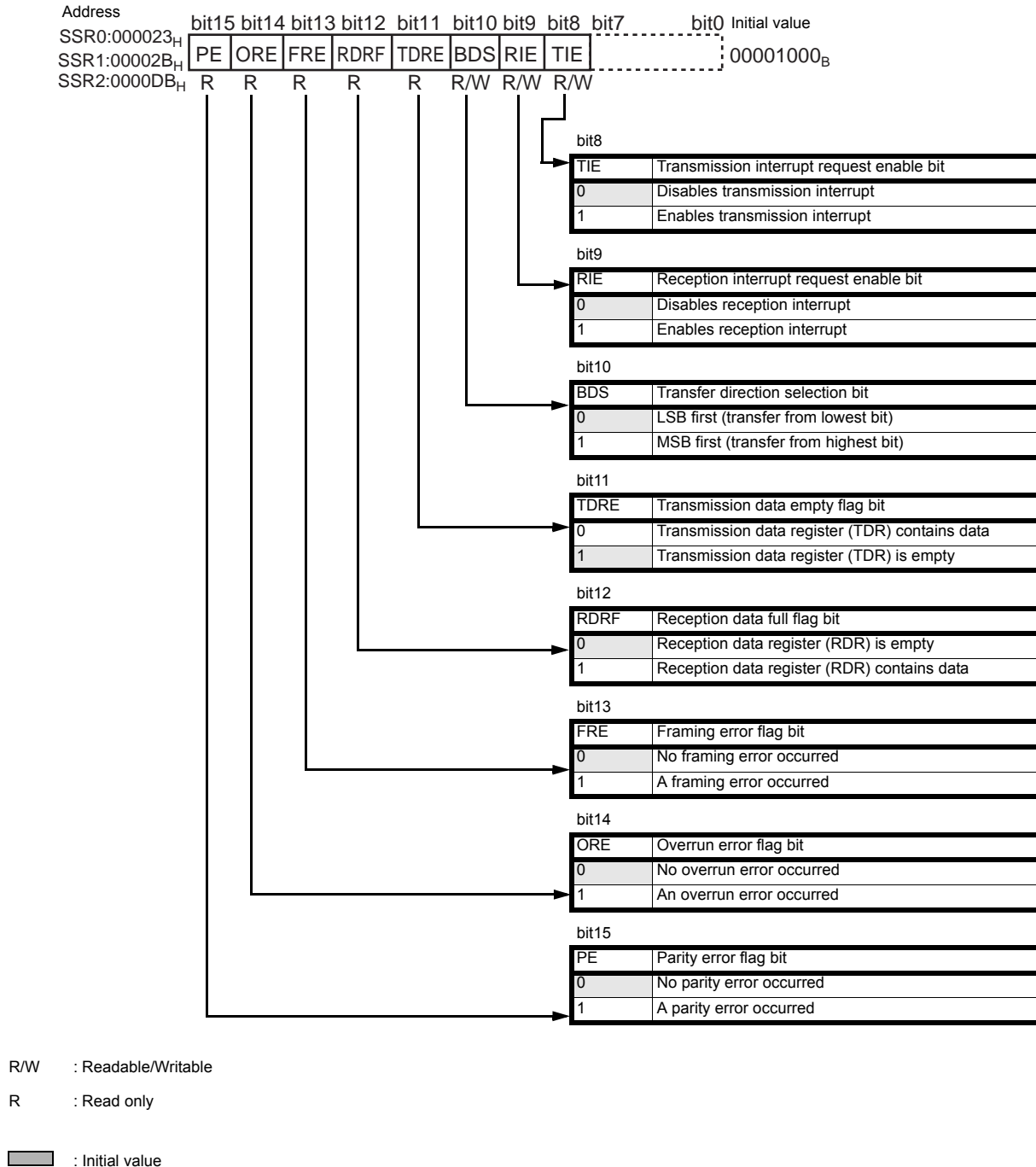


Table 20-8. Function of Each Bit in Serial Status Register (SSR)

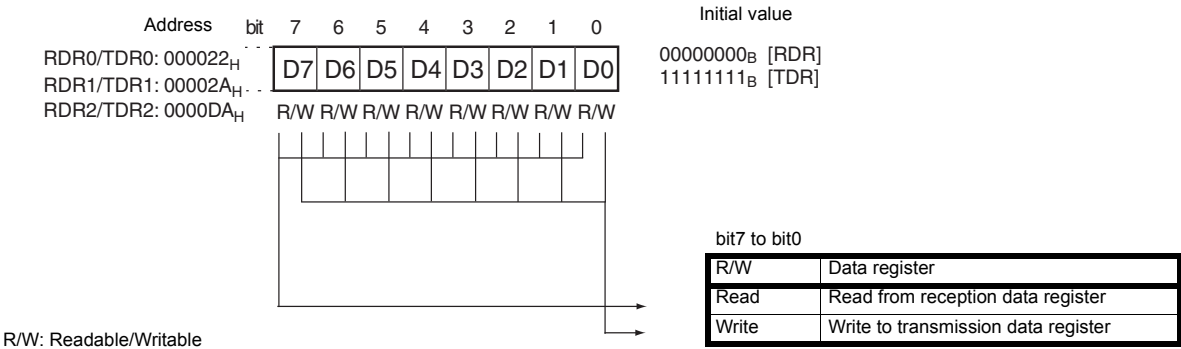
Bit name		Function
bit15	PE: Parity error flag bit	<ul style="list-style-type: none"> •This bit is set to "1" when a parity error occurs during reception at PE=1 and is cleared when "1" is written to the CRE bit of the LIN-UART serial control register (SCR). •A reception interrupt request is outputted 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	<ul style="list-style-type: none"> •This bit is set to "1" when an overrun error occurs during reception and is cleared when "1" is written to the CRE bit of the LIN-UART serial control register (SCR). •A reception interrupt request is outputted 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	<ul style="list-style-type: none"> •This bit is set to "1" when a framing error occurs during reception and is cleared when "1" is written to the CRE bit of the LIN-UART serial control register (SCR). •A reception interrupt request is outputted when this bit and the RIE bit are "1". •Data in the reception data register (RDR) is invalid when this flag is set. <p>Note:When SCR:SBL is "1", and a framing error is detected at the first or second bit (stop bit), this bit is set to "1" regardless of which stop bit. Therefore, it is necessary to determine whether the reception data is valid or invalid at the second stop bit.</p>
bit12	RDRF: Receive data full flag bit	<ul style="list-style-type: none"> •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 outputted when this bit and the RIE bit are "1".
bit11	TDRE: Transmission data empty flag bit	<ul style="list-style-type: none"> •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 indicates that valid data exists in TDR. This bit is set to "1" when data is loaded into the transmission shift register and transmission starts and indicates that no valid data exists in TDR. •A transmission interrupt request is generated if both this bit and the TIE bit are "1". •When the TDRE bit is "1", setting the LBR bit in the extended communication control register (ECCR) to "1" changes the TDRE bit to "0". Then, the TDRE bit goes back to "1" when TDR does not have valid data after LIN Synch break is generated. <p>Note:This bit is set to "1" (TDR empty) as its initial value.</p>
bit10	BDS: Transfer direction selection bit	<ul style="list-style-type: none"> •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). It is fixed to "0" in mode 3. <p>Note:Data values are exchanged between the upper and lower when reception data is written to the reception data register (RDR). Consequently, if the BDS bit is rewritten after reception data is written to RDR, the RDR data will be invalid.</p>
bit9	RIE: Reception interrupt request enable bit	<ul style="list-style-type: none"> •This bit enables or disables the reception interrupt request output to the CPU. •If any of the PE, ORE and FRE bits is set to "1" or if this bit and RDRF bit are "1", then a reception interrupt request is outputted.
bit8	TIE: Transmission request interrupt enable bit	<ul style="list-style-type: none"> •This bit enables or disables the transmission interrupt request output to the CPU. •A transmission interrupt request is outputted when this bit and the TDRE bit are "1".

20.4.4 Reception and Transmission Data Register (RDR/TDR)

Both RDR and TDR registers are located at the same address. At reading, it functions as the reception data register. At writing, it functions as the transmission data register.

Reception Data Register (RDR)

Figure 20-7. Reception and Transmission Data Registers (RDR/TDR)



RDR is the data buffer register for serial data reception. The serial data signal transmitted to the serial input pin (SINn pin) is converted in the shift register and stored in RDR register. When the data length is 7 bits, the uppermost bit (RDR: D7) contains "0". When the data is stored in this register and the reception data full flag bit (SSR: RDRF) is set to "1". If a reception interrupt is enabled (SSR: RIE=1) at this point, a reception interrupt request occurs.

Read RDR when the RDRF bit of the serial 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 Data Register (TDR)

TDR is the data buffer register for serial data transmission. When data to be transmitted is written to the transmission data register (TDR) in transmission enable state (SCR: TXE=1), it is transferred to the transmission shift register, then converted to serial data, and transmitted from the serial data output pin (SOTn pin). If the data length is 7 bits, the uppermost bit (TDR: D7) is invalid data.

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 completed and transmission starts, the bit is set to "1". When the TDRE bit is "1", the next part of transmission data can be written. If transmission interrupts 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.

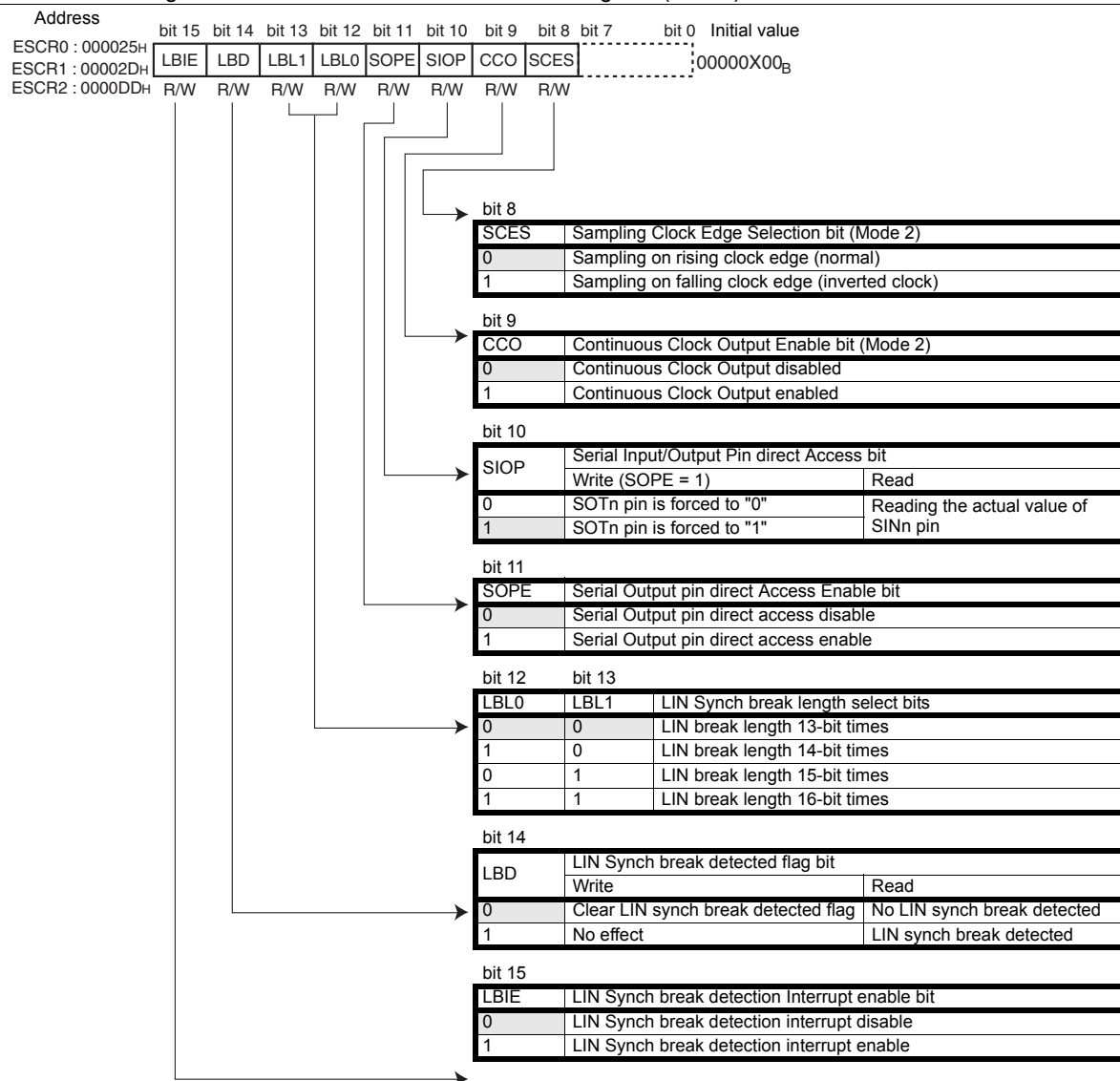
20.4.5 Extended Status/Control Register (ESCR)

This register provides the settings for enabling/disabling LIN Synch break interrupts LIN Synch break length selection, LIN Synch break detection, direct access to the SINn and SOTn pins and continuous clock output in LIN-UART synchronous clock mode and sampling clock edge.

■ Bit Configuration of Extended Status/Control Register (ESCR)

[Figure 20-8](#). shows the bit configuration of the extended status/control register (ESCR), and [Table 20-9](#). shows the function of each bit.

Figure 20-8. Bit Configuration of the Extended Status/Control Register (ESCR)



R/W : Readable/Writable


 : Initial value

Table 20-9. Function in 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 detection interrupt. When the LBD bit is set to "1" and this bit is "1", an interrupt is generated. This bit is fixed to "0" in operation mode 1 and 2.
bit14	LBD: LIN synch break detected flag bit	This bit goes "1" if a LIN synch break was detected in operating mode 3 (the serial input is "0" when bit width is 11 bits or more). Writing "0" to it clears this bit and the corresponding interrupt, if it is enabled. Read-modify-write instructions always return "1". Note that this does not indicate a LIN synch break detection. Note:When LIN synch break detection is performed, disable reception (SCR: RXE=0) after enabling LIN synch break detection interrupt (LBIE=1).
bit13, bit12	LBL1/LBL0: LIN synch break length selection bits	These bits specify the bit length for the LIN Synch break generation time. Receiving a LIN synch break is always fixed to 11 bit times.
bit11	SOPE: Serial Output pin direct access enable bit *	Setting this bit to "1" enables the direct write to the SOTn pin, if SOE = 1 (SMR). *
bit10	SIOP: Serial Input/Output Pin direct access bit *	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. Note: The bit operation instruction returns the bit value of the SOTn pin in the read cycle.*
bit9	CCO: Continuous Clock Output enable bit	This bit enables a continuous serial clock output at the SCKn pin if LIN-UART operates in master operation 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: Sampling clock edge selection bit	When SCES is set to "1" in operating mode 2 (synchronous) with the slave setting, the sampling edge switches from the rising edge to the falling edge. When the SCKn pin is set to output clock in operating mode 2 with the master setting (ECCR:MS=0), the internal serial clock and the output clock signal are inverted. During operation mode 0,1,3, please set this bit to "0".

*: Refer to the following table.

Table 20-10. Description of the Interaction of SOPE and SIOP

SOPE	SIOP	Writing to SIOP	Reading from SIOP
0	R/W	No effect (but the write value is retained)	Returns current value of SINn
1	R/W	Write "0" or "1" to SOTn	Returns current value of SINn
0	RMW	No effect (but the write value is retained)	Returns current value of SOTn
1	RMW	Write "0" or "1" to SOTn	Returns current value of SOTn

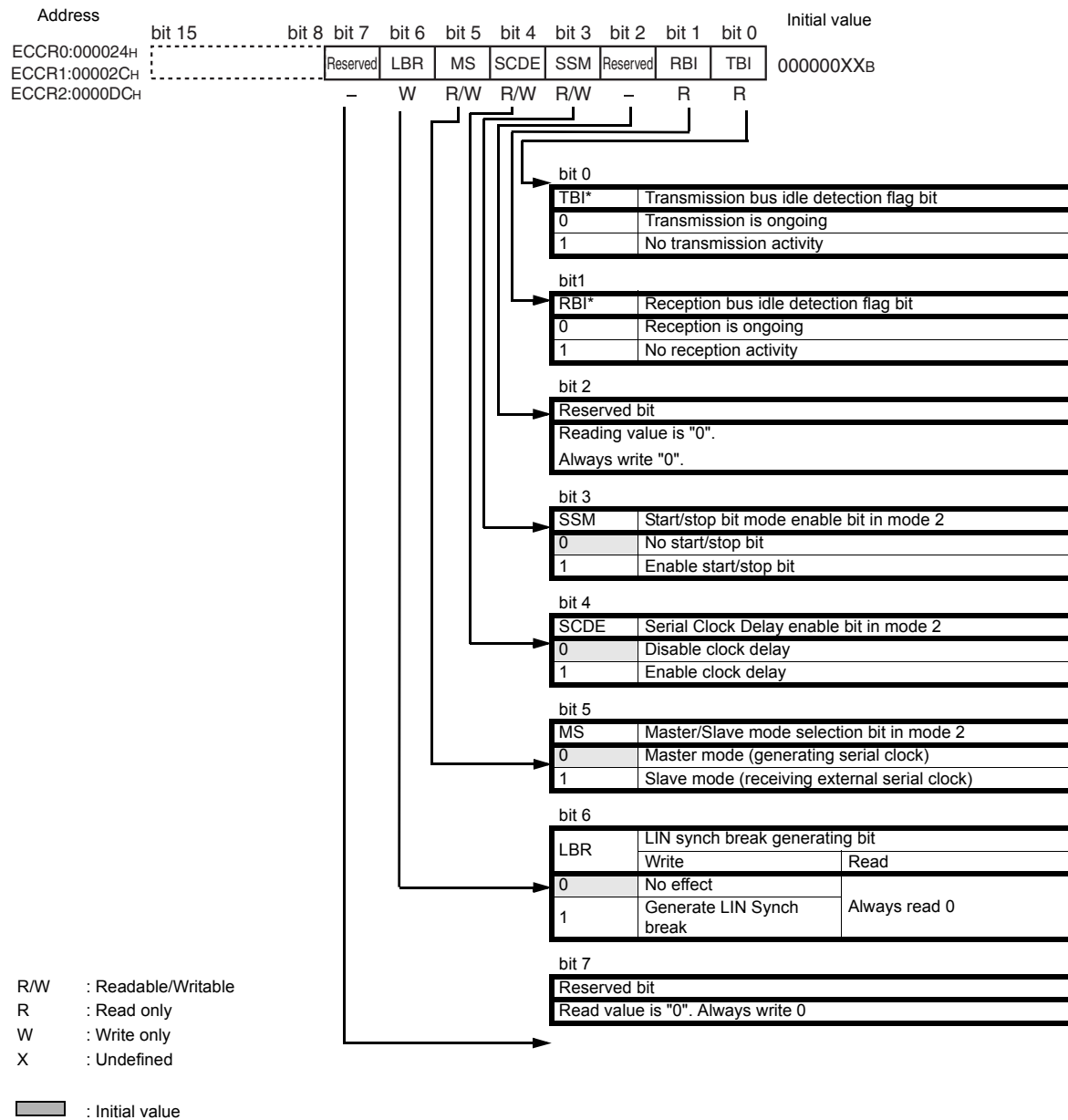
20.4.6 Extended Communication Control Register (ECCR)

The extended communication control register (ECCR) provides bus idle detection, synchronous clock settings, and the LIN synch break generation.

■ Bit Configuration of Extended Communication Control Register (ECCR)

Figure 20-9. shows the bit configuration of the extended communication control register (ECCR), and Table 20-11. shows the function of each bit.

Figure 20-9. Bit Configuration of the Extended Communication Control Register (ECCR)



*: Not used in operation mode 2 when SSM = 0

Table 20-11. Function of Each Bit in the Extended Communication Control Register (ECCR)

Bit name		Function
bit7	Reserved bit	Reading value is "0". Always write "0".
bit6	LBR: LIN Synch break Generating bit	Writing "1" to this bit generates a LIN synch break of the length selected by the LBL0/LBL1 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 LIN-UART in mode 2. If master is selected LIN-UART generates the synchronous clock by itself. If slave mode is selected, LIN-UART receives external serial clock. This bit is fixed to "0" in operation mode 0, 1 and 3. Change this bit, when the SCR: TXE bit is "0". Note: If slave mode is selected, the clock source must be external and enabled the external clock input (SMR: SCKE = 0, EXT = 1, OTO = 1).
bit4	SCDE: Serial clock delay enable bit	If this bit is set to "1" the serial output clock is delayed as shown in Figure 20-21. if LIN-UART operates in master mode 2. This bit is enabled to SPI. This bit is fixed to "0" in operation mode 0, 1, and 3.
bit3	SSM: Start/Stop bit mode enable bit	This bit adds start and stop bits to the synchronous data format if the bit is set to "1" in operation mode 2. This bit is fixed to "0" in operation mode 0, 1, and 3.
bit2	Reserved bit	Reading value is "0". Always write to "0".
bit1	RBI: Reception bus idle detection flag bit	This bit is "1" if there is no reception activity on the SIN pin and it is kept at "H". Do not use this bit in mode 2.
bit0	TBI: Transmission bus idle detection 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 MS=0.

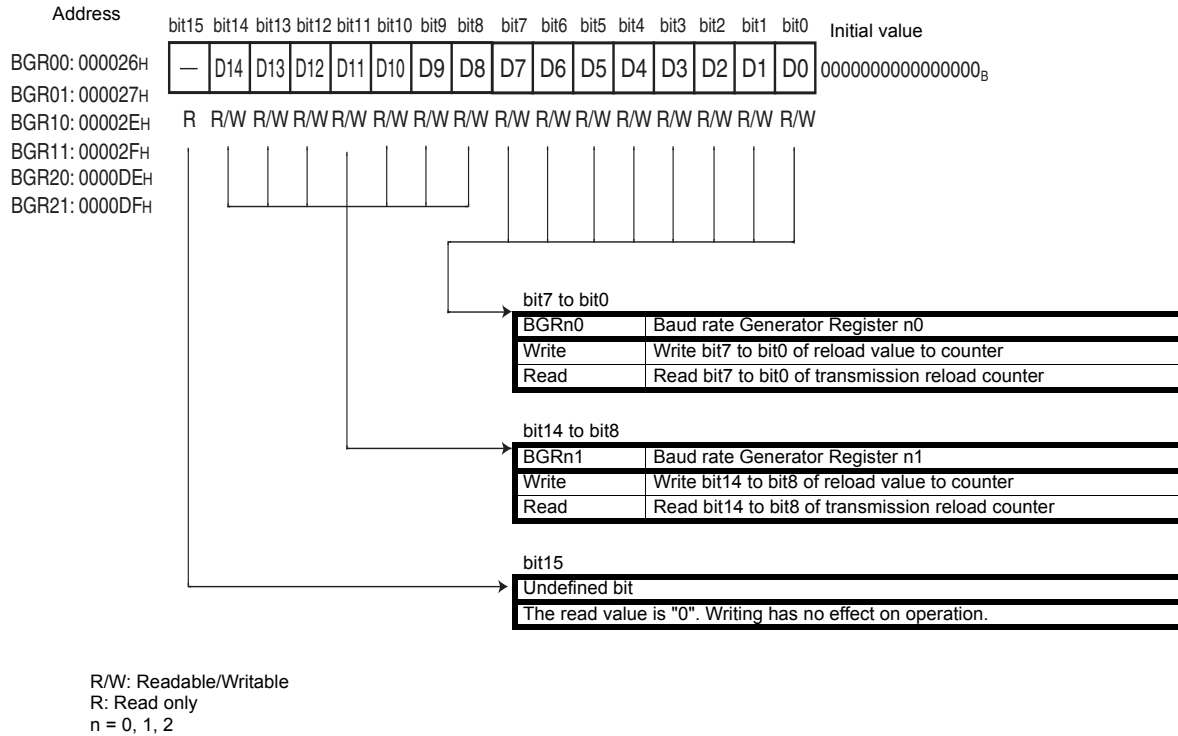
20.4.7 Baud Rate Generator Register 0 and 1 (BGR0/BGR1)

The baud rate generator register 0 and 1 (BGR0/BGR1) set the division ratio for the serial clock. Also, the actual count of the transmission reload counter can be read.

■ Bit Configuration of Baud Rate Generator Register (BGR0/BGR1)

Figure 20-10. shows the bit configuration of the baud rate generator register (BGR0/BGR1).

Figure 20-10. Bit Configuration of Baud Rate Generator Register (BGR0/BGR1)



The baud rate generator registers determine the division ratio for the serial clock.

The BGRn1 corresponds to the upper bit and BGRn0 to lower bit, writing of the reload value to counter and reading of the transmission reload counter value are allowed. Also, byte and word access are enabled. Writing a reload value other than "0" to the baud rate generator registers causes the reload counter to start counting.

20.5 LIN-UART Interrupts

LIN-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 transmission is started.
- A LIN Synch break is detected.

The extended intelligent I/O service (EI²OS) is available for these interrupts.

■ LIN-UART Interrupts

Table 20-12. shows the interrupt control bits and interrupt cause of the LIN-UART.

Table 20-12. Interrupt Control Bits and Interrupt Cause 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 flag
			0	1	2	3			
Reception	RDRF	SSR	m	m	m	m	Receive data is written to RDR.	SSR:RIE	Receive data is read.
	ORE	SSR	m	m	m	m	Overrun error		"1" is written to reception error flag clear bit (SCR: CRE).
	FRE	SSR	m	m	D	m	Framing error		
	PE	SSR	m	'	D	'	Parity error		
	LBD	ESCR	'	'	'	m	LIN Synch break detected	ESCR:LBIE	"0" is written to ESCR: LBD.
Transmission	TDRE	SSR	m	m	m	m	TDR empty	SSR:TIE	Writing transmit data; writing "1" to LIN Synch break generation bit (ECCR:LBR)
Input Capture	ICP0/ICP1	ICS01	'	'	'	m	1st falling edge of LIN synch field	ICS01: ICE0/ICE1	Disable ICP0/ICP1
	ICP0/ICP1	ICS01	'	'	'	m	5th falling edge of LIN synch field		

○:Used bit

×:Unused bit

Δ: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 completed

- When the reception data is transferred from the reception shift register to the reception data register (RDR) (RDRF = 1)

- Overrun error

- When RDRF is 1, and the next reception data is transferred from the reception shift register to the reception data register (RDR) (ORE=1) without RDR being read by the CPU

- Framing error, Stop bit reception error (FRE=1)

- Parity error, Parity detection error (PE=1)

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". The error flags are cleared to "0", if "1" is written to the reception error flag clear bit (CRE) of the serial control register (SCR).

Note:

The CRE flag is "write only" and by writing "1" to it, it is internally held to "1" for one clock cycle.

□ Transmission Interrupt

If transmission data is transferred from the transmission data register (TDR) to the transfer shift register and transmission is started, the transmission data register empty flag bit (TDRE) of the serial status register (SSR) is set to "1". In this case a transmission interrupt request is generated, if the transmission interrupt enable (TIE) bit of the SSR was set to "1" before.

Note:

The initial value of TDRE (after hardware or software reset) is "1". So an interrupt is generated immediately then, if the TIE bit 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 LIN-UART operates in mode 3 as a LIN slave.

If the bus (serial input) goes "0" for more than 11 bit times, the LIN synch break detected flag bit (LBD) of the extended status/control register (ESCR) is set to "1".

The LIN synch break interrupt and the LBD flag are cleared after writing "0" to the LBD flag. The LBD flag has to be performed before capture interrupt for LIN synch field.

When LIN synch break detection is performed, it is necessary to disable the reception (SCR: RXE=0).

□ LIN Synchronization Field Edge Detection Interrupts

This works for LIN slave operation in operation mode 3.

After a LIN Synch break is detected, the internal signal is set to "1" at the first falling edge of the LIN Synch field, and clear to "0" after the fifth falling edge. When the capture side is configured to input the

internal signal (ICU0/ICU1) and to detect both edges, an input capture interrupt is generated if enabled.

The difference in the count values detected by the capture function corresponds to the 8 bits in the master serial clock. The new baud rate can be calculated from this value.

When the falling edge of the start bit is detected, the reload counter restarts automatically.

■ LIN-UART Interrupts and EI²OS

Table 20-13. LIN-UART Interrupts and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI ² OS
		Register name	Address	Lower	Upper	Bank	
LIN-UART0 reception	#35(23 _H)	ICR12	0000BC _H	FFFF70 _H	FFFF71 _H	FFFF72 _H	*1
LIN-UART0 transmission	#36(24 _H)	ICR12	0000BC _H	FFFF6C _H	FFFF6D _H	FFFF6E _H	*2
LIN-UART1 reception	#37(25 _H)	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	*1
LIN-UART1 transmission	#38(26 _H)	ICR13	0000BD _H	FFFF64 _H	FFFF65 _H	FFFF66 _H	*2
UART2 reception	#39(27 _H)	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	*1
UART2 transmission	#40(28 _H)	ICR14	0000BE _H	FFFF5C _H	FFFF5D _H	FFFF5E _H	*2

*1:ICR12 to ICR14 are shared with multiple interrupt sources; therefore, they are only available when such sources are not used as interrupts.

Provided with a function that detects a LIN-UART reception error and stops EI²OS.

*2:ICR12 to ICR14 are shared with multiple interrupt sources; therefore, they are only available when such sources are not used as interrupts.

■ LIN-UART EI²OS Functions

LIN-UART has a circuit for operating EI²OS, which can be started up for either reception or transmission interrupts.

□ For Reception

EI²OS can be used if other interrupt is not enabled because the UART shares the interrupt control registers with transmission interrupt and other UART.

□ For Transmission

LIN-UART shares the interrupt control registers with the LIN-UART reception interrupts and other UART. Therefore, EI²OS can be used if other interrupt is not enabled.

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

The received data is stored in the RDR register 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).

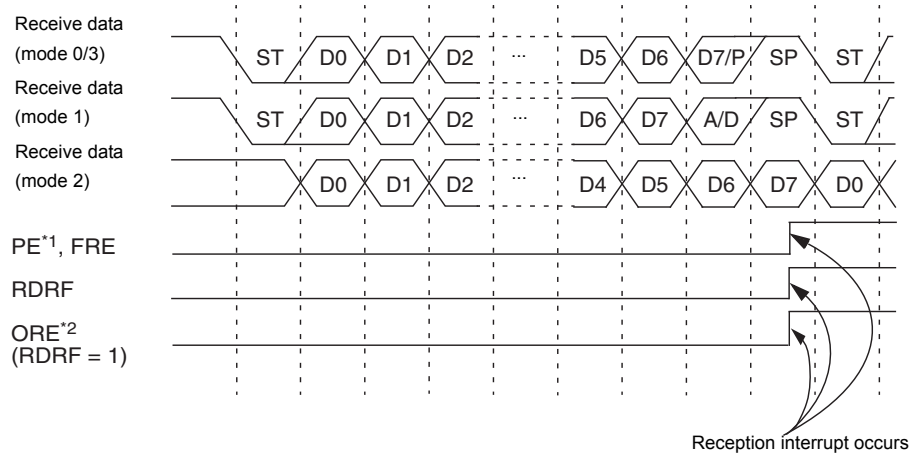
Each flag is set if the received data is completed (RDRF = 1) and the reception error (PE, ORE, FRE) of the serial status register (SSR) was set to "1". In this case, if the reception interrupt is enabled (SSR: RIE=1), reception interrupt occurs.

Note:

If a reception error has occurred in each mode, the reception data register (RDR) contains invalid data.

Figure 20-11. shows the reception operation and flag set timing.

Figure 20-11. Reception Operation and Flag Set Timing



*1: The PE flag will always remain "0" in mode 1 or 3.

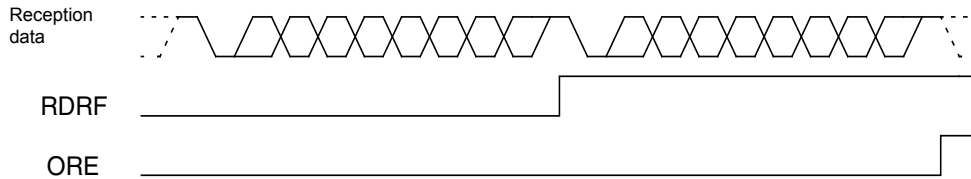
*2: An overrun error only occurs, if next data is transferred before the reception data is read (RDRF=1).

ST: Start Bit SP: Stop Bit A/D: Mode 1 (multiprocessor) address/data selection bit

Note:

The example in Figure 20-11. does not show all possible reception options for mode 0. Here it is: "7P1" and "8N1" (P = "E" [even] or "O" [odd]).

Figure 20-12. ORE Flag Set Timing



20.5.2 Transmission Interrupt Generation and Flag Set Timing

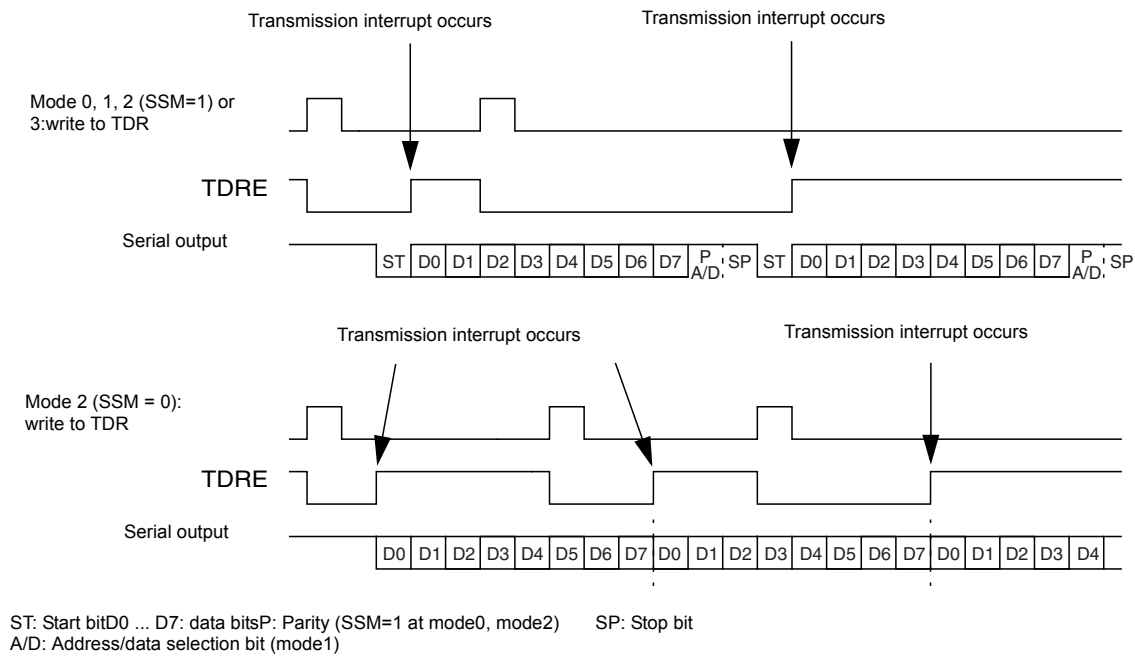
A transmission interrupt is generated when the transmission data is transferred from transmission data register (TDR) to transmission shift register and transmission is started.

■ Transmission Interrupt Generation and Flag Set Timing

When the data written to the TDR register is transferred to the transmission shift register and the transmission is started, next data to be written is enabled (SSR: TDRE=1). Then, if transmission interrupt is enabled (SSR: TIE=1), the transmission interrupt occurs. Because the TDRE bit is "read only", it only can be cleared to "0" by writing data into TDR.

The following figure demonstrates the transmission operation and flag set timing for the four modes of LIN-UART.

Figure 20-13. Transmission Operation and Flag Set Timing


Note:

The example in [Figure 20-13](#) does not show all possible transmission options for mode 0. Here it is: "8p1" (p = "E" [even] or "O" [odd]).

No parity bit or address data selection bit are transmitted in mode 3, or in mode 2 with 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 is generated.

Note:

A transmission interrupt is generated immediately after the transmission interrupt is enabled (SSR: TIE=1) because the TDRE bit is set to "1" as its initial value. Be careful with the timing for enabling the transmit interrupt since the TDRE bit can be cleared only by writing new data to the transmit data register (TDR). Carefully specify the transmission interrupt enable timing.

20.6 LIN-UART Baud Rates

One of the following can be selected for the LIN-UART transmission/reception clock source:

- Dedicated baud rate generator (Reload Counter)
- Input external clock to baud rate generator (Reload Counter)
- External clock (directly use SCKn pin input clock)

■ LIN-UART Baud Rate Selection

The baud rate selection circuit is designed as shown in [Figure 20-14](#). One of the following three types of baud rates can be selected:

- Baud rates determined using the dedicated baud rate generator (reload counter) with internal clock

LIN-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/BGR1).

The reload counter divides the internal clock by set value.

It is used in asynchronous or synchronous (master) mode. Internal clock and baud rate generator clock are selected for the setting of clock source (SMR: EXT=0, OTO=0).

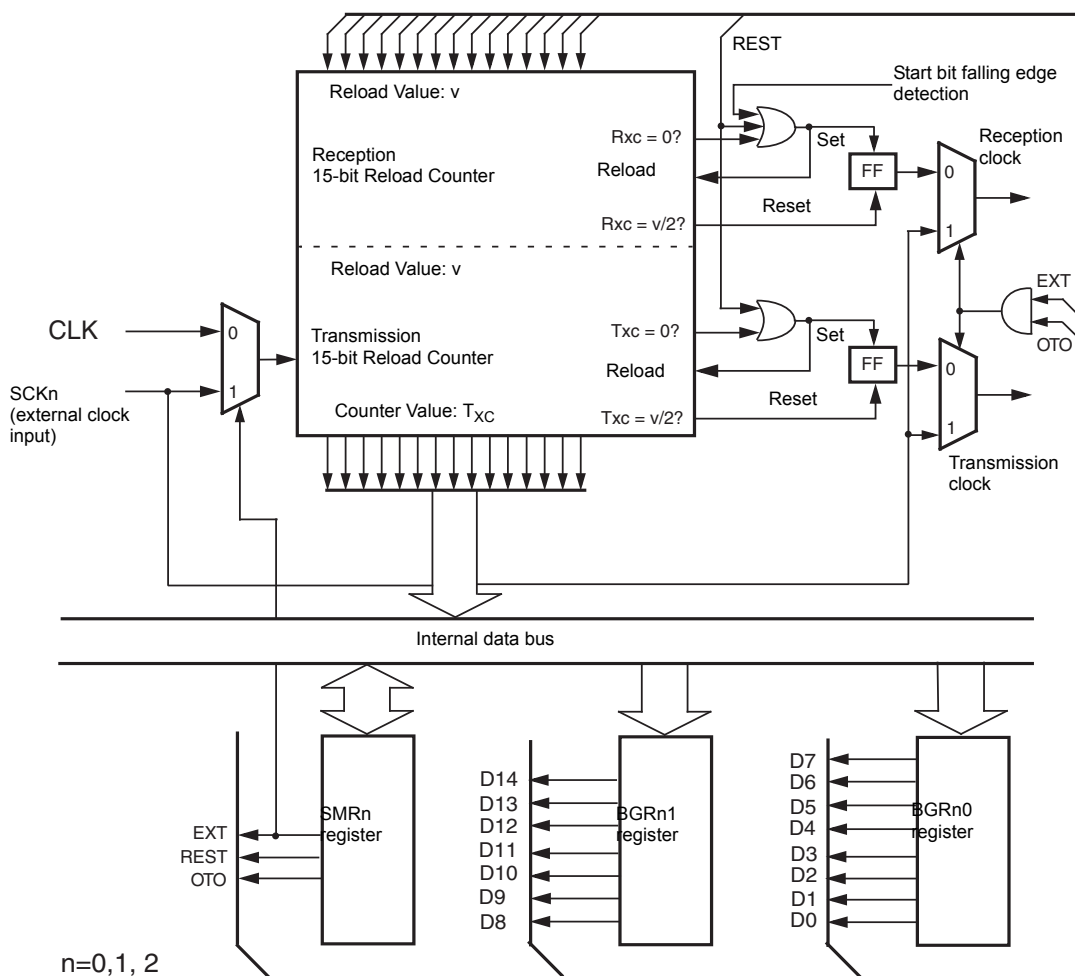
- Baud rates determined using the dedicated baud rate generator (reload counter) with external clock

The external clock is used as the clock source for the reload counter. The baud rate can be selected via the 15-bit reload value determined by the baud rate generator register 0 and 1 (BGR0/BGR1). The reload counter divides the external clock by set value. It is used in asynchronous mode. External clock and baud rate generator clock are selected for the setting of the clock source (SMR: EXT=1, OTO=0). This was designed to use oscillators with special frequencies and having the possibility to divide them.

- Baud rates determined using external clock (one-to-one mode)

The clock input from LIN-UART clock pulse input pins (SCKn) is used as it is (synchronous mode 2 slave operation (ECCR: MS=1)). It is used in synchronous mode (slave). External clock and direct use of external clock are selected for the setting of clock source (SMR: EXT=1, OTO=1).

Figure 20-14. Baud Rate Selection Circuit of LIN-UART



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 (BGRn1/BGRn0). The following calculation formula should be used to set the desired baud rate:

Reload Value:

$$v = [\phi / b] - 1,$$

where v is the reload value, ϕ is the machine clock or external clock frequency, b is the baud rate.

□ Example of calculation

If the machine clock is 16 MHz, the internal clock is used, and the desired baud rate is 19200 bps baud then the reload value v is:

$$v = [16 \times 10^6 / 19200] - 1 = 832$$

The exact baud rate can then be recalculated: $b = \phi / (v + 1)$, here it is: $16 \times 10^6 / 833 = 19207.6831$

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 oversampling is performed to determine the reception value internally.

■ Reload Value and Baud Rate for Each Clock Speed

Table 20-14. shows the reload value and baud rate for each clock speed.

Table 20-14. Reload Value and Baud Rate

Baud rate (bps)	8 MHz		10 MHz		16 MHz		20 MHz		24 MHz		32 MHz	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
4M	-	-	-	-	-	-	-	-	5	0	7	0
2M	-	-	-	-	7	0	9	0	11	0	15	0
1M	7	0	9	0	15	0	19	0	23	0	31	0
500000	15	0	19	0	31	0	39	0	47	0	63	0
460800	-	-	-	-	-	-	-	-	51	-0.16	68	-0.64
250000	31	0	39	0	63	0	79	0	95	0	127	0
230400	-	-	-	-	-	-	-	-	103	-0.16	138	0.08
153600	51	-0.16	64	-0.16	103	-0.16	129	-0.16	155	-0.16	207	-0.16
125000	63	0	79	0	127	0	159	0	191	0	255	0
115200	68	-0.64	86	0.22	138	0.08	173	0.22	207	-0.16	277	0.08
76800	103	-0.16	129	-0.16	207	-0.16	259	-0.16	311	-0.16	416	0.08
57600	138	0.08	173	0.22	277	0.08	346	-0.06	416	0.08	555	0.08
38400	207	-0.16	259	-0.16	416	0.08	520	0.03	624	0	832	-0.04
28800	277	0.08	346	<0.01	554	-0.01	693	-0.06	832	-0.03	1110	-0.01
19200	416	0.08	520	0.03	832	-0.03	1041	0.03	1249	0	1666	0.02
10417	767	<0.01	959	<0.01	1535	<0.01	1919	<0.01	2303	<0.01	3071	<0.01
9600	832	0.04	1041	0.03	1666	0.02	2083	0.03	2499	0	3332	-0.01
7200	1110	<0.01	1388	<0.01	2221	<0.01	2777	<0.01	3332	<0.01	4443	-0.01
4800	1666	0.02	2082	-0.02	3332	<0.01	4166	<0.01	4999	0	6666	<0.01

Table 20-14. Reload Value and Baud Rate

Baud rate (bps)	8 MHz		10 MHz		16 MHz		20 MHz		24 MHz		32 MHz	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
2400	3332	<0.01	4166	<0.01	6666	<0.01	8332	<0.01	9999	0	13332	<0.01
1200	6666	<0.01	8334	0.02	13332	<0.01	16666	<0.01	19999	0	26666	<0.01
600	13332	<0.01	16666	<0.01	26666	<0.01	-	-	-	-	-	-
300	26666	<0.01	-	-	-	-	-	-	-	-	-	-

The unit of frequency deviation (dev.) is %.

The maximum baud rate for synchronous mode is 1/6 of the machine clock (value=5).

■ Using External Clock

If the EXT bit of the SMR is set to "1", an external clock is selected. The external clock is used in the same way as the internal clock to the baud rate generator.

If One-to-one external clock input mode (SMR: OTO=1) is selected the external clock input to the SCKn signal is directly connected to the LIN-UART serial clock inputs. This is needed for the LIN-UART synchronous mode 2 operating as slave device.

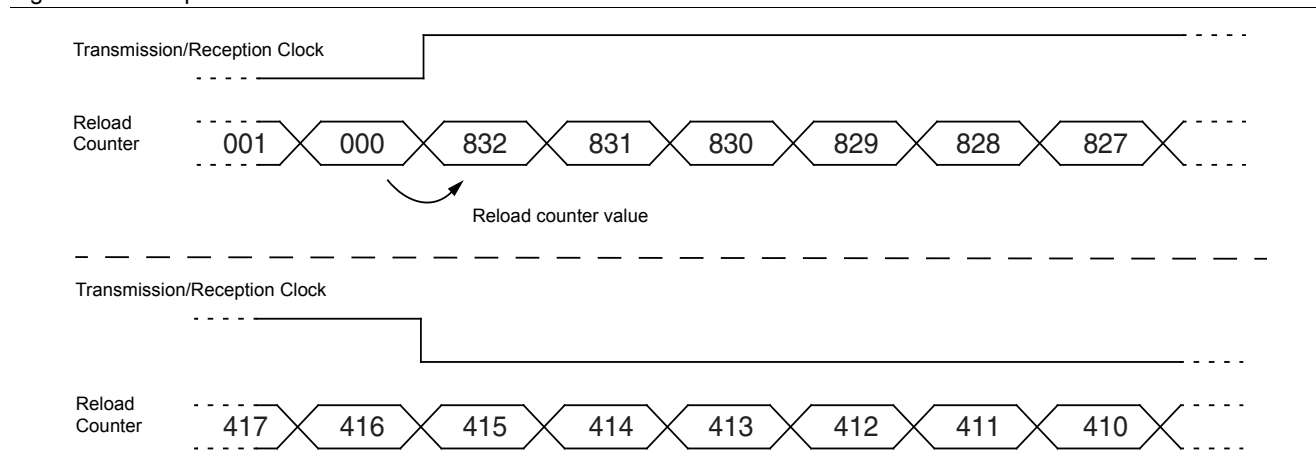
Note:

In any case the resulting clock signal is synchronized to the internal clock in the LIN-UART module. This means that indivisible clock rates will result in phase unstable signals.

■ Operation of Reload Counters

Assume the reload value is 832. Figure 20-15. demonstrates the operation of both reload counters:

Figure 20-15. Operation of the Reload Counters



Note:

The falling edge of the serial clock signal always occurs after $((v + 1) / 2)$.

20.6.2 Reload Counter

The reload counter is a 15-bit reload counter that functions as dedicated baud rate generator. The transmission/reception clock is generated by the external or internal clock. Also, the count value of the transmission reload counter can be read by the baud rate generator register (BGR1, BGR0).

■ Function of Reload Counter

The reload counter has the transmission and reception reload counters and functions as dedicated baud rate generator. It consists of a 15-bit register for the reload value and generates the transmission/reception clocks by the external or internal clock. Also, the count value of the transmission reload counter can be read by the baud rate generator register (BGR1, BGR0)

- Count start

When the reload value is written to the baud rate generator register (BGR1, BGR0), the reload counter starts counting.

- Restart

The reload counter is restarted under the following conditions.

For both transmit and receive reload counters:

- UART programmable reset (SMR: UPCL bit)
- Programmable restart (SMR: REST bit)

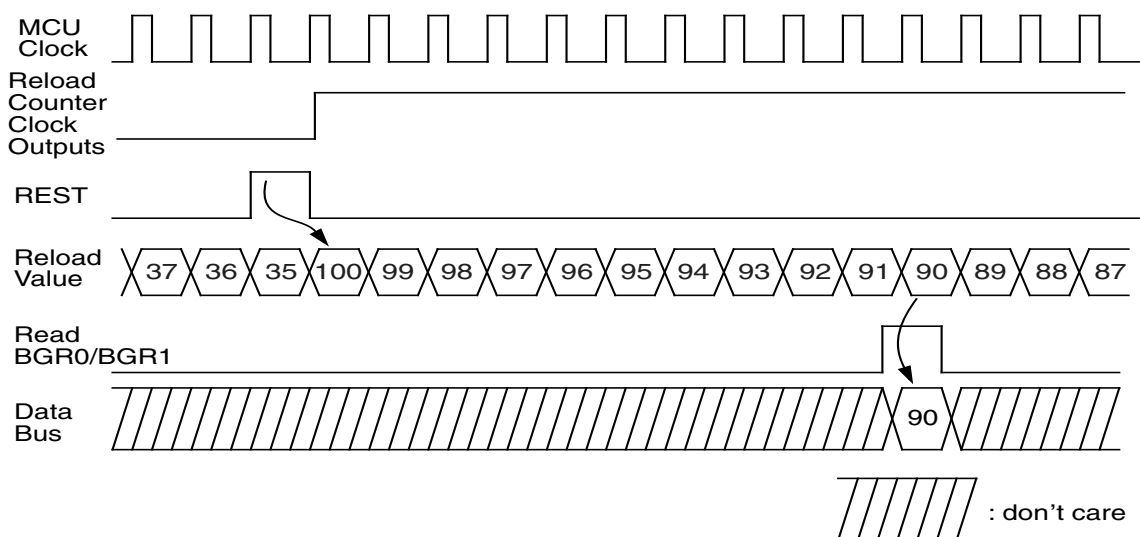
For a receive reload counter:

- Detection of the falling edge of the start bit in asynchronous mode

If the REST bit of the serial mode register (SMR) is set to "1", both reload counters are restarted at the next clock cycle. This feature is intended to use the transmission reload counter as a simple timer.

Figure 20-16. illustrates a possible usage of this feature (assume that the reload value is 100.)

Figure 20-16. Example of Using a Simple Timer by Restarting the Reload Timer



In this example the number of machine cycles (cyc) after restart 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 LIN-UART is reset by setting SMR:UPCL to "1", the reload counters will restart, too.

Automatic restart (reception reload counter only)

In asynchronous LIN-UART mode, if a falling edge of a start bit is detected, the reception reload counter is restarted. This is intended to synchronize the reception shift register to the reception data.

- Clearing reload counters

When a reset occurs, the reload values in the baud rate generator registers (BRG1, BGR0) and the reload counter are cleared to "00_H", and the reload counter are cleared to "00_H", and the reload counter halts.

Although the counter value is temporarily cleared to "00_H" by the LIN-UART reset (writing "1" to SMR:UPCL), the reload counter restarts since the reload value is retained. The counter value is not cleared to "00_H" by the restart setting (writing "1" to SMR:REST), and the reload counter restarts.

20.7 Operation of LIN-UART

LIN-UART operates in operation mode 0 for 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 LIN-UART

□ Operation modes

There are four LIN-UART operation modes: modes 0 to 3. As listed in Table 20-15., LIN-UART can select inter-CPU connection methods and data transfer method.

Table 20-15. Operation Mode of LIN-UART

Operation mode		Data length		Synchronization of mode	Length of stop bit	Data bit format
		Parity disabled	Parity enabled			
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits	LSB first MSB first
1	Multiprocessor mode	7 or 8 bits + 1*	-	Asynchronous		
2	Normal mode	8 bits		Synchronous	None, 1 or 2 bits	LSB first
3	LIN mode	8 bits	-	Asynchronous	1 bit	

-.Setting disabled

*: "+1" means the address/data selection bit (A/D) used for controlling communication in the multiprocessor mode.

Note:

Mode 1 operation is supported both for master or slave operation of LIN-UART in a master-slave connection system. In Mode 3, the LIN-UART function is locked to the communication format 8N1-Format, LSB first.

If the mode is changed, LIN-UART cuts off all possible transmission or reception and awaits then new action.

■ Inter-CPU Connection Methods

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, LIN-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 LIN-UART itself if operating as master.

■ Signal Mode

LIN-UART can treat data only in non-return to zero (NRZ) format.

■ Permission of Transmission and Reception

LIN-UART controls both transmission and reception using the operation enable bit for transmission (SCR: TXE) and reception (SCR: RXE). To disable transmission or reception, follow the procedure described below.

- ❑ If reception operation is disabled during reception, finish 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, wait until the transmission is completed before stopping the transmission operation.

20.7.1 Operation in Asynchronous Mode (Operation Modes 0 and 1)

When LIN-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

- ❑ Transmission and reception data format

Transmission/reception data always begins with the start bit ("L" level) followed by transmission/reception for a specified data bit length, and ends with at least one stop bit ("H" 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 bits 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 bits 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 transmission/reception frame is:

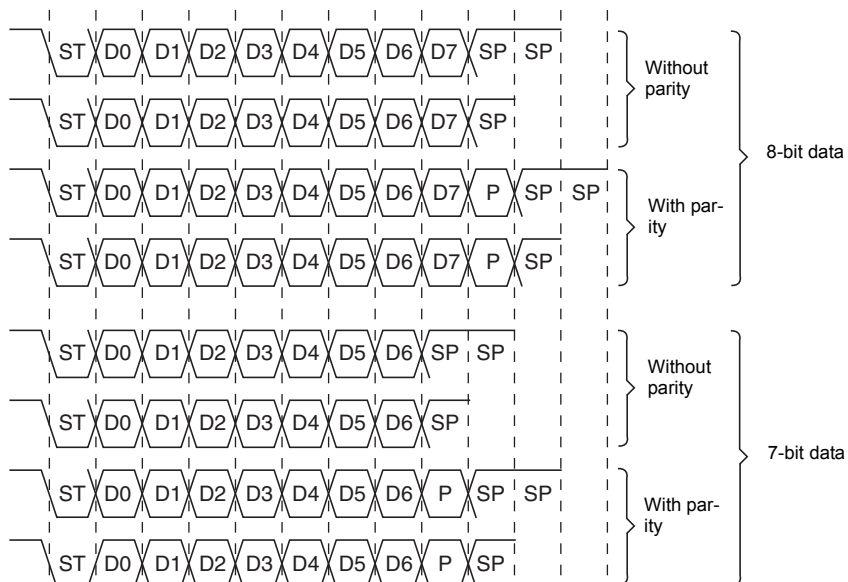
$$\text{Length} = 1 + d + p + s$$

(d = number of data bits [7 or 8], p = parity [0 or 1], s = number of stop bits [1 or 2])

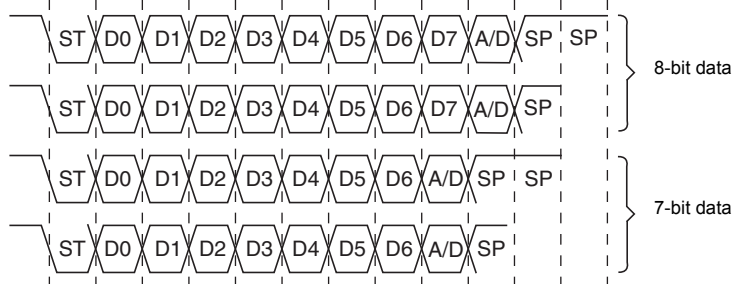
Figure 20-17. shows the data format in the asynchronous mode.

Figure 20-17. Transmission/Reception Data Format (Operation Modes 0 and 1)

[Operation mode 0]



[Operation mode 1]



ST: Start bit
 SP: Stop mode
 P: Parity bit
 A/D: Address/data bit

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).

□ Transmission operation

If the transmission data register empty flag bit (TDRE) 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.

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 data length is set to 7 bits (CL=0), the unused bit of the TDR is always the MSB, independently from the transfer direction selection in the BDS bit (LSB first or MSB first).

Note:

As the initial value of transmission data empty flag bit (SSR: TDRE) is "1" if the transmission interrupt is enabled (SSR: TIE=1), the interrupt occurs immediately.

□ 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 data format specified by the SCR. In case of errors, the corresponding error flags are set (SSR: PE, ORE, FRE). After the reception of the data frame, the data is transferred from the reception shift register to the reception data register (RDR) and the receive data register full flag bit (RDRF) of the SSR is set to "1". In this case, if the reception interrupt request is enabled (SSR: RIE=1), the reception interrupt request is occurred. When reading data after reception of one frame data, check the error flag state and read reception data from the RDR register if the reception is performed normally. If the reception error occurs, perform error processing. The data then has to be read by the CPU. By doing so, the RDRF flag of SSR is cleared to "0".

When the data length is set to 7 bits (CL=0), the unused bit of the TDR is always the MSB, independently from the transfer direction selection in the BDS bit (LSB first or MSB first).

Note:

Only when the RDRF flag bit of SSR is set to "1" and no errors have occurred (SSR: PE, ORE, FRE=0) the reception data register (RDR) contains valid data.

□ Used clock

Use the internal clock or external clock. Select the baud rate generator (SMR: EXT = 0 or 1, OTO = 0) for desired baud rate.

□ Stop bit

1- or 2-stop bit can be selected at the transmission. When 2-stop bit is selected, both stop bits are detected at the reception. When first stop bit is detected, the RDRF bit of SSR is "1". Then, when the start bit is not detected, the RBI bit of ECCR is set to "1", indicating no reception operation.

□ Error detection

In mode 0, the parity, overrun, and framing errors can be detected.

In mode 1, the overrun and framing errors can be detected, and the parity error cannot be detected.

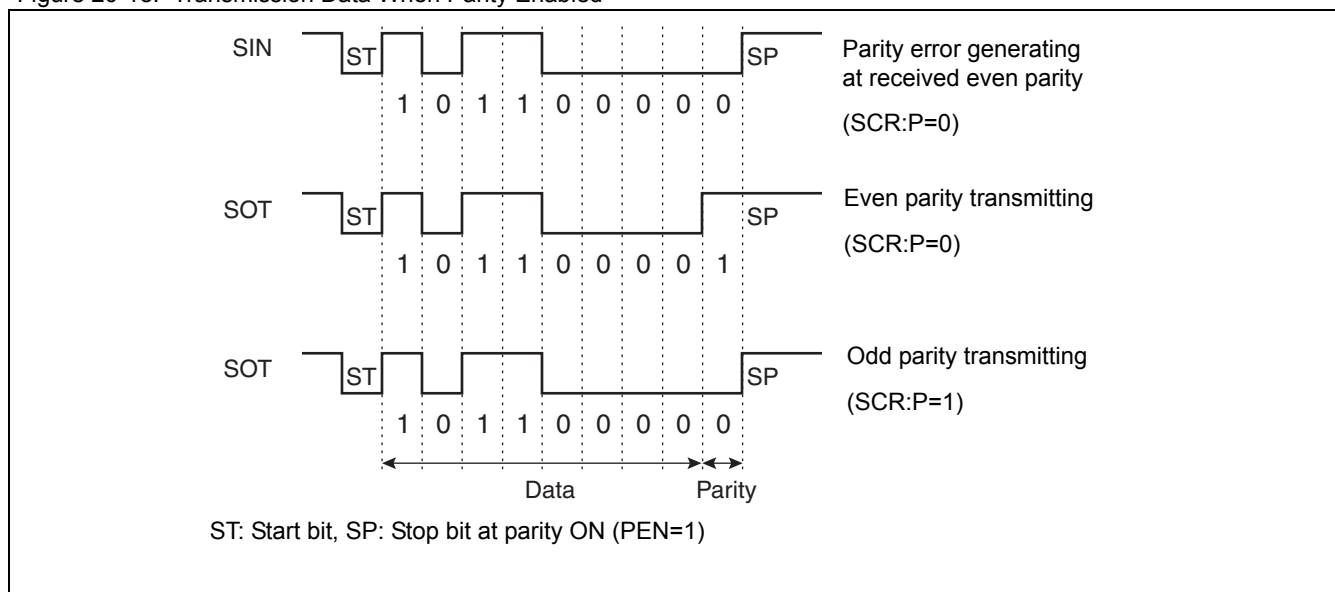
□ Parity

Parity can set to add (transmission) or detect (reception) the parity bit.

The parity enable bit (SCR: PEN) is used to specify whether there is parity or not, and parity selection bit (SCR: P) is selected the even/odd parity.

In operation mode 1, the parity cannot be used.

Figure 20-18. Transmission Data When Parity Enabled



□ Data signal type

The data signal type is NRZ data format.

□ Data transition method

The data bit transfer method can be selected by LSB first or MSB first.

20.7.2 Operation in Synchronous Mode (Operation Mode 2)

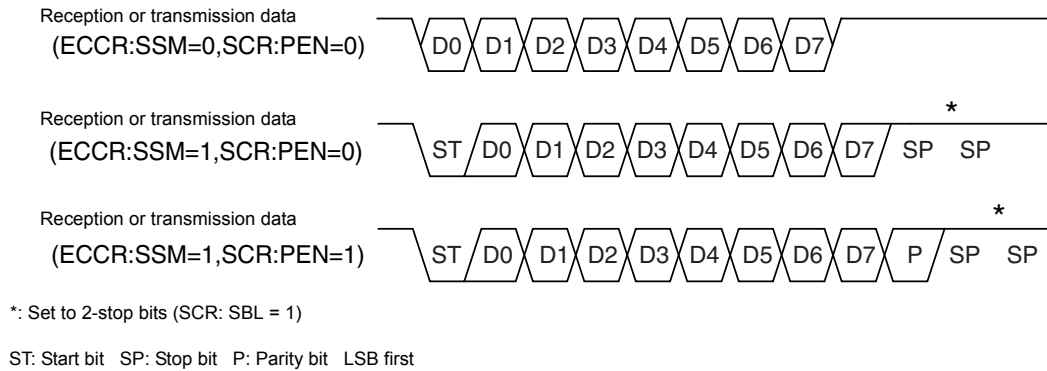
The clock synchronous transfer method is used for LIN-UART operation mode 2 (normal mode).

■ Operation in Synchronous Mode (Operation Mode 2)

□ Transmission/Reception

In the synchronous mode, 8-bit data is transmitted/received without start or stop bits if the SSM bit of the extended communication control register (ECCR) is "0". Also, when the start/stop bit is provided (ECCR: SSM = 1), presence or absence of the parity bit can be selected (SCR: PEN). The figure below illustrates the data format in the synchronous operation mode.

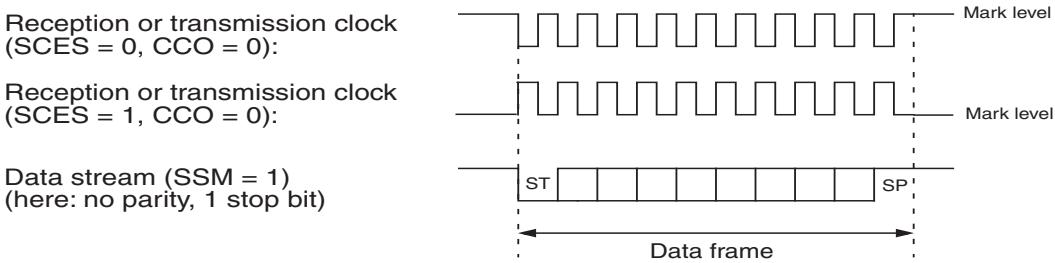
Figure 20-19. Transmission/Reception Data Format (Operation Mode 2)



□ Clock inversion function

If the SCES bit of the extended status/control register (ECCR) is set to "1", the serial clock is inverted. Therefore, in slave mode LIN-UART samples the data bits at the falling edge of the received serial clock. Note, that in master mode if SCES is set to "1", the clock signal's mark level is "0".

Figure 20-20. Transfer Data Format with Clock Inversion



□ Start/stop bits

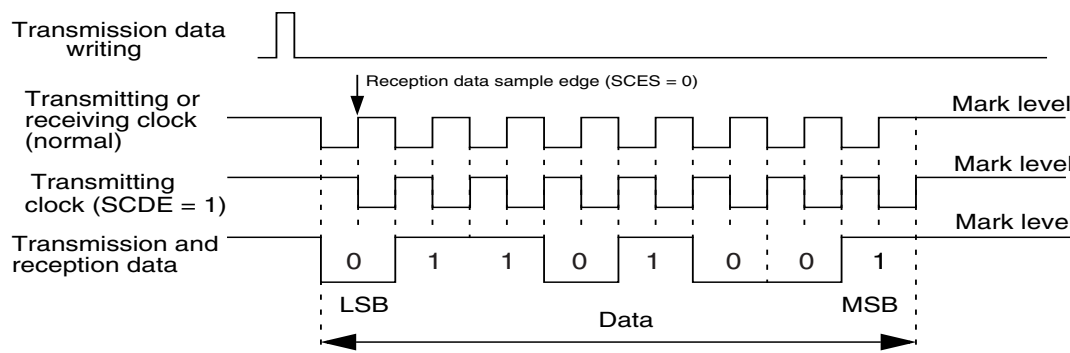
If the SSM bit of the extended communication control register (ECCR) is set to "1", the data format gets additional start and stop bits like in asynchronous mode.

□ Clock supply

In clock synchronous mode (normal), the number of clock cycles for the clock signal must be the same as the number of bits for the transmission and reception. If the start/stop bits are enabled, it must be matched the additional start/stop bits. If the MS bit of the ECCR register is "0" (master mode) and the SCDE bit of the SMR register is "1" (serial clock output enabled), the consistent clock cycles are generated automatically at transmission/reception. If the MS bit of the ECCR register is "1" (slave mode), or if the SCDE bit of SMR is "0" (serial clock output disabled), the clock for each bit of transmission/reception data must be supplied from outside. While there is no transmission/reception, the clock signal must be kept at "H" as the mark level

If the SCDE bit of the ECCR register is "1", a delayed transmit clock is output as shown in [Figure 20-21](#).. The operation is prepared for reception devices which use the rising or falling edge of the clock signal for the data sampling.

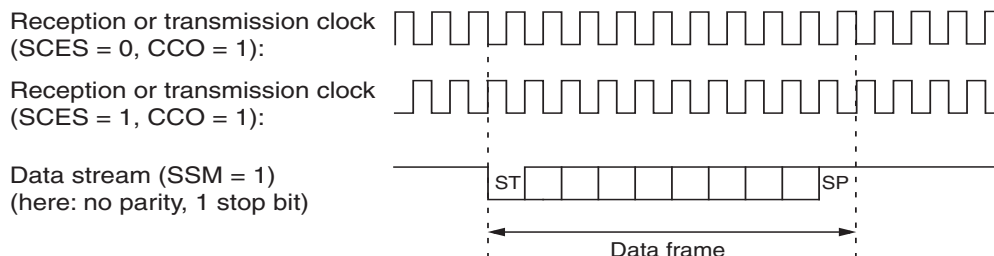
Figure 20-21. Delayed Transmitting Clock Signal (SCDE=1)



If the SCES bit of the ESCR register is "1", the UART clock signal is inverted. Receiving data is sampled at the falling edge of the clock.

In this case, the serial data must be valid value at the falling edge of the clock. If the CCO bit of ESCR register is "1" (master mode), the serial clock output of the SCKn pin is supplied continuously. In this mode, be sure to add the start/stop bits (SSM = 1) to identify the start and end of data frame. [Figure 20-22](#) shows the operation of this function.

Figure 20-22. Continuous Clock Supply in Mode 2



□ Error detection

If no start/stop bits are selected (ECCR: SSM = 0) only overrun errors are detected.

- Communication setting for synchronous mode

For communication of the synchronous mode, following settings have to be done:

Baud rate generator registers (BGR0/BGR1):

Set the desired value for the dedicated baud rate reload counter.

Serial mode 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 of these bits 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. Suspend transmission and reception.

-- when SSM=0:

PEN, P, SBL: don't care because parity bit and stop bit are not used

-- when SSM=1:

PEN: "1" if parity bit is added/detected, "0" if not

P: "1" for even parity, "0" 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" reception interrupts are enabled; "0" reception interrupts are disabled.

TIE: "1" transmission interrupts are enabled; "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 (extended function)

MS: "0" for master mode (LIN-UART generates the serial clock); "1" for slave mode (LIN-UART receives serial clock from the master device)

Note:

To start the communication, write data to TDR register. Only when receiving the data, disable the serial output (SMR: SOE = 0) and write the dummy data to TDR. By enabling the continuous clock and start/stop bits, in the bi-directional communication the same as the asynchronous mode is allowed.

20.7.3 Operation with LIN Function (Operation Mode 3)

LIN-UART can be used either as LIN-Master or LIN-Slave. Setting the LIN-UART to mode 3 configures the data format to 8N1-LSB-first format.

■ Operation in Asynchronous LIN Mode

□ LIN-UART as LIN master

In LIN master mode, the master determines the baud rate of the whole bus, therefore slaves devices have to synchronize to the master. Therefore, the desired baud rate remains fixed in master operation after initialization.

Writing "1" to the LBR bit in the extended communication control register (ECCR) outputs 13 to 16 bits at the "L" level from the SOTn pin. These bits are the LIN Synch break indicating the beginning of a LIN message. The TDRE flag bit in the serial status register (SSR) is set to "0". After the break, it is set to "1" (initial value), if no valid data is contained in the transmit data register (TDR). If the TIE bit in SSR is "1" at this time, a transmit interrupt is output.

The length of the LIN break to be sent can be determined by the LBL1/LBL0 bits of the ESCR as follows:

Table 20-16. LIN Break Length

LBL0	LBL1	Break length
0	0	13 bits
1	0	14 bits
0	1	15 bits
1	1	16 bits

The Synch Field is sent as byte data of 55_H after the LIN break. To prevent a transmission interrupt, the 55_H can be written to the TDR just after writing "1" to the LBR bit, although the TDRE flag is "0".

□ LIN-UART as LIN slave

In LIN slave mode, LIN-UART has to synchronize to the master's baud rate. If reception is disabled (RXE = 0) but LIN break interrupt is enabled (LBIE = 1) LIN-UART will generate a reception interrupt, and indicates the LBD flag of the ESCR is set to "1". Writing "0" to this bit clears the reception interrupt request flag. For the calculation of the baud rate, the UART0 operation is explained as an example. When UART0 detects first falling edge of synch field, set the internal signal inputted to the input capture (ICU0) to "H" and start ICU0. This internal signal is set to "L" at fifth falling edge, ICU0 must be set to the LIN mode (ICE01). Also, the ICU0 interrupt must be set to enable and to detect both edges (ICS01).

The time when the ICU0 input signal is "1" is the value in which eight baud rates are multiplied. Therefore, baud rate setting value is summarized as follows:

without free-run timer overflow : $BGR \text{ value} = \{(b-a) \times Fe / (8 \times \phi)\} - 1$

with free-run timer overflow : $BGR \text{ value} = \{(\max + b-a) \times Fe / (8 \times \phi)\} - 1$

max: Maximum value of free-run timer

a : ICU data register value after the 1st interrupt

b : ICU data register value after the 2nd interrupt

f : Machine clock frequency (MHz)

Fe : External clock frequency (MHz)

Calculation based on the internal baud rate generator in use (EXT=0),
and $Fe = f$

Note:

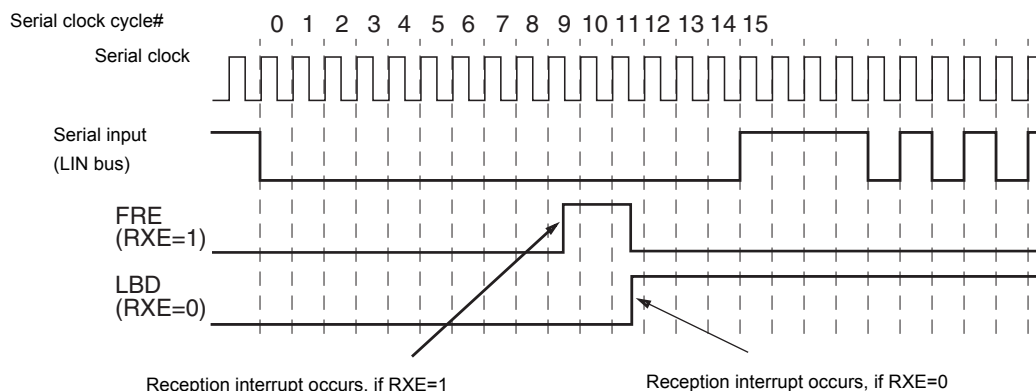
As shown in the LIN slave mode, when the BGR value newly calculated by synch field generates $\pm 15\%$ or more baud rate error, do not set the baud rate.

For the correspondence between other LIN-UARTs and ICUs, see Sections "13.5 Explanation of Operation of 16-bit Free-run Timer" and "13.6 Explanation of Operation of Input Capture".

□ LIN Synch Break Detection Interrupt and Flags

If a LIN Synch break is detected in the slave mode, the LIN break detected flag (LBD) of the ESCR is set to "1". This causes an interrupt, if the LIN break interrupt enable bit (LBIE) is set to "1".

Figure 20-23. LIN Synch Break Detection and Flag Set Timing



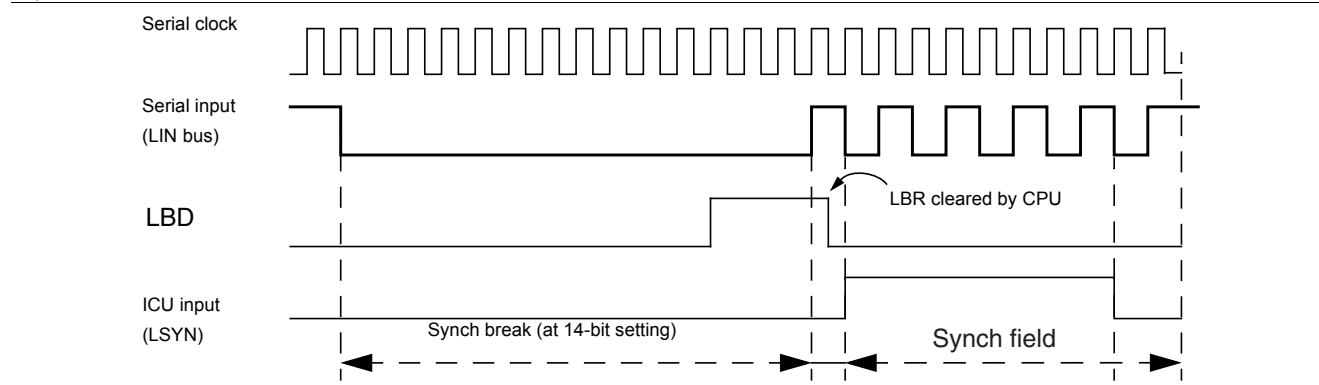
The figure above demonstrates the LIN synch break detection and flag set timing.

The data framing error flag bit (FRE) 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.

LIN synch break detection is only supported in operation mode 3.

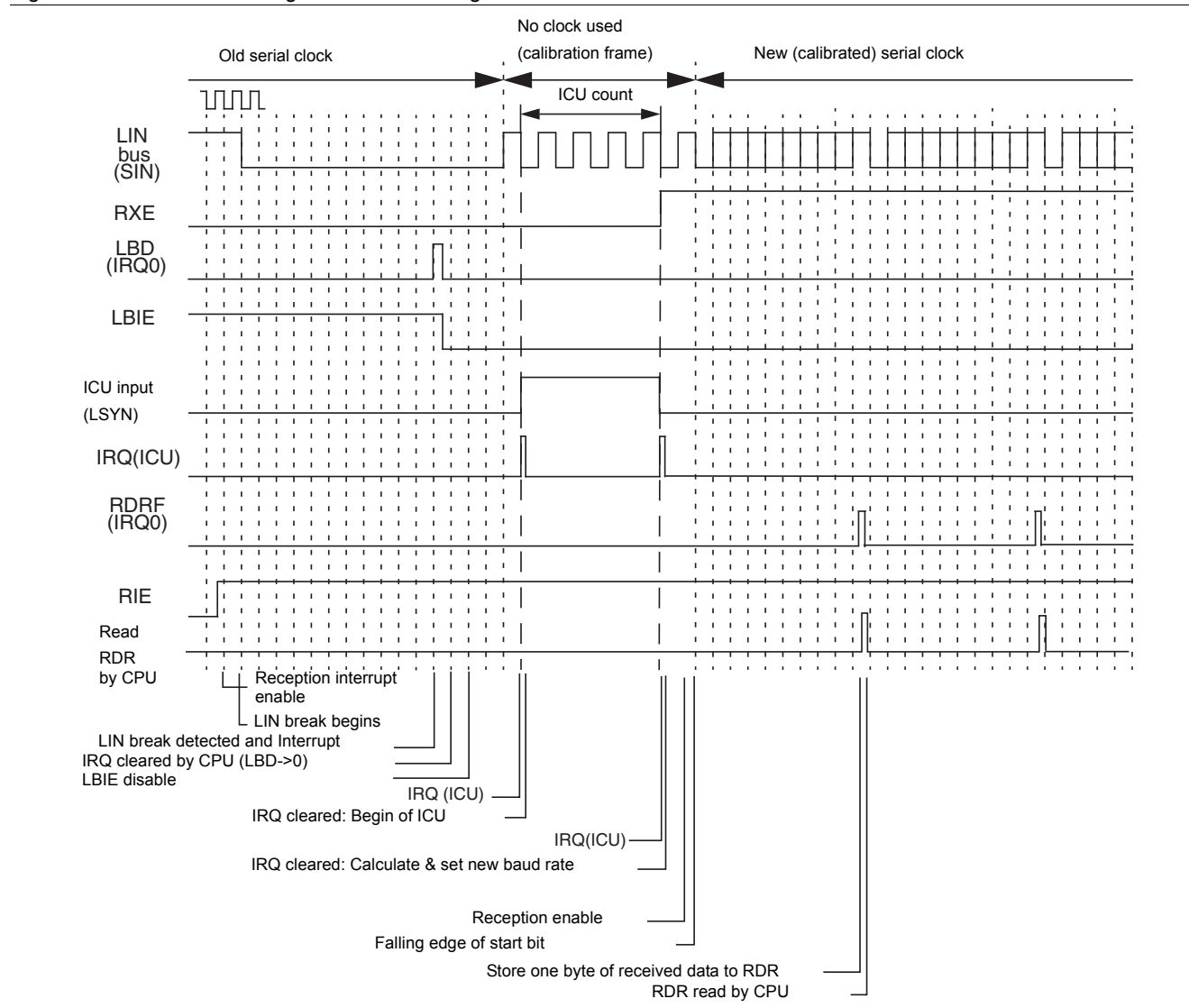
Figure 20-24. shows a typical start of a LIN message and the operation of the LIN-UART.

Figure 20-24. LIN-UART Operation as Slave in LIN Mode



□ LIN bus timing

Figure 20-25. LIN Bus Timing and LIN-UART Signals



20.7.4 Direct Access to Serial Pins

LIN-UART allows the user to directly access to the transmission pin (SOTn) or the reception pin (SINn).

■ LIN-UART Pin Direct Access

The LIN-UART provides the ability for the programmer to access directly to serial input or output pin. The status of the serial input pin (SINn) can be read by using the serial I/O pin direct access bit (ESCR:SIOP). If direct write to the serial output pin (SOTn) is allowed (ESCR: SOPE = 1) when the serial output is enabled (SMR: SOE = 1) after "0" or "1" is written to the SIOP bit of the ESCR register, the SOTn value can be set arbitrarily.

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:

- Direct access is enabled only when the transmission is not ongoing (transmission shift register is empty).
- Write a value to the SIOP bit of ESCR register before enabling the transmission (SMR: SOE = 1). This prevents the signal of the unexpected level from being outputted because the SIOP bit retains the previous value.
- During a read-modify-write (RMW) instruction 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 20-26](#) are required to operate LIN-UART in normal mode (operation mode 0 or 2).

Figure 20-26. Settings for LIN-UART Operation Mode 0 and 2

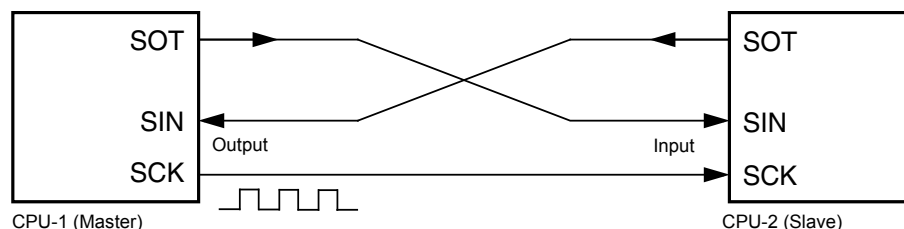
		bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SCRn, SMRn		PEN	P	SBL	CL	A/D	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 0	→	○	○	○	○	x	0	○	○	0	0	0	○	0	0	0	○
Mode 2	→	□	□	□	+	x	0	○	○	1	0	○	○	0	0	○	○
SSRn, TDRn/RDRn		PE	ORE	FRE	RDRF	TDRE	BDS	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	○	○	x	+	0	0	x	x	x	0	○	○
Mode 2	→	x	x	x	x	○	○	□	○	0	x	○	○	○	0	□	□

○ : Used bit
 x : Unused bit
 1 : Set to "1"
 0 : Set to "0"
 □ : Used if SSM = 1 (Synchronous start-/stop-bit mode)
 + : Bit automatically set to correct value
 n = 0, 1, 2

□ Inter-CPU connection

As shown in [Figure 20-27](#)., interconnect two CPUs in bidirectional communication.

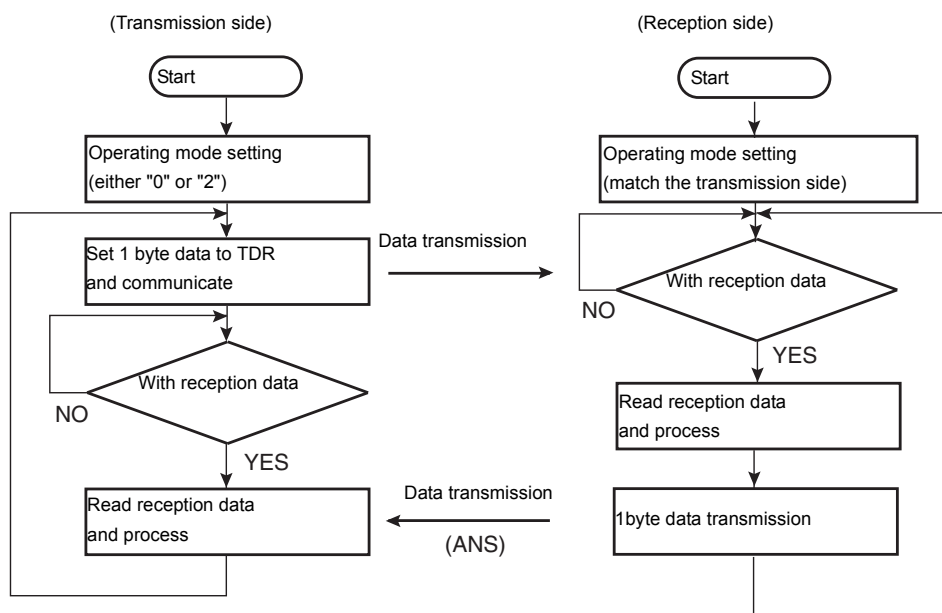
Figure 20-27. Connection Example of LIN-UART Mode 2 Bidirectional Communication



Communication procedure

Communication starts at arbitrary timing from the transmission side when the transmission data is provided. When the transmission data is received at the reception side, ANS (per one byte in example) is returned periodically. Figure 20-28. shows an example of the bi-directional communication flowchart.

Figure 20-28. Example of Bi-directional Communication Flowchart



20.7.6 Master-Slave Communication Function (Multiprocessor Mode)

LIN-UART communication with multiple CPUs connected in master-slave mode is available for both master or slave systems in the operation mode 1.

Master-slave Communication Function

The settings shown in Figure 20-29. are required to operate LIN-UART in multiprocessor mode (operation mode 1).

Figure 20-29. Settings for LIN-UART Operation Mode 1

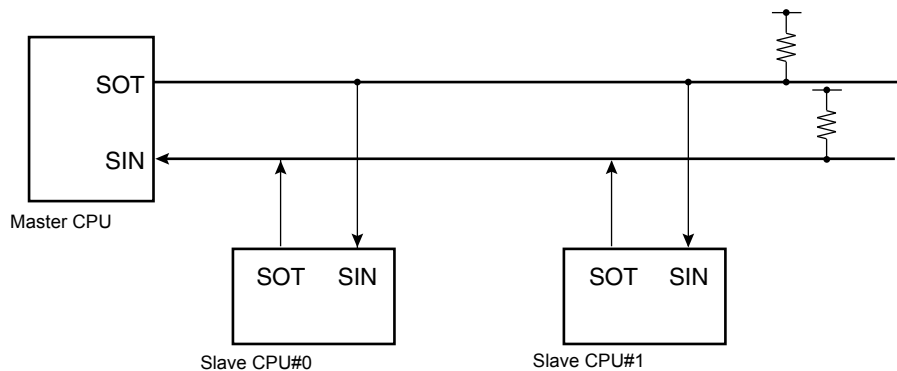
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SCRn, SMRn	PEN	P	SBL	CL	A/D	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 1	+	x	⊙	⊙	⊙	0	⊙	⊙	0	1	0	⊙	0	0	0	⊙
SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)							
Mode 1	x	⊙	⊙	⊙	⊙	⊙	⊙	⊙								
ESCRn, ECCRn	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES	/	LBR	MS	SCDE	SSM	/	RBI	TBI
Mode 1	x	x	x	x	⊙	⊙	x	+	0	x	x	x	x	0	⊙	⊙

⊙ : Used bit
 x : Unused bit
 1 : Set to "1"
 0 : Set to "0"
 + : Bit automatically set to correct value
 n = 0, 1, 2

□ Inter-CPU connection

As shown in [Figure 20-30.](#), a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. LIN-UART can be used for the master or slave CPU.

Figure 20-30. Connection Example of LIN-UART Master-slave Communication



□ Function selection

Select the operation mode and data transfer mode for master-slave communication as shown in [Table 20-17](#).

Table 20-17. Selection of the Master-slave Communication Function

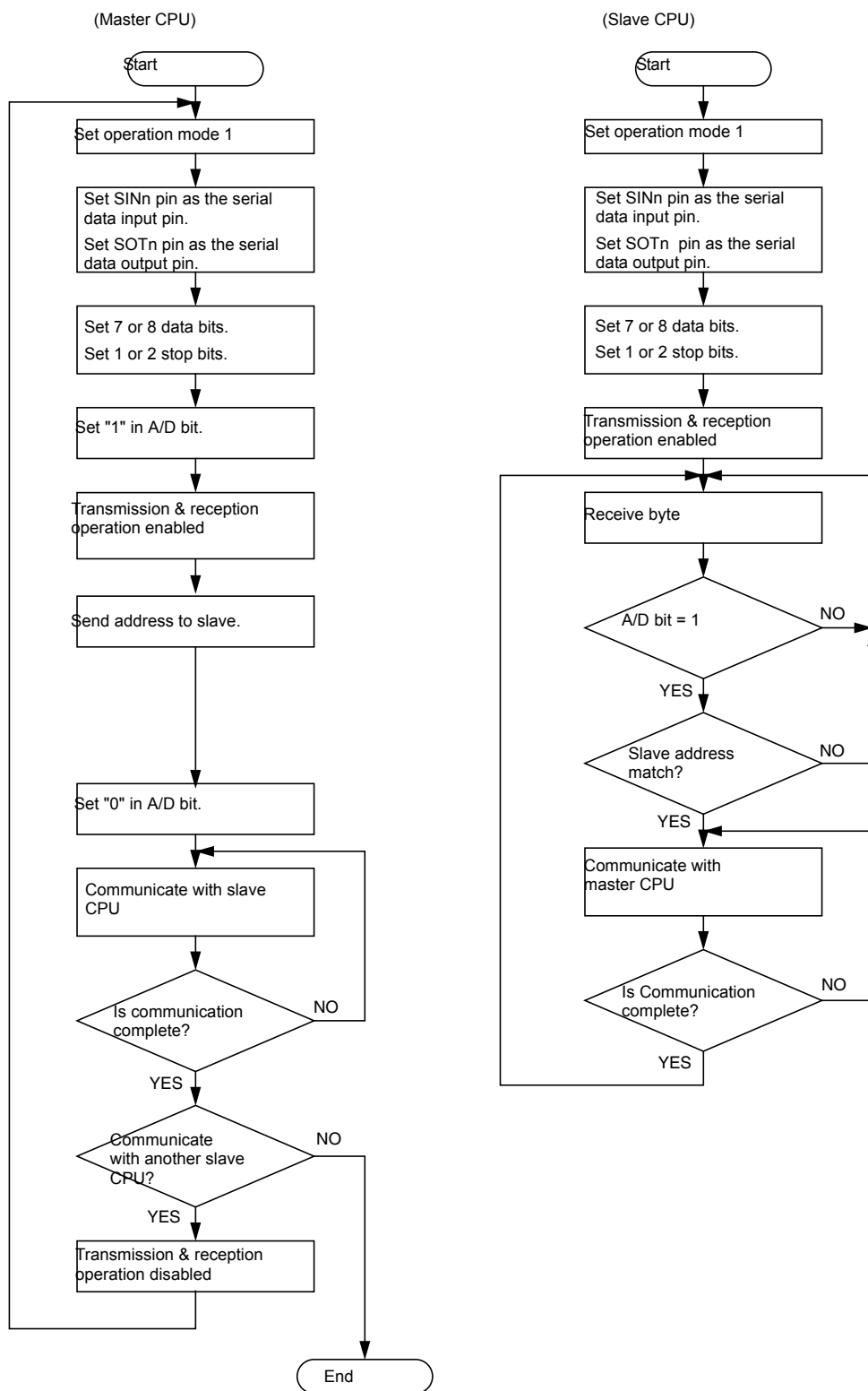
	Operation mode		Data	Parity	Synchroni- zation method	Stop bit	Bit direction
	Master CPU	Slave CPU					
Address trans- mission and reception	Mode 1 (transmit/receive A/D bit)	Mode 1 (transmit/receive A/D bit)	A/D=1 + 7- or 8-bit address	None	Asynchro- nous	1 bit or 2 bits	LSB first or MSB first
Data transmis- sion and recep- tion			A/D=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 20-31](#). shows a flowchart of master-slave communication (multiprocessor mode).

Figure 20-31. Master-slave Communication Flowchart



20.7.7 LIN Communication Function

LIN-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 LIN-UART in LIN communication mode (operation mode 3).

Figure 20-32. Settings for LIN-UART in Operation Mode 3 (LIN)

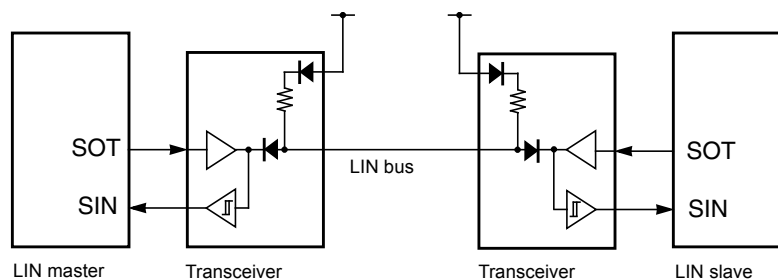
SCRn, SMRn	PEN	P	SBL	CL	A/D	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 3	+	x	+	+	x	0	⊙	⊙	1	1	0	⊙	0	0	0	⊙
SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)							
Mode 3	x	⊙	⊙	⊙	⊙	+	⊙	⊙								
ESCRx, ECCRx	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES	—	LBR	MS	SCDE	SSM	—	RBI	TBI
Mode 3	⊙	⊙	⊙	⊙	⊙	⊙	x	+	0	⊙	x	x	x	0	⊙	⊙

⊙: Used bit
 x: Unused bit
 1: Set to "1"
 0: Set to "0"
 +: Bit automatically set to correct value
 n = 0, 1, 2

□ LIN device connection

The figure below shows a communication system of one LIN-Master device and a LIN-Slave device. LIN-UART can operate both as LIN-Master or LIN-Slave.

Figure 20-33. Connection Example of a LIN-Bus System

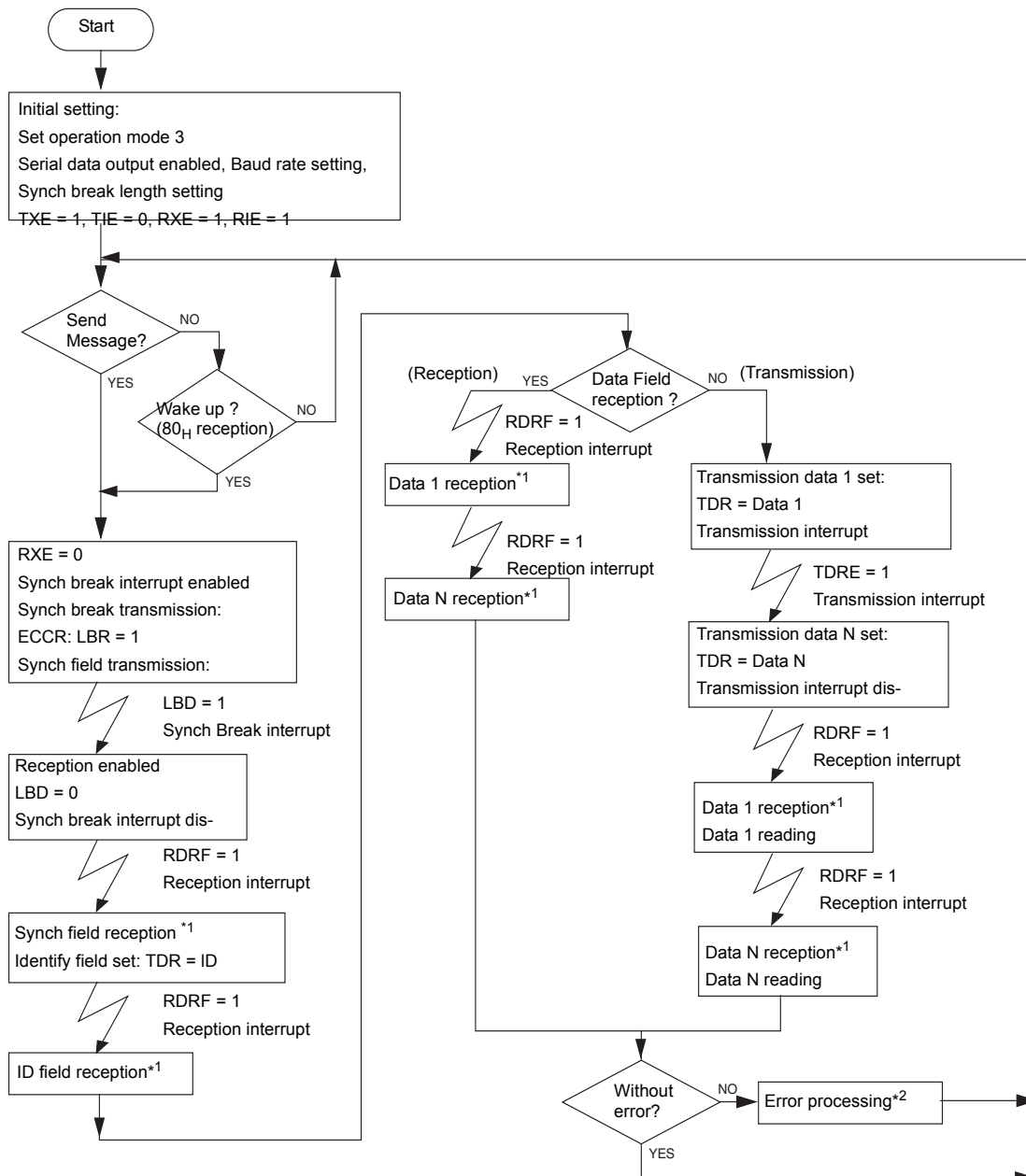


20.7.8 Sample Flowcharts for LIN-UART in LIN communication (Operation Mode 3)

This section contains sample flowcharts for LIN-UART in LIN communication.

■ LIN Master Device

Figure 20-34. LIN Master Flowchart



*1: If an error occurs, perform the error processing.

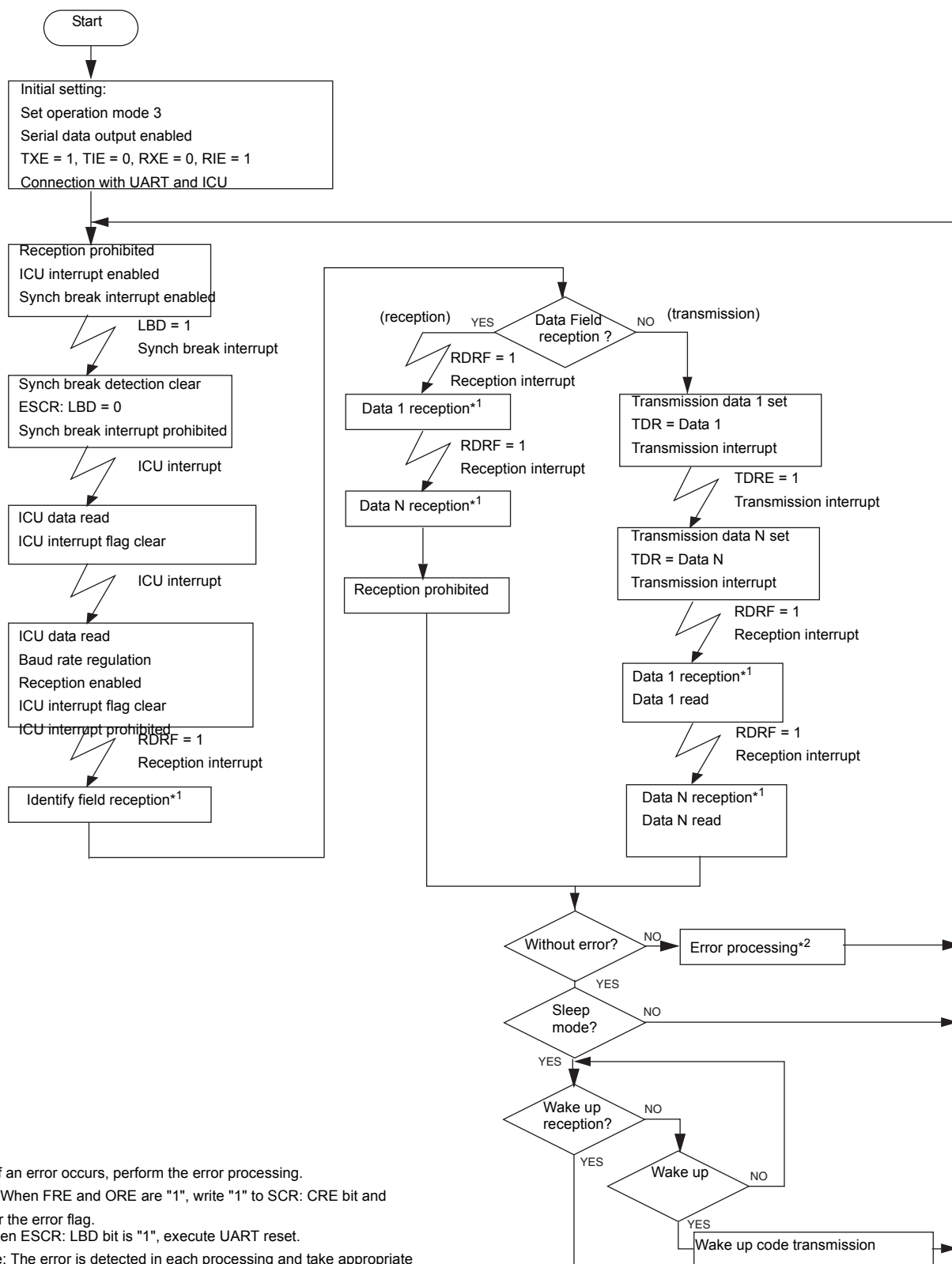
*2: •When FRE and ORE are "1", write "1" to SCR: CRE bit and clear the error flag.

•When ESCR: LBD bit is "1", execute UART reset.

Note: The error is detected in each processing and take appropriate measures.

■ LIN Slave Device

Figure 20-35. LIN Slave Flowchart



20.8 Notes on Using LIN-UART

Notes on using LIN-UART are given below.

■ Notes on Using LIN-UART

□ Enabling operations

In LIN-UART, the serial 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 transfer.

□ Communication mode setting

Set the communication mode while the LIN-UART 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.

□ Changing operation settings

Please reset LIN-UART after changing operation settings. Particularly if (for example) start-/stop-bits added to or removed from the data format.

If settings in the LIN-UART serial mode register (SMR) are desired, it is not useful to set the UPCL bit to "1" at the same time to reset LIN-UART. The correct operation settings are not guaranteed in this case. Thus please set the bits of the SMR and then reset them again plus the UPCL bit.

□ Using LIN function

The LIN features are available in mode 3, but using mode 3 sets the UART data format automatically to LIN format (8-bit data, no parity, 1 stop bit, LSB first). Note that the length of the synch break for transmission is variable but for reception it is fixed 11-bit time.

□ LIN slave settings

Set the baud rate before receiving the first LIN synch break for the slave operation. This is needed to detect the minimum of 13-bit time of a LIN synch break surely.

□ Software compatibility

Although this LIN-UART is similar to other FJ-UART in other microcontrollers, it is not 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 is not available in synchronous mode 2, when SSM is 0.

□ A/D bit (serial control register (SCR): address/data type select bit)

The A/D bit is used to select the address/data for transmission in write operation, and to read the A/D bit received last in read operation. Internally, the A/D bit values for transmission and reception are stored in separate registers.

The transmit A/D bit value is read when read-modify-write (RMW) instructions are used. Otherwise, the received A/D data is read.

At transmission, when the TDRE bit changes from "0" to "1", the transmit A/D bit is also loaded to the transmit shift register along with the data in the transmit data register (TDR). Therefore, set the transmit A/D bit before writing to the transmit data register (TDR).

□ Software reset of LIN-UART

Perform the LIN-UART software reset (SMR: UPCL=1), when the TXE bit of the SCR register is "0".

□ Synch Break detection

In mode 3 (LIN mode), the LBD bit in the ESCR register is set to "1" (Synch Break detection) if the serial input signal is kept at "0" for more than equal to 11-bit time. Then the LIN-UART waits for the following synch field to be received. If the LIN-UART

is set into this state for other reasons than the synch break, it recognizes that synch break is inputted ($LBD = 1$) and waits for synch field.

In this case, execute the LIN-UART reset (SMR: $UPCL = 1$).

21. Can Controller



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

- 21.1 Features of CAN Controller
- 21.2 Block Diagram of CAN Controller
- 21.3 List of Registers
- 21.4 Classifying CAN Controller Registers
- 21.5 Transmission of CAN Controller
- 21.6 Reception of CAN Controller
- 21.7 Reception Flowchart of CAN Controller
- 21.8 How to Use CAN Controller
- 21.9 Procedure for Transmission by Message Buffer (x)
- 21.10 Procedure for Reception by Message Buffer (x)
- 21.11 Setting Configuration of Multi-level Message Buffer
- 21.12 CAN Direct Mode Register
- 21.13 Notes on Using CAN Controller

21.1 Features of CAN Controller

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

- Conforms to CAN Specification Version 2.0 Part A and Part 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

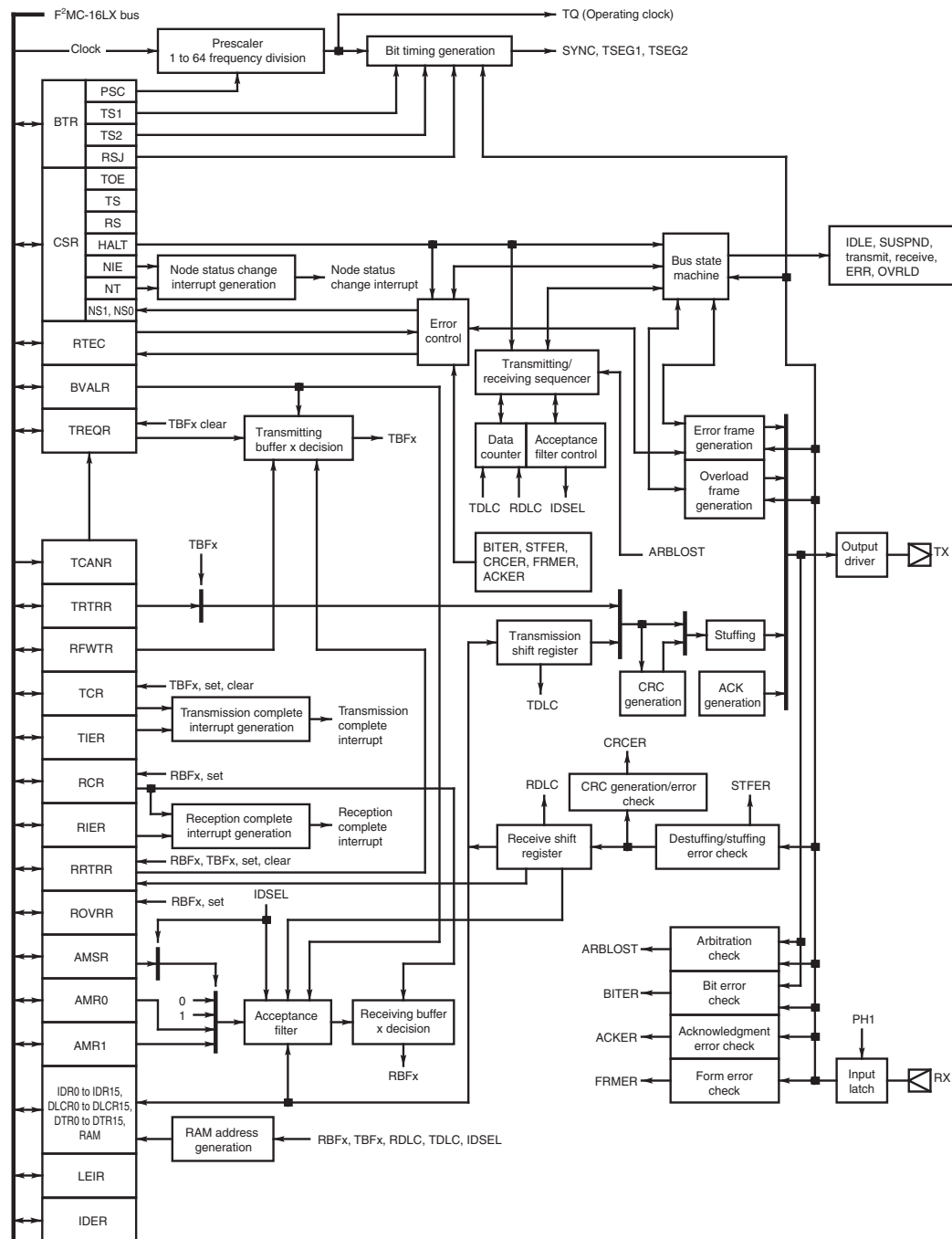
- Bit rate programmable from 10 kbps to 1 Mbps (Minimum 8 MHz machine clock is required at using 1 Mbps. Also, maximum 16 MHz machine clock is required at using 10 kbps.)

21.2 Block Diagram of CAN Controller

Figure 21-1. shows a block diagram of the CAN controller.

■ Block Diagram of CAN Controller

Figure 21-1. Block Diagram of CAN Controller



21.3 List of Registers

The following tables list the registers.

■ List of Overall Control Registers

Table 21-1. List of Overall Control Registers (Sheet 1 of 2)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
000080 _H	Message buffer valid register	BVALR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
000081 _H				
000082 _H	Transmit request register	TREQR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
000083 _H				
000084 _H	Transmit cancel register	TCANR	W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
000085 _H				
000086 _H	Transmit complete register	TCR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
000087 _H				
000088 _H	Receive complete register	RCR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
000089 _H				
00008A _H	Remote request receiving register	RRTRR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
00008B _H				
00008C _H	Receive overrun register	ROVRR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
00008D _H				
00008E _H	Receive interrupt enable register	RIER	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
00008F _H				
007D00 _H	Control status register	CSR	R/W	0 0 XXX 0 0 0 0 XXXX0 X1 _B
007D01 _H				
007D02 _H	Last event indicator register	LEIR	R/W	XXXXXXXX 0 0 0 X0 0 0 0 _B
007D03 _H				
007D04 _H	Receive/transmit error counter	RTEC	R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
007D05 _H				
007D06 _H	Bit timing register	BTR	R/W	X1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 _B
007D07 _H				
007D08 _H	IDE register	IDER	R/W	XXXXXXXX XXXXXXXX _B
007D09 _H				
007D0A _H	Transmit RTR register	TRTRR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
007D0B _H				
007D0C _H	Remote frame receive waiting register	RFWTR	R/W	XXXXXXXX XXXXXXXX _B
007D0D _H				
007D0E _H	Transmit interrupt enable register	TIER	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 _B
007D0F _H				
007D10 _H	Acceptance mask select register	AMSR	R/W	XXXXXXXX XXXXXXXX _B
007D11 _H				XXXXXXXX XXXXXXXX _B
007D12 _H				
007D13 _H				

Table 21-1. List of Overall Control Registers (Sheet 2 of 2)

Address CAN1	Register	Abbreviation	Access	Initial Value
007D14 _H	Acceptance mask register 0	AMR0	R/W	XXXXXXXX XXXXXXXX _B
007D15 _H				XXXXXXXX XXXXXXXX _B
007D16 _H				
007D17 _H				
007D18 _H	Acceptance mask register 1	AMR1	R/W	XXXXXXXX XXXXXXXX _B
007D19 _H				XXXXXXXX XXXXXXXX _B
007D1A _H				
007D1B _H				

■ List of Message Buffers (ID Registers)

Table 21-2. List of Message Buffers (ID Registers) (Sheet 1 of 2)

Address CAN1	Register	Abbreviation	Access	Initial Value
007C00 _H to 007C1F _H	General-purpose RAM	—	R/W	XXXXXXXX _B to XXXXXXXX _B
007C20 _H 007C21 _H 007C22 _H 007C23 _H	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C24 _H 007C25 _H 007C26 _H 007C27 _H	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C28 _H 007C29 _H 007C2A _H 007C2B _H	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C2C _H 007C2D _H 007C2E _H 007C2F _H	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C30 _H 007C31 _H 007C32 _H 007C33 _H	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C34 _H 007C35 _H 007C36 _H 007C37 _H	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C38 _H 007C39 _H 007C3A _H 007C3B _H	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C3C _H 007C3D _H 007C3E _H 007C3F _H	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B
007C40 _H 007C41 _H 007C42 _H 007C43 _H	ID register 8	IDR8	R/W	XXXXXXXX XXXXXXXX _B XXXXXXXX XXXXXXXX _B

Table 21-2. List of Message Buffers (ID Registers) (Sheet 2 of 2)

Address CAN1	Register	Abbreviation	Access	Initial Value
007C44 _H	ID register 9	IDR9	R/W	XXXXXXXX XXXXXXXX _B
007C45 _H				XXXXXXXX XXXXXXXX _B
007C46 _H				
007C47 _H				
007C48 _H	ID register 10	IDR10	R/W	XXXXXXXX XXXXXXXX _B
007C49 _H				XXXXXXXX XXXXXXXX _B
007C4A _H				
007C4B _H				
007C4C _H	ID register 11	IDR11	R/W	XXXXXXXX XXXXXXXX _B
007C4D _H				XXXXXXXX XXXXXXXX _B
007C4E _H				
007C4F _H				
007C50 _H	ID register 12	IDR12	R/W	XXXXXXXX XXXXXXXX _B
007C51 _H				XXXXXXXX XXXXXXXX _B
007C52 _H				
007C53 _H				
007C54 _H	ID register 13	IDR13	R/W	XXXXXXXX XXXXXXXX _B
007C55 _H				XXXXXXXX XXXXXXXX _B
007C56 _H				
007C57 _H				
007C58 _H	ID register 14	IDR14	R/W	XXXXXXXX XXXXXXXX _B
007C59 _H				XXXXXXXX XXXXXXXX _B
007C5A _H				
007C5B _H				
007C5C _H	ID register 15	IDR15	R/W	XXXXXXXX XXXXXXXX _B
007C5D _H				XXXXXXXX XXXXXXXX _B
007C5E _H				
007C5F _H				

■ List of Message Buffers (DLC Registers)

Table 21-3. List of Message Buffers (DLC Registers)

Address CAN1	Register	Abbreviation	Access	Initial Value
007C60 _H	DLC register 0	DLCR0	R/W	XXXXXXXX _B
007C61 _H				
007C62 _H	DLC register 1	DLCR1	R/W	XXXXXXXX _B
007C63 _H				
007C64 _H	DLC register 2	DLCR2	R/W	XXXXXXXX _B
007C65 _H				
007C66 _H	DLC register 3	DLCR3	R/W	XXXXXXXX _B
007C67 _H				
007C68 _H	DLC register 4	DLCR4	R/W	XXXXXXXX _B
007C69 _H				

Table 21-3. List of Message Buffers (DLC Registers)

Address CAN1	Register	Abbreviation	Access	Initial Value
007C6A _H 007C6B _H	DLC register 5	DLCR5	R/W	XXXXXXXX _B
007C6C _H 007C6D _H	DLC register 6	DLCR6	R/W	XXXXXXXX _B
007C6E _H 007C6F _H	DLC register 7	DLCR7	R/W	XXXXXXXX _B
007C70 _H 007C71 _H	DLC register 8	DLCR8	R/W	XXXXXXXX _B
007C72 _H 007C73 _H	DLC register 9	DLCR9	R/W	XXXXXXXX _B
007C74 _H 007C75 _H	DLC register 10	DLCR10	R/W	XXXXXXXX _B
007C76 _H 007C77 _H	DLC register 11	DLCR11	R/W	XXXXXXXX _B
007C78 _H 007C79 _H	DLC register 12	DLCR12	R/W	XXXXXXXX _B
007C7A _H 007C7B _H	DLC register 13	DLCR13	R/W	XXXXXXXX _B
007C7C _H 007C7D _H	DLC register 14	DLCR14	R/W	XXXXXXXX _B
007C7E _H 007C7F _H	DLC register 15	DLCR15	R/W	XXXXXXXX _B

■ List of Message Buffers (Data Registers)

Table 21-4. List of Message Buffers (Data Registers)

Address CAN1	Register	Abbreviation	Access	Initial Value
007C80 _H to 007C87 _H	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX _B to XXXXXXXX _B
007C88 _H to 007C8F _H	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX _B to XXXXXXXX _B
007C90 _H to 007C97 _H	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX _B to XXXXXXXX _B
007C98 _H to 007C9F _H	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX _B to XXXXXXXX _B
007CA0 _H to 007CA7 _H	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX _B to XXXXXXXX _B
007CA8 _H to 007CAF _H	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX _B to XXXXXXXX _B
007CB0 _H to 007CB7 _H	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX _B to XXXXXXXX _B
007CB8 _H to 007CBF _H	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX _B to XXXXXXXX _B

Table 21-4. List of Message Buffers (Data Registers)

Address CAN1	Register	Abbreviation	Access	Initial Value
007CC0 _H to 007CC7 _H	Data register 8 (8 bytes)	DTR8	R/W	XXXXXXXX _B to XXXXXXXX _B
007CC8 _H to 007CCF _H	Data register 9 (8 bytes)	DTR9	R/W	XXXXXXXX _B to XXXXXXXX _B
007CD0 _H to 007CD7 _H	Data register 10 (8 bytes)	DTR10	R/W	XXXXXXXX _B to XXXXXXXX _B
007CD8 _H to 007CDF _H	Data register 11 (8 bytes)	DTR11	R/W	XXXXXXXX _B to XXXXXXXX _B
007CE0 _H to 007CE7 _H	Data register 12 (8 bytes)	DTR12	R/W	XXXXXXXX _B to XXXXXXXX _B
007CE8 _H to 007CEF _H	Data register 13 (8 bytes)	DTR13	R/W	XXXXXXXX _B to XXXXXXXX _B
007CF0 _H to 007CF7 _H	Data register 14 (8 bytes)	DTR14	R/W	XXXXXXXX _B to XXXXXXXX _B
007CF8 _H to 007CFF _H	Data register 15 (8 bytes)	DTR15	R/W	XXXXXXXX _B to XXXXXXXX _B

21.4 Classifying CAN Controller Registers

There are 3 types of CAN controller registers;

- Overall control registers
- Message buffer control registers
- Message buffers

■ Overall Control Registers

The overall control registers are the following 4 registers;

- ❑ Control status register (CSR)
- ❑ Last event indicator register (LEIR)
- ❑ Receive and transmit error counters (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 3 registers;

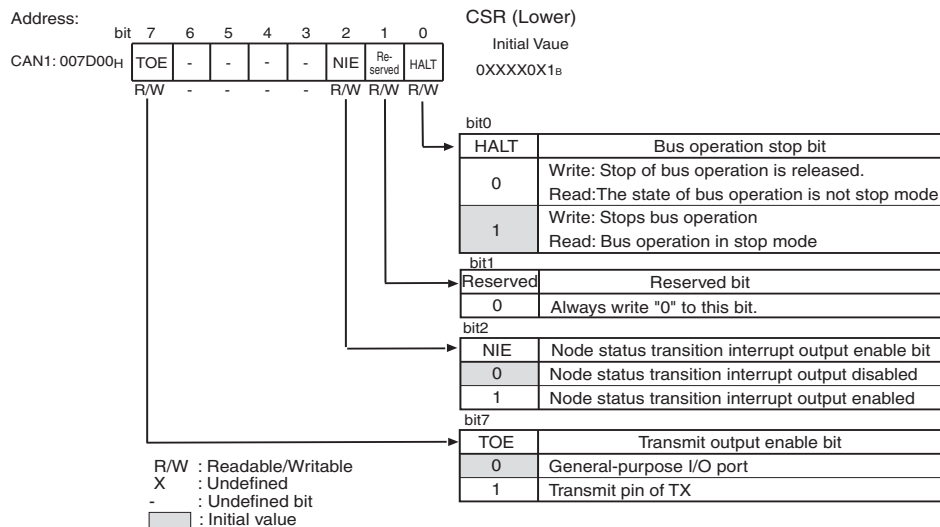
- ❑ ID register x (x = 0 to 15) (IDRx)
- ❑ DLC register x (x = 0 to 15) (DLCRx)
- ❑ Data register x (x = 0 to 15) (DTRx)

21.4.1 Configuration of Control Status Register (CSR)

This register indicates bus operation, node status, transmit output enable and transmit/receive status. The lower 8-bit with the control status register (CSR) is prohibited from executing any bit manipulation instructions (Read-Modify-Write (RMW) instructions). Only in the case of HALT bits unchanged (initialization of the macro, etc.), there is no problem even if any bit manipulation instructions is used.

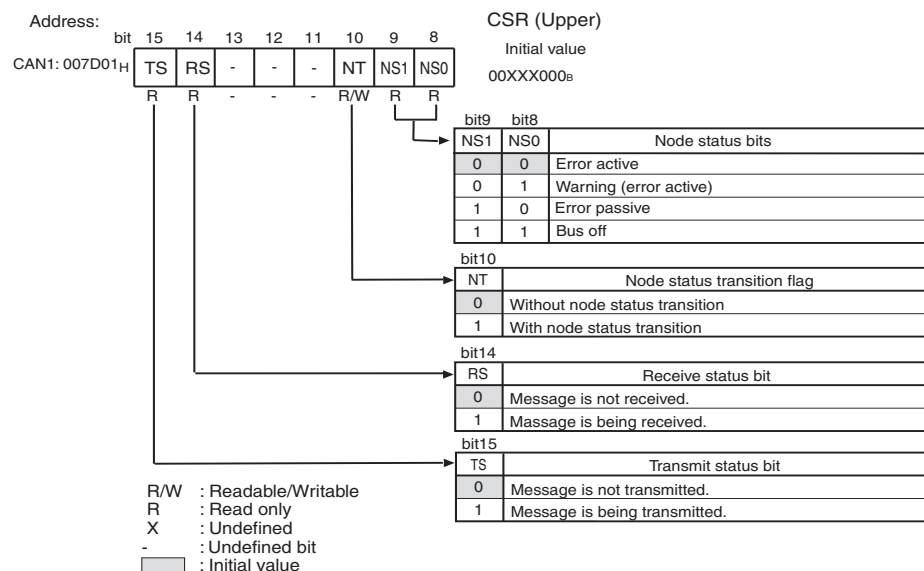
■ Control Status Register (CSR) (Lower)

Figure 21-2. Configuration of the Control Status Register (Lower Byte) (CSR: L)



■ Control Status Register (CSR) (Upper)

Figure 21-3. Configuration of the Control Status Register (Upper Byte) (CSR: H)



21.4.2 Function of Control Status Register (CSR)

The operating status of the register's each bit is confirmed by following;

- Setting "0" or "1"
- Function control by writing
- Read

■ Control Status Register (CSR) (Lower)

Table 21-5. Function of Each Bit of the Control Status Register (CSR:L)

Bit Name		Function
bit7	TOE: Transmit output enable bit	This bit switches from a general-purpose I/O port to a transmit pin TX. When setting to "0" : Functions as general-purpose I/O port. When setting to "1" : Functions as transmit pin TX.
bit6 to bit3	Undefined bits	When reading : Value is undefined. When writing : No effect
bit2	NIE: Node status transition interrupt output enable bit	This bit controls a node status transition interrupt generation when a node status is transferred (CSR: NT = 1). When setting to "0" : Interrupt generation is disabled. When setting to "1" : Interrupt generation is enabled.
bit1	Reserved: Reserved bit	This bit is always set to "0". When reading : The value is always "0".
bit0	HALT: Bus operation halt 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 from this bit: 0: Bus operation not in stop state 1: Bus operation in stop state Note: After ensuring that "1" is written to this bit, write "0" to this bit if the node status is Bus Off. Example program: switch (IO_CANCT1.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_CANCT1.CSR.bit.HALT == 0); i++); IO_CANCT1.CSR.word = 0x0084; /* HALT = 0 */ break; } *:The variable "i" is used for fail-safe. For detail information, see Section "21.4.4 Notes on Using Bus Operation Stop Bit (HALT = 1)".

■ Control Status Register (CSR) (Upper)

Table 21-6. Function of Each Bit of the Control Status Register (CSR:H)

Bit Name		Function
bit15	TS: Transmit status bit	This bit indicates whether a message is being transmitted. At read: 0: Message not being transmitted 1: Message being transmitted This bit is cleared "0" even while error and overload frames are transmitted.

Table 21-6. Function of Each Bit of the Control Status Register (CSR:H)

Bit Name	Function
bit14 RS: Receive status bit	<p>This bit indicates whether a message is being received.</p> <p>At read:</p> <p>0: Message not being received</p> <p>1: Message being received</p> <ul style="list-style-type: none"> ❑ 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 = 1); the bus is in the intermission/bus idle or a error/overload frame is on the bus.
bit13 to bit11	<p>When reading: The value is undefined.</p> <p>When writing: No effect</p>
bit10 NT: Node status transition flag	<p>When the node status changes from increment transition or bus off into error active, this bit is set to "1". The condition that this bit is set to "1" is as follows. At this time, the interruption is generated for the node status interruption permission bit (NIE) = 1.</p> <ul style="list-style-type: none"> ❑ Error active ("00_B") → Warning ("01_B") ❑ Warning ("01_B") → Error passive ("10_B") ❑ Error passive ("10_B") → Bus off ("11_B") ❑ Bus off ("11_B") → Error active ("00_B") <p>Note:</p> <p>In parentheses, the value of the NS1 and NS0 bits is indicated.</p> <p>At Write:</p> <p>"0": Cleared</p> <p>"1": Not possible to set (No effect)</p> <p>At read by the instruction of the read-modify-write (RMW):</p> <p>Always read "1".</p>
bit9, bit8	<p>NS1, NS0: Node status bits</p> <p>These bits indicate the current node status.</p> <p>For detail information, see Section "21.4.3 Correspondence between Node Status Bits and Node Status".</p>

21.4.3 Correspondence between Node Status Bits and Node Status

Node status bits show the node status by two bits (NS1 and NS0).

■ Correspondence between Node Status Bits (NS1 and NS0) and Node Status

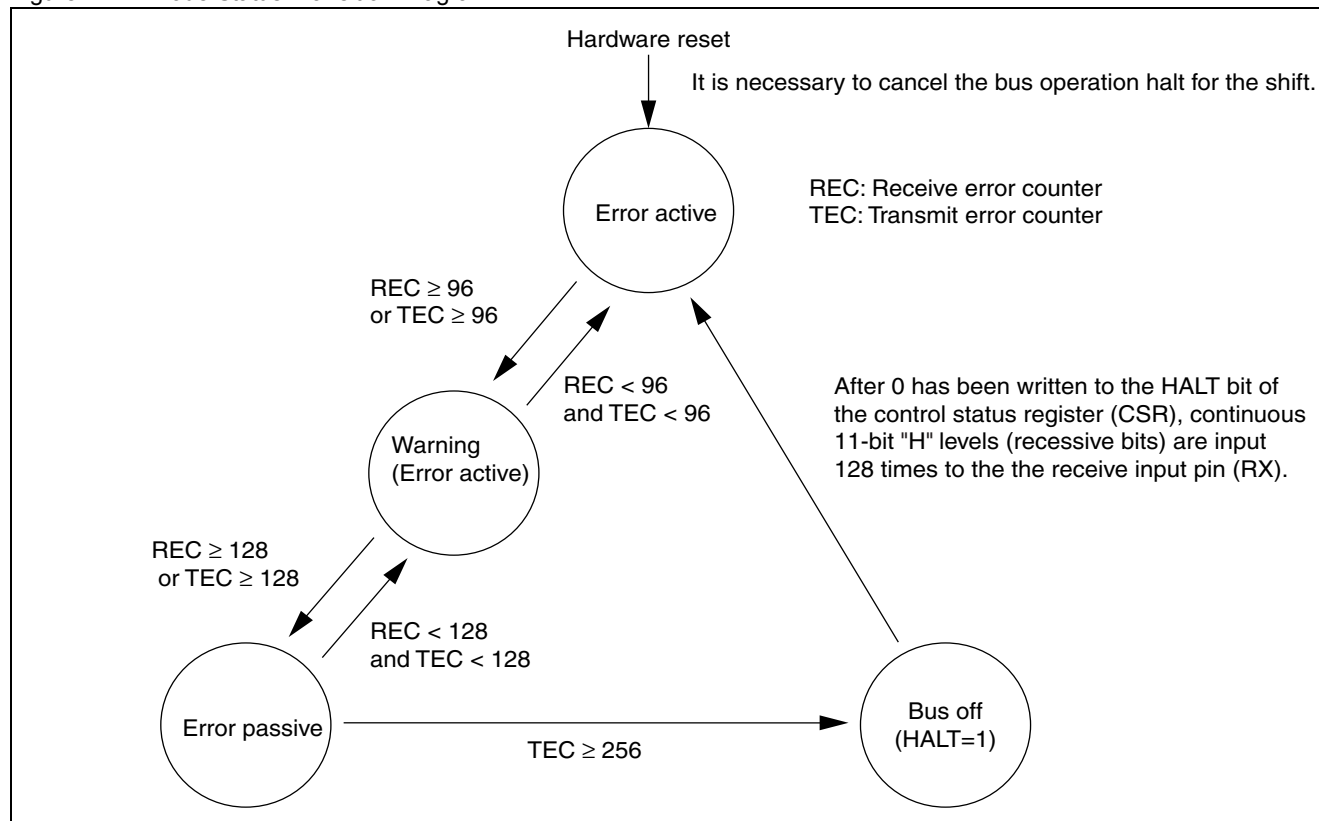
Table 21-7. Correspondence between NS1 and NS0 and Node Status

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 ver. 2.0 Part B for the node status, however, indicates that the transmit error counter or receive error counter has exceeded 96. The node status transition diagram is shown in Figure 21-4. .

Figure 21-4. Node Status Transition Diagram



21.4.4 Notes on Using Bus Operation Stop Bit (HALT = 1)

The bus operation stop bit is set by writing to the bit, hardware reset and the node status. The stop operation of the bus operation is different according to the state of the message buffer.

■ Conditions for Setting Bus Operation Stop (HALT=1)

There are 3 conditions for setting bus operation stop (HALT = 1):

- Hardware reset

- ❑ When node status changed to bus off
- ❑ By writing "1" to HALT bit

Notes:

- 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 clock 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)

The condition for canceling the bus operation stop (HALT=0) is writing "0" to HALT.

Notes:

- 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 "H" levels (recessive bits) have been input to the receive input pin (RX).
- 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 "H" levels (recessive bits) have been input 128 times to the receive input pin (RX). Then, the values of both transmit and receive error counters reach "0" and the node status is changed to error active.
- When writing "0" to HALT bit 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 "H" 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).

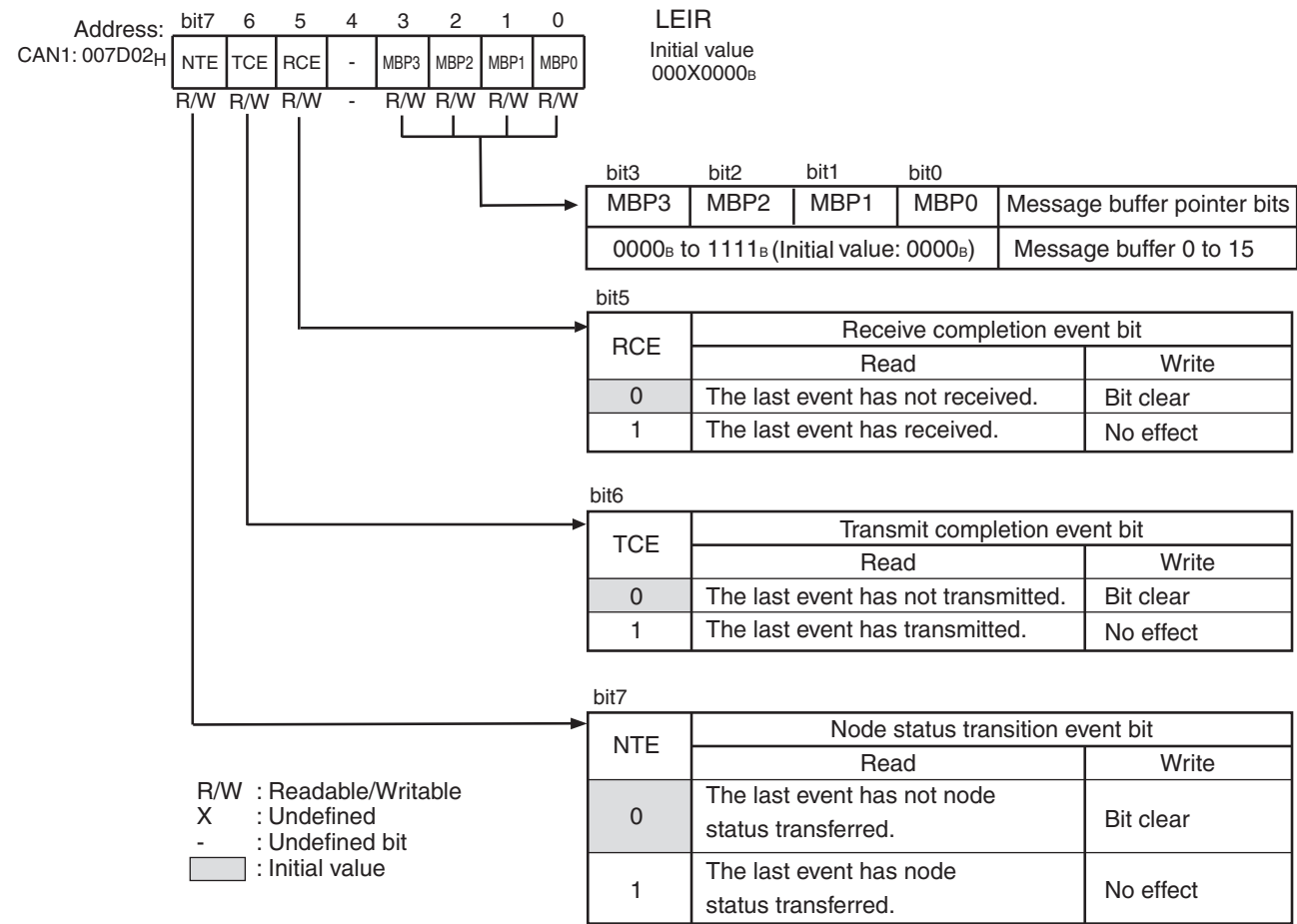
21.4.5 Last Event Indicator Register (LEIR)

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 cleared to "0".

■ Register Configuration

Figure 21-5. Configuration of the Last Event Indicator Register (LEIR)



■ Register Function

Table 21-8. Function of Each Bit of the Last Event Indicator Register (LEIR)

Bit Name		Function
bit7	NTE: Node status transition event bit	When this bit is "1", node status transition is the last event. This bit is set to "1" after set either of bit of the control status register to "1" (CSR:NTx=1). This setting is not related to the setting of NIE bit of the control status register (CSR). At Write: "0": Cleared "1": No effect At read by the instruction of the read-modify-write (RMW): Always read "1".

Table 21-8. Function of Each Bit of the Last Event Indicator Register (LEIR)

Bit Name		Function
bit6	TCE: Transmit completion event bit	<p>When this bit is "1", it indicates that transmit completion is the last event.</p> <p>This bit is set to "1" after set either of bit of the transmit completion register to "1" (TCR:TCx=1).</p> <ul style="list-style-type: none"> <input type="checkbox"/> This setting is not related to the setting of the transmit complete interrupt enable register (TIER). <input type="checkbox"/> When this bit is "1", MBP3 to MBP0 bits show the message buffer number (x) to complete the transmission of the message in the last event. <p>At Write: "0": Cleared "1": No effect</p> <p>At read by the instruction of the read-modify-write (RMW): Always read "1".</p>
bit5	RCE: Receive completion event bit	<p>When this bit is "1", it indicates that receive completion is the last event.</p> <p>This bit is set to "1" after set either of bit of the receive complete register to "1" (RCR:RCx=1).</p> <ul style="list-style-type: none"> <input type="checkbox"/> This setting is not related to the setting of the receive complete interrupt enable register (RIER). <input type="checkbox"/> When this bit is "1", MBP3 to MBP0 bits show the message buffer number (x) to complete the reception of the message in the last event. <p>At Write: "0": Cleared "1": No effect</p> <p>At read by the instruction of the read-modify-write (RMW): Always read "1".</p>
bit4	Undefined bit	<p>When reading: The value is undefined.</p> <p>When writing: No effect</p>
bit3 to bit0	MBP3 to MBP0: Message buffer pointer bits	<p>When TCE bit or RCE bit is "1", these bits show the message buffer number (x) to generating of the corresponding last event. If the NTE bit is set to "1", these bits have no meaning.</p> <p>At Write: "0": Cleared "1": No effect</p> <p>At read by the instruction of the read-modify-write (RMW): Always read "1".</p> <p>Note: If LEIR is accessed within an CAN interrupt handling, the event causing the interrupt is not necessarily the same as indicated by LEIR. In the time from interrupt generation to the LEIR access by the interrupt handler there may occur other CAN events.</p>

21.4.6 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.

■ Register Configuration

Figure 21-6. Configuration of the Receive and Transmit Error Counters (RTEC)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RTEC (Upper)
CAN1: 007D05 _H	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	Initial value 00000000 _B
	R	R	R	R	R	R	R	R	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RTEC (Lower)
CAN1: 007D04 _H	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	Initial value 00000000 _B
	R	R	R	R	R	R	R	R	

R: Read only

■ Register Function

Table 21-9. Function of Each Bit of the Receive and Transmit Error Counters (RTEC)

Bit Name		Function
bit15 to bit8	TEC7 to TEC0: 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 _B).
bit7 to bit0	REC7 to REC0: 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 _B).

21.4.7 Bit Timing Register (BTR)

Bit timing register (BTR) sets the prescaler and bit timing.

■ Register Configuration

Figure 21-7. Configuration of the Bit Timing Register (BTR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	BTR (Upper)
CAN1: 007D07 _H	-	TS2.2	TS2.1	TS2.0	TS1.3	TS1.2	TS1.1	TS1.0	Initial value X1111111 _B
	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	BTR (Lower)
CAN1: 007D06 _H	RSJ1	RSJ0	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0	Initial value 1111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable
 X: Undefined
 -: Undefined bit

■ Register Function

Table 21-10. Function of Each Bit of the Bit Timing Register (BTR)

Bit Name		Function
bit14 to bit12	TS2.2 to TS2.0: Time segment 2 setting bits 2 to 0	These bits define the number of the time quanta (TQ's) by dividing [(TS 2.2 to TS 2.0)+1] for the time segment 2 (TSEG2). The time segment 2 is equal to the phase buffer segment 2 (PHASE_SEG2) in the CAN specification.
bit11 to bit8	TS1.3 to TS1.0: Time segment 1 setting bits 3 to 0	These bits define the number of the time quanta (TQ's) by dividing [(TS 1.3 to TS 1.0)+1] 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.
bit7, bit6	RSJ1, RSJ0: Resynchronization jump width setting bits 1, 0	These bits define the number of the time quanta (TQ's) by dividing [(RSJ1 to RSJ0)+1] for the resynchronization jump width (RSJW).
bit5 to bit0	PSC5 to PSC0: Prescaler setting bits 5 to 0	These bits define the time quanta (TQ) of the CAN controller by dividing system clock.

Note:

Please set the bit timing register (BTR) after stopping the bus operation (CSR: HALT=1). Please release the bus operation stop by writing "0" in the HALT bit of the control status register after the setting of bit timing register (BTR) is ended.

21.4.8 Prescaler Setting by Bit Timing Register (BTR)

The setting of bit timing register (BTR) corresponds to the bit time segments of prescaler in the CAN specification and the CAN controller.

■ Prescaler Settings

The bit time segments defined in the CAN specification and the CAN controller are shown in [Figure 21-8.](#) and [Figure 21-9.](#) respectively.

Figure 21-8. Bit Time Segment in CAN Specification

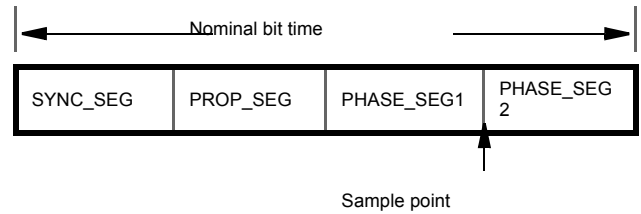
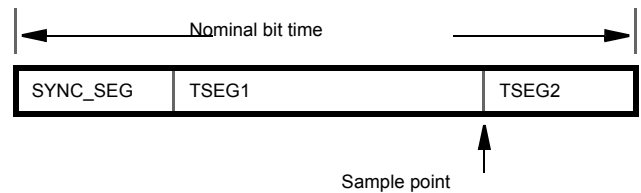


Figure 21-9. Bit Time Segment in CAN Controller



The relationship between PSC = PSC5 to PSC0, TS1 = TS1.3 to TS1.0, TS2 = TS2.2 to TS2.0, and RSJ = RSJ1, RSJ0 is shown below.

$$\begin{aligned} \text{TQ} &= (\text{PSC} + 1) \times \text{CLK} \\ \text{BT} &= \text{SYNC_SEG} + \text{TSEG1} + \text{TSEG2} \\ &= (1 + (\text{TS1} + 1) + (\text{TS2} + 1)) \times \text{TQ} \\ &= (3 + \text{TS1} + \text{TS2}) \times \text{TQ} \\ \text{RSJW} &= (\text{RSJ} + 1) \times \text{TQ} \end{aligned}$$

CLK: input clock

TQ: time quanta

BT: bit time

SYNC_SEG: synchronous segment

TSEG1 and TSEG2: time segment 1 and 2

resynchronization jump width [(RSJ1 and RSJ0) + 1] frequency division

For correct operation, the following conditions should be met.

$$\begin{aligned} &\text{For } 1 \leq \text{PSC} \leq 63 \\ &\quad \text{TSEG1} \geq 2\text{TQ} \\ &\quad \text{TSEG1} \geq \text{RSJW} \\ &\quad \text{TSEG2} \geq 2\text{TQ} \\ &\quad \text{TSEG2} \geq \text{RSJW} \\ &\text{For } \text{PSC} = 0 \\ &\quad \text{TSEG1} \geq 5\text{TQ} \\ &\quad \text{TSEG2} \geq 2\text{TQ} \\ &\quad \text{TSEG2} \geq \text{RSJW} \end{aligned}$$

In order to meet the bit timing requirements defined in the CAN specification, additions have to be met, e.g. delay time has to be considered.

21.4.9 Message Buffer Valid Register (BVALR)

Message buffer valid register (BVALR) sets the validity of the message buffers (x) or displays their state.

■ Register Configuration

Figure 21-10. Configuration of the Message Buffer Valid Register (BVALR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	BVALR (Upper)
CAN1: 000081 _H	BVAL1 5	BVAL1 4	BVAL1 3	BVAL1 2	BVAL1 1	BVAL1 0	BVAL9	BVAL8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	BVALR (Lower)
CAN1: 000080 _H	BVAL7	BVAL6	BVAL5	BVAL4	BVAL3	BVAL2	BVAL1	BVAL0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

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.

Notes:

- 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 cleared 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 "21.13 Notes on Using CAN Controller".

21.4.10 IDE Register (IDER)

This register sets the frame format used by the message buffers (x) during transmission/reception.

■ Register Configuration

Figure 21-11. Configuration of the IDE Register (IDER)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	IDER (Upper)
CAN1: 007D09 _H	IDE15	IDE14	IDE13	IDE12	IDE11	IDE10	IDE9	IDE8	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	IDER (Lower)
CAN1: 007D08 _H	IDE7	IDE6	IDE5	IDE4	IDE3	IDE2	IDE1	IDE0	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W:Readable/Writable
X:Undefined

■ Register Function

0: The standard frame format (ID11 bits) is used for the message buffer (x).

1: The extended frame format (ID29 bits) is used for the message buffer (x).

Notes:

- 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 "21.13 Notes on Using CAN Controller".

21.4.11 Transmission Request Register (TREQR)

Transmission request register (TREQR) sets transmission requests to the message buffers (x) or displays their state.

■ Register Configuration

Figure 21-12. Configuration of the Transmission Request Register (TREQR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TREQR (Upper)
CAN1: 000083 _H	TREQ 15	TREQ 14	TREQ 13	TREQ 12	TREQ 11	TREQ 10	TREQ 9	TREQ 8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TREQR (Lower)
CAN1: 000082 _H	TREQ 7	TREQ 6	TREQ 5	TREQ 4	TREQ 3	TREQ 2	TREQ 1	TREQ 0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

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 "[21.4.12 Transmission RTR Register \(TRTRR\)](#)" and "[21.4.13 Remote Frame Receiving Wait Register \(RFWTR\)](#)".

^{*2}: For cancellation of transmission, see Sections "[21.4.14 Transmission Cancel Register \(TCANR\)](#)" and "[21.4.15 Transmission Complete Register \(TCR\)](#)".

Writing "0" to TREQx is ignored.

"0" is read when a read-modify-write (RMW) 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 1 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.

21.4.12 Transmission RTR Register (TRTRR)

This register sets the transmission RTR (Remote Transmission Request) bits for the message buffers (x).

■ Register Configuration

Figure 21-13. Configuration of the Transmission RTR Register (TRTRR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TRTRR (Upper)
CAN1: 007D0B _H	TRTR1 5	TRTR 14	TRTR1 3	TRTR 12	TRTR1 1	TRTR 10	TRTR9	TRTR 8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TRTRR (Lower)
CAN1: 007D0A _H	TRTR7	TRTR 6	TRTR5	TRTR 4	TRTR3	TRTR 2	TRTR1	TRTR 0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

0: Transmit data frame.

1: Transmit remote frame.

21.4.13 Remote Frame Receiving Wait Register (RFWTR)

Remote frame receiving wait register (RFWTR) sets 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 transmission RTR register (TRTRR) is "0").

■ Register Configuration

Figure 21-14. Configuration of the Remote Frame Receiving Wait Register (RFWTR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RFWTR (Upper)
CAN1: 007D0D _H	RFWT 15	RFWT 14	RFWT 13	RFWT 12	RFWT 11	RFWT 10	RFWT 9	RFWT 8	Initial value XXXXXXXX _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RFWTR (Lower)
CAN1: 007D0C _H	RFWT 7	RFWT 6	RFWT 5	RFWT 4	RFWT 3	RFWT 2	RFWT 1	RFWT 0	Initial value XXXXXXXX _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable
 X: Undefined

■ Register Function

0: Transmission starts immediately

1: Transmission starts after waiting until remote frame received (RRTRx of remote request receiving register (RRTRR) becomes "1")

Notes:

- Transmission starts immediately if RRTRx has been already "1" when a request for transmission is set.
- For remote frame transmission, do not set RFWTx to "1".

21.4.14 Transmission Cancel Register (TCANR)

This register cancels a pending request for transmission to the message buffer (x).

■ Register Configuration

Figure 21-15. Configuration of the Transmission Cancel Register (TCANR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TCANR (Upper)
CAN1: 000085 _H	TCAN 15	TCAN 14	TCAN 13	TCAN 12	TCAN 11	TCAN 10	TCAN 9	TCAN 8	Initial value 00000000 _B
	W	W	W	W	W	W	W	W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TCANR (Lower)
CAN1: 000084 _H	TCAN 7	TCAN 6	TCAN 5	TCAN 4	TCAN 3	TCAN 2	TCAN 1	TCAN 0	Initial value 00000000 _B
	W	W	W	W	W	W	W	W	

W: Write only

■ Register Function

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".

21.4.15 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.

■ Register Configuration

Figure 21-16. Configuration of the Transmission Complete Register (TCR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TCR (Upper)
CAN1: 000087 _H	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TCR (Lower)
CAN1: 000086 _H	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable
X: Undefined

■ Register Function

- 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 cleared it to "0". Writing "1" to TCx is ignored.

"1" is read when a read-modify-write (RMW) 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".

21.4.16 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").

■ Register Configuration

Figure 21-17. Configuration of the Transmission Interrupt Enable Register (TIER)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TIER (Upper)
CAN1: 007D0F _H	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TIER (Lower)
CAN1: 007D0E _H	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

0: Transmission interrupt disabled.

1: Transmission interrupt enabled.

21.4.17 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.

■ Register Configuration

Figure 21-18. Configuration of the Reception Complete Register (RCR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RCR (Upper)
CAN1: 000089 _H	RC15	RC14	RC13	RC12	RC11	RC10	RC9	RC8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RCR (Lower)
CAN1: 000088 _H	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

- Conditions for RCx = 0

Write "0" to RCx.

After completion of handling received message, write "0" to RCx to cleared it to "0". Writing "1" to RCx is ignored.

"1" is read when a read-modify-write (RMW) 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".

21.4.18 Remote Request Receiving Register (RRTRR)

After the received remote frame is stored in the message buffer (x), RRTRx becomes "1" (at the same time as RCx setting to "1").

■ Register Configuration

Figure 21-19. Configuration of the Remote Request Receiving Register (RRTRR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RRTRR (Upper)
CAN1: 00008B _H	RRTR 15	RRTR 14	RRTR 13	RRTR 12	RRTR 11	RRTR 10	RRTR 9	RRTR 8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RRTRR (Lower)
CAN1: 00008A _H	RRTR 7	RRTR 6	RRTR 5	RRTR 4	RRTR 3	RRTR 2	RRTR 1	RRTR 0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

- 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 (RMW) 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".

21.4.19 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.

■ Register Configuration

Figure 21-20. Configuration of the Receive Overrun Register (ROVRR)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	ROVRR (Upper)
CAN1: 00008D _H	ROVR 15	ROVR 14	ROVR 13	ROVR 12	ROVR 11	ROVR 10	ROVR 9	ROVR 8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	ROVRR (Lower)
CAN1: 00008C _H	ROVR 7	ROVR 6	ROVR 5	ROVR 4	ROVR 3	ROVR 2	ROVR 1	ROVR 0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

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 (RMW) instruction is performed.

Note:

If setting to "1" by generation of the receive overrun and clearing to "0" by writing occur at the same time, the bit is set to "1".

21.4.20 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").

■ Register Configuration

Figure 21-21. Configuration of the Reception Interrupt Enable Register (RIER)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RIER (Upper)
CAN1: 00008F _H	RIE15	RIE14	RIE13	RIE12	RIE11	RIE10	RIE9	RIE8	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RIER (Lower)
CAN1: 00008E _H	RIE7	RIE6	RIE5	RIE4	RIE3	RIE2	RIE1	RIE0	Initial value 00000000 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

■ Register Function

0: Reception interrupt disabled.

1: Reception interrupt enabled.

21.4.21 Acceptance Mask Select Register (AMSR)

This register selects masks (acceptance mask) for comparison between the received message ID's and the message buffer (x) ID.

■ Register Configuration

Figure 21-22. Configuration of the Acceptance Mask Select Register (AMSR)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMSR(Byte0)
CAN1: 007D10 _H	AMS3. 1	AMS3. 0	AMS2. 1	AMS2. 0	AMS1. 1	AMS1. 0	AMS0. 1	AMS0. 0	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMSR(Byte1)
CAN1: 007D11 _H	AMS7. 1	AMS7. 0	AMS6. 1	AMS6. 0	AMS5. 1	AMS5. 0	AMS4. 1	AMS4. 0	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMSR(Byte2)
CAN1: 007D12 _H	AMS11. 1	AMS1. 1.0	AMS1. 0.1	AMS1. 0.0	AMS9. 1	AMS9. 0	AMS8. 1	AMS8. 0	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMSR(Byte3)
CAN1: 007D13 _H	AMS1. 5.1	AMS1. 5.0	AMS1. 4.1	AMS1. 4.0	AMS1. 3.1	AMS1. 3.0	AMS1. 2.1	AMS1. 2.0	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W:Readable/Writable

X:Undefined

■ Register Function

Table 21-11. 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)

Notes:

- 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 ["21.13 Notes on Using CAN Controller"](#).

21.4.22 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)

There are two acceptance mask registers, 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.

■ Register Configuration

Figure 21-23. Configuration of the Acceptance Mask Register 0 (AMR0)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR0 (Byte0)
CAN1: 007D14 _H	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR0 (Byte1)
CAN1: 007D15 _H	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR0 (Byte2)
CAN1: 007D16 _H	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR0 (Byte3)
CAN1: 007D17 _H	AM4	AM3	AM2	AM1	AM0	-	-	-	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	-	-	-	


R/W : Readable/Writable
 X : Undefined
 - : Undefined bit
 : Used bit in standard frame format

Figure 21-24. Configuration of the Acceptance Mask Register 1 (AMR1)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR1 (Byte0)
CAN1: 007D18 _H	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR1 (Byte1)
CAN1: 007D19 _H	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR1 (Byte2)
CAN1: 007D1A _H	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR1 (Byte3)
CAN1: 007D1B _H	AM4	AM3	AM2	AM1	AM0	-	-	-	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	-	-	-	

R/W : Readable/Writable
X : Undefined
- : Undefined bit
 : Used bit in standard frame format

Register Function

- ☐ 0: Compare

Compare the bit of the acceptance code (ID register IDR_x for comparing with the received message ID) corresponding to this bit (be set to "0") with the bit of the received message ID. If there is no match, no message is received.

- ☐ 1: Mask

Mask the bit of the acceptance code ID register (IDR_x) corresponding to this bit (be set to "1"). No comparison is made with the bit of the received message ID.

Notes:

- AMR0 and AMR1 should be set when all the message buffers (x) selecting AMR0 and AMR1 are invalid (BVAL_x of the message buffer valid register (BVALR) is "0"). Setting when the buffers are valid (BVAL_x = 1) may cause unnecessary received messages to be stored.
- To invalidate the message buffer (by setting the BVALR: BVAL bit to "0") while 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 "21.13 Notes on Using CAN Controller".

21.4.23 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

- Register Configuration
- ID register x ($x = 0$ to 15) (IDRx)

This register is a ID register of the message buffer. This register memorizes acceptance code setting, transmission message ID setting, and reception ID.

- DLC register x ($x = 0$ to 15) (DLCRx)

This register stores the DLC of the message buffer. This register sets the data length of the message when a data frame and a remote frame are transmitted and the data length of the message when a data frame or a remote frame is received.

- Data register x ($x = 0$ to 15) (DTRx)

This register is a data register of the message buffer. This register memorizes the setting of the reception message data or the transmission message data.

- 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 1 message buffer, transmission is performed, starting with the lowest-numbered message buffer (See Section "21.5 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 1 message buffer, the received message is stored in the lowest-numbered message buffer (See Section "21.6 Reception of CAN Controller").

- Message buffer that can be used as multi level message buffer

When the same acceptance filter is set in "1" or more message buffers, the message buffer can be used as a multi level message buffer.

As a result, the reserve to the reception time is given. (See Section "21.10 Procedure for Reception by Message Buffer (x)").

Notes:

- 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 (Address 007A00_H to 007A1F_H, 007C00_H to 007C1F_H, 007E00_H to 007E1F_H).

21.4.24 ID Register x (x = 0 to 15) (IDRx)

This register is the ID register for message buffer (x).

■ Register Configuration

Figure 21-25. Configuration of the ID Registers x (x = 0 to 15) (IDRx)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	IDRx(Byte0)
CAN1: 007C20 _H + 4 × x	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	IDRx(Byte1)
CAN1: 007C21 _H + 4 × x	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	IDRx(Byte2)
CAN1: 007C22 _H + 4 × x	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	IDRx(Byte3)
CAN1: 007C23 _H + 4 × x	ID4	ID3	ID2	ID1	ID0	-	-	-	Initial value XXXXXXXXB
	R/W	R/W	R/W	R/W	R/W	-	-	-	

x	: x = 0 to 15
R/W	: Readable/Writable
X	: Undefined
-	: Undefined bit
<div style="display: inline-block; width: 15px; height: 10px; background-color: #cccccc; border: 1px solid black;"></div>	: Used bit in standard frame format

■ Register Function

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 by acceptance filter). In the standard frame format, ID17 to ID0 store image of old message left in the receive shift register.

Notes:

- 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 "21.13 Notes on Using CAN Controller".

21.4.25 DLC Register x (x = 0 to 15) (DLCRx)

This register stores the DLC for message buffer (x).

■ Register Configuration

Figure 21-26. Configuration of the DLC Registers x (x = 0 to 15) (DLCRx)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DLCRx (Lower)
CAN1: 007C60 _H + 2 × x	-	-	-	-	DLC3	DLC2	DLC1	DLC0	Initial value XXXXXXXX _B
x = 0 to 15	-	-	-	-	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									
X: Undefined									
-: Undefined bit									

■ Register Function

- 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_B" to "1000_B" (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.

21.4.26 Data Register x (x = 0 to 15) (DTRx)

This register 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.

■ Register Configuration

Figure 21-27. Configuration of the Data Registers x (x = 0 to 15) (DTRx)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx(Byte0)
CAN1: 007C80 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx(Byte1)
CAN1: 007C81 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx(Byte2)
CAN1: 007C82 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx(Byte3)
CAN1: 007C83 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx(Byte4)
CAN1: 007C84 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx(Byte5)
CAN1: 007C85 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx(Byte6)
CAN1: 007C86 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx(Byte7)
CAN1: 007C87 _H + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	XXXXXXXXB

x = 0 to 15

R/W: Readable/Writable

X: Undefined

■ Register Function

- 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.

21.5 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 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 1 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 Transmission Request from 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:

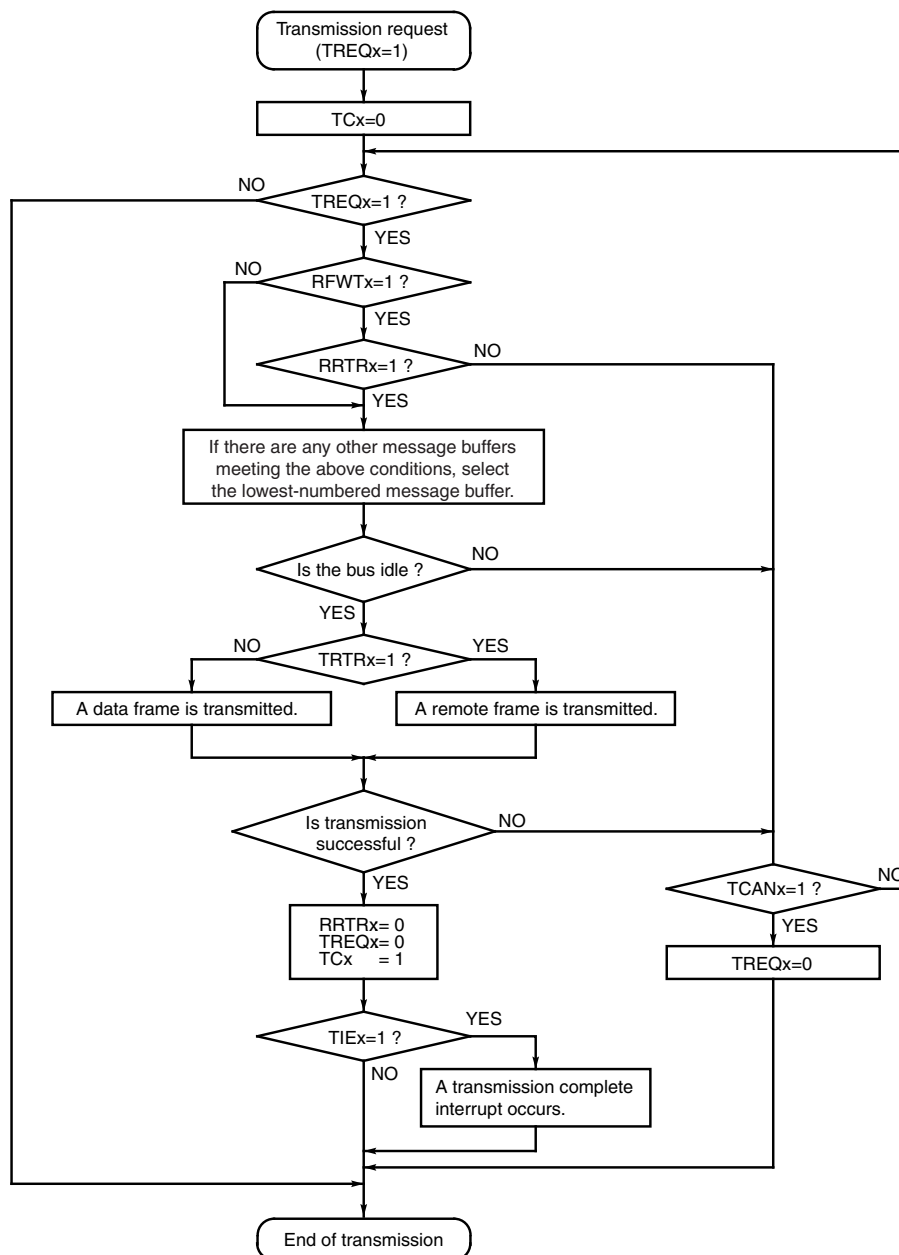
The transmission request is canceled by storing either data frames or remote frames.

■ Completing Transmission of 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 CAN Controller

Figure 21-28. Transmission Flowchart of the CAN Controller



21.6 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 (IDRx) is "0"). The received message in the extended frame format is compared with the message buffer (x) set in the extended frame format (IDEx is "1").

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.

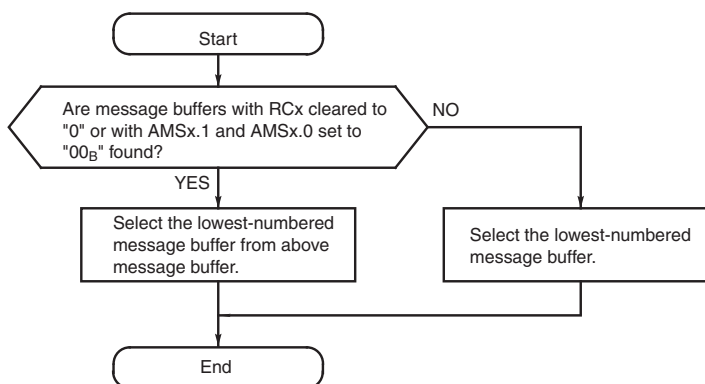
When receiving remote frames, received messages are stored only in the IDRx and DLCRx, and the DTRx remains unchanged.

If there is more than 1 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_B), 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 21-29. shows a flowchart for determining the message buffer (x) where received messages are to be stored. For message buffers please arrange 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 21-29. Flowchart Determining Message Buffer (x) where Received Messages Stored



■ Receive Overrun

When the received message is stored in the message buffer (x) with the corresponding RCx bit of the reception complete register (RCR) 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.

Notes:

- A request for data frame transmission is not canceled.
- For cancellation of a transmission request, see Section "21.5 Transmission of 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:

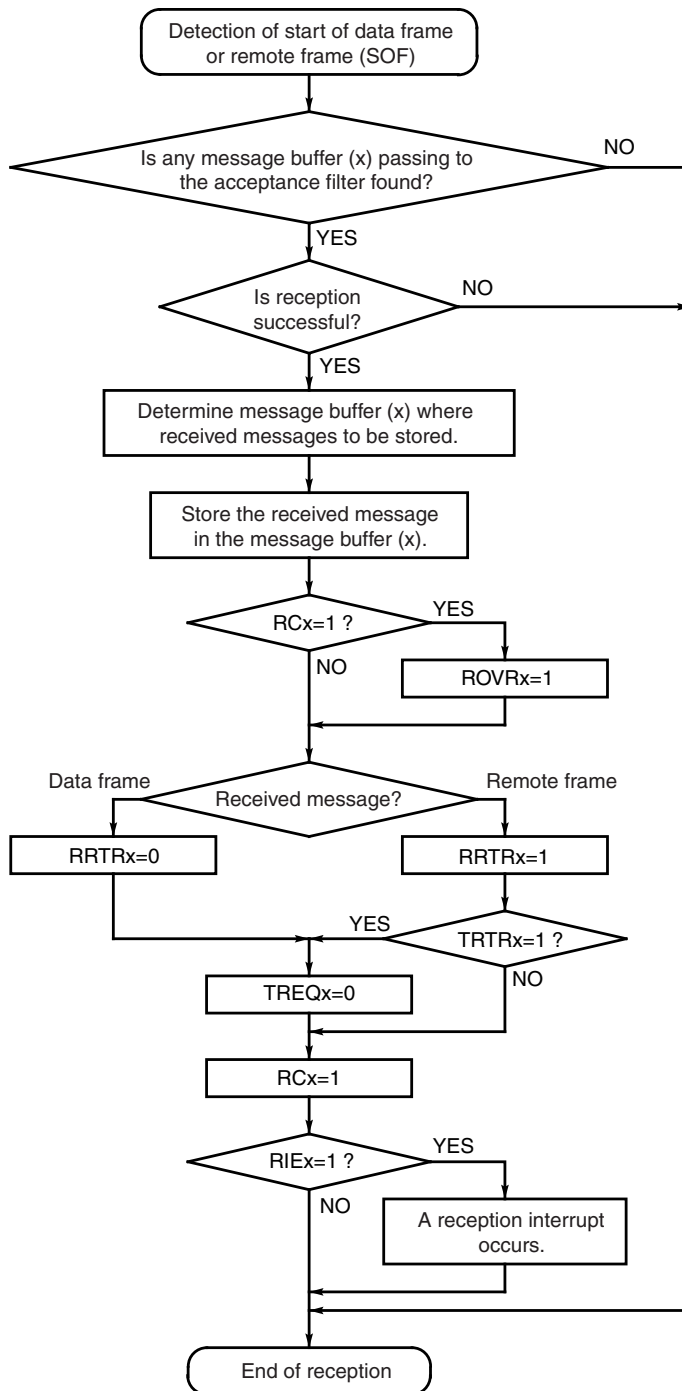
CAN controller will not receive any messages transmitted by itself.

21.7 Reception Flowchart of CAN Controller

Figure 21-30. shows a reception flowchart of the CAN controller.

■ Reception Flowchart of the CAN Controller

Figure 21-30. Reception Flowchart of the CAN Controller



21.8 How to Use 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 IDEx of the IDE register (IDER) to "0". When using the extended frame format, set IDEx to "1".

This setting should be made 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.

■ Setting ID

Set the message buffer (x) ID to ID28 to ID0 of ID register (IDRx). The message buffer (x) ID need not be set to ID17 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 (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.

■ Setting Acceptance Filter

The acceptance filter of the message buffer (x) is set by an acceptance code and acceptance mask. It should be set when the acceptance 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.

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 "21.4.21 Acceptance Mask Select Register (AMSR)" and "21.4.22 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 watch 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).

21.9 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_B" to "1000_B" (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 written while the TREQx bit of the transmission request register (TREQR) is 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 cannot be executed, 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 request cancellation is terminated or transmission is completed. Check TCx of the transmission complete register (TCR). For TCx = 0, transmission request 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.

- Cancel the request for data frame transmission by reception of data frame
- Cancel the request for remote frame transmission by reception of data frame
- Cancel the 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.

21.10 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.

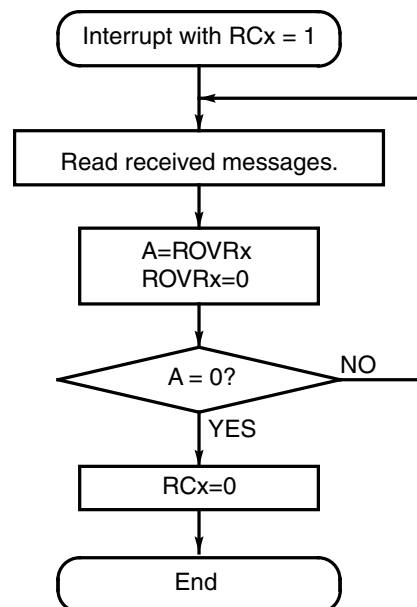
After completion of processing the received message, check ROVRx of the reception overrun register (ROVRR).

If ROVRx = 0, the processed received message is valid. Write "0" to RCx 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 21-31. shows an example of receive interrupt processing.

Figure 21-31. Example of Receive Interrupt Processing



21.11 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 1 message buffer can be combined into a multi-level message buffer to provide allowance for processing time of the received message by CPU.

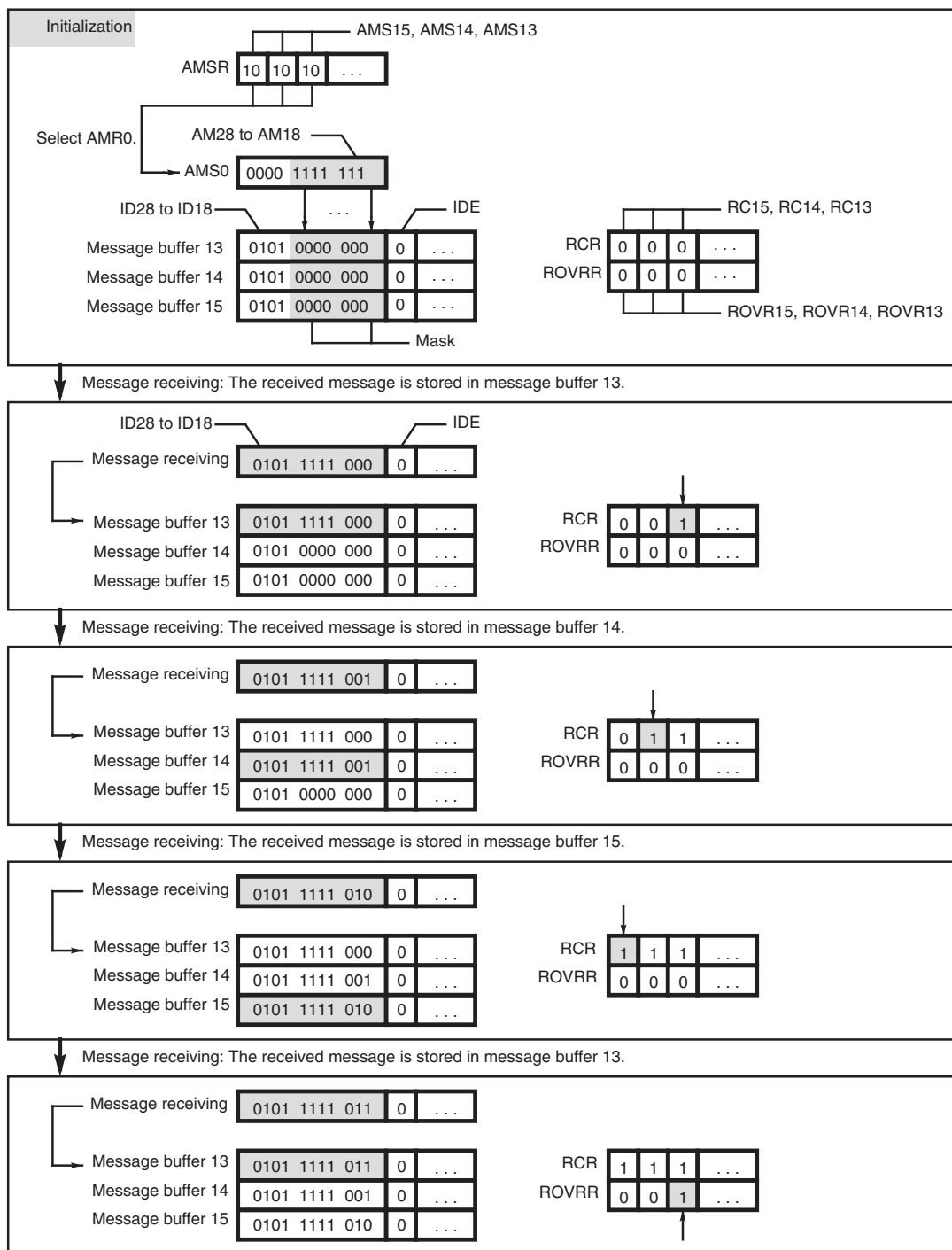
■ Setting Configuration of Multi-level Message Buffer

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 ((AMsx.1, AMsx.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 1 message buffer. Therefore, all bits compare and identical acceptance code should not be specified for more than 1 message buffer.

Figure 21-32. shows operational examples of multi-level message buffers.

Figure 21-32. 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.

21.12 CAN Direct Mode Register

To operate CAN normally, this register must be set correctly.

■ CAN Direct Mode Register (CDMR)

Figure 21-33. CAN Direct Mode Register (CDMR)

	bit 7	6	5	4	3	2	1	0	CDMR
Address: 00796E _H	—	—	—	—	—	—	—	DIRECT	Initial value 11111110 _B
	—	—	—	—	—	—	—	R/W	

R/W : Readable/Writable
 — : Undefined bit

Table 21-12. Function of CAN Direct Mode Register (CDMR)

Bit Name		Function
bit 7 to bit 1	Undefined bits	-
bit 0	DIRECT: Direct mode bit	This bit should be set to "1".

Note:

When using CAN controller in the evaluation product, the DIRECT bit of this register must be set to "1". If setting the bit to "0", operation is different from that in the flash memory product. In the flash memory product, setting of the DIRECT bit has no effect on operation. Operation is the same as that in the evaluation product with DIRECT set to "1".

21.13 Notes on Using CAN Controller

Use of the CAN controller requires the following notes.

■ Note for Disabling Message Buffers by BVAL Bits

When the message buffer is disabled by using the BVAL bit, CAN controller may not perform the proper receive/transmit operation. This section shows the work around of this malfunction.

□ Condition

When following 2 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 participating in CAN bus communication to enable to transmit and receive messages).
- Message buffers are read or written when BVAL bits disable the message buffers.

□ Operation

Operation for suppressing transmission request

Do not use BVAL bit for suppressing or stopping transmission request, but 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, perform the either of the following operations before disabling the message buffer by writing "0" to BVAL bit.

- Read the transmission request bit (TREQ) to check that there is no transmit request (TREQ=0)
- Read the transmission complete bit (TC) to check that transmission has been completed (TC=1)

In case a transmission buffer needs to be disabled, ensure that no transmission request is pending (if it was requested before). Do not write "0" to BVALx bit until confirming no transmission in progress.

1) Cancel the transmission request: TCANx=1 (if necessary)

2) and wait for the transmission completion (while (TREQx=1);) by polling or interrupt.

Be sure to make the transmission buffer ignored (BVALx=0) after executing the above operations.

Note:

For case 1), 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.

22. Address Match Detection Function



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

22.1 Overview of Address Match Detection Function

22.2 Block Diagram of Address Match Detection Function

22.3 Configuration of Address Match Detection Function

22.4 Explanation of Operation of Address Match Detection Function

22.5 Program Example of Address Match Detection Function

22.1 Overview of Address Match Detection Function

If the address of the instruction to be processed next to the instruction currently processed by the program matches the address set in the detect address setting registers, the address match detection function forcibly replaces the next instruction to be processed by the program with the INT9 instruction to branch to the interrupt processing program. Since the address match detection function can use the INT9 interrupt for processing, the program can be corrected by patch processing.

■ Overview of Address Match Detection Function

- The address of the instruction to be processed next to the instruction currently processed by the program is always held in the address latch through the internal data bus. The address match detection function always compares the value of the address held in the address latch with that of the address set in the detect address setting registers. When these compared values match, the next instruction to be processed by the CPU is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.
- There are six detect address setting registers (PADR0 to PADR5), each of which has an interrupt enable bit. The generation of an interrupt due to a match between the address held in the address latch and the address set in the detect address setting registers can be enabled or disabled for each register.

22.2 Block Diagram of Address Match Detection Function

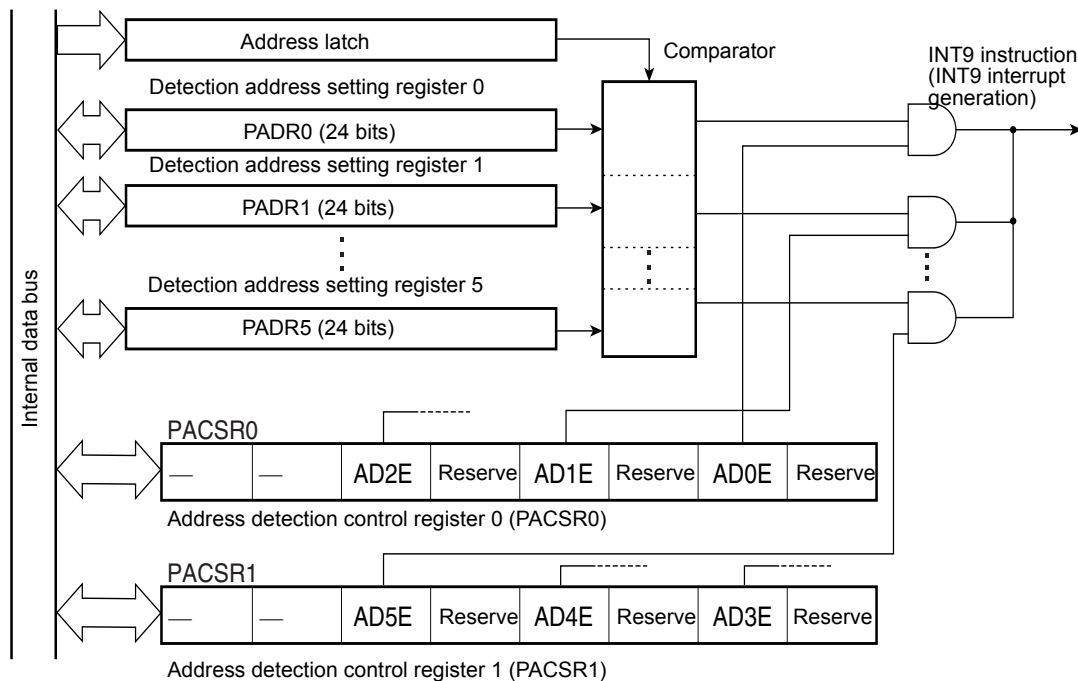
The address match detection module consists of the following blocks:

- Address latch
- Address detection control register (PACSR0/PACSR1)
- Detect address setting registers (PADR0 to PADR5)

■ Block Diagram of Address Match Detection Function

Figure 22-1. shows the block diagram of the address match detection function.

Figure 22-1. Block Diagram of the Address Match Detection Function



□ Address latch

The address latch stores the value of the address output to the internal data bus.

□ Address detection control register (PACSR0/PACSR1)

The address detection control register enables or disables output of an interrupt at an address match.

□ Detect address setting registers (PADR0 to PADR5)

The detect address setting registers set the address that is compared with the value of the address latch.

22.3 Configuration of Address Match Detection Function

This section lists and details the registers used by the address match detection function.

■ List of Registers and Initial Values of Address Match Detection Function

Figure 22-2. List of Registers and Initial Values of Address Match Detection Function

Address detection control register 0(PACSR0)	bit	7	6	5	4	3	2	1	0
		1	1	0	0	0	0	0	0
Address detection control register 1(PACSR1)	bit	15	14	13	12	11	10	9	8
		1	1	0	0	0	0	0	0
Detection address setting register 0(PADR0): Low	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 0(PADR0): Middle	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 0(PADR0): High	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 1(PADR1): Low	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 1(PADR1): Middle	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 1(PADR1): High	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 2(PADR2): Low	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 2(PADR2): Middle	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 2(PADR2): High	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 3(PADR3): Low	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 3(PADR3): Middle	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 3(PADR3): High	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 4(PADR4): Low	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 4(PADR4): Middle	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 4(PADR4): High	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 5(PADR5): Low	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x
Detection address setting register 5(PADR5): Middle	bit	15	14	13	12	11	10	9	8
		x	x	x	x	x	x	x	x
Detection address setting register 5(PADR5): High	bit	7	6	5	4	3	2	1	0
		x	x	x	x	x	x	x	x

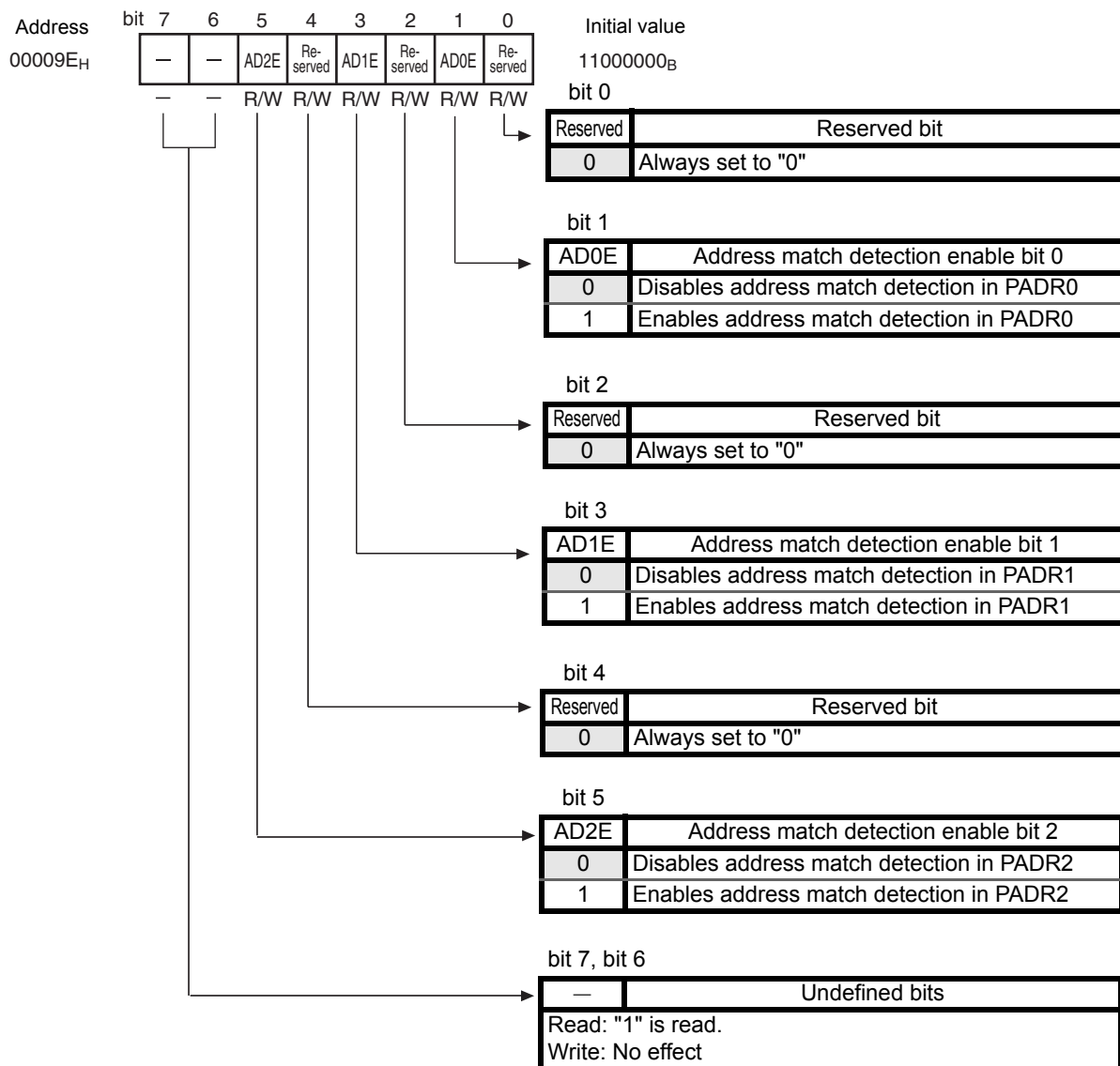
x: Undefined

22.3.1 Address Detection Control Register (PACSR0/PACSR1)

The address detection control register enables or disables output of an interrupt at an address match. When an address match is detected when output of an interrupt at an address match is enabled, the INT9 interrupt is generated.

■ Address Detection Control Register 0 (PACSR0)

Figure 22-3. Address Detection Control Register 0 (PACSR0)



R/W : Readable/Writable

- : Undefined

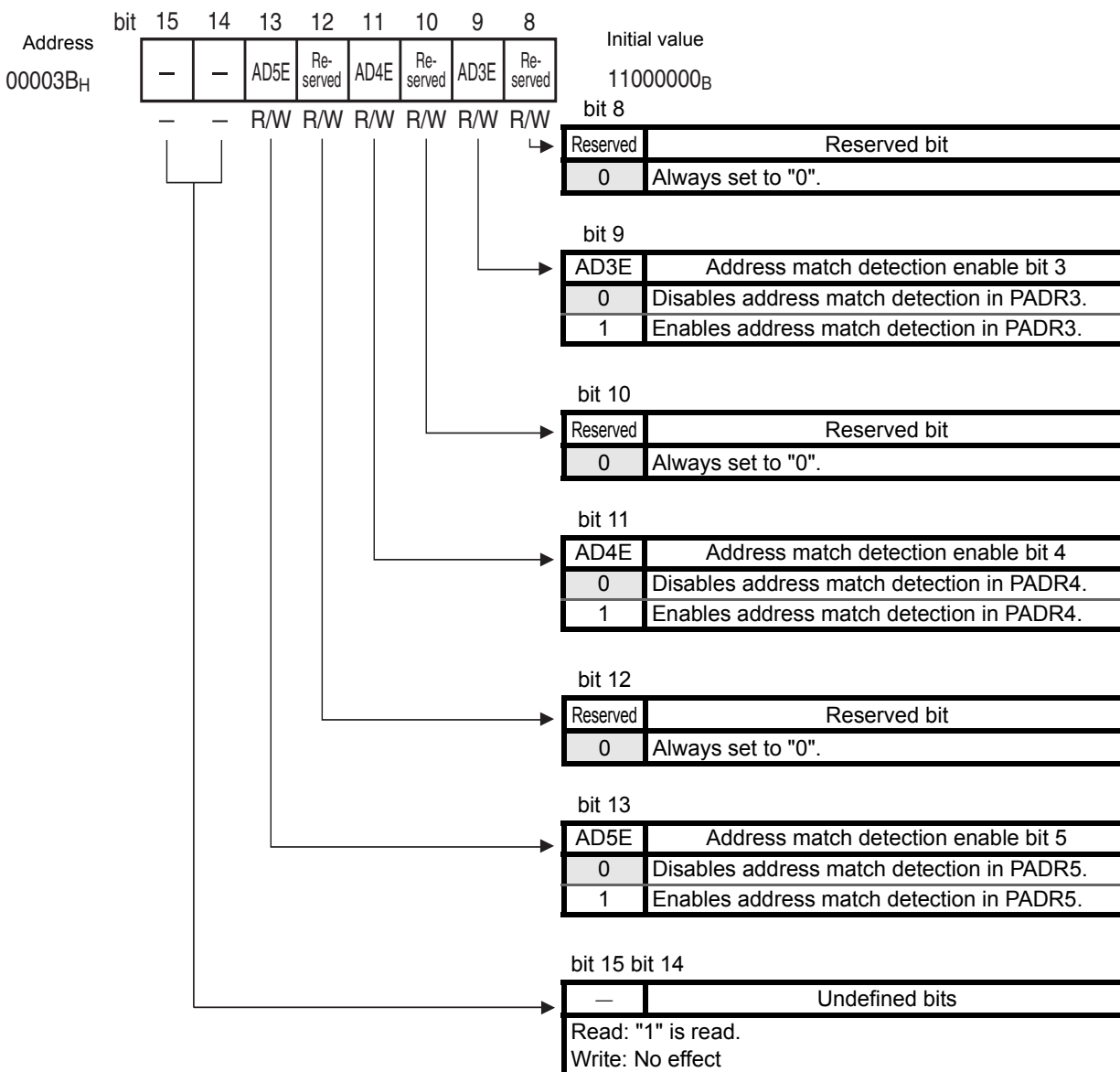
Initial value

Table 22-1. Functions of Address Detection Control Register (PACSR0)

Bit Name		Function
bit7, bit6	Undefined : Undefined bits	Read: "1" is read. Write: No effect
bit5	AD2E: Address match detection enable bit 2	The address match detection operation with the detect address setting register 2 (PADR2) is enabled or disabled. When set to "0" : Disables the address match detection operation. When set to "1" : Enables the address match detection operation. When the value of detect address setting register 2 (PADR2) matches with the value of address latch at enabling the address match detection operation (AD2E = 1), the INT9 instruction is immediately executed.
bit4	Reserved: reserved bit	Always set to "0".
bit3	AD1E: Address match detection enable bit 1	The address match detection operation with the detect address setting register 1 (PADR1) is enabled or disabled. When set to "0" : Disables the address match detection operation. When set to "1" : Enables the address match detection operation. When the value of detect address setting register 1 (PADR1) matches with the value of address latch at enabling the address match detection operation (AD1E = 1), the INT9 instruction is immediately executed.
bit2	Reserved: reserved bit	Always set to "0".
bit1	AD0E: Address match detection enable bit 0	The address match detection operation with the detect address setting register 0 (PADR0) is enabled or disabled. When set to "0" : Disables the address match detection operation. When set to "1" : Enables the address match detection operation. When the value of detect address setting register 0 (PADR0) matches with the value of address latch at enabling the address match detection operation (AD0E = 1), the INT9 instruction is immediately executed.
bit0	Reserved: reserved bit	Always set to "0".

■ Address Detection Control Register 1 (PACSR1)

Figure 22-4. Address Detection Control Register 1 (PACSR1)



R/W : Readable/Writable

- : Undefined

Initial value

Table 22-2. Functions of Address Detection Control Register (PACSR1)

Bit Name		Function
bit15, bit14	Undefined: Undefined bits	Read: "1" is read. Write: No effect
bit13	AD5E: Address match detection enable bit 5	The address match detection operation with the detect address setting register 5 (PADR5) is enabled or disabled. When set to "0" : Disables the address match detection operation. When set to "1" : Enables the address match detection operation. When the value of detect address setting register 5 (PADR5) matches with the value of address latch at enabling the address match detection operation (AD5E = 1), the INT9 instruction is immediately executed.
bit12	Reserved: reserved bit	Always set to "0".
bit11	AD4E: Address match detection enable bit 4	The address match detection operation with the detect address setting register 4 (PADR4) is enabled or disabled. When set to "0" : Disables the address match detection operation. When set to "1" : Enables the address match detection operation. When the value of detect address setting register 4 (PADR4) matches with the value of address latch at enabling the address match detection operation (AD4E = 1), the INT9 instruction is immediately executed.
bit10	Reserved: reserved bit	Always set to "0".
bit9	AD3E: Address match detection enable bit 3	The address match detection operation with the detect address setting register 3 (PADR3) is enabled or disabled. When set to "0" : Disables the address match detection operation. When set to "1" : Enables the address match detection operation. When the value of detect address setting register 3 (PADR3) matches with the value of address latch at enabling the address match detection operation (AD3E = 1), the INT9 instruction is immediately executed.
bit8	Reserved: reserved bit	Always set to "0".

22.3.2 Detect Address Setting Registers (PADR0 to PADR5)

The value of an address to be detected is set in the detect address setting registers. When the address of the instruction processed by the program matches the address set in the detect address setting registers, the next instruction is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.

Detect Address Setting Registers (PADR0 to PADR5)

Figure 22-5. Detect Address Setting Registers (PADR0 to PADR5)

PADR5: High PADR2: High	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	0079F8H 0079E8H	D23	D22	D21	D20	D19	D18	D17	D16	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR5: Middle PADR2: Middle	0079F7H 0079E7H	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
		D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR5: Low PADR2: Low	0079F6H 0079E6H	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
		D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR4: High PADR1: High	0079F5H 0079E5H	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
		D23	D22	D21	D20	D19	D18	D17	D16	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR4: Middle PADR1: Middle	0079F4H 0079E4H	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
		D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR4: Low PADR1: Low	0079F3H 0079E3H	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
		D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR3: High PADR0: High	0079F2H 0079E2H	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
		D23	D22	D21	D20	D19	D18	D17	D16	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR3: Middle PADR0: Middle	0079F1H 0079E1H	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
		D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PADR3: Low PADR0: Low	0079F0H 0079E0H	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
		D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W:Readable/Writable
X:Undefined

■ Functions of Detect Address Setting Registers

- There are six detect address setting registers (PADR0 to PADR5) that consist of a high byte (bank), middle byte, and low byte, totaling 24 bits.

Table 22-3. Address Setting of Detect Address Setting Registers

Register Name	Interrupt Output Enable	Address Setting	
Detect address setting register 0 (PADR0)	PACSR0: AD0E	High	Set the upper 8 bits of detect address 0 (bank).
		Middle	Set the middle 8 bits of detect address 0.
		Low	Set the lower 8 bits of detect address 0.
Detect address setting register 1 (PADR1)	PACSR0: AD1E	High	Set the upper 8 bits of detect address 1 (bank).
		Middle	Set the middle 8 bits of detect address 1.
		Low	Set the lower 8 bits of detect address 1.
Detect address setting register 2 (PADR2)	PACSR0: AD2E	High	Set the upper 8 bits of detect address 2 (bank).
		Middle	Set the middle 8 bits of detect address 2.
		Low	Set the lower 8 bits of detect address 2.
Detect address setting register 3 (PADR3)	PACSR1: AD3E	High	Set the upper 8 bits of detect address 3 (bank).
		Middle	Set the middle 8 bits of detect address 3.
		Low	Set the lower 8 bits of detect address 3.
Detect address setting register 4 (PADR4)	PACSR1: AD4E	High	Set the upper 8 bits of detect address 4 (bank).
		Middle	Set the middle 8 bits of detect address 4.
		Low	Set the lower 8 bits of detect address 4.
Detect address setting register 5 (PADR5)	PACSR1: AD5E	High	Set the upper 8 bits of detect address 5 (bank).
		Middle	Set the middle 8 bits of detect address 5.
		Low	Set the lower 8 bits of detect address 5.

In the detect address setting registers (PADR0 to PADR5), starting address (first byte) of instruction to be replaced by INT9 instruction should be set.

Figure 22-6. Setting of Starting Address of Instruction Code to be Replaced by INT9 Instruction

Address	Instruction code	Mnemonic	
FF001C :	A8 00 00	MOVW	RW0,#0000
FF001F :	4A 00 00	MOVW	A,#0000
FF0022 :	4A 80 08	MOVW	A,#0880

Set to detect address (High: FF_H, Middle: 00_H, Low: 1F_H)

Notes:

- When an address other than the first byte is set to the detect address setting registers (PADR0 to PADR5), the instruction code is not replaced by INT9 instruction and a program of an interrupt processing is not performed. When the address is set to the second byte or subsequent, the address set by the instruction code is replaced by "01_H" (INT9 instruction code) and, which may cause malfunction.
- The detect address setting registers (PADR0 to PADR5) should be set after disabling the address match detection (PACSR: AD0E to AD5E=0) of corresponding address match control registers. If the detect address setting registers are changed without disabling the address match detection, the address match detection function will work immediately after an address match occurs during writing address, which may cause malfunction.
- The address match detection function can be used only for addresses of the internal ROM area. If addresses of the external memory area are set, the address match detection function will not work and the INT9 instruction will not be executed.

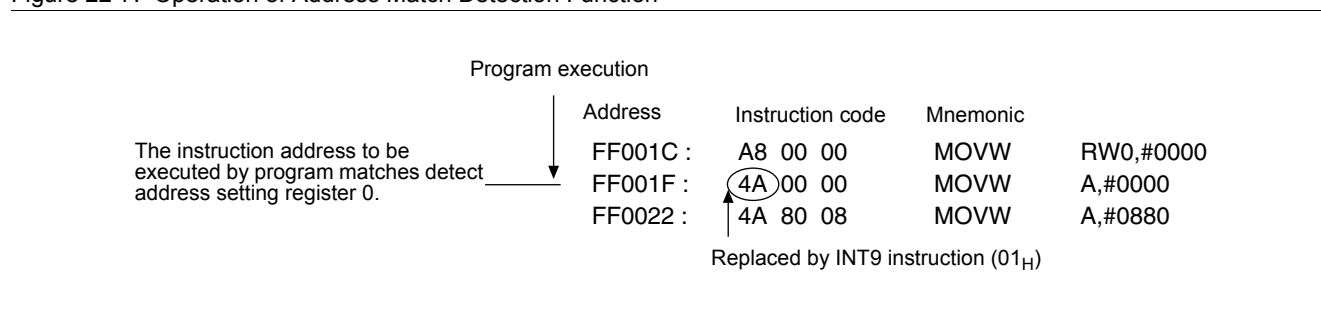
22.4 Explanation of Operation of Address Match Detection Function

If the addresses of the instructions executed in the program match those set in the detection address setting registers (PADR0 to PADR5), the address match detection function will replace the first instruction code executed by the CPU with the INT9 instruction (01_H) to branch to the interrupt processing program.

■ Operation of Address Match Detection Function

Figure 22-7. shows the operation of the address match detection function when the detect addresses are set and an address match is detected.

Figure 22-7. Operation of Address Match Detection Function



■ Setting Detect Address

- 1) Disable the detection address setting register 0 (PADR0) where the detect address is set for address match detection (PACSR0: AD0E=0).
- 2) Set the detect address in the detection address setting register 0 (PADR0). Set "FF_H" at the higher bits, "00_H" at the middle bits, and "1F_H" at the lower bits of the detection address setting register 0 (PADR0).
- 3) Enable the detect address setting register 0 (PADR0) where the detect address is set for address match detection (PACSR0: AD0E=1).

■ Program Execution

- 1) If the address of the instruction to be executed in the program matches the set detect address, the first instruction code at the matched address is replaced by the INT9 instruction code ("01_H").
- 2) INT9 instruction is executed. INT9 interrupt is generated and then interrupt processing program is executed.

22.4.1 Example of Using Address Match Detection Function

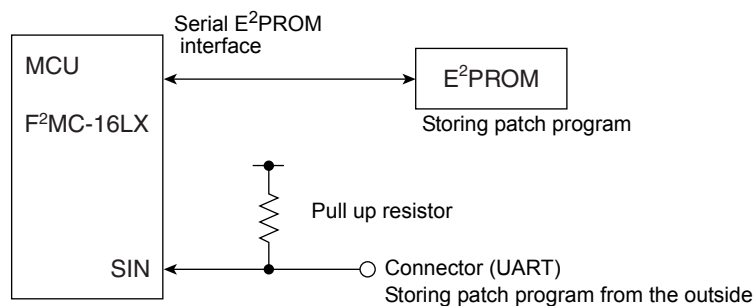
This section gives an example of patch processing for program correction using the address match detection function.

■ System Configuration and E²PROM Memory Map

- System configuration

Figure 22-8. gives an example of the system configuration using the address match detection function.

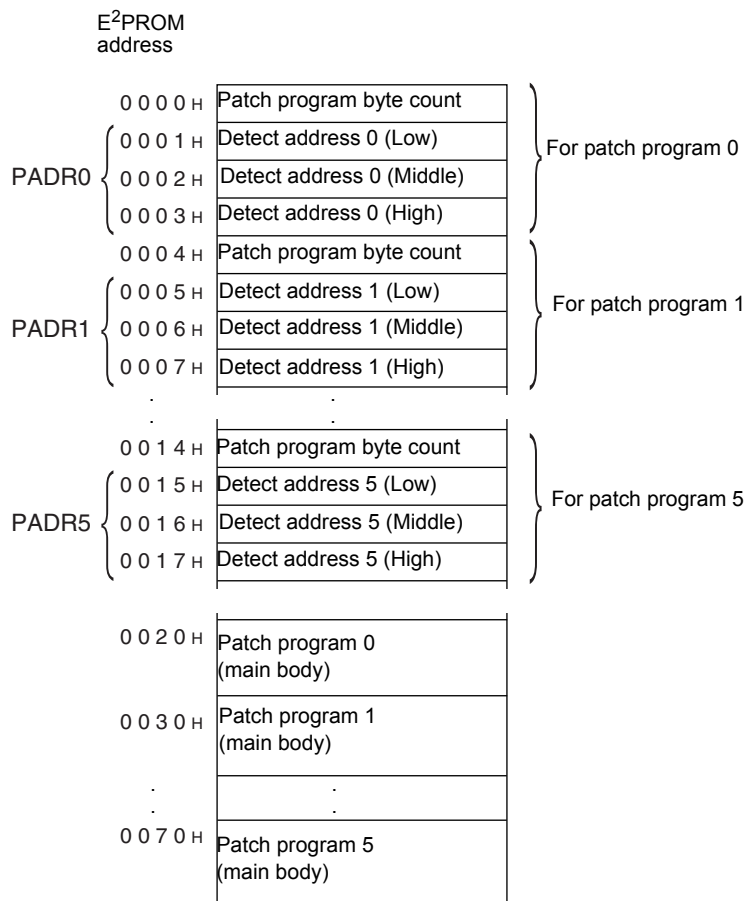
Figure 22-8. Example of System Configuration Using Address Match Detection Function



■ E²PROM Memory Map

Figure 22-9. shows the allocation of the patch program and data at storing the patch program in E²PROM.

Figure 22-9. Allocation of E²PROM Patch Program and Data



□ Patch program byte count

The total byte count of the patch program (main body) is stored. If the byte count is "00_H", it indicates that no patch program is provided.

□ Detect address (24 bits)

The address where the instruction code is replaced by the INT9 instruction code due to program error is stored. This address is set in the detection address setting registers (PADR0 to PADR5).

□ Patch program (main body)

The program executed by the INT9 interrupt processing when the program address matches the detect address is stored. Patch program 0 is allocated from any predetermined address. Patch program 1 is allocated from the address indicating <starting address of patch program 0 + total byte count of patch program 0>. It is similar for the patch program 2 to 5.

■ Setting and Operating State

□ Initialization

E²PROM data are all cleared to "00_H".

□ Occurrence of program error

By using the connector (UART), information about the patch program is transmitted to the MCU (F²MC-16LX) from the outside according to the allocation of the E²PROM patch program and data.

The MCU (F²MC-16LX) stores the information received from outside in the E²PROM.

□ Reset sequence

After reset, the MCU (F²MC-16LX) reads the byte count of the E²PROM patch program to check the presence or absence of the patch program.

If the byte count of the patch program is not "00_H", the higher, middle and lower bits at detect addresses 0 to 5 are read and set in the detection address setting registers 0 to 5 (PADR0 to PADR5). The patch program (main body) is read according to the byte count of the patch program and written to RAM in the MCU (F²MC-16LX).

The patch program (main body) is allocated to the address where the patch program is executed in the INT9 interrupt processing by the address match detection function.

Address match detection is enabled (PACSR: AD0E=1, AD1E=1 ... AD5E=1).

□ INT9 Interrupt processing

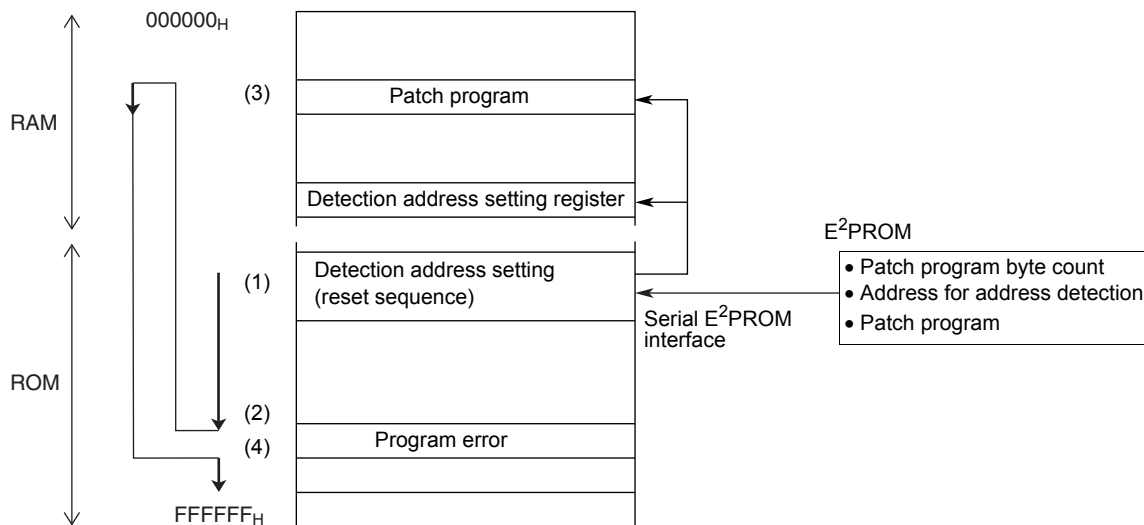
Interrupt processing is performed by the INT9 instruction. The MB90990 series has no interrupt request flag by address match detection. Therefore, if the stack information in the program counter is discarded, the detect address cannot be checked. When checking the detect address, check the value of program counter stacked in the interrupt processing routine.

The patch program is executed, branching to the normal program.

■ Operation of Address Match Detection Function at Storing Patch Program in E²PROM

Figure 22-10. shows the operation of the address match detection function at storing the patch program in E²PROM.

Figure 22-10. Operation of Address Match Detection Function at Storing Patch Program in E²PROM

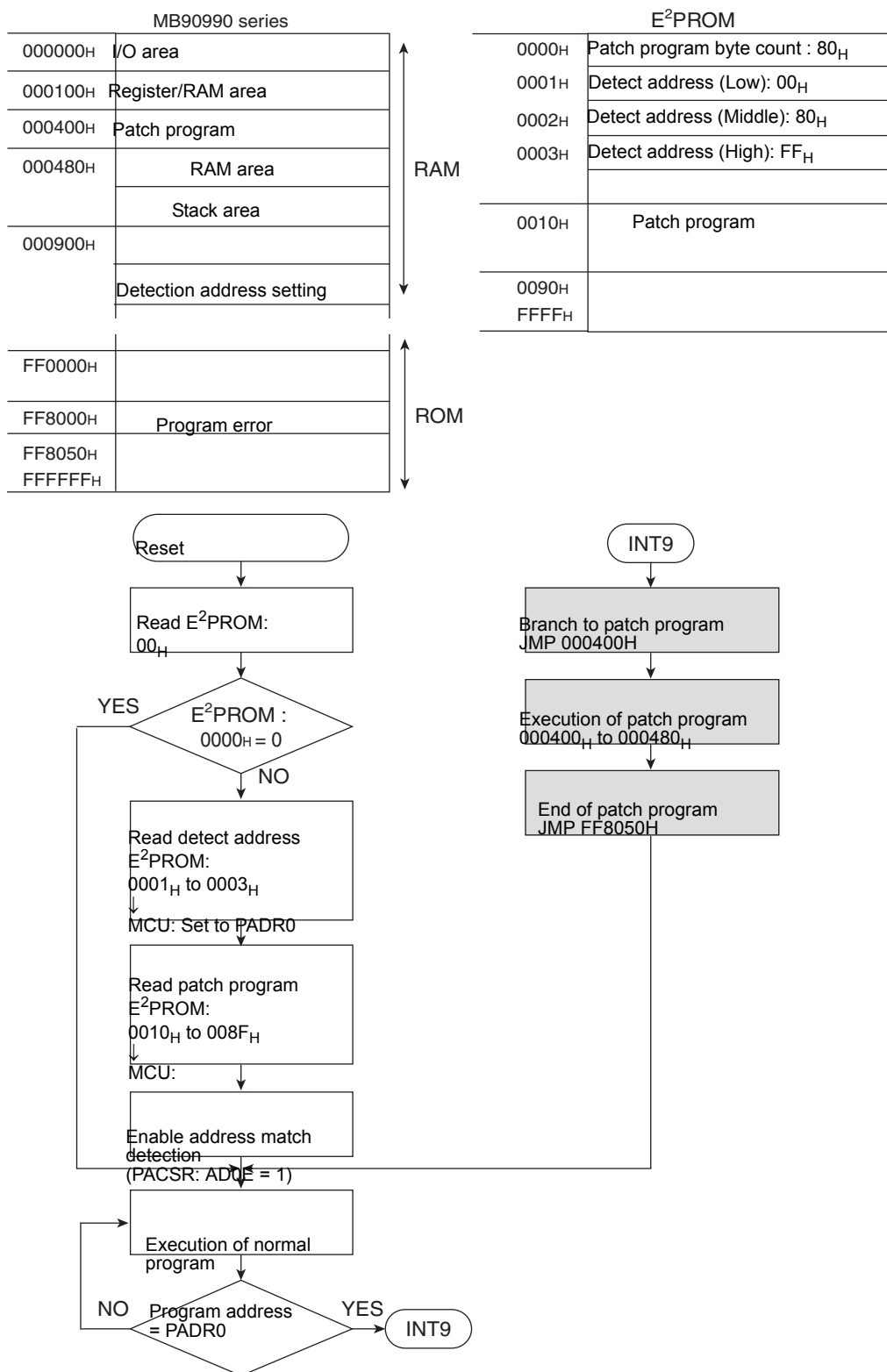


- (1) Execution of detection address setting of reset sequence and normal program
- (2) Branch to patch program which expanded in RAM with INT9 interrupt processing by address match detection
- (3) Patch program execution by branching of INT9 processing
- (4) Execution of normal program which branches from patch program

■ Flow of Patch Processing for Patch Program

Figure 22-11. shows the flow of patch processing for patch program using the address match detection function.

Figure 22-11. Flow of Patch Processing for Patch Program



22.5 Program Example of Address Match Detection Function

This section gives a program example for the address match detection function.

■ Program Example for Address Match Detection Function

□ Processing specifications

If the address of the instruction to be executed by the program matches the address set in the detection address setting register (PADR0), the INT9 instruction is executed.

□ Coding example

```

PACSR0 EQU 00009EH      ;Address detection control register 0
PADRL EQU 0079E0H       ;Detection address setting register 0
                          (Low)
PADRM EQU 0079E1H       ;Detection address setting register 0
                          (Middle)
PADRH EQU 0079E2H       ;Detection address setting register 0
                          (High)

;
;-----Main program-----
CODE CSEG
START:
                          ;Stack pointer (SP), etc.,
                          ;already initialized
      MOV PADRL,#00H      ;Set address detection register 0
                          (Low)
      MOV PADRM,#00H      ;Set address detection register 0
                          (Middle)
      MOV PADRH,#00H      ;Set address detection register 0
                          (High)

;
      MOV I:PACSR0,#00000010B ;Enable address match
      .
      Processing by user
      .
LOOP:
      .
      Processing by user
      .
      BRA LOOP
;-----Interrupt program-----
WARI:
      .
      Processing by user
      .
      RETI                ;Return from interrupt processing
CODE ENDS
;-----Vector setting-----
VECT CSEG ABS=0FFH
      ORG 00FFD8H
      DSL WARI
      ORG 00FFDCH          ;Set reset vector
      DSL START
      DB 00H                ;Set to single-chip mode
VECT ENDS
      END START
  
```

23. ROM Mirroring Function Select Module



This chapter describes the functions and operations of the ROM mirroring function select module.

23.1 Overview of ROM Mirroring Function Select Module

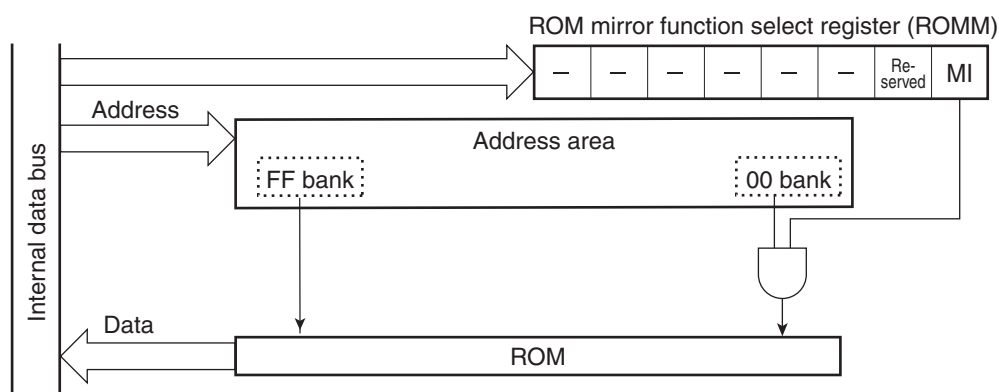
23.2 ROM Mirroring Function Select Register (ROMM)

23.1 Overview of ROM Mirroring Function Select Module

The ROM mirroring function select module provides a setting so that ROM data in the FF bank can be read by access to the 00 bank.

■ Block Diagram of ROM Mirroring Function Select Module

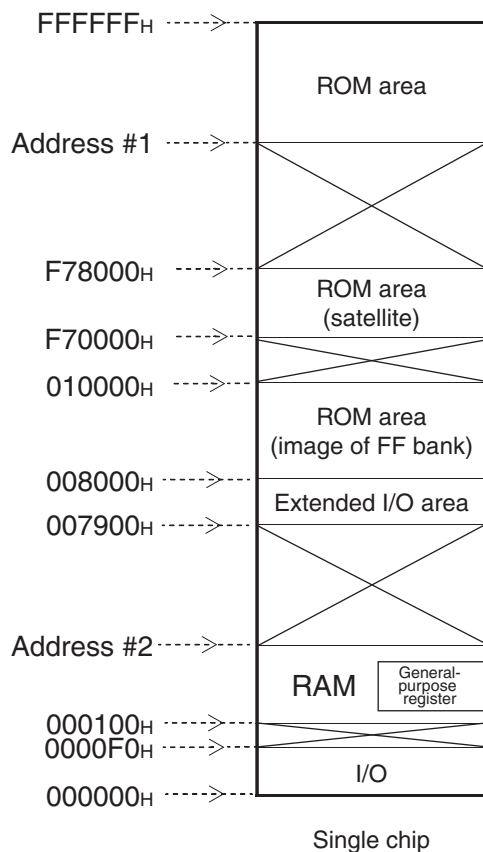
Figure 23-1. Block Diagram of ROM Mirroring Function Select Module



■ Memory Space When ROM Mirroring Function Enabled/Disabled

Figure 23-2. shows the availability of access to memory space when the ROM mirroring function is enabled or disabled.

Figure 23-2. Memory Space When ROM Mirroring Function Enabled/Disabled



Product type	Address #1	Address #2
MB90F997JBS, MB90F997MBS	FE0000 _H	0020FF _H
MB90V950AJAS, MB90V950AMAS	F80000 _H	007900 _H

■ List of Registers and Initial Values of ROM Mirroring Function Select Module

Figure 23-3. List of Registers and Initial Values of ROM Mirroring Function Select Module

bit	15	14	13	12	11	10	9	8
ROM mirror function select register (ROMM)	1	1	1	1	1	1	0	1

23.2 ROM Mirroring Function Select Register (ROMM)

The ROM mirroring function select register (ROMM) enables or disables the ROM mirroring function. When the ROM mirroring function is enabled, ROM data in the FF bank can be read by access to the 00 bank.

■ ROM Mirroring Function Select Register (ROMM)

Figure 23-4. ROM Mirroring Function Select Register (ROMM)

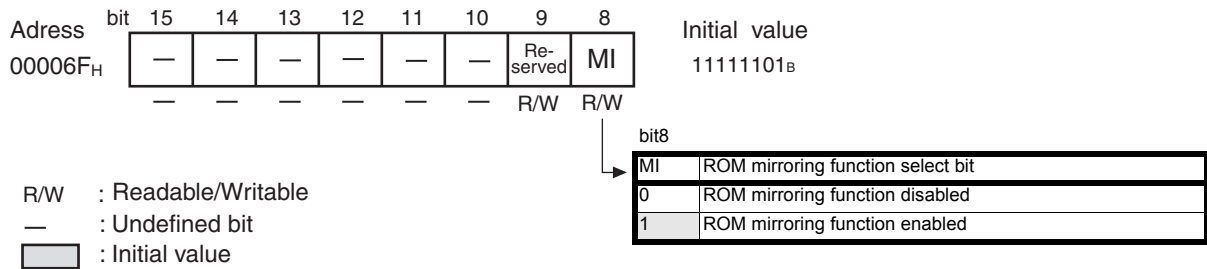


Table 23-1. Functions of ROM Mirroring Function Select Register (ROMM)

Bit Name		Function
bit15 to bit10	Undefined bits	Read: Value is undefined. Write: No effect
bit9	Reserved bit	Always write "0" to this bit.
bit8	MI: ROM mirroring func- tion select bit	This bit enables or disables the ROM mirroring function. When set to "0": Disables ROM mirroring function. When set to "1": Enables ROM mirroring function.

Note:

While the ROM area at addresses "008000_H" to "00FFFF_H" is being used, access to the ROM mirroring function select register (ROMM) is prohibited.

24. Flash Memory



This chapter explains the functions and operation of the flash memory.

The following three methods are available for writing data to and erasing data from the flash memory:

- **Parallel programmer**
- **Serial programmer**
- **Executing programs to write/erase data**

This chapter explains “Executing programs to write/erase data”.

[24.1 Overview of Flash Memory](#)

[24.2 Block Diagram of the Entire Flash Memory](#)

[24.3 Sector Configuration of the Flash Memory](#)

[24.4 Write/Erase Modes](#)

[24.5 Flash Memory Control Status Register \(FMCS\)](#)

[24.6 Flash Memory Write Control Register \(FWR0/FWR1\)](#)

[24.7 Starting the Flash Memory Automatic Algorithm](#)

[24.8 Confirming the Automatic Algorithm Execution State](#)

[24.9 Writing to and Erasing Flash Memory](#)

[24.10 Notes on Using Flash Memory](#)

[24.11 Flash Security Feature](#)

24.1 Overview of Flash Memory

The MB90990 series has two sets of flash memory. The first set of flash memory is called "main flash", which has a sector configuration of "48K × 2 + 8K × 4" and is located in the FF_H to FE_H banks on the CPU memory map. The second set of flash memory is called "satellite flash", which has a sector configuration of "8K × 4" and is located in the F7_H bank on the CPU memory map. This configuration allows the user to execute a flash writing program on the main flash program and write data to or erase data from the satellite flash. The function of the flash memory interface circuit enables read access and program access from the CPU, just like the mask ROM. Data can be written to or erased from the flash memory via the flash memory interface circuit by an instruction operation from the CPU. This allows data to be rewritten during the implementation state by the internal CPU control, resulting in efficient program and data improvements.

■ Features of Flash Memory

- Use of automatic program algorithm (Embedded Algorithm: Equivalent to MBM29LV200)
- Erase pause/restart function 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

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 same flash memory cannot be used to write/erase and read data at the same time. This means that when writing and erasing data from the main flash memory, only the write operation can be performed without accessing the program from the flash memory, by first copying the program stored in the flash memory and then executing it through RAM. When writing and erasing data from the satellite flash memory, the writing can be done by either executing the program on the main flash memory, or first copying the program stored in the flash memory and then executing it through RAM.

Note:

Do not place any program in the satellite flash memory.

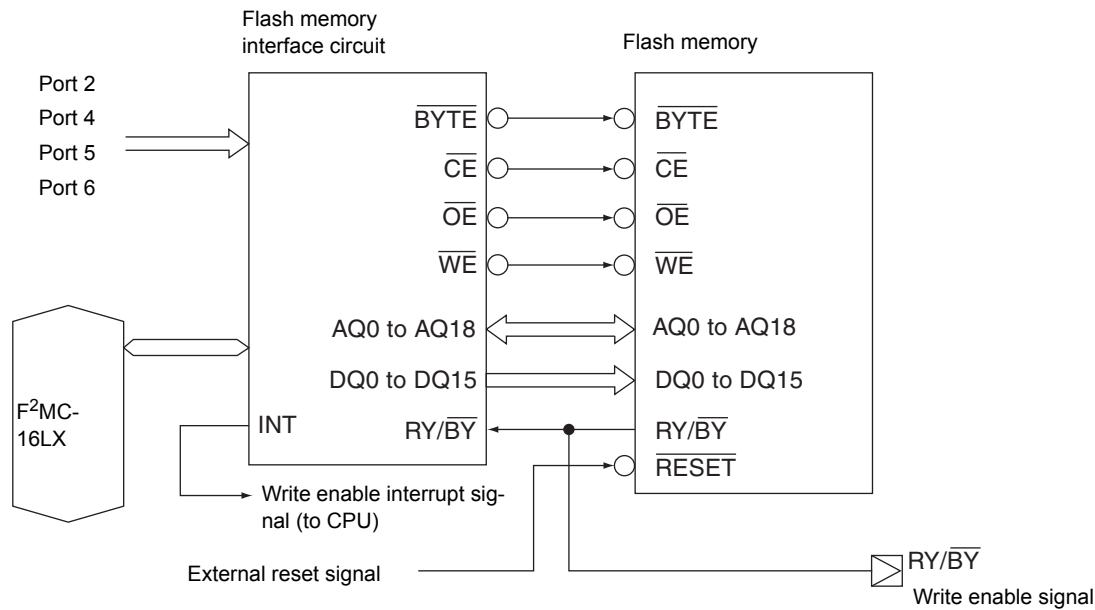
24.2 Block Diagram of the Entire Flash Memory

This section shows a block diagram of the entire flash memory .

■ Block Diagram of the Entire Flash Memory

Figure 24-1. shows a block diagram of the entire flash memory with the flash memory interface circuit included.

Figure 24-1. Block Diagram of the Entire Flash Memory



24.3 Sector Configuration of the Flash Memory

This section shows the sector configuration of the flash memory .

■ Sector Configuration of the Flash Memory

Figure 24-2. shows the sector configuration of the flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

Figure 24-2. Sector Configuration of the Flash Memory

	Writer Address*	CPU Address
SA5(8Kbytes)	FFFFF _H	FFFFFF _H
SA4(8Kbytes)	FDFFF _H	FFDFFF _H
SA3(48Kbytes)	FBFFF _H	FFBFFF _H
SA2(48Kbytes)	EFFFF _H	FEFFFF _H
SA1(8Kbytes)	E3FFF _H	FE3FFF _H
SA0(8Kbytes)	E1FFF _H	FE1FFF _H
Interval area	DFFFF _H	FDFFFF _H
SB3(8Kbytes)	77FFF _H	F77FFF _H
SB2(8Kbytes)	75FFF _H	F75FFF _H
SB1(8Kbytes)	73FFF _H	F73FFF _H
SB0(8Kbytes)	71FFF _H	F71FFF _H
	70000 _H	F70000 _H

*: The Writer 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.

24.4 Write/Erase Modes

The flash memory can be accessed in 2 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_B" while the reset signal is asserted. The flash memory interface circuit is connected directly to ports 2, 4, 5, 6 and 8, 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 F7/FE bank to FF bank 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.

Note:

Sector protection function such as standard flash memory MBM29LV200 does not contain in the flash memory mode.

24.5 Flash Memory Control Status Register (FMCS)

This section shows the function of the flash memory control status register (FMCS).

■ Flash Memory Control Status Register (FMCS)

Figure 24-3. Flash Memory Control Status Register (FMCS)

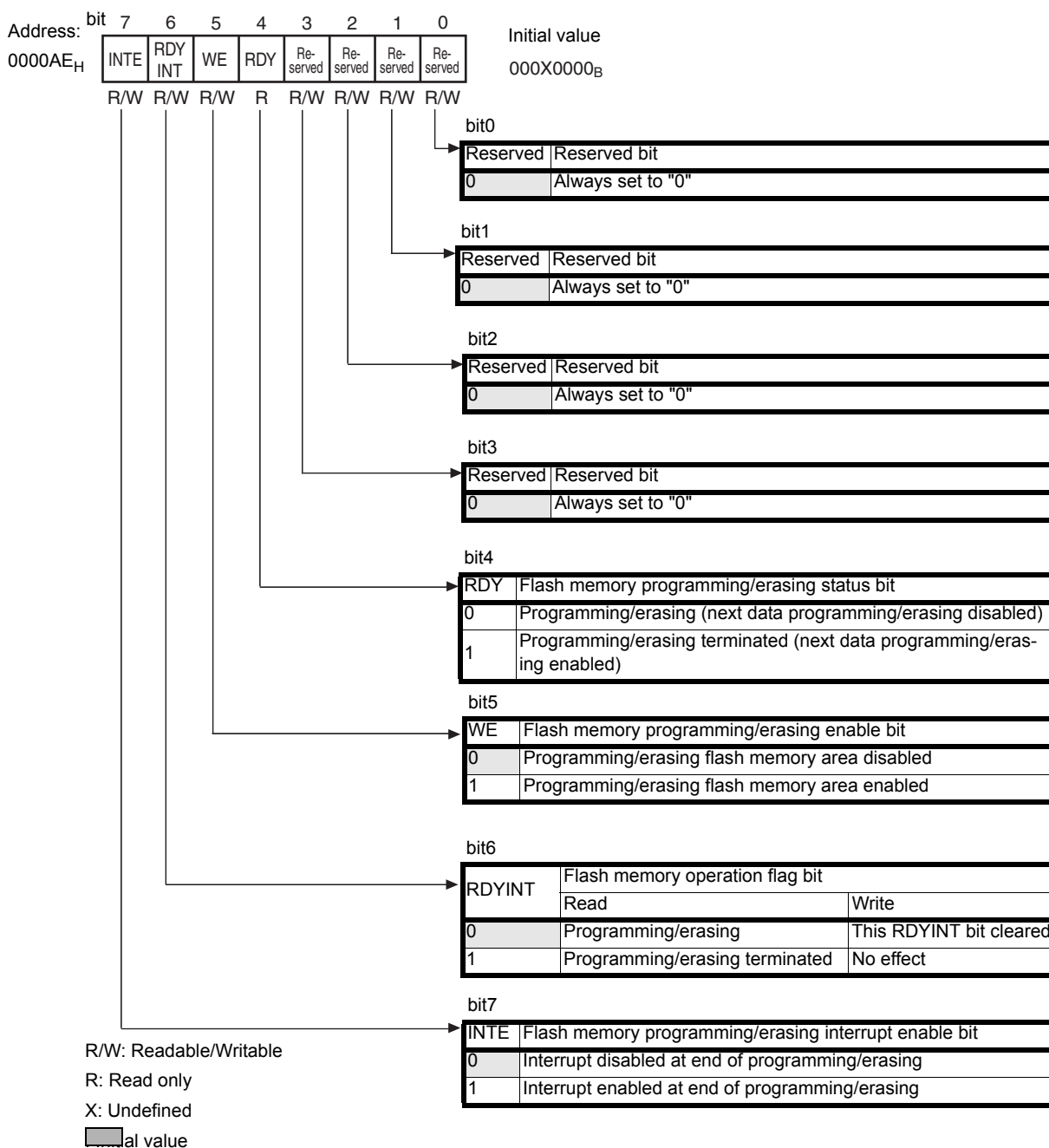


Table 24-1. Functions of Flash Memory Control Status Register (FMCS)

Bit Name		Function
bit7	INTE: Flash memory programming/erasing interrupt enable bit	<p>This bit enables or disables an interrupt request as programming/erasing flash memory is terminated.</p> <p>When set to "1":</p> <p>If the flash memory operation flag bit is set to "1" (FMCS: RDYINT=1), an interrupt is requested.</p>
bit6	RDYINT: Flash memory operation flag bit	<p>This bit shows the operating state of flash memory.</p> <p>If programming/erasing flash memory is terminated, the RDYINT bit is set to "1" in timing of termination of the flash memory automatic algorithm.</p> <ul style="list-style-type: none"> If the RDYINT bit is set to "1" when an interrupt as programming/erasing flash memory is terminated is enabled (FMCS:INTE = 1), an interrupt is requested. If the RDYINT bit is "0", programming/erasing flash memory is disabled. <p>When set to "0": Cleared.</p> <p>When set to "1": No effect</p> <p>If the read-modify-write (RMW) instructions are used, "1" is always read.</p>
bit5	WE: Flash memory programming/erasing enable bit	<p>This bit enables or disables the programming/erasing of flash memory area.</p> <p>The WE bit should be set before starting the command to program/erase flash memory.</p> <p>When set to "0":</p> <p>No program/erase signal is generated even if inputting to program/erase command to the F7/FE bank to FF bank.</p> <p>When set to "1":</p> <p>Programming/erasing flash memory is enabled after inputting program/erase command to the F7/FE bank to FF bank.</p> <p>When not performing programming/erasing, the WE bit should be set to "0" so as not to accidentally program or erase flash memory.</p>
bit4	RDY: Flash memory programming/erasing status bit	<p>This bit shows the programming/erasing status of flash memory.</p> <p>If the RDY bit is "0", programming/erasing flash memory is disabled.</p> <p>Even while this bit is "0", a read/reset command and sector erase pause command are accepted.</p> <p>When writing/erasing operation is ended, "1" is set.</p>
bit3 to bit0	Reserved: Reserved bits	Always set to "0".

Notes:

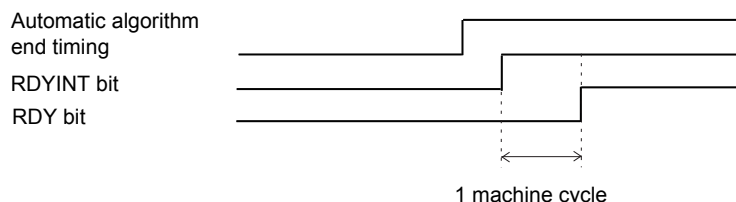
This register can be accessed only in byte-access mode.

■ Automatic Algorithm End Timing

Figure 24-4. illustrates the relationship between the automatic algorithm end timing and the respective RDYINT and RDY bits.

The RDYINT and RDY bits do not simultaneously change. Create the program to determine the end of the automatic algorithm with either of the bits.

Figure 24-4. Relationship between Automatic Algorithm End Timing and RDYINT and RDY bits



24.6 Flash Memory Write Control Register (FWR0/FWR1)

The flash memory write control register (FWR0/FWR1) is a register located on the flash memory interface, which is used to set the function that prevents accidental writing to the flash memory.

■ Flash Memory Write Control Register (FWR0/FWR1)

The flash memory write control register (FWR0/FWR1) contains write-enable/disable setting bits that correspond to each sector (SA0 to SA7, SB0 to SB3). The initial value indicates "0", which disables write operations. Writing "1" enables write operations for the corresponding sector. Writing "0" turns on the function to prevent accidental write operations. Therefore, once "0" is written, even if "1" is written afterwards, data cannot be written to that sector. If data must be rewritten, a reset must be performed.

Figure 24-5. 1M-bit Flash Memory Write Control Register (FWR0/FWR1)

FWR0	bit	7	6	5	4	3	2	1	0
Address:0079A2 _H		-	-	SA5E	SA4E	SA3E	SA2E	SA1E	SA0E
		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

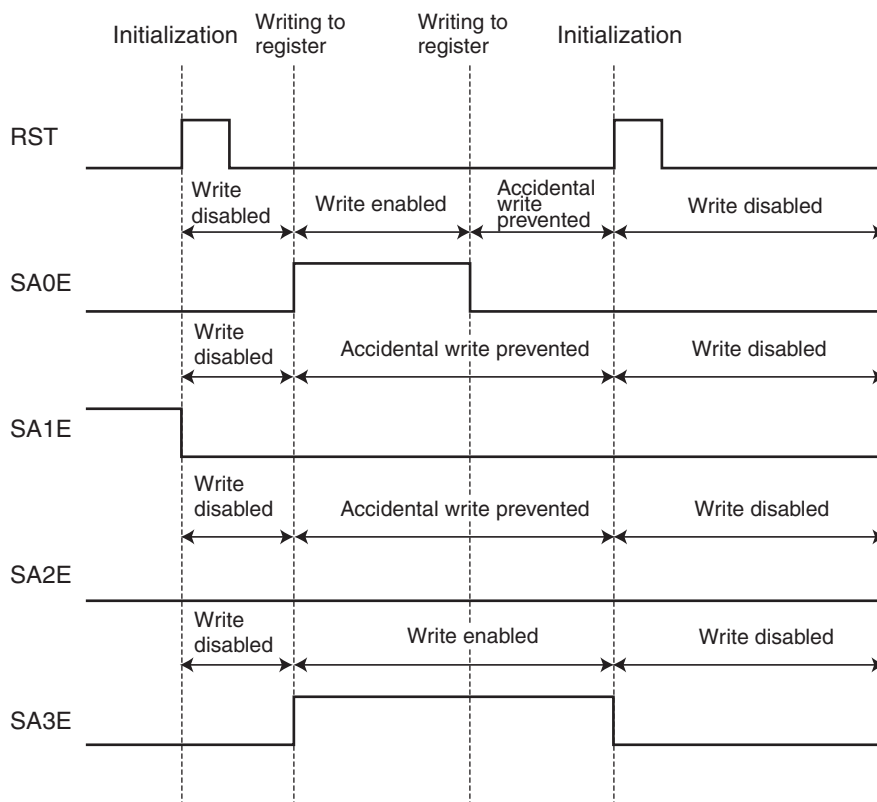
FWR1	bit	15	14	13	12	11	10	9	8
Address:0079A3 _H		SB3E	SB2E	SB1E	SB0E	-	-	-	-
		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

R/W : Readable/Writable
 0 : Write disabled [initial value]

Table 24-2. Function of Flash Memory Write Control Register (FWR0/FWR1)

Bit Name		Function																																	
bit11 to bit6	Reserved bits	Be sure to write "0" to these bits. The read value is undefined.																																	
bit15 to bit12, bit5 to bit0	SB3E to SB0E, SA5E to SA0E Accidental write prevention function setting bits	<p>These bits set the accidental write prevention function for each corresponding sector of the flash memory. Writing "1" enables writing to the corresponding sector. Writing "0" activates the accidental write prevention function for the corresponding sector. In addition, these bits are initialized to "0" (writing disabled) at a reset.</p> <p>Flash sector correspondence table of accidental write prevention function setting bits (1M-bit flash memory)</p> <table border="1"> <thead> <tr> <th>Bit</th><th>Bit Name</th><th>Corresponding sector of flash memory</th></tr> </thead> <tbody> <tr><td>15</td><td>SB3E</td><td>SB3</td></tr> <tr><td>14</td><td>SB2E</td><td>SB2</td></tr> <tr><td>13</td><td>SB1E</td><td>SB1</td></tr> <tr><td>12</td><td>SB0E</td><td>SB0</td></tr> <tr><td>5</td><td>SA5E</td><td>SA5</td></tr> <tr><td>4</td><td>SA4E</td><td>SA4</td></tr> <tr><td>3</td><td>SA3E</td><td>SA3</td></tr> <tr><td>2</td><td>SA2E</td><td>SA2</td></tr> <tr><td>1</td><td>SA1E</td><td>SA1</td></tr> <tr><td>0</td><td>SA0E</td><td>SA0</td></tr> </tbody> </table> <p>Write disabled:"0" state. It is possible to write ("1") the register corresponding to each sector while "0" is not written in the flash memory write control register (FWR0/FWR1) (post-reset state).</p> <p>Write enabled:"1" state. It is possible to write data to the corresponding sector.</p> <p>Accidental write prevented:"0" state. Write operations cannot be enabled ("1"), even if "1" is written to the register corresponding to each sector while "0" is written in the flash memory write control register (FWR0/ FWR1).</p>	Bit	Bit Name	Corresponding sector of flash memory	15	SB3E	SB3	14	SB2E	SB2	13	SB1E	SB1	12	SB0E	SB0	5	SA5E	SA5	4	SA4E	SA4	3	SA3E	SA3	2	SA2E	SA2	1	SA1E	SA1	0	SA0E	SA0
Bit	Bit Name	Corresponding sector of flash memory																																	
15	SB3E	SB3																																	
14	SB2E	SB2																																	
13	SB1E	SB1																																	
12	SB0E	SB0																																	
5	SA5E	SA5																																	
4	SA4E	SA4																																	
3	SA3E	SA3																																	
2	SA2E	SA2																																	
1	SA1E	SA1																																	
0	SA0E	SA0																																	

Figure 24-6. Examples of Write-disabled/-enabled State and Accidental Write Prevented State to Flash Memory Using Flash Memory Write Control Register (FWR0/FWR1)



Write disabled:

"0" state. It is possible to write ("1") the register corresponding to each sector while "0" is not written in the flash memory write control register (FWR0/FWR1) (post-reset state).

Write enabled:

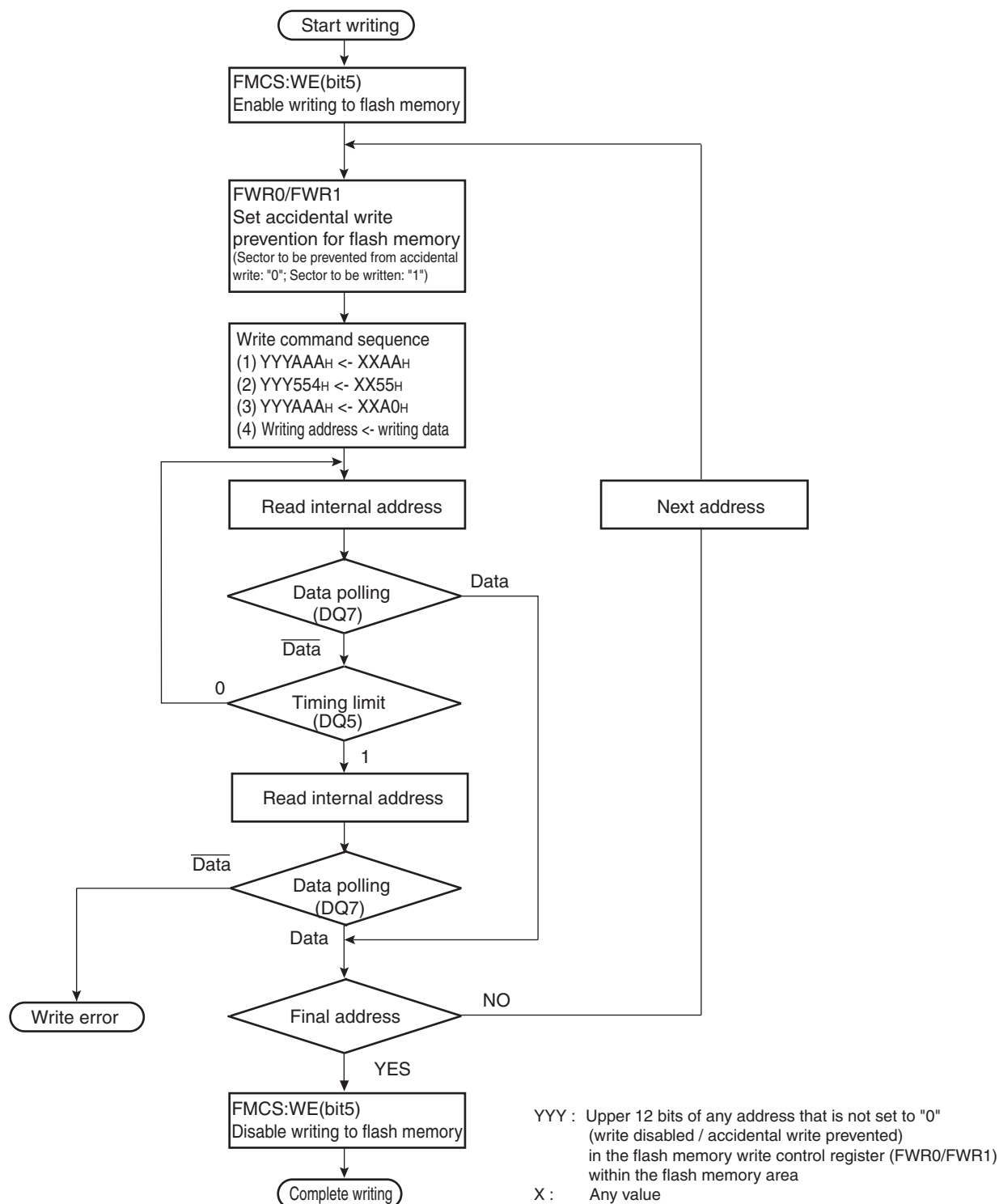
"1" state. It is possible to write data to the corresponding sector.

Accidental write prevented:

"0" state. Write operations cannot be enabled ("1"), even if "1" is written to the register corresponding to each sector while "0" is written in the flash memory write control register (FWR0/ FWR1).

■ Setup Flow of Flash Memory Write Control Register (FWR0/FWR1)

After setting the FMCS:WE bit, set the flash memory write control register (FWR0/FWR1) to "1" for a sector to be written, and set the register to "0" for a sector to be prevented from being accidentally written. Also, the setup by a bit manipulation instruction is prohibited.



■ Setting FMCS:WE

When writing to the flash memory, set FMCS:WE to "1" to enable write operations first, then set the flash memory write control register (FWR0/FWR1). When FMCS:WE is set to disable write operations ("0"), no write operation is performed for the flash memory, even if such write operation is enabled by the flash memory write control register (FWR0/FWR1).

24.7 Starting the Flash Memory Automatic Algorithm

Four types of commands are available for starting the flash memory automatic algorithm: Read/Reset, Write, Chip Erase, and Sector Erase. Control of suspend and restart is enabled for Sector Erase.

■ Command Sequence Table

Table 24-3. lists the commands used for flash memory write/erase. All of the data written to the command register is in bytes, but use word access to write. The data of the high-order bytes at this time is ignored.

Table 24-3. 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	YYYXXH	XXF0H	-	-	-	-	-	-	-	-	-	-
Read/reset*	4	YYYYAAH	XXAAH	YYY554H	XX55H	YYYYAAH	XXF0H	RA	RD	-	-	-	-
Write	4	YYYYAAH	XXAAH	YYY554H	XX55H	YYYYAAH	XXA0H	PA	PD	-	-	-	-
Chip erase	6	ZZZAAAH	XXAAH	ZZZ554H	XX55H	ZZZAAAH	XX80H	ZZZAAAH	XXAAH	ZZZ554H	XX55H	ZZZAAAH	XX10H
Sector erase	6	YYYYAAH	XXAAH	YYY554H	XX55H	YYYYAAH	XX80H	YYYYAAH	XXAAH	YYY554H	XX55H	SA	XX30H
Suspension of Sector Erase		Entering address YYYXXH and data "XXB0H" suspends erase operation during Sector Erase.											
Restart of Sector Erase		Entering address YYYXXH and data "XX30H" restarts Erase after suspension of Sector Erase.											

RA: Reading address

PA: Writing address

SA: Sector address (specify any address in the sector)

RD: Reading data

PD: Writing data

YYY: Upper 12 bits of any address that is not set to "0" (write disabled / accidental write prevented) in the flash memory write control register (FWR0/FWR1) within the flash memory area

ZZZ: Upper 12 bits of any address within the flash memory area

*: Both of the two types of read/reset commands can reset the flash memory to the read mode.

Notes:

- Addresses in the table are the values in the CPU memory map. All addresses and data are hexadecimal values, where "x" is any value.
- If the Chip Erase command is issued by accessing a write-enabled sector when write-enabled and write-disabled sectors coexist, the content of all the sectors is erased, including the write-disabled sectors.

■ Notes on Issuing Commands

The following points must be noted when issuing the commands on the command sequence table.

- Enable write operations for each sector before issuing the 1st command.

If the above action is not taken, the command will not be recognized properly and it will be necessary to perform a reset to initialize the command sequencer in the flash memory.

- If a Chip Erase command is issued for the main flash area (SA0 to SA5), the main flash area (SA0 to SA5) will be erased. If a Chip Erase command is issued for the satellite flash area (SB0 to SB3), the satellite flash area (SB0 to SB3) will be erased.

- If a Read/Reset command is issued during the execution of the automatic algorithm, the Reset command will be ignored and the currently executing command will continue.

24.8 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 flag.

■ Hardware Sequence Flags

The hardware sequence flags are configured from the four-bit output of DQ7, DQ6, DQ5, and DQ3.

The functions of these bits are those of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit exceeded flag (DQ5), and sector erase timer flag (DQ3). 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 referred by read-accessing the addresses of the target sectors in the flash memory after setting of the command sequence (see [Table 24-3](#) in Section "24.7 Starting the Flash Memory Automatic Algorithm"). [Table 24-4](#) lists the bit assignments of the hardware sequence flags.

Table 24-4. Bit Assignments of Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

To determine whether automatic writing or chip 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 a second or subsequent sector erase code write is valid. The following sections describe each hardware sequence flag separately. [Table 24-5](#) lists the functions of the hardware sequence flags.

Table 24-5. Hardware Sequence Flag Functions

State		DQ7	DQ6	DQ5	DQ3
State change for normal operation	Write → Write completed (write address specified)	$\overline{DQ7} \rightarrow \text{DATA:7}$	Toggle → DATA:6	0 → DATA:5	0 → DATA:3
	Chip erase → Sector erase completed	0 → 1	Toggle	0 → 1	1
	Sector erase → Erase completed	Timeout period	1	Toggle	0
		Erase	0 → 1	Toggle	0 → 1
	Sector erase wait → Erase started	1 → 0	Toggle	0	0 → 1
	Erase → Sector erase suspended (sector being erased)	0 → 1	Toggle → 1	0	1 → 0
	Sector erase suspend → Erase restarted (sector being erased)	1 → 0	1 → Toggle	0	0 → 1
	Sector erase suspended (sector not being erased)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Write	DQ7	Toggle	1	0
	Chip erase	0	Toggle	1	1

24.8.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) uses the data polling function to post that the automatic algorithm is being executed or has terminated.

■ Data Polling Flag (DQ7)

Table 24-6. and Table 24-7. list the state transitions of the data polling flag.

Table 24-6. State Transition of Data Polling Flag (State Change at Normal Operation)

Operating State	Programming → Completed	Chip/sector Erasing → 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	$\overline{DQ7} \rightarrow \text{DATA:7}$	0 → 1	0	0 → 1	1 → 0	DATA:7

Table 24-7. State Transition of Data Polling Flag (State Change at Abnormal Operation)

Operating State	Programming	Chip/Sector Erasing
DQ7	$\overline{DQ7}$	0

□ Write

Read-access during execution of the automatic write algorithm causes the flash memory to output the opposite data of bit7 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 bit7 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 erase 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 bit7 (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 sector erase suspended state and which sector is being erased.

Note:

When the automatic algorithm is being started, read-access to the specified address is ignored. Since termination of the data polling flag (DQ7) can be accepted for a data read and other bits output, data read after the automatic algorithm has terminated should be performed after read-access has confirmed that data polling has terminated.

24.8.2 Toggle Bit Flag (DQ6)

Like the data polling flag (DQ7), the toggle bit flag (DQ6) uses the toggle bit function to post that the automatic algorithm is being executed or has terminated.

■ Toggle Bit Flag (DQ6)

Table 24-8. and Table 24-9. list the state transitions of the toggle bit flag.

Table 24-8. State Transition of Toggle Bit Flag (State Change at Normal Operation)

Operating State	Programming → Completed	Chip/sector Erasing → Completed	Sector erase wait → Started	Sector erase → Erase suspend (sector being erased)	Sector erase suspend → Restarted (sector being erased)	Sector erase suspend (sector not being erased)
DQ6	Toggle → DATA:6	Toggle → Stop	Toggle	Toggle → 1	1 → Toggle	DATA:6

Table 24-9. State Transition of Toggle Bit Flag (State Change at Abnormal Operation)

Operating State	Programming	Chip/Sector Erasing
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 bit6 and output bit6 (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 bit6 (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 2 μ s and without any data being rewritten.

For an erase, if all of the selected sectors are rewrite-protected, the toggle bit performs toggling for approximately 100 μ s and then returns to the read/reset state without any data being rewritten.

24.8.3 Timing Limit Exceeded Flag (DQ5)

The timing limit exceeded flag (DQ5) is used to post that execution of the automatic algorithm has exceeded the time (internal pulse count) prescribed in the flash memory.

■ **Timing Limit Exceeded Flag (DQ5)**

Table 24-10. and Table 24-11. list the state transitions of the timing limit exceeded flag.

Table 24-10. State Transition of Timing Limit Exceeded Flag (State Change at Normal Operation)

Operating State	Programming → Completed	Chip/Sector Erasing → Completed	Sector erase wait → Started	Sector erase → Erase suspend (sector being erased)	Sector erase suspend → Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ5	0 → DATA:5	0 → 1	0	0	0	DATA:5

Table 24-11. State Transition of Timing Limit Exceeded Flag (State Change at Abnormal Operation)

Operating State	Programming	Chip/Sector Erasing
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. In rare cases normal termination may be seen as with the case where "1" can be written. As a result, valid data will not be outputted 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 not been used correctly. When this state occurs, execute the Reset command.

24.8.4 Sector Erase Timer Flag (DQ3)

The sector erase timer flag (DQ3) is used to post whether the automatic algorithm is being executed during the sector erase wait period after the Sector Erase command has been started.

■ Sector Erase Timer Flag (DQ3)

Table 24-12. and Table 24-13. list the state transitions of the sector erase timer flag.

Table 24-12. State Transitions of Sector Erase Timer Flag (State Change for Normal Operation)

Operating State	Programming → Completed	Chip/sector Erasing → Completed	Sector erase wait ® Started	Sector erase ® Erase suspend (sector being erased)	Sector erase suspend ® Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ3	0 → DATA:3	1	0 → 1	1 → 0	0 → 1	DATA:3

Table 24-13. State Transitions of Sector Erase Timer Flag (State Change for Abnormal Operation)

Operating State	Programming	Chip/Sector Erasing
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.

When 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, please check the state of this flag 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.

□ Suspending sector erase

Read-access during execution of sector erase suspend causes the flash memory to output "0" if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit3 (DATA: 3) of the read value of the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

24.9 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 Writing to and Erasing Flash Memory

The flash memory executes the automatic algorithm by issuing a command sequence (see [Table 24-3](#) in Section "24.7 Starting the Flash Memory Automatic Algorithm") for a write cycle to the bus to perform Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, or Sector Erase Restart operations. Each bus write cycle must be performed continuously. In addition, whether the automatic algorithm has terminated can be determined using the data polling or other function. At normal termination, the flash memory is returned to the read/reset state.

Each operation of the flash memory is described in the following order:

- ❑ Setting the read/reset state
- ❑ Writing data
- ❑ Erasing all data (erasing chips)
- ❑ Erasing optional data (erasing sectors)
- ❑ Suspending sector erase
- ❑ Restarting sector erase

24.9.1 Setting the Flash Memory to the Read/Reset State

This section describes the procedure for issuing the Read/Reset command to set the flash memory to the read/reset state.

■ Setting the Flash Memory to the Read/reset State

The flash memory can be set to the read/reset state by sending the Read/Reset command in the command sequence table (see [Table 24-3](#) in Section "24.7 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 fourth bus operations. However, there are no essential differences between these command sequences.

The read/reset state is the initial state of the flash memory. When the power is turned on and when a command terminates normally, the flash memory is set to the read/reset state. In the read/reset state, other commands wait for input.

In the read/reset state, data is read by regular read-access. As with the mask ROM, program access from the CPU is enabled. The Read/Reset command is not required to read data by a regular read. The Read/Reset command is mainly used to initialize the automatic algorithm in such cases as when a command does not terminate normally.

24.9.2 Writing Data to the Flash Memory

This section describes the procedure for issuing the Write command to write data to the flash memory.

■ Writing Data to the Flash Memory

The data write automatic algorithm of the flash memory can be started by sending the Write command in the command sequence table (see [Table 24-3](#) in section "24.7 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 address can be specified as the write address specified in a write data cycle, odd addresses cannot be written correctly. That is, writing to even addresses must be done in units of word data.

Writing can be done in any order of addresses or even if the sector boundary is exceeded. However, the Write command writes only data of one word for each execution.

- ❑ Notes on writing data

Writing cannot return data 0 to data 1. When data 1 is written to data 0, the data polling algorithm (DQ7) or toggle operation (DQ6) does not terminate and the flash memory elements are determined to be faulty. If the time prescribed for writing is thus exceeded, the timing limit exceeded flag (DQ5) is determined to be an error. Otherwise, the data is viewed as if dummy data 1 had been written. However, when data is read in the read/reset state, the data remains "0". Data 0 can be set to data 1 only by erase operations.

All commands are ignored during execution of the automatic write algorithm. If a hardware reset is started during writing, the data of the written addresses will be unpredictable.

■ Writing to the Flash Memory

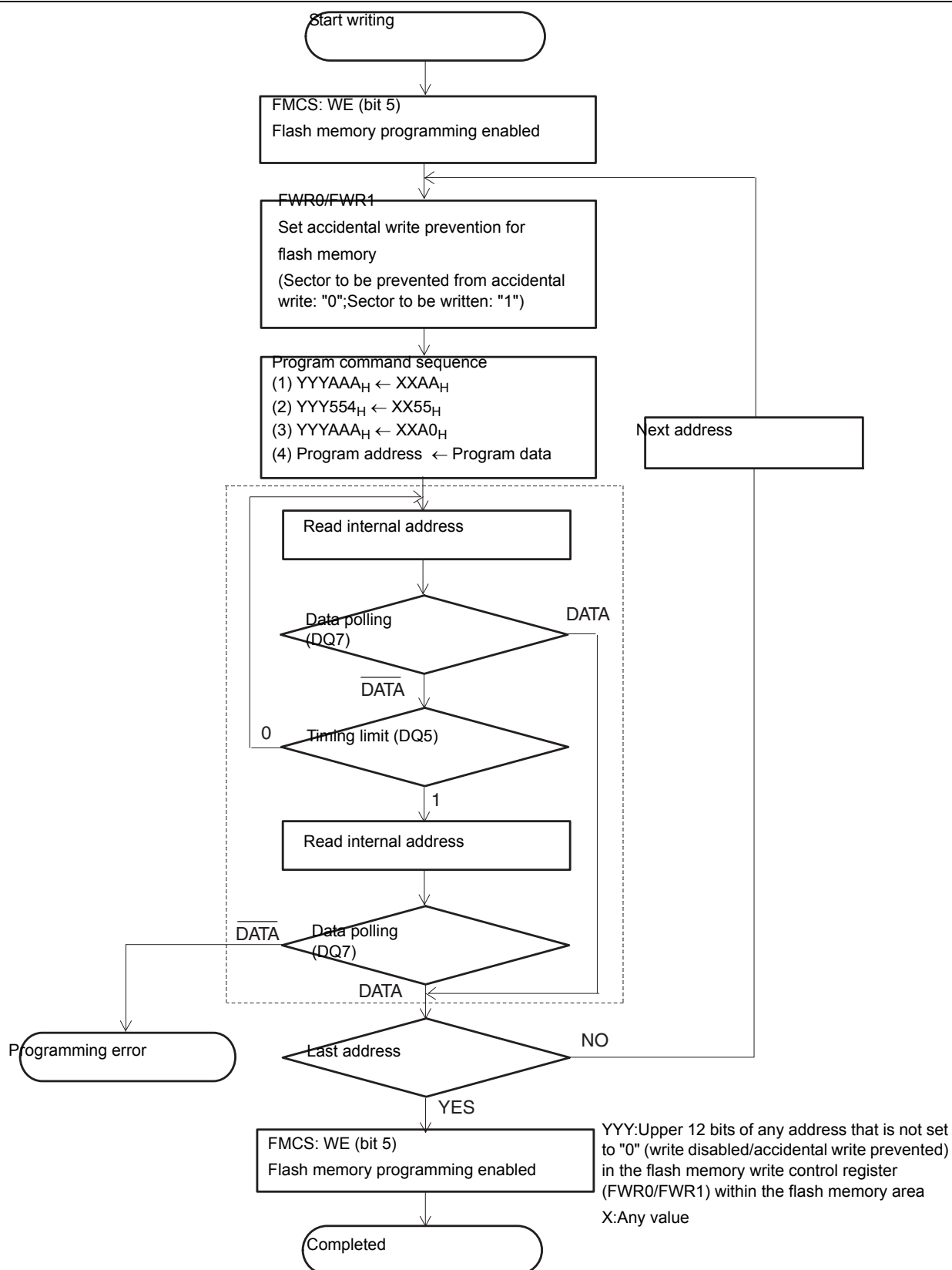
Figure 24-8. is an example of the procedure for writing to the flash memory. The hardware sequence flags (see Section "24.8 Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Here, the data polling flag (DQ7) is used to confirm that writing has terminated.

The data read to check the flag is read from the address written to last.

The data polling flag (DQ7) changes at the same time that the timing limit exceeded flag (DQ5) changes. For example, even if the timing limit exceeded flag (DQ5) is "1", the data polling flag bit (DQ7) must be rechecked.

Also for the toggle bit flag (DQ6), the toggle operation stops at the same time that the timing limit exceeded flag bit (DQ5) changes to "1". The toggle bit flag (DQ6) must therefore be rechecked.

Figure 24-8. Example of the Flash Memory Write Procedure



24.9.3 Erasing All Data in the Flash Memory (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 repeatedly the Chip Erase command in the command sequence table (see [Table 24-3](#) in Section "24.7 Starting the Flash Memory Automatic Algorithm") to the target sectors of both the main flash memory and the satellite flash memory. The Chip Erase command is executed in 6 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.

■ Notes on Chip Erase

- Areas to be erased by the Chip Erase command

When the Chip Erase command is issued for the main flash area (SA0 to SA5), the main flash area (SA0 to SA5) will be erased. If the Chip Erase command is issued for the satellite flash area (SB0 to SB3), the satellite flash area (SB0 to SB3) will be erased.

- Note on coexistence of write-enabled and -disabled sectors

If the Chip Erase command is issued by accessing a write-enabled sector when write-enabled and write-disabled sectors coexist, the content of all the sectors is erased, including the write-disabled sectors.

24.9.4 Erasing Optional Data in the Flash Memory (Erasing Sectors)

This section describes the procedure for issuing the Sector Erase command to erase optional data in the flash memory (erase sector). Individual sectors can be erased. Multiple sectors can also be specified at one time.

■ Erasing Optional Data in the Flash Memory (Erasing Sectors)

Optional sectors in the flash memory can be erased by sending the Sector Erase command in the command sequence table (see [Table 24-3](#) in Section "24.7 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 minimum 50 μ s is started by writing the sector erase code (30_H) to an accessible even-numbered address in the target sector in the sixth cycle. To erase multiple sectors, write the erase code (30_H) 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 minimum 50 μ s terminates after the final sector erase code has been written. That is, to erase multiple sectors at one time, the address of the next erase sector and the erase code (sixth cycle of the command sequence) must be written within 50 μ s. 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.

■ Sector Erase Procedure for Flash Memory

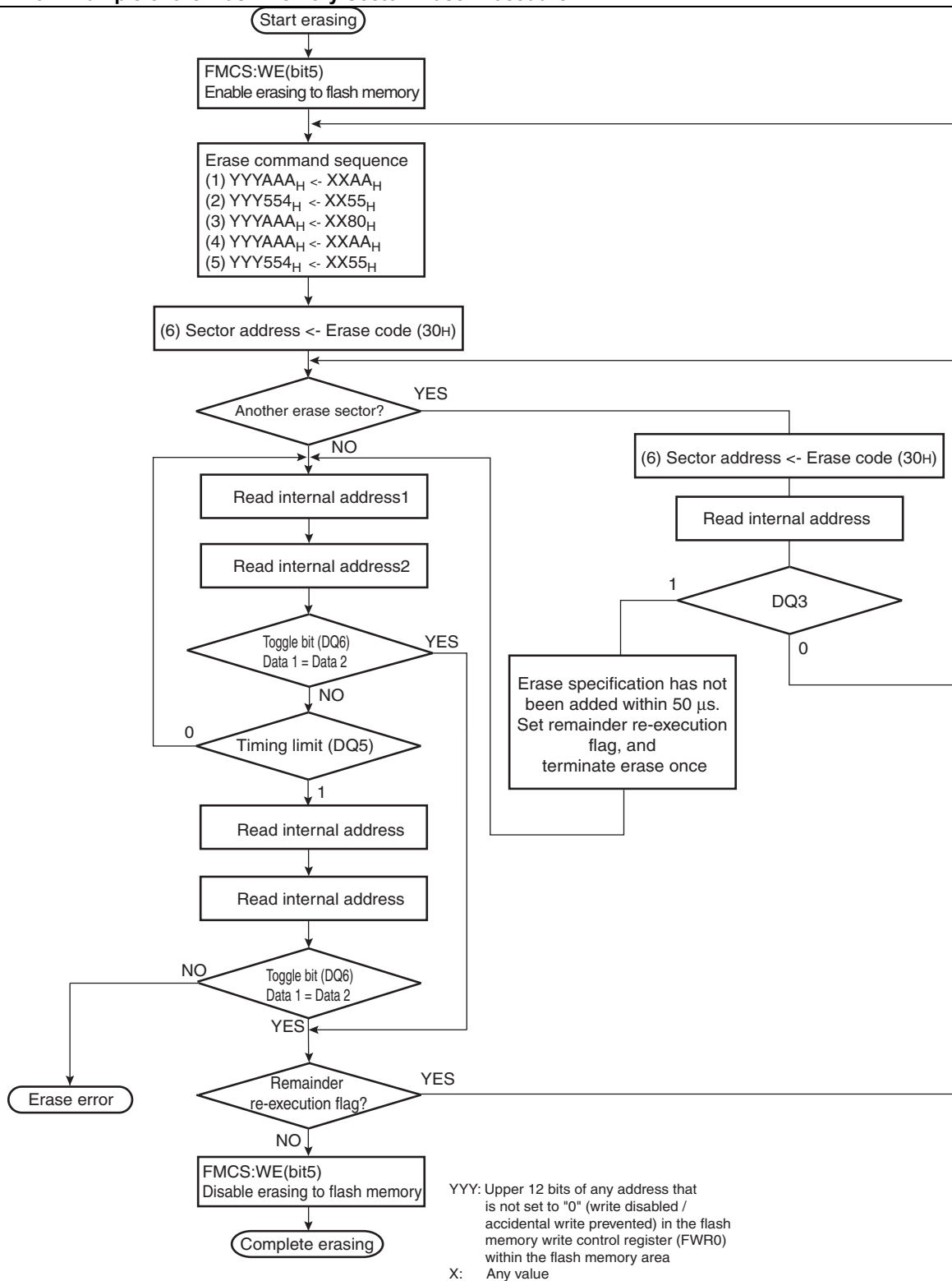
The hardware sequence flags (see Section "24.8 Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 24-9. shows an example of the procedure for erasing sectors in the flash memory. Here, the toggle bit flag (DQ6) is used to confirm that erasing has terminated.

Note that 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. Therefore, even if the timing limit exceeded flag (DQ5) is 1, the toggle bit flag (DQ6) must be rechecked.

The data polling flag (DQ7) also changes at the same time that the timing limit exceeded flag bit (DQ5) changes. As a result, the data polling flag (DQ7) must be rechecked.

Figure 24-9. Example of the Flash Memory Sector Erase Procedure



24.9.5 Suspending Sector Erase of Flash Memory

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 Sector Erase of Flash Memory

Erasing of flash memory sectors can be suspended by sending the Sector Erase Suspend command in the command sequence table (see [Table 24-3](#) in Section "24.7 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 an erase wait time. The command will be ignored during chip erase or write operations.

This command is implemented by writing the erase suspend code ($B0_H$). 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 20 μ s has elapsed. Sector Erase Suspend command should be entered more than 20 μ s after Sector Erase command or Sector Erase Restart command is issued.

If Sector Erase Restart command will be issued less than 20 μ s after Sector Erase Suspend command is issued, operations after restarting cannot be guaranteed. In the evaluation product, however, the restart command which is issued less than 20 μ s is neglected.

24.9.6 Restarting Sector Erase of Flash Memory

This section describes the procedure for issuing the Sector Erase Restart command to restart suspended erasing of flash memory sectors.

■ Restarting Sector Erase of Flash Memory

Suspended erasing of flash memory sectors can be restarted by sending the Sector Erase Restart command in the command sequence table (see [Table 24-3](#) in Section "24.7 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 (30_H). At this time, specify an optional address in the flash memory area for the address.

If a Sector Erase Restart command is issued during sector erase, the command will be ignored.

24.10 Notes on Using Flash Memory

This section contains notes on using flash memory.

■ Notes on Using Flash Memory

- Entering hardware reset (\overline{RST})

The "L" level width must be at least 500 ns to enter a hardware reset.

- When a reset is generated during a write/erase operation

The write/erase sequence is initialized when a hardware reset or a low-voltage detection reset is generated. After recovering from the reset, the normal operation is performed. The data currently being written/erased becomes indefinite. Therefore, perform the write/erase operation again. Up to 20 μ s is required for the state machine of the flash memory that receives a hard-

ware reset or a low-voltage detection reset to complete the reset and become readable. This is designed to discharge a high voltage, if such a voltage is used inside the flash memory. It is however only needed when a high voltage is used (when a command other than read/reset command is issued). In case of a reset in any state other than above, up to 500ns is required to complete a flash memory reset. It takes 200ns from the cancellation of a reset until it becomes readable.

The write/erase sequence is not initialized, when a software reset, watchdog timer reset or CPU operation detection reset is generated. After the completion of the write/erase sequence, the normal operation is performed.

❑ Program access to the main flash memory

Read access to the flash memory during the operation of the automatic algorithm is prohibited. This means that no program must be executed in the main flash memory when the main flash memory is being written/erased. To write to or erase the main flash memory, it is necessary to start the write/erase operation after switching the program area to another area such as RAM. In this case, if a sector containing an interrupt vector is erased, the write/erase interrupt process cannot be executed. Do not place any program in the satellite flash memory.

❑ 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.

❑ Transition during write/erase mode

Never transit to subclock mode and standby mode (time-base timer mode, watch mode, or stop mode) during the writing mode to flash memory (the WE bit of the control status register (FMCS) is "1").

24.11 Flash Security Feature

The flash security feature provides possibilities to protect the content of the flash memory.

■ Overview

By writing the protection code of "01_H" to the security bit in the flash memory, 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 program and data stored in the flash memory.

Address of the security bit depends on the size of built-in flash memory. [Table 24-14](#). shows the address of security bit.

Table 24-14. Address of Flash Security Bit

	Flash memory size	Address of security bit
MB90F997JBS, MB90F997MBS	Built-in 1M-bit flash memory	FE0001 _H

■ How to Enable the Security

After writing the protection code "01_H" to the security bit, the following external reset or power-on enables the flash security feature.

■ How to Disable the Security

The chip erase is executed, and after external reset or the power supply is turned on again, security is released.

■ Operation of the Security

Read operation: invalid data read

Write operation: ignored

■ Others

- About configuration of the general-purpose parallel programmer, please follow to the specification of parallel programmer.
- 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.

Notes:

- The security bit is allocated in the flash memory area. When the protection code "01_H" is written to the security bit, it is locked by the security. So, when not using the security function, do not write "01_H" to this address.
See [Table 24-14](#). for the address of the security bit.
- By specifying the sector of the flash memory, it cannot be locked the security per sector. It is the security function for all areas in the flash memory.
- The FLASH memory trouble in the state to put security cannot be analyzed at all.

25. Examples of Serial Programming Connection for Flash Memory Products



This chapter shows an example of a serial programming connection when the AF220/AF210/AF120/AF110 flash microcontroller programmer made by Yokogawa Digital Computer Corporation is used.

25.1 Basic Configuration of Serial Programming Connection

25.2 Example of Serial Programming Connection (User Power Supply Used)

25.3 Example of Serial Programming Connection (Power Supplied from Programmer)

25.4 Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)

25.5 Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)

25.1 Basic Configuration of Serial Programming Connection

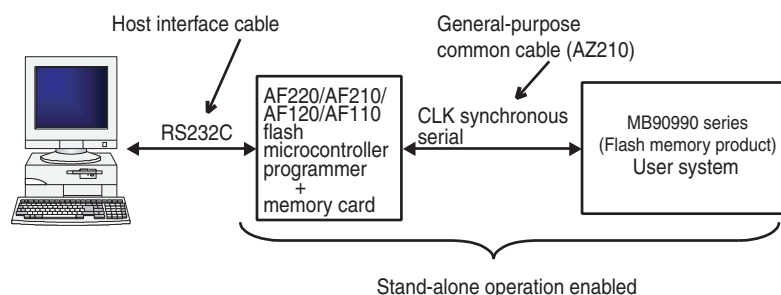
The support serial on-board writing (Cypress standard) of the flash ROM. This section provides the related specifications.

■ Basic Configuration of Serial Programming Connection

Cypress standard serial on-board writing uses the Yokogawa Digital Computer Corporation AF220/AF210/AF120/AF110 flash microcontroller programmer.

Figure 25-1. shows the basic configuration for the serial programming connection.

Figure 25-1. Basic Configuration of Serial Programming Connection



Note:

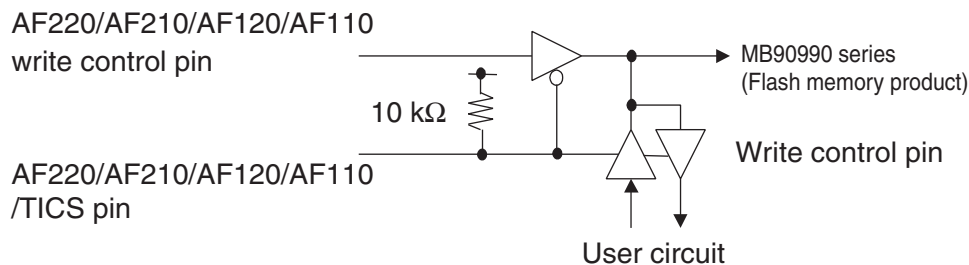
For information on the functions of and operational procedures related to the flash microcontroller programmer (AF220/AF210/AF120/AF110), the general-purpose common cable (AZ210) for connection, and the connector, contact Yokogawa Digital Computer Corporation.

Table 25-1. Pin Used for Fujitsu Standard Serial On-board Programming

Pin	Function	Supplementary information
MD2, MD1, MD0	Mode pins	Controls programming mode from the flash microcontroller 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.
P83, P84	Programming activation pins	Input "L" level to P83 and "H" level to P84.
RST	Reset pin	-
SIN1	Serial data input pin	Use UART1 as CLK synchronous mode.
SOT1	Serial data output pin	
SCK1	Serial clock signal input pin	
C	C pin	This pin is used to stabilize the power supply. Connect to a external ceramic capacitor of approximately 0.1 μ F or more.
V _{CC}	Power voltage supply pin	Programming voltage (5 V \pm 10%)
V _{SS}	GND pin	Common to the GND of the flash microcontroller programmer.

Even if the P83, P84, SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in Figure 25-2. is required. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25-2. Control Circuit



Sections "25.2 Example of Serial Programming Connection (User Power Supply Used)" to "25.5 Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)" present examples for the following 4 types of serial programming connection. See each Section as required.

- Example of serial programming connection in internal vector mode (user power supply used)
- Example of serial programming connection in internal vector mode (power supplied from the programmer)
- Example of minimum connection to the flash microcontroller programmer (user power supply used)
- Example of minimum connection to the flash microcontroller programmer (power supplied from the programmer)

Table 25-2. System Configuration of Flash Microcontroller Programmers (Manufactured by Yokogawa Digital Computer Corporation)

Model		Function
Main unit	AF220/AC4P	Ethernet interface built-in model and 100 V to 220 V AC power adapter
	AF210/AC4P	Standard model and 100 V to 220 V AC power adapter
	AF120/AC4P	Single-key Ethernet interface built-in model and 100 V to 220 V AC power adapter
	AF110/AC4P	Single-key model and 100 V to 220 V AC power adapter
AZ221		RS232C cable for programmer PC/AT
AZ210		Standard target probe (a) cable length: 1 m
FF201		Fujitsu F ² MC-16LX flash microcontroller control module
AZ290		Remote controller
/P2		2MB PC Card (optional) for flash memory sizes up to 128 KB
/P4		4MB PC Card (optional) for flash memory sizes up to 512 KB

Inquiries: Yokogawa Digital Computer Corporation

Telephone number: (81)-42-333-6224

Note:

Although the AF200 flash microcontroller programmer is no longer manufactured, the programmer still can be used in combination with the FF201 control module.

Examples of serial programming connection can correspond to the connection example as shown in "■ Oscillating Clock Frequency and Serial Clock Input Frequency".

■ 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 Flash memory product.

Serial clock frequency that can be input = $0.125 \times$ example of oscillation clock frequency

Set an appropriate serial clock input frequency in the flash microcontroller programmer according to the oscillating clock frequency in use.

Table 25-3. Examples of Serial Clock Frequencies That can be Input

Oscillating clock frequency	Maximum serial clock frequency that can be input for microcontrollers	Maximum serial clock frequency that can be used for the AF220, AF210, AF120 and AF110	Maximum serial clock frequency that can be used for the AF200
4 MHz	500 kHz	500 kHz	500 kHz
8 MHz	1 MHz	850 kHz	500 kHz
16 MHz	2 MHz	1.25 MHz	500 kHz

25.2 Example of Serial Programming Connection (User Power Supply Used)

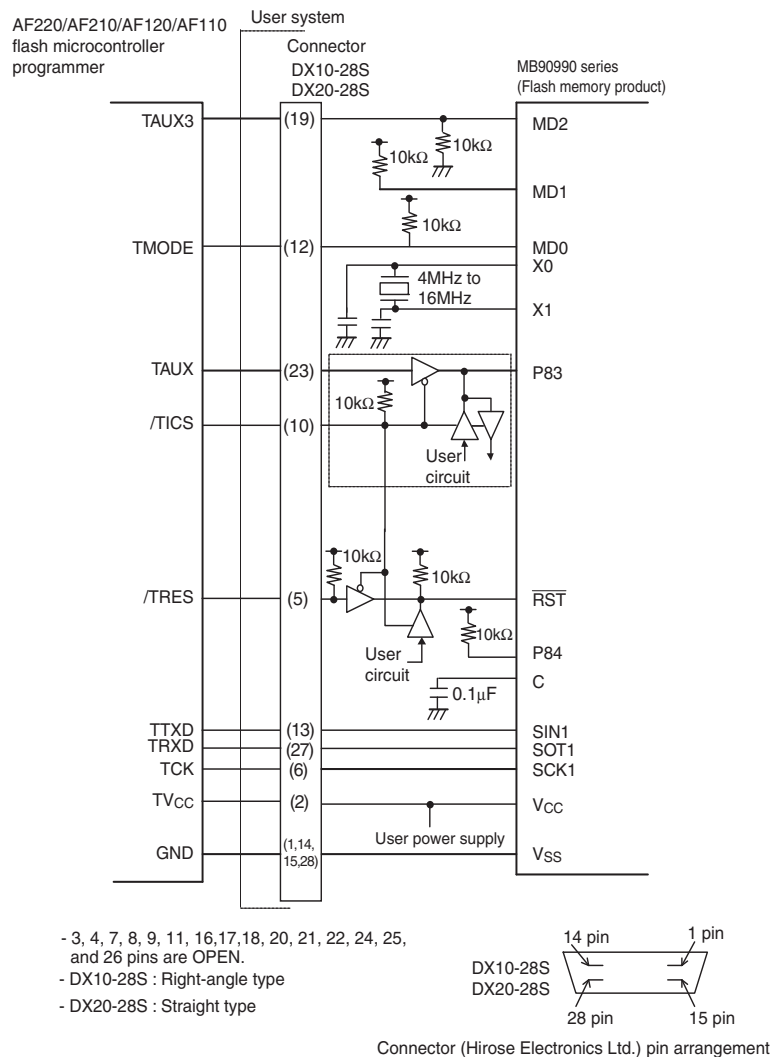
Figure 25-3. shows an example of a serial programming connection when the supply voltage of the microcontroller is supplied from the user power supply.

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

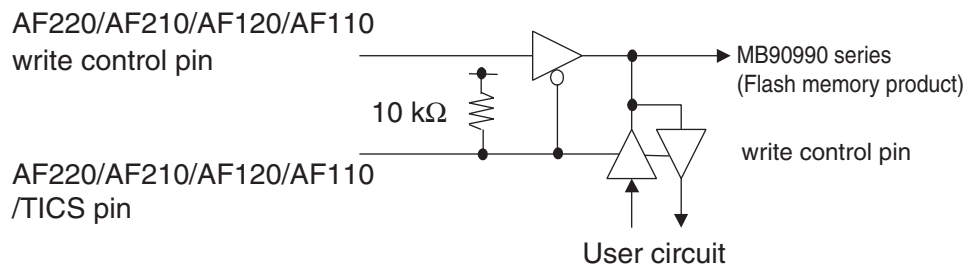
■ Example of Serial Programming Connection (User Power Supply Used)

Figure 25-3. Example of Serial Programming Connection for Single-chip Modes (User Power Supply Used)



- Even if the SIN1, SOT1, and SCK1 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 P83. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25-4. Control Circuit



- Connect the AF220/AF210/AF120/AF110 while the user power is off.

25.3 Example of Serial Programming Connection (Power Supplied from Programmer)

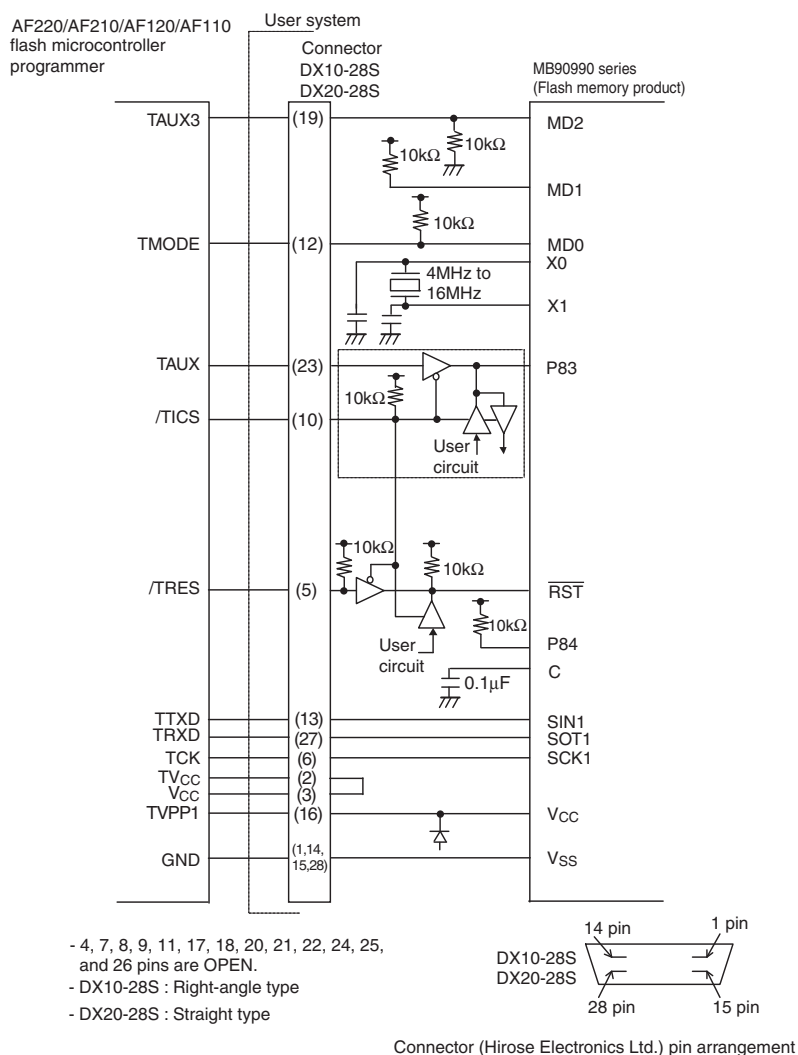
Figure 25-5. shows an example of a serial programming connection when the supply voltage of the microcontroller is supplied from the programmer power supply.

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

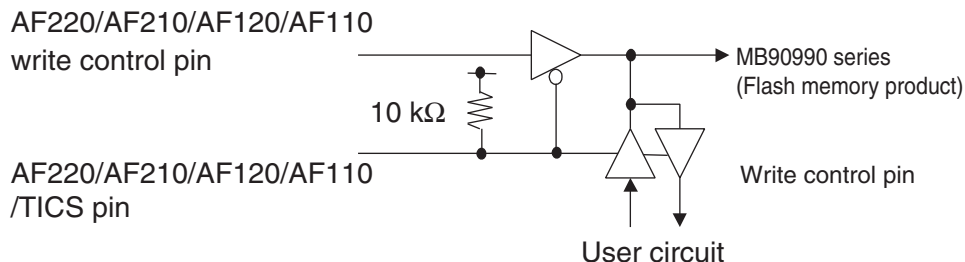
■ Example of Serial Programming Connection (Power Supplied from Programmer)

Figure 25-5. Example of Serial Programming Connection for Single-chip Modes (Power Supplied from Programmer)



- Even if the SIN1, SOT1, and SCK1 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 P83. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25-6. Control Circuit



- Connect the AF220/AF210/AF120/AF110 while the user power is off.
- When the programming power is supplied from the AF220/AF210/AF120/AF110, be careful not to short-circuit the user power supply.

25.4 Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)

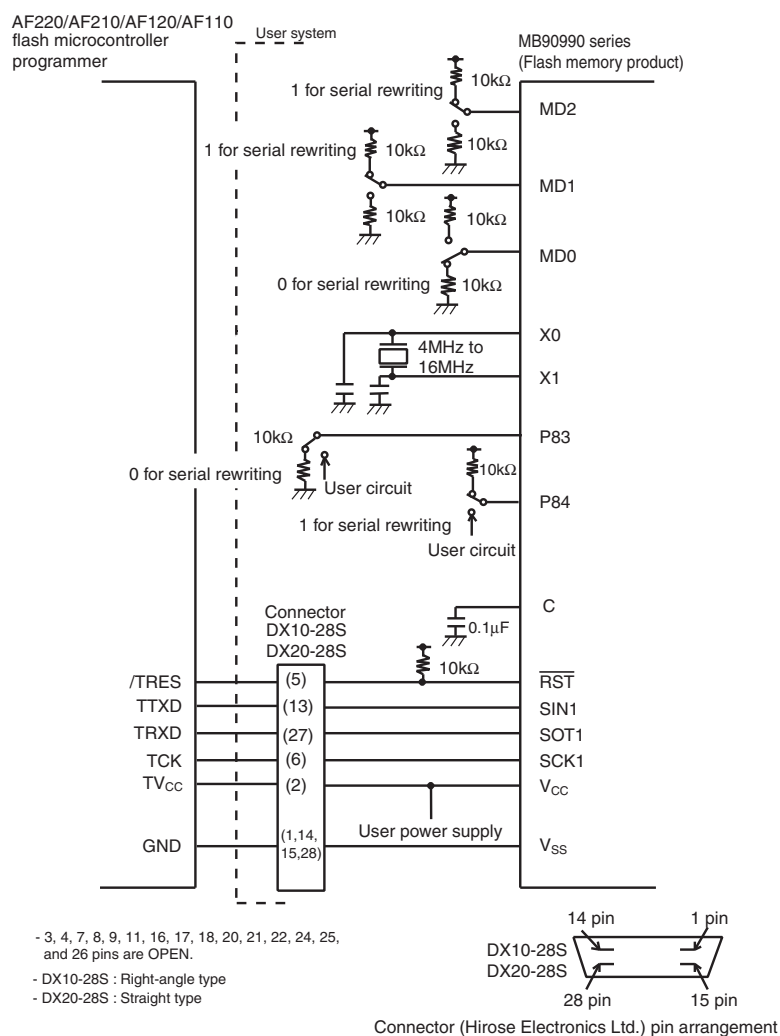
Figure 25-7. shows an example of the minimum connection to the flash microcontroller programmer when the supply voltage of the microcontroller is supplied from the user power supply.

Serial reprogramming mode: MD2, MD1, MD0 = 110_B

■ Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)

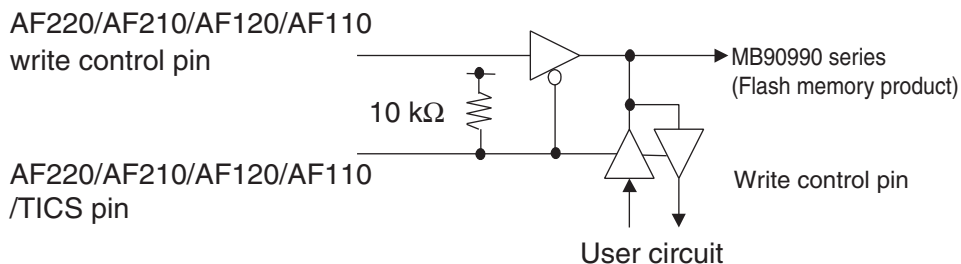
For a flash memory write, the MD2, MD1, MD0, P83, and P84 pins and flash microcontroller programmer need not be connected if the pins are set as Figure 25-7. .

Figure 25-7. Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)



- Even if the SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in the figure below is required. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25-8. Control Circuit



- Connect the AF220/AF210/AF120/AF110 while the user power is off.

25.5 Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)

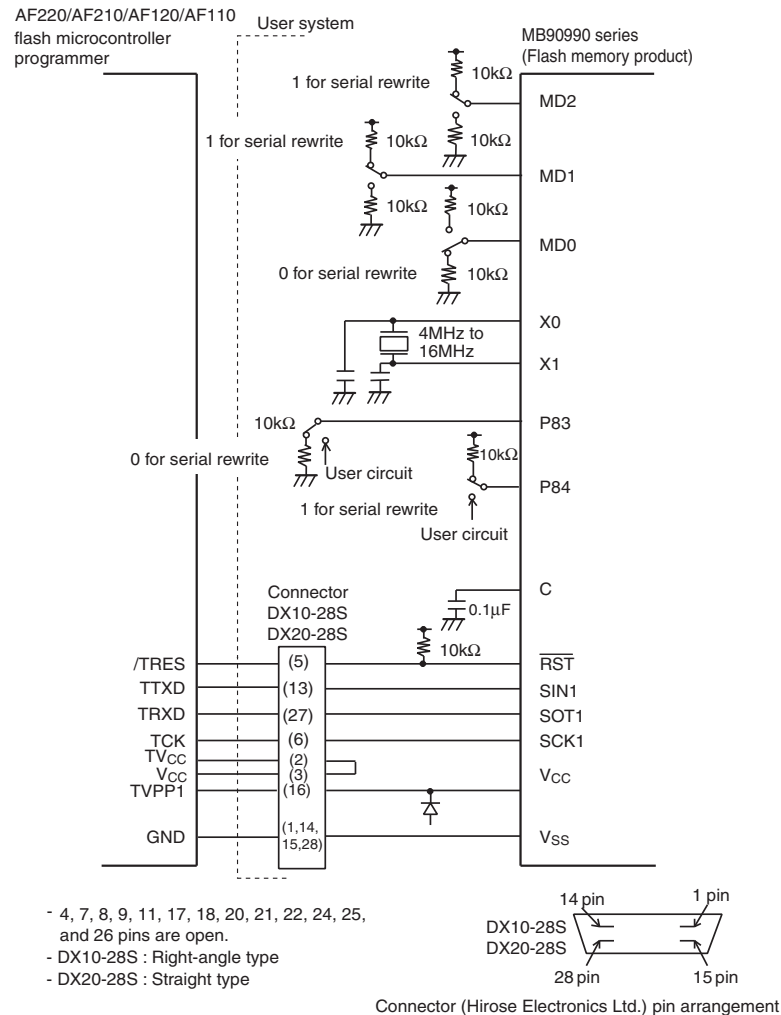
Figure 25-9. shows an example of the minimum connection to the flash microcontroller programmer when the supply voltage of the microcontroller is supplied from the programmer power supply.

Serial reprogramming mode: MD2, MD1, MD0 = 110_B

- **Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)**

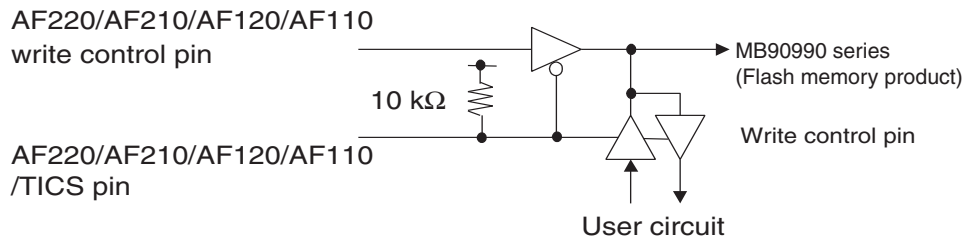
For a flash memory write, the MD2, MD1, MD0, P83, and P84 pins and flash microcontroller programmer need not be connected if the pins are set as Figure 25-9. .

Figure 25-9. Example of Minimum Connection to the Flash Microcontroller Programmer (Power Supplied from Programmer)



- Even if the SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in the figure below is required. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25-10. Control Circuit



- 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.

26. Clock Calibration Unit



This chapter outlines the clock calibration unit and describes its register configuration and functions. This function can be used by only the products with "J"-suffix.

Evaluation products don't correspond to the CR clock calibration.

[26.1 Overview of Clock Calibration Unit](#)

[26.2 Register of Clock Calibration Unit](#)

[26.3 Application Notes](#)

26.1 Overview of Clock Calibration Unit

By using the clock calibration module, it is possible to calibrate the CR oscillation clock using the main oscillation clock as the standard.

The CR oscillation clock can be improved the accuracy by the trimming.

■ Features of the Calibration Unit

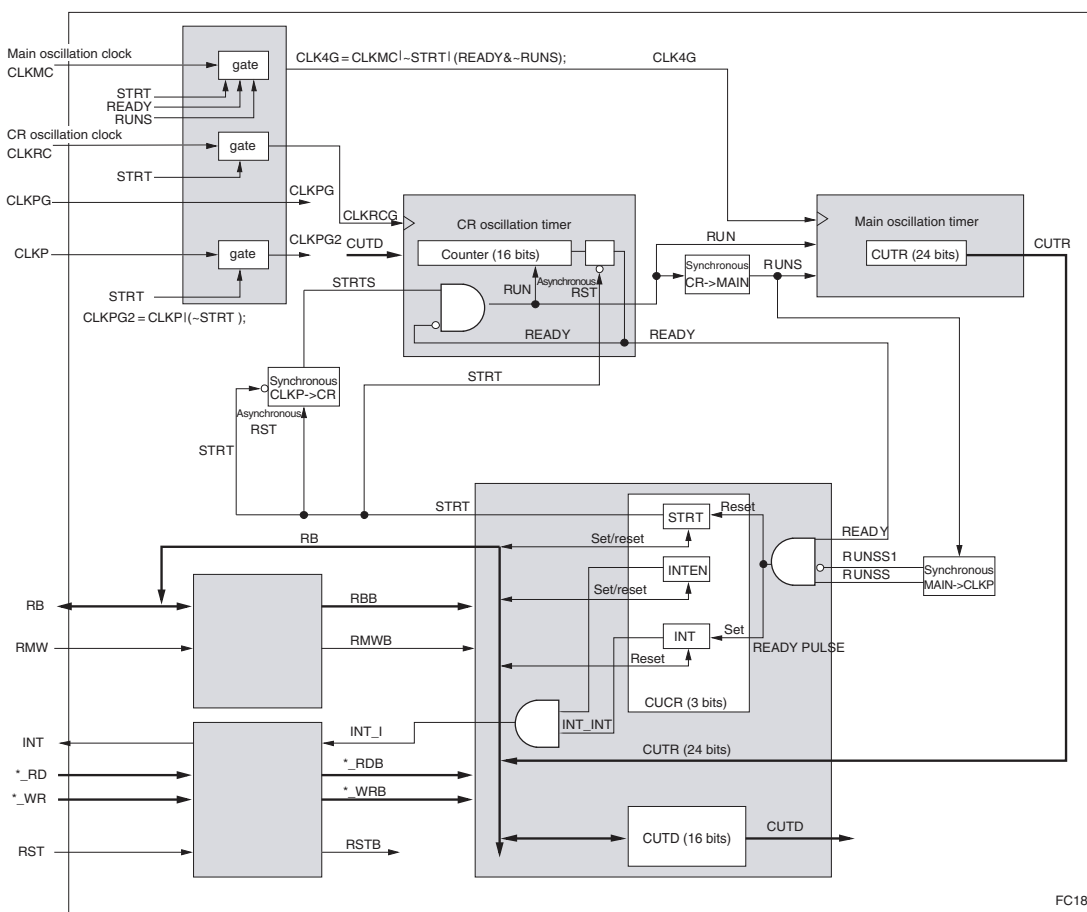
When this module is used, by means of the software, the time generated by the CR oscillation clock can be measured by the main oscillation clock.

The accuracy of the CR oscillation clock can approach the accuracy of the main oscillation clock via processing by software and the use of this module. The measurement by the clock calibration module can be processed by software. This module is composed of two timers: the timer operating with the CR oscillation clock and the timer operating with the main oscillation clock. The CR oscillation timer triggers the main oscillation timer and the values of the main oscillation timer is stored in the register.

The CR oscillation circuit can be calibrated by processing the values in the register via software and setting it to the trimming register of the CR oscillation circuit.

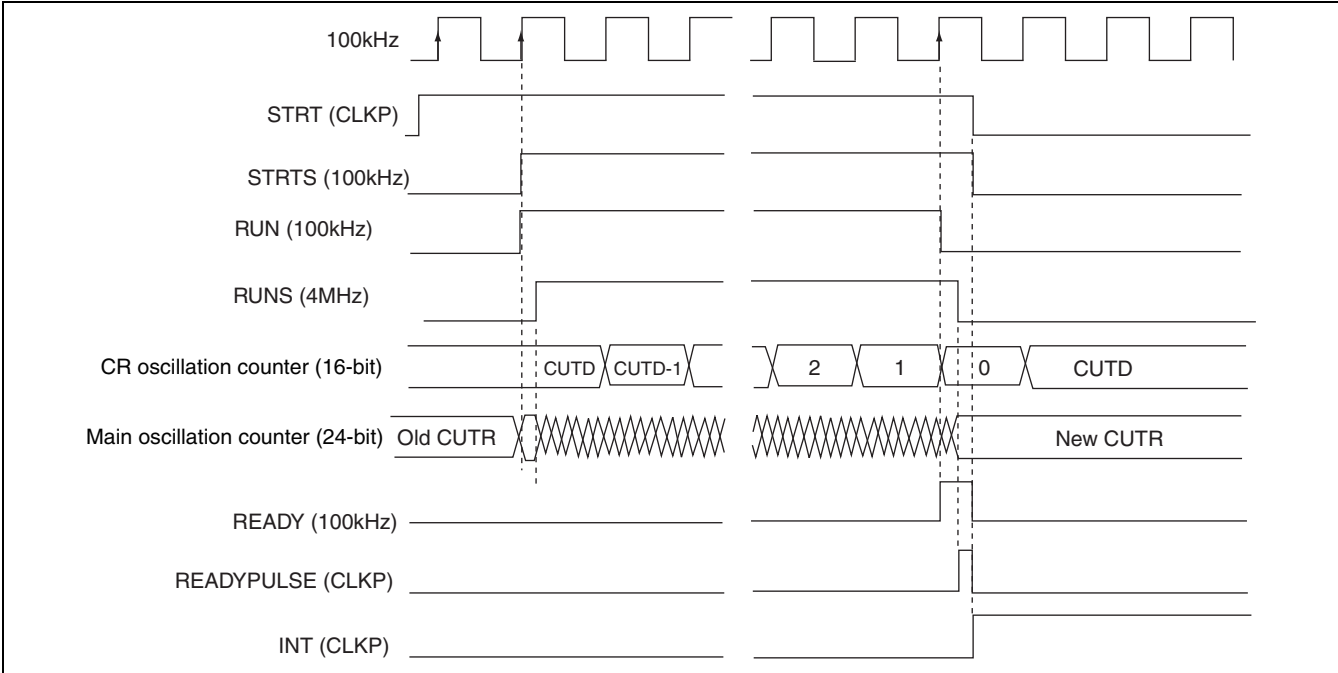
Since the clock calibration unit uses the main oscillation as the standard, it cannot use in the state that the main oscillation has stopped.

Figure 26-1. Block Diagram of the Clock Calibration Unit



■ Timing of Clock Calibration Unit

Figure 26-2. Measurement Processing Timing



■ Clock Timing of Clock Calibration Unit

The module is operated by 3 different clocks: the main oscillation clock CLKMC, the CR oscillation clock CLKRC, and the machine clock CLKP. Each domain is made to conform by a synchronous circuit.

All three clocks are gate-controlled. When STRT is "0", the CLKRC and CLKMC clocks are switched off.

The clock frequency must satisfy the following conditions.

Clock ratio

$$T_{CLKRC} > 2 \times T_{CLKMC} + 3 \times T_{CLKP}$$

$$T_{CLKMC} < 1/2 \times T_{CLKRC} - 3/2 \times T_{CLKP}$$

$$T_{CLKP} < 1/3 \times T_{CLKRC} - 2/3 \times T_{CLKMC}$$

Table 26-1. Example of Valid Clock Ratio

	CLKRC		CLKMC		CLKP	
Normal Operation	100kHz	100μs	4MHz	250ns	2MHz or more	500ns or less

26.2 Register of Clock Calibration Unit

This section shows the register list of the calibration unit and gives a detailed explanation of each register.

■ Register List of the Calibration Unit

- CR oscillation trimming setting register (CRTR)

Figure 26-3. Bit Configuration of CR Oscillation Trimming Setting Register (CRTR)

bit	15	14	13	12	11	10	9	8
Address: 0079B9 _H	-	-	-	-	TRD3	TRD2	TRD1	TRD0
R/W →	(R)	(R)	(R)	(R)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(1)	(1)	(1)	(1)	(0)	(1)	(1)	(1)

- Calibration unit control register (CUCR)

Figure 26-4. Bit Configuration of Calibration Unit Control Register (CUCR)

bit	7	6	5	4	3	2	1	0
Address: 0079B8 _H	-	-	-	STRT	-	-	INT	INTEN
R/W →	(R)	(R)	(R)	(R/W)	(R)	(R)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

- CR oscillation timer data register (CUTD)

Figure 26-5. Bit Configuration of CR Oscillation Timer Data Register (CUTD)

CUTDH		bit	15	14	13	12	11	10	9	8
Address: 0079BB _H			TDD15	TDD14	TDD13	TDD12	TDD11	TDD10	TDD9	TDD8
R/W →			(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →			(1)	(1)	(0)	(0)	(0)	(0)	(1)	(1)
CUTDL		bit	7	6	5	4	3	2	1	0
Address: 0079BA _H			TDD7	TDD6	TDD5	TDD4	TDD3	TDD2	TDD1	TDD0
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)	(0)	(0)	(0)

- Main oscillation timer data register (CUTR)

Figure 26-6. Bit Configuration of Main Oscillation Timer Data Register (CUTR)

CUTR1Hbit		15	14	13	12	11	10	9	8
Address: 0079BD _H		-	-	-	-	-	-	-	-
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
CUTR1Lbit		7	6	5	4	3	2	1	0
Address: 0079BC _H		TDR23	TDR22	TDR21	TDR20	TDR19	TDR18	TDR17	TDR16
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
CUTR2Hbit		15	14	13	12	11	10	9	8
Address: 0079BF _H		TDR15	TDR14	TDR13	TDR12	TDR11	TDR10	TDR9	TDR8
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
CUTR2Lbit		7	6	5	4	3	2	1	0
Address: 0079BE _H		TDR7	TDR6	TDR5	TDR4	TDR3	TDR2	TDR1	TDR0
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

■ CR Oscillation Trimming Setting Register (CRTR)

Figure 26-7. Bit Configuration of CR Oscillation Trimming Setting register (CRTR)

bit		15	14	13	12	11	10	9	8
Address: 0079B9 _H		-	-	-	-	TRD3	TRD2	TRD1	TRD0
R/W →		(R)	(R)	(R)	(R)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →		(1)	(1)	(1)	(1)	(0)	(1)	(1)	(1)

TRD3	TRD2	TRD1	TRD0	Trimming value	n value
0	0	0	0	-43.75%	0
0	0	0	1	-37.50%	1
0	0	1	0	-31.25%	2
0	0	1	1	-25.00%	3
0	1	0	0	-18.75%	4
0	1	0	1	-12.50%	5
0	1	1	0	-6.25%	6
0	1	1	1	0.0% [initial value]	7
1	0	0	0	+6.25%	8
1	0	0	1	+12.5%	9

TRD3	TRD2	TRD1	TRD0	Trimming value	n value
1	0	1	0	+18.75%	10
1	0	1	1	+25.00%	11
1	1	0	0	+31.25%	12
1	1	0	1	+37.50%	13
1	1	1	0	+43.75%	14
1	1	1	1	+50.00%	15

The trimming value is calculated as follows:

1) Set TRD[3:0]=0111_B, operate the calibration unit and get the CUTR value.

Subsequent processing differs whether the CR oscillation frequency derived from the CUTR value is larger than/equal to/smaller than 100kHz.

The following expression can be used to obtain the CR oscillation frequency from the CUTR value.

CR oscillation frequency = (CUTD value ´ main oscillation clock frequency) / CUTR value

2-1) CR oscillation frequency (get by "1") > 100 kHz

This CR oscillation frequency is to be Fmax.

Set TRD[3:0]=1111_B, operate the calibration unit and get the CUTR value.

The CR oscillation frequency get by this CUTR value is to be Fmin.

3-1) Enter 0 to 7 to the "n" in the following expression.

The "n" that Fer becomes a minimum point is the trimming value.

$F_{step} = (F_{max} - F_{min}) / 7$

$F_{er} = 100 \text{ kHz} - (F_{min} + F_{step} \cdot n) : n = 0 \text{ to } 7$

2-2) CR oscillation frequency (get by "1") < 100 kHz

This CR oscillation frequency is to be Fmin.

Set TRD[3:0]=1111_B, operate the calibration unit and get the CUTR value.

The CR oscillation frequency get by this CUTR value is to be Fmax.

3-2) Enter 7 to 15 to the "n" in the following expression.

The "n" that Fer becomes a minimum point is the trimming value.

$F_{step} = (F_{max} - F_{min}) / 8$

$F_{er} = 100 \text{ kHz} - (F_{min} + F_{step} \cdot (n - 7)) : n = 7 \text{ to } 15$

2-3) CR oscillation frequency (get by "1") = 100 kHz

There is no need to calculate the trimming value. The initial value can be used.

■ Calibration Unit Control Register (CUCR)

Figure 26-8. Bit Configuration of Calibration Unit Control Register (CUCR)

bit	7	6	5	4	3	2	1	0
Address: 0079B8 _H	-	-	-	STRT	-	-	INT	INTEN
R/W →	(R)	(R)	(R)	(R/W)	(R)	(R)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

The calibration unit control register (CUCR) has the following functions.

- ☐ Start/stop calibration measurements
- ☐ Enable/disable interrupts
- ☐ Show the end of calibration measurements

[bit4] STRT - Start calibration

STRT	Function
0	Stop calibration, module switch off (Initial value)
1	Start calibration

Calibration starts when the STRT bit is set to "1" by software. The CR oscillation timer starts its countdown from the value stored in the CR oscillation timer data register, and the main oscillation timer starts its count up from "0".

When the CR oscillation timer reaches "0", this bit is cleared to "0" by hardware.

Calibration will stop immediately if software writes "0" in this bit during the calibration. If a software write of "0" and a hardware cleared to "0" are generated simultaneously, the hardware operation is given priority by the software operation.

Namely, the calibration ends properly and the INT bit is set to "1". Even if "1" is written in this bit during calibration, it is not affected.

[bit1] INT: Interrupt

INT	Function
0	Calibrating/module inactive (Initial value)
1	Calibration complete

This bit shows the completion of the calibration. After calibration starts, when the CR oscillation timer reaches "0", the main oscillation timer data register stores the final main oscillation timer value and the INT bit is set to "1". When read-modify-write (RMW) instruction is performed for this bit, "1" is read. The flag is cleared when "0" is written in this bit (INT=0). Even if "1" is written in this bit, it is not affected. The INT interrupt flag is not reset by hardware. Therefore it is necessary to reset by software before starting a new calibration. Without a reset, the calibration completion is only available from the STRT bit (the INT flag remains "1" even during the calibration).

[bit0] INTEN: Interrupt enable

INTEN	Function
0	Interrupt disable (initial value)
1	Interrupt enable

This is the interrupt enable bit corresponding to the INT bit.

When this bit is set to "1" and the INT is set by hardware, the calibration module sends an interrupt signal to the CPU. The INT bit itself is not affected by the INTEN bit and is set by hardware even when interrupts are disabled (INTEN=0).

■ CR Oscillation Timer Data Register(16-bit) (CUTD)

Figure 26-9. Bit Configuration of CR Oscillation Timer Data Register (16-bit) (CUTD)

CUTDH		15	14	13	12	11	10	9	8
Address: 0079BB _H		TDD15	TDD14	TDD13	TDD12	TDD11	TDD10	TDD9	TDD8
R/W →		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →		(1)	(1)	(0)	(0)	(0)	(0)	(1)	(1)
CUTDL		7	6	5	4	3	2	1	0
Address: 0079BA _H		TDD7	TDD6	TDD5	TDD4	TDD3	TDD2	TDD1	TDD0
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)	(0)	(0)	(0)

The CR oscillation timer data register (CUTD) retains the value that determines the calibration period (100kHz reload value).

The default value when using the 100kHz crystal is "C350_H" and corresponds to a measurement duration of 0.5s.

Other than when calibration is inactive (STRT=0), this register cannot be written.

The CR oscillation timer register stores the value that designates the duration of the calibration. When the calibration starts, the stored value is loaded into the CR oscillation timer and a countdown until the timer reaches "0" starts.

When CUTD is initialized to "0000_H", underflow is generated and the measurement value becomes $(FFFF_H + 1) \times T_{CLKRC}$.

It is necessary to set "C350_H"=50000 (decimal) to the CUTD register to obtain a measurement duration of 0.5s.

The ideal values of measurements when using the 4.00MHz crystal are shown in Table 27.2-1

Table 26-2. Ideal Measurements Depending on Measurement Duration

Calibration time(s)	CUTD value	CUTR value
0.5	C350 _H	1E8480 _H
0.25	61A8 _H	0F4240 _H
0.125	30D4 _H	07A120 _H
0.1	2710 _H	061A80 _H

The overall processing time from writing "1" into the STRT bit to the reset by hardware of STRT is longer than the actual calibration duration in order to synchronize the different clock domains.

The processing time becomes $(CUTD + 3) \times T_{CLKRC}$.

The exact calibration duration becomes $CUTD \times T_{CLKRC}$.

■ Main Oscillation Timer Data Register(24-bit) (CUTR)

Figure 26-10. Bit Configuration of Main Oscillation Timer Data Register (24-bit) (CUTR)

CUTR1Hbit		15	14	13	12	11	10	9	8
Address: 0079BD _H		-	-	-	-	-	-	-	-
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
CUTR1Lbit		7	6	5	4	3	2	1	0
Address: 0079BC _H		TDR23	TDR22	TDR21	TDR20	TDR19	TDR18	TDR17	TDR16
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
CUTR2Hbit		15	14	13	12	11	10	9	8
Address: 0079BF _H		TDR15	TDR14	TDR13	TDR12	TDR11	TDR10	TDR9	TDR8
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
CUTR2Lbit		7	6	5	4	3	2	1	0
Address: 0079BE _H		TDR7	TDR6	TDR5	TDR4	TDR3	TDR2	TDR1	TDR0
R/W →		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value →		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

The main oscillation timer data register (CUTR) retains the value of the calibration results.

The end of the calibration is indicated by the INT bit and the STRT bit of the CUCR register.

The CUTR value becomes valid when INT changes from "0" to "1" and STRT changes from "1" to "0".

The main oscillation timer data register stores the calibration results. When the calibration starts, the main oscillation timer starts to count up from zero. The main oscillation timer stops when the CR oscillation timer reaches "0", and the register retains the calibration results until the next calibration is triggered by software.

If this register is read during calibration, the value obtained will be random.

There is no effect even if the register is written in by software.

Note:

If this register is read during calibration, the value obtained will be random.

26.3 Application Notes

This section shows application notes dealing with calibration accuracy, power dissipation and measurement time.

■ Configuration of CR Oscillation Timer Data Register

The CR oscillation timer data register setting can be calculated according to the method below.

When a duration of 0.5s is required for the calibration, it is necessary to set C350_H=50,000 (decimal) in the CR oscillation timer data register. This shows the 50,000 pulse of the 100kHz oscillation clock.

Approximately "1E8480_H" values are stored within the main oscillation timer data register by this setting. These values show 2,000,000 pulses of the 4 MHz oscillator.

Table 26-3. Ideal Measurements for the 100kHz CR Oscillator and the 4.0 MHz Oscillator (CUTR)

Calibration time(s)	CUTD setting value	CUTR ideal value
0.5	C350 _H	1E8480 _H
0.25	61A8 _H	0F4240 _H
0.125	30D4 _H	07A120 _H
0.1	2710 _H	061A80 _H

Calibration accuracy together with power dissipation is important when using the calibration module.

The trimming should be executed in the actual operation status.

27. D/A Converter



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

27.1 Overview of D/A Converter

27.2 Block Diagram of D/A Converter

27.3 Configuration of D/A Converter

27.4 D/A Converter Registers

27.5 Sample Program for the D/A Converter

27.1 Overview of D/A Converter

The digital/analog (D/A) converter converts an 8-bit digital input into an analog output by using R-2R method. The D/A converter has one channels. Output control can be individually executed for each channel by using its D/A control register.

■ Function and Operation of the D/A Converter

This circuit is used to generate an analog output from an 8-bit digital input. Setting the enable bit in the D/A control register (DAC0) to "1" will enable the corresponding D/A output channel. Hence, setting this bit to "0" will disable that channel.

If D/A output is disabled, the analog switch inserted to the output of each D/A converter channel in series is turned off. In the D/A converter, the bit is cleared to 0 and the direct-current path is shut off. The above is also true in the stop mode.

The output voltage of the D/A converter ranges from 0 V to $255/256 \times AV_{CC}$. To change the output voltage range, adjust the AV_{CC} voltage externally.

The D/A converter output does not have the internal buffer amplifier. The analog switch ($= 100 \Omega$) is inserted to the output in series. To apply load to the output externally, estimate a sufficient stabilizing time.

Table 27-1. lists the theoretical values of output voltage for the D/A converter.

Table 27-1. Theoretical Values of Output Voltage for the D/A Converter

Value written to DA07 to DA00	Theoretical value of output voltage
00 _H	$0/256 \times AV_{CC}(= 0V)$
01 _H	$1/256 \times AV_{CC}$
02 _H	$2/256 \times AV_{CC}$
:	:
FD _H	$253/256 \times AV_{CC}$
FE _H	$254/256 \times AV_{CC}$

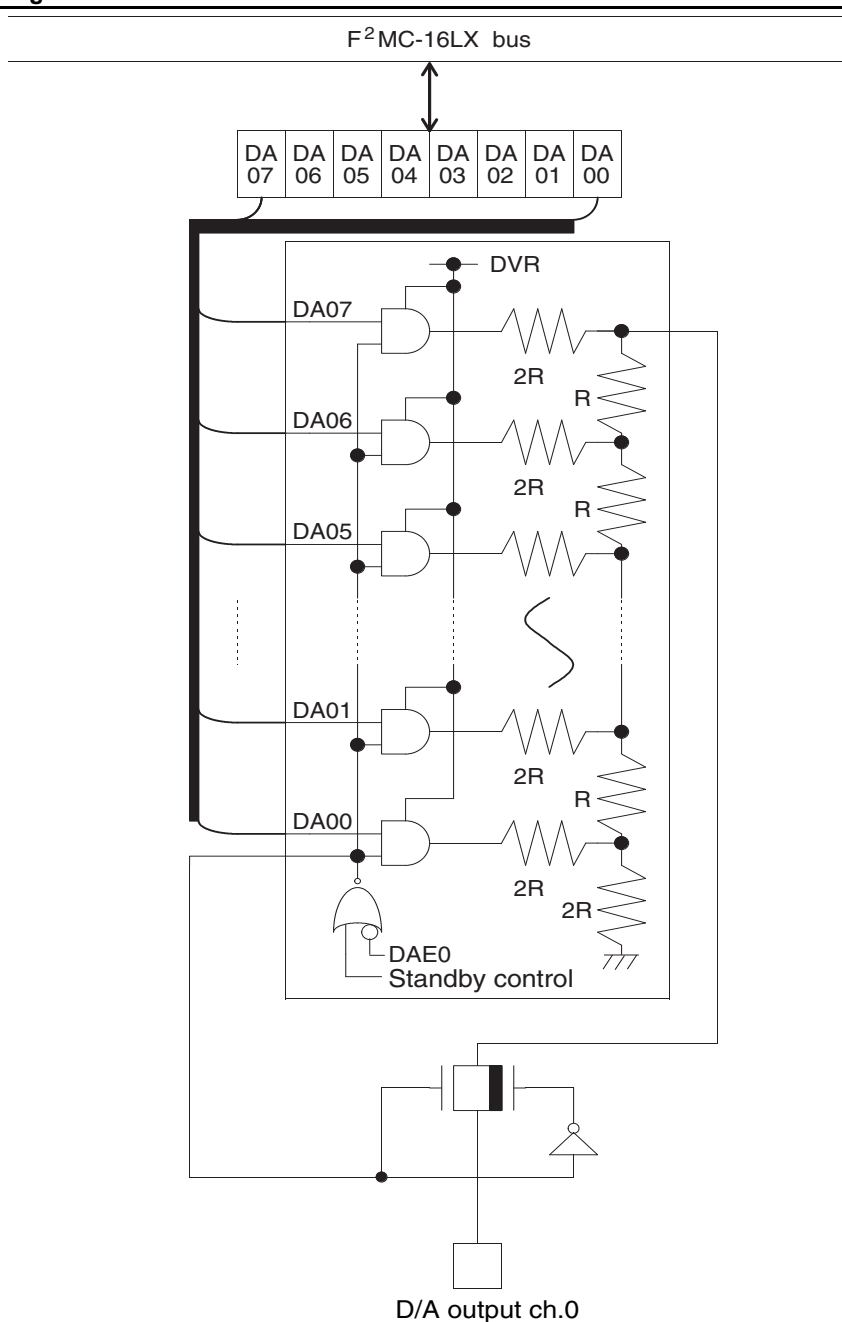
Table 27-1. Theoretical Values of Output Voltage for the D/A Converter

Value written to DA07 to DA00	Theoretical value of output voltage
FF _H	$255/256 \times AV_{CC}$

27.2 Block Diagram of D/A Converter

This section shows the block diagram of D/A converter.

■ Block Diagram of D/A Converter

Figure 27-1. Block Diagram of D/A Converter


27.3 Configuration of D/A Converter

This section describes the pins of the D/A converter and provides a pin block diagram.

■ D/A Converter Pins

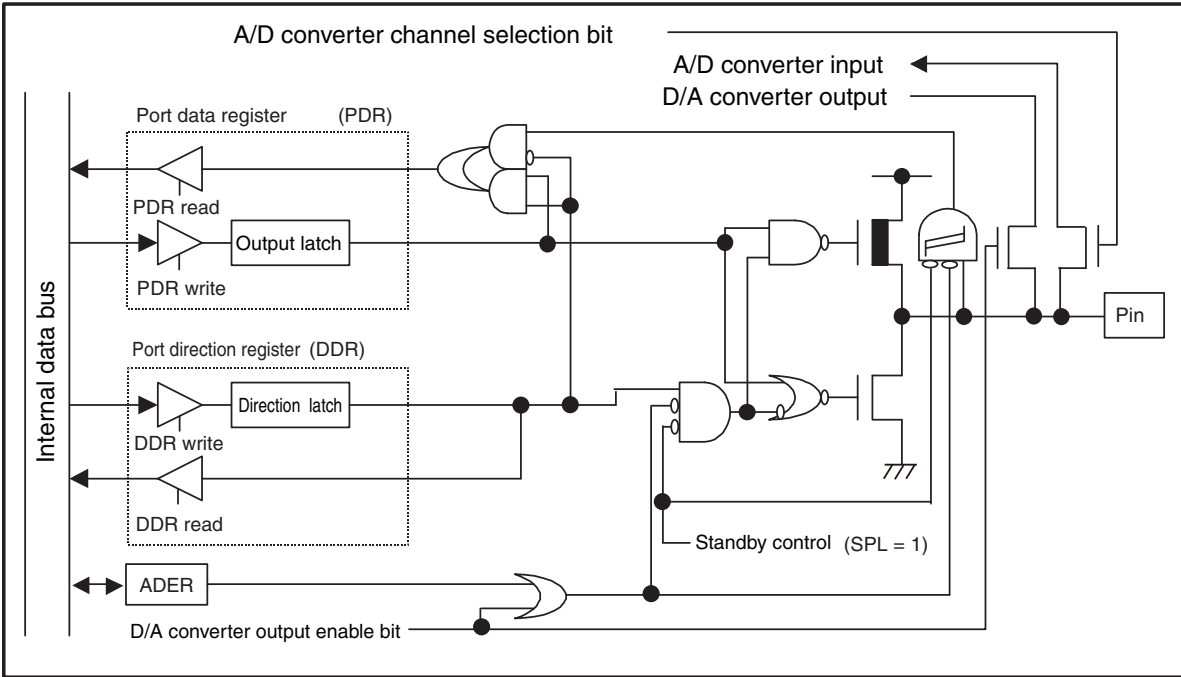
The pins of the D/A converter are shared with the general-purpose I/O ports. Table 28.3-1 lists the functions of the pins, I/O format, and settings required to use the D/A converter.

Table 27-2. D/A Converter Pin

Pin name	Pin function	I/O format	Pull-up option	Standby	Settings required to use D/A converter
P56/AN14/ INT11/DA0	Port 5 I/O/ Analog input/External interrupt/Analog output pin	Analog/CMOS output / CMOS hysteresis input	Not provided	Provided	DACR0 : DAE0 = 1

■ Block Diagram of the D/A Converter Pin

Figure 27-2. Block Diagram of the D/A Converter Pin



27.4 D/A Converter Registers

The D/A converter has the following two types of registers:

- D/A converter register (DAT0)
- D/A control register(DACR0)

■ D/A Converter Registers

Figure 27-3. D/A Converter Registers

D/A data register 0								Initial value
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00	XXXXXXXX _B
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

D/A control register 0								Initial value
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
-	-	-	-	-	-	-	DAE0	00000000 _B
-	-	-	-	-	-	-	R/W	

27.4.1 D/A Converter Register 0 (DAT0)

The D/A converter register 0 (DAT0) is used to set the digital input data for ch.0, which will be converted into an analog output.

■ D/A Converter Register 0 (DAT0)

Figure 27-4. D/A Converter Register 0 (DAT0)

D/A converter register 0								Initial value
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00	XXXXXXXX _B
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable
X : Undefined

27.4.2 D/A Control Register 0 (DACR0)

The D/A control register 0 is used to keep the output enable signal for ch.0.

■ D/A Control Register 0 (DACR0)

Figure 27-5. D/A Control Register 0 (DACR0)

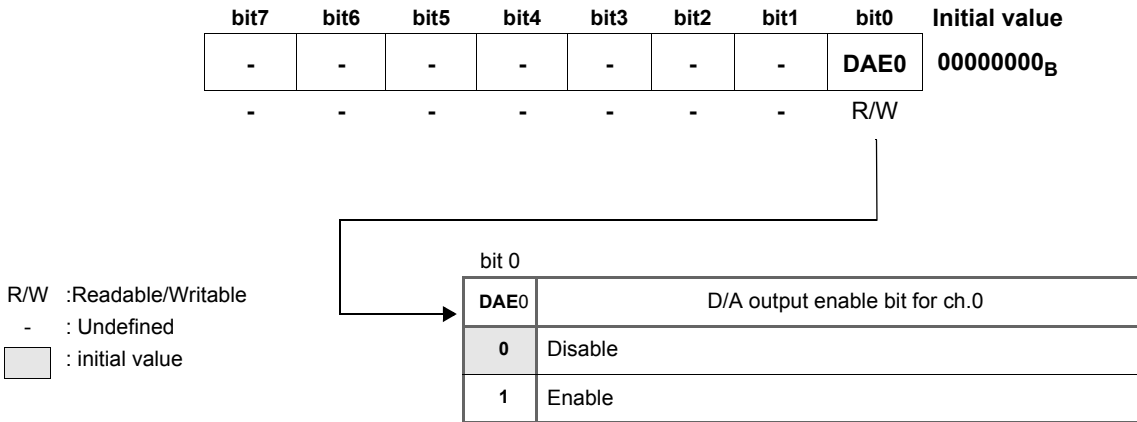


Table 27-3. D/A Control Register 0 (DACR0)

Bit name		Function
bit7 to bit1	Undefined bits	Read: Value is undefined Write: No effect
bit0	DAE0: D/A output enable bit for ch.0	This bit is used to determine if D/A converter output is enabled or disabled. DAE0 is responsible for channel 0. Writing "1" to this bit enables D/A output; similarly "0" disables D/A output. This bit is initialized to "0" at reset. This bit can be read and written.

27.5 Sample Program for the D/A Converter

This section contains a sample program for the D/A converter.

■ Sample Program for the D/A Converter

- Processing specification

Convert a byte to analog and output to Port 7 bit 0.

- Coding example

```

/* Insert initialization routines below for:
    User/System stacks,
    Direct page register,

```

```
Register bank,  
Interrupt level mask register,  
main() routine start address - reset vector starting address */  
  
/* Insert definition in register structure below:  
    structure of DACR0 */  
  
/* Insert definition in register location below:  
    DAT0, DACR0 */  
  
void main(void)  
{  
    IO_DACR0.byte = 1;    /*enable DAC ch.0*/  
    IO_DAT0 = 0;          /*output 0V*/  
    __asm("nop");  
    __asm("nop");        /*short delay*/  
    __asm("nop");  
    IO_DAT0 = 0x80;       /*output 1/2 AVCC*/  
    __asm("nop");  
    __asm("nop");        /*short delay*/  
    __asm("nop");  
    IO_DAT0 = 0xff;       /*output AVCC*/  
    __asm("nop");  
    __asm("nop");        /*short delay*/  
    __asm("nop");  
}
```

Appendix



The appendices provide I/O maps, instructions by F²MC-16LX, and other information.

["A. I/O Maps"](#)

["APPENDIX B Instructions"](#)

["APPENDIX C List of MB90990 Series Interrupt Vectors"](#)

A. I/O Maps

Table 0-1 lists addresses to be assigned to the registers in the peripheral blocks.

■ I/O Maps (Addresses:000000_H to 0000FF_H)

Table 0-1 I/O Map (000000_H to 0000FF_H) (Sheet 1 of 5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
000000 _H , 000001 _H	Reserved				
000002 _H	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX _B
000003 _H	Reserved				
000004 _H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX _B
000005 _H	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX _B
000006 _H	Port 6 data register	PDR6	R/W	Port 6	XXXXXXXX _B
000007 _H	Reserved				
000008 _H	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX _B
000009 _H , 00000A _H	Reserved				
00000B _H	Analog input enable register 5	ADER5	R/W	Port 5, A/D	1 1 1 1 1 1 1 _B
00000C _H	Analog input enable register 6	ADER6	R/W	Port 6, A/D	1 1 1 1 1 1 1 _B
00000D _H	Reserved				
00000E _H	Input level select register 0	ILSR0	R/W	Ports	XXXXXXXX _B
00000F _H	Input level select register 1	ILSR1	R/W	Ports	XXXX0XXX _B
000010 _H , 000011 _H	Reserved				
000012 _H	Port 2 direction register	DDR2	R/W	Port 2	0 0 0 0 0 0 0 _B

Table 0-1 I/O Map (000000_H to 0000FF_H) (Sheet 2 of 5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
000013 _H	Reserved				
000014 _H	Port 4 direction register	DDR4	R/W	Port 4	0 0 0 0 0 0 0 0 _B
000015 _H	Port 5 direction register	DDR5	R/W	Port 5	0 0 0 0 0 0 0 0 _B
000016 _H	Port 6 direction register	DDR6	R/W	Port 6	0 0 0 0 0 0 0 0 _B
000017 _H	Reserved				
000018 _H	Port 8 direction register	DDR8	R/W	Port 8	0 0 0 0 0 0 0 0 _B
000019 _H	Reserved				
00001A _H	Port A direction register	DDRA	W	Port A	0 0 0 0 0 1 1 1 _B
00001B _H to 00001D _H	Reserved				
00001E _H	Port 2 pull-up control register	PUCR2	R/W	Port 2	0 0 0 0 0 0 0 0 _B
00001F _H	Reserved				
000020 _H	Serial mode register 0	SMR0	W, R/W	UART0	0 0 0 0 0 0 0 0 _B
000021 _H	Serial control register 0	SCR0	W, R/W		0 0 0 0 0 0 0 0 _B
000022 _H	Reception/transmission data register 0	RDR0/TDR0	R/W		0 0 0 0 0 0 0 0 _B / 1 1 1 1 1 1 1 1 _B
000023 _H	Serial status register 0	SSR0	R, R/W		0 0 0 0 1 0 0 0 _B
000024 _H	Extended communication control register 0	ECCR0	R, W, R/W		0 0 0 0 0 0 X X _B
000025 _H	Extended status control register 0	ESCR0	R/W		0 0 0 0 0 X 0 0 _B
000026 _H	Baud rate generator register 00	BGR00	R/W, R		0 0 0 0 0 0 0 0 _B
000027 _H	Baud rate generator register 01	BGR01	R/W, R		0 0 0 0 0 0 0 0 _B
000028 _H	Serial mode register 1	SMR1	W, R/W	UART1	0 0 0 0 0 0 0 0 _B
000029 _H	Serial control register 1	SCR1	W, R/W		0 0 0 0 0 0 0 0 _B
00002A _H	Reception/transmission data register 1	RDR1/TDR1	R/W		0 0 0 0 0 0 0 0 _B / 1 1 1 1 1 1 1 1 _B
00002B _H	Serial status register 1	SSR1	R, R/W		0 0 0 0 1 0 0 0 _B
00002C _H	Extended communication control register1	ECCR1	R, W, R/W		0 0 0 0 0 0 X X _B
00002D _H	Extended status control register 1	ESCR1	R/W		0 0 0 0 0 X 0 0 _B
00002E _H	Baud rate generator register 10	BGR10	R/W, R		0 0 0 0 0 0 0 0 _B
00002F _H	Baud rate generator register 11	BGR11	R/W, R		0 0 0 0 0 0 0 0 _B
000030 _H to 00003A _H	Reserved				
00003B _H	Address detection control register 1	PACSR1	R/W	Address Match Detection 1	1 1 0 0 0 0 0 0 _B
00003C _H to 000043 _H	Reserved				

Table 0-1 I/O Map (000000_H to 0000FF_H) (Sheet 3 of 5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
000044 _H	PPGA operation mode control register	PPGCA	R/W	16-bit PPGA/B	0 1 0 0 0 1 1 1 _B
000045 _H	PPGB operation mode control register	PPGCB	R/W		0 1 0 0 0 0 0 1 _B
000046 _H	PPGA/B count clock selection register	PPGAB	R/W		0 0 0 0 0 0 1 0 _B
000047 _H	Reserved				
000048 _H	PPGC operation mode control register	PPGCC	W, R/W	16-bit PPGC/D	0 1 0 0 0 1 1 1 _B
000049 _H	PPGD operation mode control register	PPGCD	W, R/W		0 1 0 0 0 0 0 1 _B
00004A _H	PPGC/D count clock selection register	PPGCD	R/W		0 0 0 0 0 0 1 0 _B
00004B _H	Reserved				
00004C _H	PPGE operation mode control register	PPGCE	W, R/W	16-bit PPGE/F	0 1 0 0 0 1 1 1 _B
00004D _H	PPGF operation mode control register	PPGCF	W, R/W		0 1 0 0 0 0 0 1 _B
00004E _H	PPGE/F count clock selection register	PPGEF	R/W		0 0 0 0 0 0 1 0 _B
00004F _H	Reserved				
000050 _H	Input capture control status register 0/1	ICS01	R/W	Input Capture 0/1	0 0 0 0 0 0 0 0 _B
000051 _H	Input capture edge register 0/1	ICE01	R/W, R		1 1 1 0 1 0 XX _B
000052 _H	Input capture control status register 2/3	ICS23	R/W	Input Capture 2/3	0 0 0 0 0 0 0 0 _B
000053 _H	Input capture edge register 2/3	ICE23	R		1 1 1 1 1 1 XX _B
000054 _H to 000063 _H	Reserved				
000064 _H	Timer control status register 2	TMCSR2	R/W	16-bit Reload Timer 2	0 0 0 0 0 0 0 0 _B
000065 _H	Timer control status register 2	TMCSR2	R/W		1 1 1 1 0 0 0 0 _B
000066 _H	Timer control status register 3	TMCSR3	R/W	16-bit Reload Timer 3	0 0 0 0 0 0 0 0 _B
000067 _H	Timer control status register 3	TMCSR3	R/W		1 1 1 1 0 0 0 0 _B
000068 _H	A/D control status register 0	ADCS0	R/W	A/D Converter	0 0 0 1 1 1 1 0 _B
000069 _H	A/D control status register 1	ADCS1	R/W, W		0 0 0 0 0 0 0 1 _B
00006A _H	A/D data register 0	ADCR0	R		0 0 0 0 0 0 0 0 _B
00006B _H	A/D data register 1	ADCR1	R		1 1 1 1 1 1 0 0 _B
00006C _H	A/D setting register 0	ADSR0	R/W		0 0 0 0 0 0 0 0 _B
00006D _H	A/D setting register 1	ADSR1	R/W		0 0 0 0 0 0 0 0 _B
00006E _H	Low-voltage/CPU operation detection reset control register	LVRC	R/W, W	Low-voltage/CPU Operation Detection Reset	0 0 1 1 1 0 0 0 _B
00006F _H	ROM mirror function select register	ROMM	W	ROM Mirror	1 1 1 1 1 1 0 1 _B
000070 _H to 00007F _H	Reserved				
000080 _H to 00008F _H	Reserved for CAN controller (For more information, see Table 21-1..)				

Table 0-1 I/O Map (000000_H to 0000FF_H) (Sheet 4 of 5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
000090 _H to 00009D _H	Reserved				
00009E _H	Address detection control register 0	PACSR0	R/W	Address Match Detection 0	1 1 0 0 0 0 0 0 _B
00009F _H	Delayed interrupt request generate/cancel register	DIRR	R/W	Delayed Interrupt Generation Module	1 1 1 1 1 1 1 0 _B
0000A0 _H	Low-power consumption mode control register	LPMCR	W, R/W	Low-power Consumption Controller	0 0 0 1 1 0 0 0 _B
0000A1 _H	Clock selection register	CKSCR	R, R/W	Low-power Consumption Controller	1 1 1 1 1 1 0 0 _B
0000A2 _H to 0000A7 _H	Reserved				
0000A8 _H	Watchdog timer control register	WDTC	R, W	Watchdog Timer	X 1 XXX1 1 1 _B
0000A9 _H	Time-base timer control register	TBTC	W, R/W	Time-base Timer	1 1 1 0 0 1 0 0 _B
0000AA _H	Watch timer control register	WTC	R, R/W	Watch Timer	1 X 0 0 1 0 0 0 _B
0000AB _H to 0000AD _H	Reserved				
0000AE _H	Flash memory control status register (Flash device only)	FMCS	R, R/W	Flash Memory	0 0 0 X 0 0 0 0 _B
0000AF _H	Reserved				
0000B0 _H	Interrupt control register 00	ICR00	W, R/W	Interrupt control	0 0 0 0 0 1 1 1 _B
0000B1 _H	Interrupt control register 01	ICR01	W, R/W		0 0 0 0 0 1 1 1 _B
0000B2 _H	Interrupt control register 02	ICR02	W, R/W		0 0 0 0 0 1 1 1 _B
0000B3 _H	Interrupt control register 03	ICR03	W, R/W		0 0 0 0 0 1 1 1 _B
0000B4 _H	Interrupt control register 04	ICR04	W, R/W		0 0 0 0 0 1 1 1 _B
0000B5 _H	Interrupt control register 05	ICR05	W, R/W		0 0 0 0 0 1 1 1 _B
0000B6 _H	Interrupt control register 06	ICR06	W, R/W		0 0 0 0 0 1 1 1 _B
0000B7 _H	Interrupt control register 07	ICR07	W, R/W		0 0 0 0 0 1 1 1 _B
0000B8 _H	Interrupt control register 08	ICR08	W, R/W		0 0 0 0 0 1 1 1 _B
0000B9 _H	Interrupt control register 09	ICR09	W, R/W		0 0 0 0 0 1 1 1 _B
0000BA _H	Interrupt control register 10	ICR10	W, R/W		0 0 0 0 0 1 1 1 _B
0000BB _H	Interrupt control register 11	ICR11	W, R/W		0 0 0 0 0 1 1 1 _B
0000BC _H	Interrupt control register 12	ICR12	W, R/W		0 0 0 0 0 1 1 1 _B
0000BD _H	Interrupt control register 13	ICR13	W, R/W		0 0 0 0 0 1 1 1 _B
0000BE _H	Interrupt control register 14	ICR14	W, R/W		0 0 0 0 0 1 1 1 _B
0000BF _H	Interrupt control register 15	ICR15	W, R/W		0 0 0 0 0 1 1 1 _B
0000C0 _H	D/A converter data 0	DAT0	R/W	D/A converter	XXXXXXXX _B

Table 0-1 I/O Map (000000_H to 0000FF_H) (Sheet 5 of 5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
0000C1 _H	Reserved				
0000C2 _H	D/A control 0	DACR0	R/W	D/A converter	0 0 0 0 0 0 0 _B
0000C3 _H to 0000C9 _H	Reserved				
0000CA _H	DTP/External interrupt enable register 1	ENIR1	R/W	External Interrupt 1	0 0 0 0 0 0 0 _B
0000CB _H	DTP/External interrupt source register 1	EIRR1	R/W		XXXXXXXX _B
0000CC _H	Detection level setting register 1	ELVR1	R/W		0 0 0 0 0 0 0 _B
0000CD _H	Detection level setting register 1	ELVR1	R/W		0 0 0 0 0 0 0 _B
0000CE _H	DTP/External interrupt source select register	EISSR	R/W		0 0 0 0 0 0 0 _B
0000CF _H	PLL/sub clock control register	PSCCR	W	PLL	1 1 1 1 0 0 0 _B
0000D0 _H to 0000D7 _H	Reserved				
0000D8 _H	Serial mode register 2	SMR2	R/W	UART2	0 0 0 0 0 0 0 _B
0000D9 _H	Serial control register 2	SCR2	R/W		0 0 0 0 0 0 0 _B
0000DA _H	Receive/Transmission data register 2	RDR2/TDR2	R/W		0 0 0 0 0 0 0 _B / 1 1 1 1 1 1 1 _B
0000DB _H	Serial status register 2	SSR2	R/W		0 0 0 0 1 0 0 _B
0000DC _H	Extended communication control register 2	ECCR2	R/W		0 0 0 0 0 XX _B
0000DD _H	Extended status control register 2	ESCR2	R/W		0 0 0 0 0 X 0 0 _B
0000DE _H	Baud rate generator register 20	BGR20	R/W		0 0 0 0 0 0 0 _B
0000DF _H	Baud rate generator register 21	BGR21	R/W		0 0 0 0 0 0 0 _B
0000E0 _H to 0000FF _H	Reserved				

A.1 I/O Map (Addresses:007900_H to 007FFF_H)

Table 0-2 I/O Map (007900_H to 007FFF_H) (Sheet 1 of 4)

Address	Register	Abbreviation	Access	Peripheral	Initial value
007900 _H to 007913 _H	Reserved				
007914 _H	Reload register LA	PRLLA	R/W	16-bit PPG A/B	XXXXXXXX _B
007915 _H	Reload register HA	PRLHA	R/W		XXXXXXXX _B
007916 _H	Reload register LB	PRLLB	R/W		XXXXXXXX _B
007917 _H	Reload register HB	PRLHB	R/W		XXXXXXXX _B

Table 0-2 I/O Map (007900_H to 007FFF_H) (Sheet 2 of 4)

Address	Register	Abbreviation	Access	Peripheral	Initial value
007918 _H	Reload register LC	PRLLC	R/W	16-bit PPG C/D	XXXXXXXX _B
007919 _H	Reload register HC	PRLHC	R/W		XXXXXXXX _B
00791A _H	Reload register LD	PRLLD	R/W		XXXXXXXX _B
00791B _H	Reload register HD	PRLHD	R/W		XXXXXXXX _B
00791C _H	Reload register LE	PRLLE	R/W	16-bit PPG E/F	XXXXXXXX _B
00791D _H	Reload register HE	PRLHE	R/W		XXXXXXXX _B
00791E _H	Reload register LF	PRLLF	R/W		XXXXXXXX _B
00791F _H	Reload register HF	PRLHF	R/W		XXXXXXXX _B
007920 _H	Input capture register 0	IPCP0	R	Input Capture 0/1 *	0 0 0 0 0 0 0 0 _B
007921 _H	Input capture register 0	IPCP0	R		0 0 0 0 0 0 0 0 _B
007922 _H	Input capture register 1	IPCP1	R		0 0 0 0 0 0 0 0 _B
007923 _H	Input capture register 1	IPCP1	R		0 0 0 0 0 0 0 0 _B
007924 _H	Input capture register 2	IPCP2	R	Input Capture 2/3 *	0 0 0 0 0 0 0 0 _B
007925 _H	Input capture register 2	IPCP2	R		0 0 0 0 0 0 0 0 _B
007926 _H	Input capture register 3	IPCP3	R		0 0 0 0 0 0 0 0 _B
007927 _H	Input capture register 3	IPCP3	R		0 0 0 0 0 0 0 0 _B
007928 _H to 00793F _H	Reserved				
007940 _H	Timer data register 0	TCDT0	R/W	I/O Timer 0	0 0 0 0 0 0 0 0 _B
007941 _H	Timer data register 0	TCDT0	R/W		0 0 0 0 0 0 0 0 _B
007942 _H	Timer control status register 0	TCCSL0	R/W		0 0 0 0 0 0 0 0 _B
007943 _H	Timer control status register 0	TCCSH0	R/W		0 1 1 0 0 0 0 0 _B
*: For the evaluation product, the initial value is "XXXXXXXX _B ".					
007944 _H to 00794B _H	Reserved				
00794C _H	Timer register 2/ reload register 2	TMR2/TMRLR2	R, W	16-bit Reload Timer 2	XXXXXXXX _B
00794D _H			R, W		XXXXXXXX _B
00794E _H	Timer register 3/ reload register 3	TMR3/TMRLR3	R, W	16-bit Reload Timer 3	XXXXXXXX _B
00794F _H			R, W		XXXXXXXX _B
007950 _H to 00795F _H	Reserved				
007960 _H	Clock supervisor control register	CSVCR	R, R/W	Clock supervisor	0 0 0 1 1 1 0 0 _B
007961 _H to 00796D _H	Reserved				
00796E _H	CAN direct mode register	CDMR	R/W	CAN Clock Sync	1 1 1 1 1 1 1 0 _B

Table 0-2 I/O Map (007900_H to 007FFF_H) (Sheet 3 of 4)

Address	Register	Abbreviation	Access	Peripheral	Initial value
00796F _H to 0079A1 _H	Reserved				
0079A2 _H	FLASH writing control register 0	FWR0	R/W	FLASH	0 0 0 0 0 0 0 0 _B
0079A3 _H	FLASH writing control register 1	FWR1	R/W		0 0 0 0 0 0 0 0 _B
0079A4 _H to 0079B1 _H	Reserved				
0079B2 _H	Low-voltage/CPU operation detection reset setting register	LVRS	R/W	Low-voltage/CPU operation detection reset	1 0 0 0 0 1 1 1 _B
0079B3 _H to 0079B7 _H	Reserved				
0079B8 _H	Clock calibration unit control register	CUCR	R/W	Clock calibration unit	0 0 0 0 0 0 0 0 _B
0079B9 _H	CR trimming setting register	CRTR	R/W		1 1 1 1 0 1 1 1 _B
0079BA _H	CR oscillation timer data register	CUTDL	R/W		0 1 0 1 0 0 0 0 _B
0079BB _H	CR oscillation timer data register	CUTDH	R/W		1 1 0 0 0 0 1 1 _B
0079BC _H	Main oscillation timer data register 1	CUTR1L	R		0 0 0 0 0 0 0 0 _B
0079BD _H	Main oscillation timer data register 1	CUTR1H	R		0 0 0 0 0 0 0 0 _B
0079BE _H	Main oscillation timer data register 2	CUTR2L	R		0 0 0 0 0 0 0 0 _B
0079BF _H	Main oscillation timer data register 2	CUTR2H	R		0 0 0 0 0 0 0 0 _B
0079C0 _H to 0079DF _H	Reserved				
0079E0 _H	Detection address setting register 0	PADR0	R/W	Address Match Detection 0	XXXXXXXX _B
0079E1 _H	Detection address setting register 0	PADR0	R/W		XXXXXXXX _B
0079E2 _H	Detection address setting register 0	PADR0	R/W		XXXXXXXX _B
0079E3 _H	Detection address setting register 1	PADR1	R/W		XXXXXXXX _B
0079E4 _H	Detection address setting register 1	PADR1	R/W		XXXXXXXX _B
0079E5 _H	Detection address setting register 1	PADR1	R/W		XXXXXXXX _B
0079E6 _H	Detection address setting register 2	PADR2	R/W		XXXXXXXX _B
0079E7 _H	Detection address setting register 2	PADR2	R/W		XXXXXXXX _B
0079E8 _H	Detection address setting register 2	PADR2	R/W		XXXXXXXX _B
0079E9 _H to 0079EF _H	Reserved				

Table 0-2 I/O Map (007900_H to 007FFF_H) (Sheet 4 of 4)

Address	Register	Abbreviation	Access	Peripheral	Initial value
0079F0 _H	Detection address setting register 3	PADR3	R/W	Address Match Detection 1	XXXXXXXX _B
0079F1 _H	Detection address setting register 3	PADR3	R/W		XXXXXXXX _B
0079F2 _H	Detection address setting register 3	PADR3	R/W		XXXXXXXX _B
0079F3 _H	Detection address setting register 4	PADR4	R/W		XXXXXXXX _B
0079F4 _H	Detection address setting register 4	PADR4	R/W		XXXXXXXX _B
0079F5 _H	Detection address setting register 4	PADR4	R/W		XXXXXXXX _B
0079F6 _H	Detection address setting register 5	PADR5	R/W		XXXXXXXX _B
0079F7 _H	Detection address setting register 5	PADR5	R/W		XXXXXXXX _B
0079F8 _H	Detection address setting register 5	PADR5	R/W		XXXXXXXX _B
0079F9 _H to 007BFF _H	Reserved				
007C00 _H to 007CFF _H	Reserved for CAN controller (For more information, see Table 21-1..)				
007D00 _H to 007DFF _H	Reserved for CAN controller (For more information, see Table 21-2..)				
007E00 _H to 007FFF _H	Reserved				

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: Readable/Writable

R: Read only

W: Write only

■ 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"

Code: CM44-00202-3E

B.1 Instruction Types

The F²MC-16LX supports 351 types of instructions. Addressing is enabled by using an effective address field of each instruction or using the instruction code itself.

■ Instruction Types

The F²MC-16LX supports the following 351 types of instructions:

- 41 transfer instructions (byte)
- 38 transfer instructions (word or long word)
- 42 addition/subtraction instructions (byte, word, or long word)
- 12 increment/decrement instructions (byte, word, or long word)
- 11 comparison instructions (byte, word, or long word)
- 11 unsigned multiplication/division instructions (word or long word)
- 11 signed multiplication/division instructions (word or long word)
- 39 logic instructions (byte or word)
- 6 logic instructions (long word)
- 6 sign inversion instructions (byte or word)
- 1 normalization instruction (long word)
- 18 shift instructions (byte, word, or long word)
- 50 branch instructions
- 6 accumulator operation instructions (byte or word)
- 28 other control instructions (byte, word, or long word)
- 21 bit operation instructions
- 10 string instructions

B.2 Addressing

With the F²MC-16LX, the address format is determined by the instruction effective address field or the instruction code itself (implied). When the address format is determined by the instruction code itself, specify an address in accordance with the instruction code used. Some instructions permit the user to select several types of addressing.

■ Addressing

The F²MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)

■ Effective Address Field

Table B.2-1 lists the address formats specified by the effective address field.

Table B.2-1 Effective Address Field

Code	Representation			Address format	Default bank
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	DTB
09	@RW1				DTB
0A	@RW2				ADB
0B	@RW3				SPB
0C	@RW0+			Register indirect with post increment	DTB
0D	@RW1+				DTB
0E	@RW2+				ADB
0F	@RW3+				SPB
10	@RW0+disp8			Register indirect with 8-bit displacement	DTB
11	@RW1+disp8				DTB
12	@RW2+disp8				ADB
13	@RW3+disp8				SPB
14	@RW4+disp8			Register indirect with 8-bit displacement	DTB
15	@RW5+disp8				DTB
16	@RW6+disp8				ADB
17	@RW7+disp8				SPB
18	@RW0+disp16			Register indirect with 16-bit displacement	DTB
19	@RW1+disp16				DTB
1A	@RW2+disp16				ADB
1B	@RW3+disp16				SPB
1C	@RW0+RW7			Register indirect with index	DTB
1D	@RW1+RW7			Register indirect with index	DTB
1E	@PC+disp16			PC indirect with 16-bit displacement	PCB
1F	addr16			Direct address	DTB

B.3 Direct Addressing

An operand value, register, or address is specified explicitly in direct addressing mode.

■ Direct Addressing

● Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

Figure B.3-1 Example of Immediate Addressing (#imm)

MOVW A, #01212H (This instruction stores the operand value in A.)

Before execution	A	2 2 3 3 : 4 4 5 5	
After execution	A	4 4 5 5 : 1 2 1 2	(Some instructions transfer AL to AH.)

● Register direct addressing

Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

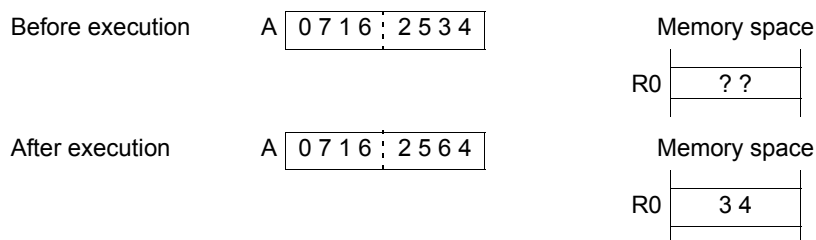
Table B.3-1 Direct Addressing Registers

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

Figure B.3-2 Example of Register Direct Addressing

MOV R0, A (This instruction transfers the eight low-order bits of A to the general-purpose register R0.)

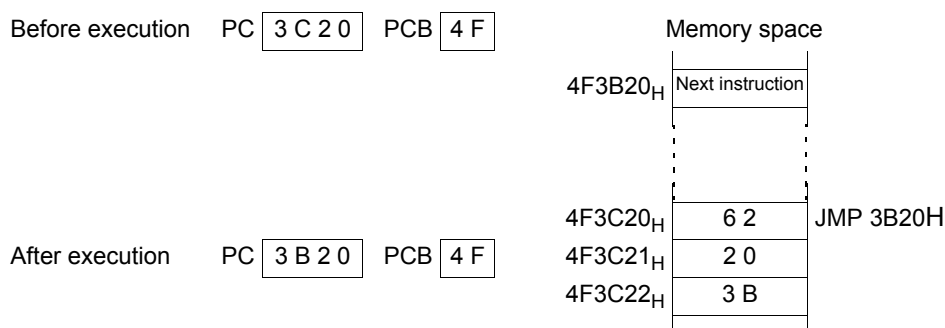


● Direct branch addressing (addr16)

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bit23 to bit16 of the address are specified by the program counter bank register (PCB).

Figure B.3-3 Example of Direct Branch Addressing (addr16)

JMP 3B20H (This instruction causes an unconditional branch by direct branch addressing in a bank.)

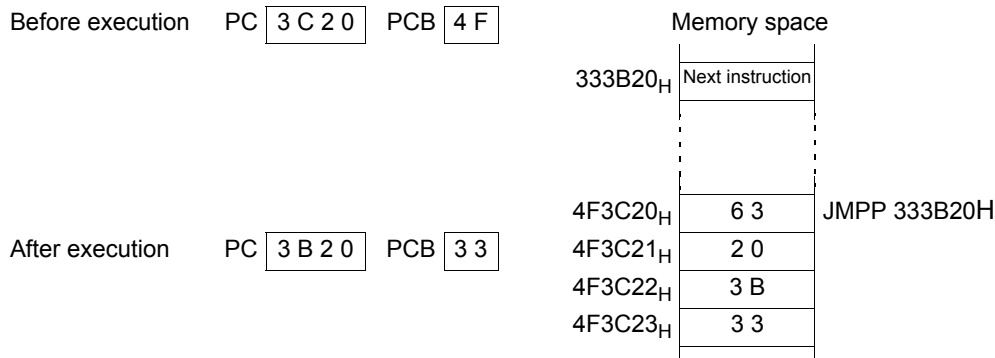


- Physical direct branch addressing (addr24)

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)

JMPP 333B20H (This instruction causes an unconditional branch by direct branch 24-bit addressing.)

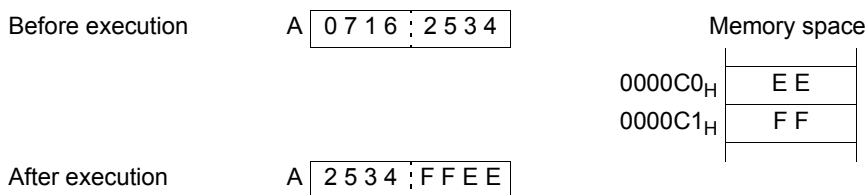


- I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000_H to 0000FF_H is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

Figure B.3-5 Example of I/O Direct Addressing (io)

MOVW A, I:0C0H (This instruction reads data by I/O direct addressing and stores it in A.)



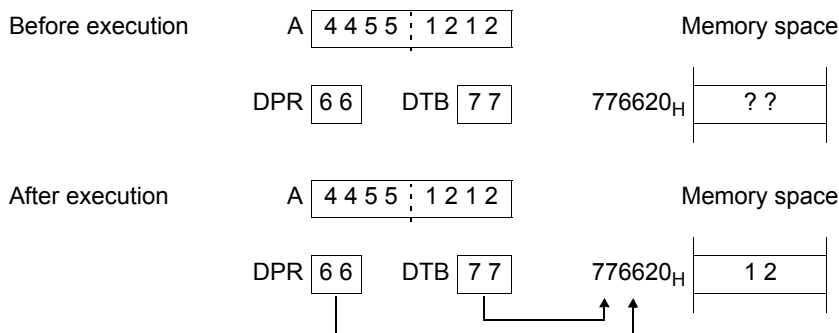
Note : "I:" is Addressing Specifier that shows the I/O Direct Addressing.

● 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)

MOV S:20H, A (This instruction writes the contents of the eight low-order bits of A in abbreviated direct addressing mode.)



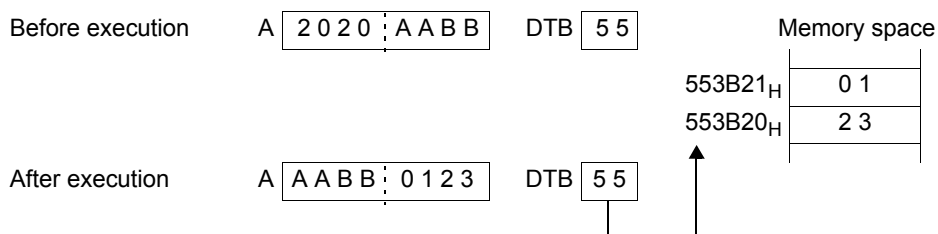
Note : "S:" is Addressing Specifier that shows the Abbreviated Direct Addressing.

● 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)

MOVW A, 3B20H (This instruction reads data by direct addressing and stores it in A.)

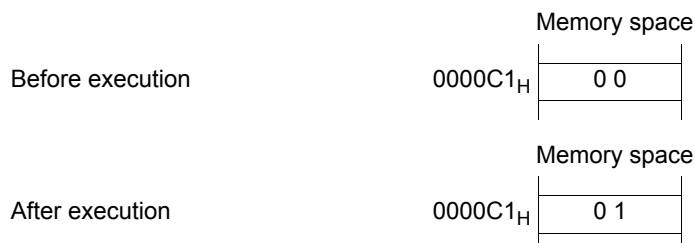


● I/O direct bit addressing (io:bp)

Specify bits in physical addresses 000000_H to 0000FF_H explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)

SETB I:0C1H:0 (This instruction sets bits by I/O direct bit addressing.)



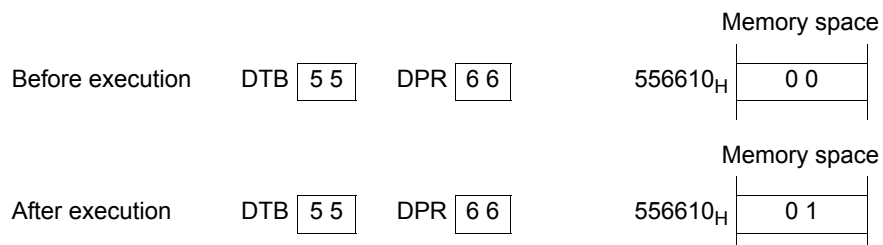
Note : "I:" is Addressing Specifier that shows the I/O Direct Addressing.

● 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)

SETB S:10H:0 (This instruction sets bits by abbreviated direct bit addressing.)



Note : "S:" is Addressing Specifier that shows the Abbreviated Direct Addressing.

● 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)

SETB 2222H : 0 (This instruction sets bits by direct bit addressing.)

Before execution	DTB	5 5	Memory space	
			552222 _H	0 0
After execution	DTB	5 5	Memory space	
			552222 _H	0 1

● 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)

CALLV #15 (This instruction causes a branch to the address indicated by the interrupt vector specified in an operand.)

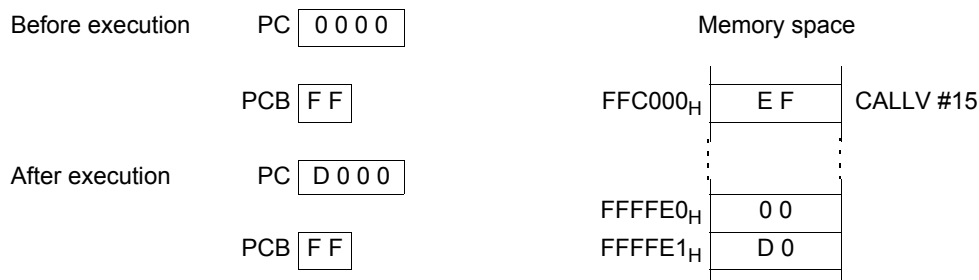


Table B.3-2 CALLV Vector List

Instruction	Vector address L	Vector address H
CALLV #0	XXFFFE _H	XXFFFF _H
CALLV #1	XXFFFC _H	XXFFFD _H
CALLV #2	XXFFFA _H	XXFFFB _H
CALLV #3	XXFFF8 _H	XXFFF9 _H
CALLV #4	XXFFF6 _H	XXFFF7 _H
CALLV #5	XXFFF4 _H	XXFFF5 _H
CALLV #6	XXFFF2 _H	XXFFF3 _H
CALLV #7	XXFFF0 _H	XXFFF1 _H
CALLV #8	XXFFEE _H	XXFFEF _H
CALLV #9	XXFFEC _H	XXFFED _H
CALLV #10	XXFFEA _H	XXFFEB _H
CALLV #11	XXFFE8 _H	XXFFE9 _H
CALLV #12	XXFFE6 _H	XXFFE7 _H
CALLV #13	XXFFE4 _H	XXFFE5 _H
CALLV #14	XXFFE2 _H	XXFFE3 _H
CALLV #15	XXFFE0 _H	XXFFE1 _H

Note: A PCB register value is set in XX.

Note:

When the program counter bank register (PCB) is FF_H, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2).

B.4 Indirect Addressing

In indirect addressing mode, an address is specified indirectly by the address data of an operand.

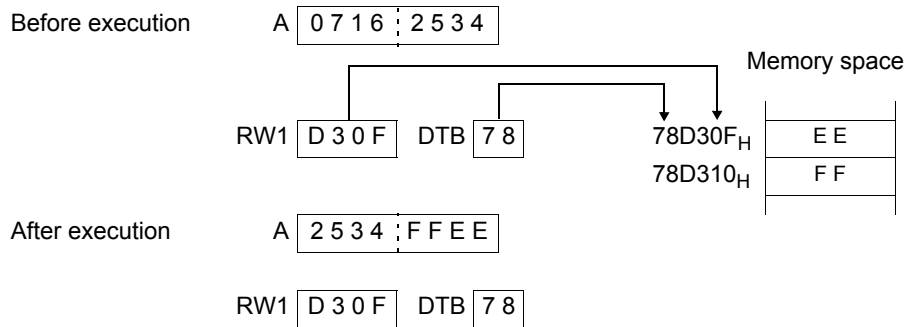
■ Indirect Addressing

● Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)

MOVW A, @RW1 (This instruction reads data by register indirect addressing and stores it in A.)



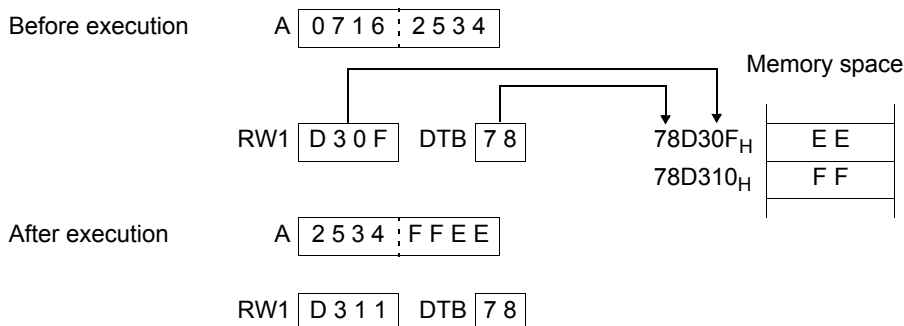
● Register indirect addressing with post increment (@RWj+ j = 0 to 3)

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)

MOVW A, @RW1+ (This instruction reads data by register indirect addressing with post increment and stores it in A.)

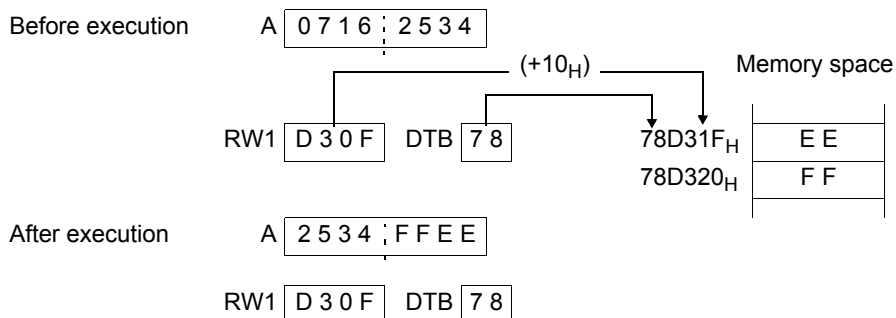


● Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)

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)

MOVW A, @RW1+10H (This instruction reads data by register indirect addressing with an offset and stores it in A.)

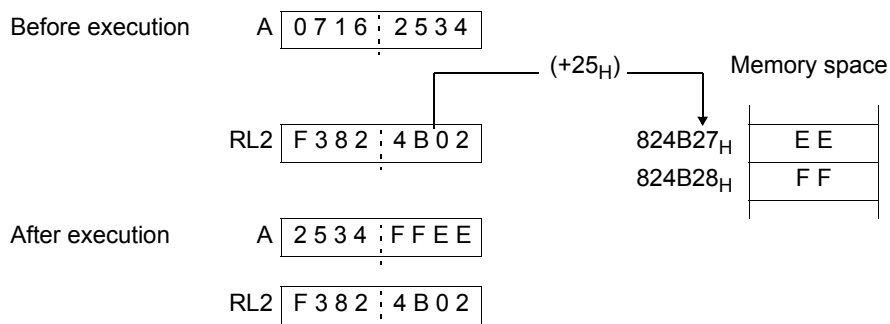


● Long register indirect addressing with offset (@RLi + disp8 i = 0 to 3)

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)

MOVW A, @RL2+25H (This instruction reads data by long register indirect addressing with an offset and stores it in A.)



● Program counter indirect addressing with offset (@PC + disp16)

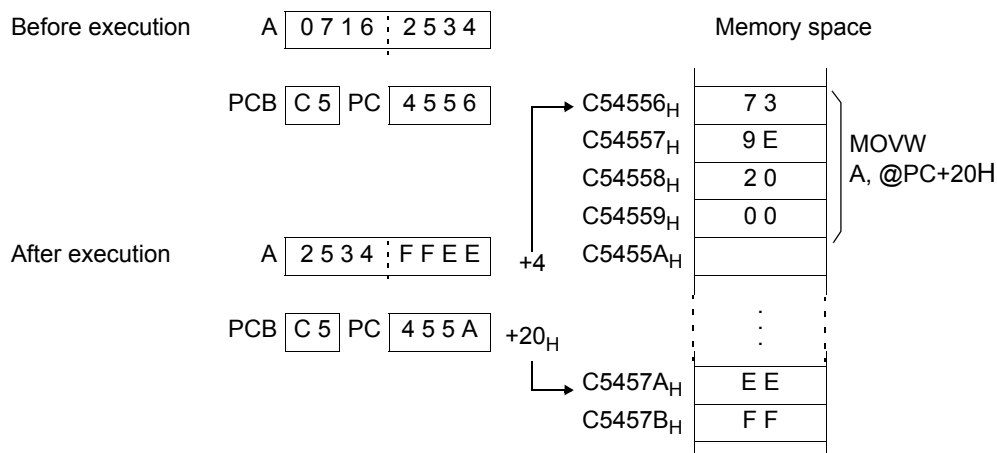
Memory is accessed using the address indicated by (instruction address + 4 + disp16). The offset is one word long. Address bits 16 to 23 are specified by the program counter bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address + disp16):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8

- MOVW eam, #imm16

Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16)

MOVW A, @PC+20H (This instruction reads data by program counter indirect addressing with an offset and stores it in A.)

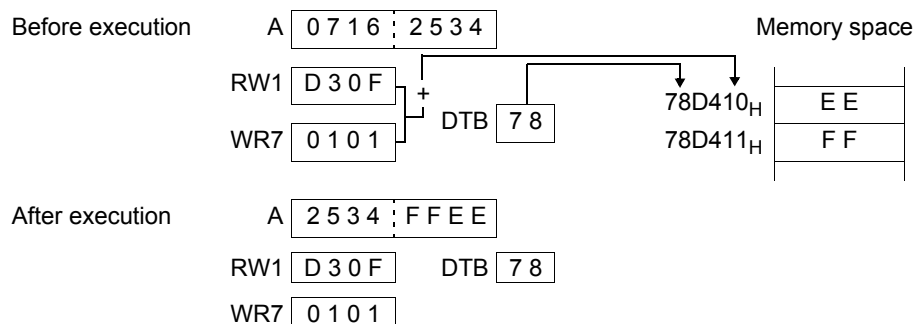


● Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)

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)

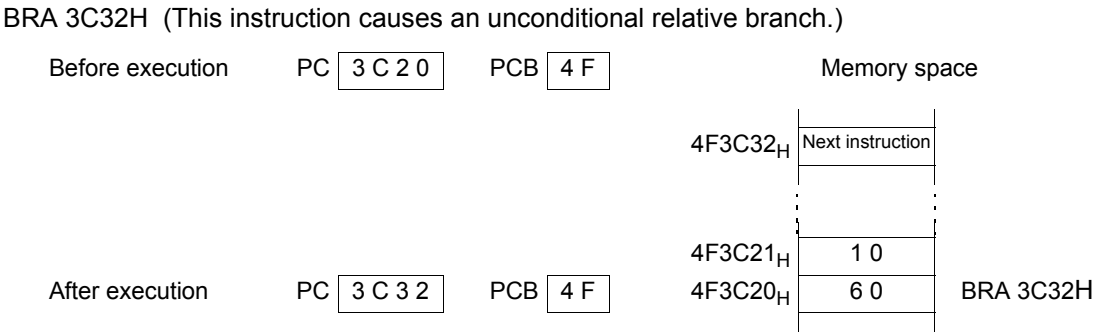
MOVW A, @RW1+RW7 (This instruction reads data by register indirect addressing with a base index and stores it in A.)



● Program counter relative branch addressing (rel)

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-kilobyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program counter bank register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



● Register list (rlst)

Specify a register to be pushed onto or popped from a stack.

Figure B.4-8 Configuration of the Register List

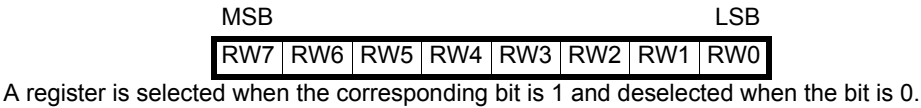
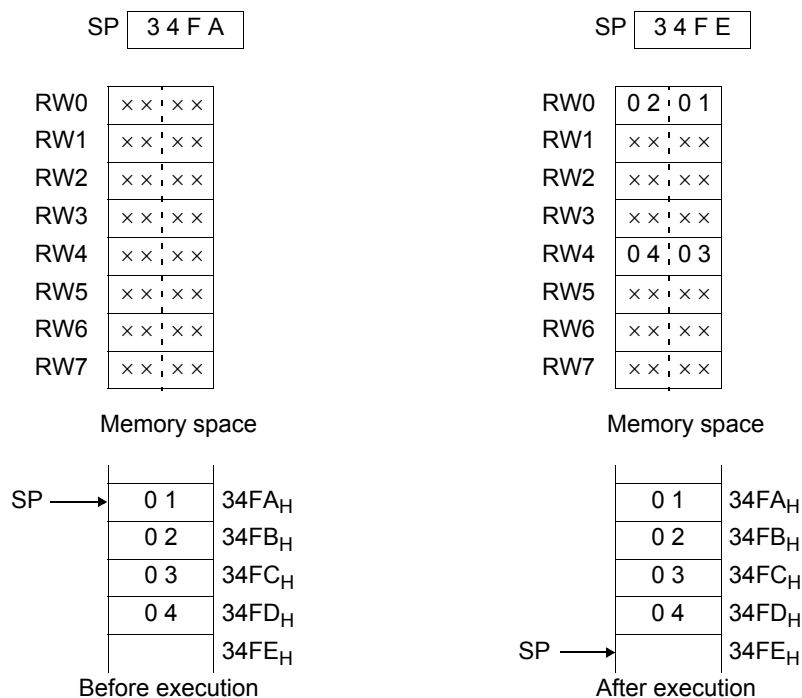


Figure B.4-9 Example of Register List (rlist)

POPW RW0, RW4 (This instruction transfers memory data indicated by the SP to multiple word registers indicated by the register list.)

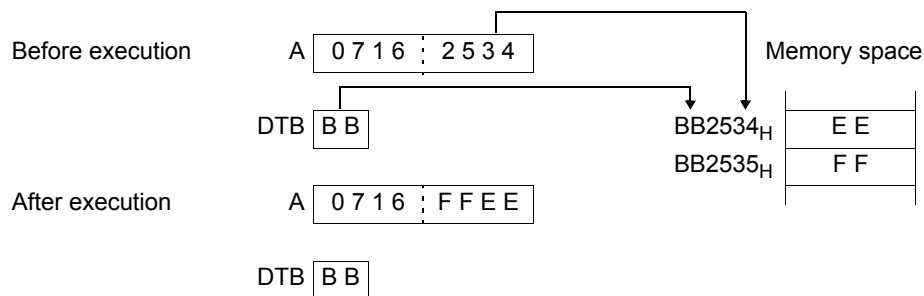


● Accumulator indirect addressing (@A)

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)

MOVW A, @A (This instruction reads data by accumulator indirect addressing and stores it in A.)

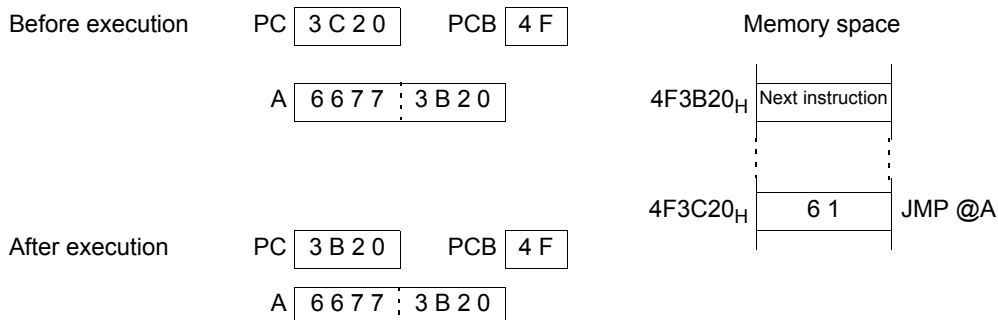


● Accumulator indirect branch addressing (@A)

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program counter bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)

JMP @A (This instruction causes an unconditional branch by accumulator indirect branch addressing.)

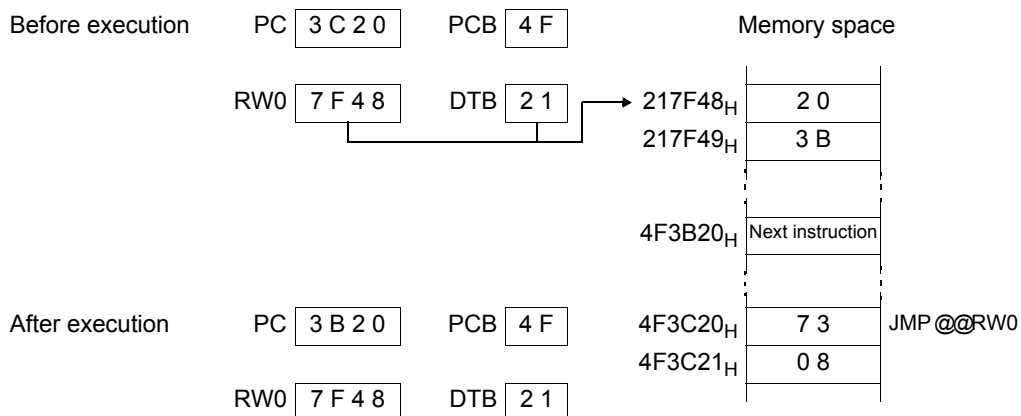


● Indirect specification branch addressing (@ear)

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)

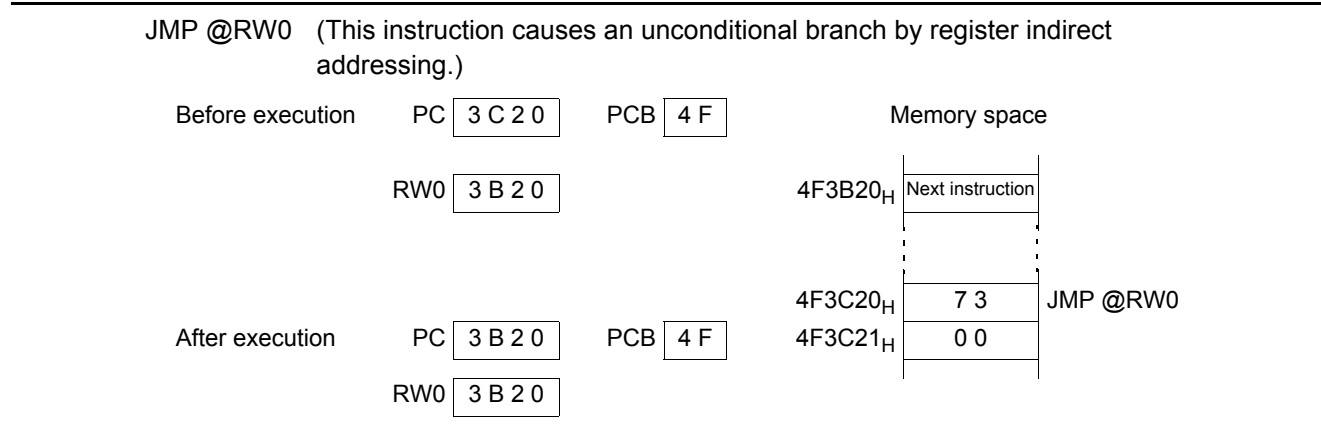
JMP @@RW0 (This instruction causes an unconditional branch by register indirect addressing.)



● Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



B.5 Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.

■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

■ Calculating the Execution Cycle Count

Table B.5-1 lists execution cycle counts and Table B.5-2 and Table B.5-3 summarize correction value data.

Table B.5-1 Execution Cycle Counts in Each Addressing Mode

Code	Operand	(a) *	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00 07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08 0B	@RWj	2	1
0C 0F	@RWj+	4	2
10 17	@RWi+disp8	2	1
18 1B	@RWi+disp16	2	1
1C	@RW0+RW7	4	2
1D	@RW1+RW7	4	2
1E	@PC+disp16	2	0
1F	addr16	1	0

*: (a) is used for ~ (cycle count) and B (correction value) in ""B.8 F2MC-16LX Instruction List"".

Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles

Operand	(b) byte *		(c) word *		(d) long *	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus 16-bit even address	+1	1	+1	1	+2	2
External data bus 16-bit odd address	+1	1	+4	2	+8	4
External data bus 8-bits	+1	1	+4	2	+8	4

*: (b), (c), and (d) are used for ~ (cycle count) and B (correction value) in ""B.8 F2MC-16LX Instruction List"".

Note:

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16-bits	-	+3
External data bus 8-bits	+3	-

Notes:

- When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.
- Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.

B.6 Effective address field

Table B.6-1 shows the effective address field.

■ Effective Address Field

Table B.6-1 Effective Address Field

Code	Representation			Address format	Byte count of extended address part *
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	0
09	@RW1				
0A	@RW2				
0B	@RW3				
0C	@RW0+			Register indirect with post increment	0
0D	@RW1+				
0E	@RW2+				
0F	@RW3+				
10	@RW0+disp8			Register indirect with 8-bit displacement	1
11	@RW1+disp8				
12	@RW2+disp8				
13	@RW3+disp8				
14	@RW4+disp8				
15	@RW5+disp8				
16	@RW6+disp8				
17	@RW7+disp8				
18	@RW0+disp16			Register indirect with 16-bit displacement	2
19	@RW1+disp16				
1A	@RW2+disp16				
1B	@RW3+disp16				
1C	@RW0+RW7			Register indirect with index	0
1D	@RW1+RW7			Register indirect with index	0
1E	@PC+disp16			PC indirect with 16-bit displacement	2
1F	addr16			Direct address	2

*1: Each byte count of the extended address part applies to + in the # (byte count) column in "B.8 F2MC-16LX Instruction List".

B.7 How to Read the Instruction List

Table B.7-1 describes the items used in ""B.8 F2MC-16LX Instruction List"", and Table B.7-2 describes the symbols used in the same list.

■ Description of Instruction Presentation Items and Symbols

Table B.7-1 Description of Items in the Instruction List (2/2)

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number of following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table B.2-1 for the alphabetical letters in items.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bit15 to bit08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00 _H or FF _H to AH after AL sign extension.
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change S: Set upon instruction execution. R: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	

Table B.7-1 Description of Items in the Instruction List (2/2)

Item	Description
RMW	<p>Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory).</p> <p>*: Read Modify Write instruction -: Not Read Modify Write instruction</p> <p>Note: Cannot be used for an address that has different meanings between read and write operations.</p>

Table B.7-2 Explanation on Symbols in the Instruction List (2/2)

Symbol	Explanation
A	<p>The bit length used varies depending on the 32-bit accumulator instruction.</p> <p>Byte: Low-order 8 bits of byte AL Word: 16 bits of word AL Long word: 32 bits of AL and AH</p>
AH	16 high-order bits of A
AL	16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	program counter bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Abbreviated direct addressing
addr16	Direct addressing

Table B.7-2 Explanation on Symbols in the Instruction List (2/2)

Symbol	Explanation
addr24	Physical direct addressing
ad24 0-15	Bit0 to bit15 of addr24
ad24 16-23	Bit16 to bit23 of addr24
io	I/O area (000000 _H to 0000FF _H)
#imm4	4-bit immediate data
#imm8	8-bit immediate data
#imm16	16-bit immediate data
#imm32	32-bit immediate data
ext (imm8)	16-bit data obtained by sign extension of 8-bit immediate data
disp8	8-bit displacement
disp16	16-bit displacement
bp	Bit offset
vct4	Vector number (0 to 15)
vct8	Vector number (0 to 255)
() b	Bit address
rel	PC relative branch
ear	Effective addressing (code 00 _H to 07 _H)
eam	Effective addressing (code 08 _H to 1F _H)
rlst	Register list

B.8 F²MC-16LX Instruction List

Table B.8-1 to Table B.8-18 list the instructions used by the F²MC-16LX.

■ F²MC-16LX Instruction List

Table B.8-1 41 Transfer Instructions (Byte)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A,dir	2	3	0	(b)	byte (A) ← (dir)	Z	*	-	-	-	*	*	-	-	-
MOV A,addr16	3	4	0	(b)	byte (A) ← (addr16)	Z	*	-	-	-	*	*	-	-	-
MOV A,Ri	1	2	1	0	byte (A) ← (Ri)	Z	*	-	-	-	*	*	-	-	-
MOV A,ear	2	2	1	0	byte (A) ← (ear)	Z	*	-	-	-	*	*	-	-	-
MOV A,eam	2+	3 + (a)	0	(b)	byte (A) ← (eam)	Z	*	-	-	-	*	*	-	-	-
MOV A,io	2	3	0	(b)	byte (A) ← (io)	Z	*	-	-	-	*	*	-	-	-
MOV A,#imm8	2	2	0	0	byte (A) ← imm8	Z	*	-	-	-	*	*	-	-	-
MOV A,@A	2	3	0	(b)	byte (A) ← ((A))	Z	-	-	-	-	*	*	-	-	-
MOV A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	Z	*	-	-	-	*	*	-	-	-
MOVN A,#imm4	1	1	0	0	byte (A) ← imm4	Z	*	-	-	-	R	*	-	-	-
MOVX A,dir	2	3	0	(b)	byte (A) ← (dir)	X	*	-	-	-	*	*	-	-	-
MOVX A,addr16	3	4	0	(b)	byte (A) ← (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX A,Ri	2	2	1	0	byte (A) ← (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX A,ear	2	2	1	0	byte (A) ← (ear)	X	*	-	-	-	*	*	-	-	-
MOVX A,eam	2+	3 + (a)	0	(b)	byte (A) ← (eam)	X	*	-	-	-	*	*	-	-	-
MOVX A,io	2	3	0	(b)	byte (A) ← (io)	X	*	-	-	-	*	*	-	-	-
MOVX A,#imm8	2	2	0	0	byte (A) ← imm8	X	*	-	-	-	*	*	-	-	-
MOVX A,@A	2	3	0	(b)	byte (A) ← ((A))	X	-	-	-	-	*	*	-	-	-
MOVX A,@RWi+disp8	2	5	1	(b)	byte (A) ← ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOV dir,A	2	3	0	(b)	byte (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV addr16,A	3	4	0	(b)	byte (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,A	1	2	1	0	byte (Ri) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV ear,A	2	2	1	0	byte (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV eam,A	2+	3 + (a)	0	(b)	byte (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV io,A	2	3	0	(b)	byte (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV @RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,ear	2	3	2	0	byte (Ri) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOV Ri,eam	2+	4 + (a)	1	(b)	byte (Ri) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOV ear,Ri	2	4	2	0	byte (ear) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV eam,Ri	2+	5 + (a)	1	(b)	byte (eam) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV Ri,#imm8	2	2	1	0	byte (Ri) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV io,#imm8	3	5	0	(b)	byte (io) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV dir,#imm8	3	5	0	(b)	byte (dir) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV ear,#imm8	3	2	1	0	byte (ear) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV eam,#imm8	3+	4 + (a)	0	(b)	byte (eam) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV @AL,AH	2	3	0	(b)	byte ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCH A,ear	2	4	2	0	byte (A) ↔ (ear)	Z	-	-	-	-	-	-	-	-	-
XCH A,eam	2+	5 + (a)	0	2 × (b)	byte (A) ↔ (eam)	Z	-	-	-	-	-	-	-	-	-
XCH Ri,ear	2	7	4	0	byte (Ri) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCH Ri,eam	2+	9 + (a)	2	2 × (b)	byte (Ri) ↔ (eam)	-	-	-	-	-	-	-	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-2 38 Transfer Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,dir	2	3	0	(c)	word (A) ← (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	4	0	(c)	word (A) ← (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	1	1	0	0	word (A) ← (SP)	-	*	-	-	-	*	*	-	-	-

Table B.8-2 38 Transfer Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,RWi	1	2	1	0	word (A) ← (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW A,ear	2	2	1	0	word (A) ← (ear)	-	*	-	-	-	*	*	-	-	-
MOVW A,eam	2+	3 + (a)	0	(c)	word (A) ← (eam)	-	*	-	-	-	*	*	-	-	-
MOVW A,io	2	3	0	(c)	word (A) ← (io)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	3	0	(c)	word (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVW A,#imm16	3	2	0	0	word (A) ← imm16	-	*	-	-	-	*	*	-	-	-
MOVW A,@RWi+disp8	2	5	1	(c)	word (A) ← ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@RLi+disp8	3	10	2	(c)	word (A) ← ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW dir,A	2	3	0	(c)	word (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW addr16,A	3	4	0	(c)	word (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW SP,A	1	1	0	0	word (SP) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,A	1	2	1	0	word (RWi) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW ear,A	2	2	1	0	word (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW eam,A	2+	3 + (a)	0	(c)	word (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW io,A	2	3	0	(c)	word (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RWi+disp8,A	2	5	1	(c)	word ((RWi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RLi+disp8,A	3	10	2	(c)	word ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,ear	2	3	2	0	word (RWi) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,eam	2+	4 + (a)	1	(c)	word (RWi) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVW ear,RWi	2	4	2	0	word (ear) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW eam,RWi	2+	5 + (a)	1	(c)	word (eam) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,#imm16	3	2	1	0	word (RWi) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW io,#imm16	4	5	0	(c)	word (io) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW ear,#imm16	4	2	1	0	word (ear) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW eam,#imm16	4+	4 + (a)	0	(c)	word (eam) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW @AL,AH	2	3	0	(c)	word ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCHW A,ear	2	4	2	0	word (A) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW A,eam	2+	5 + (a)	0	2 × (c)	word (A) ↔ (eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,ear	2	7	4	0	word (RWi) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,eam	2+	9 + (a)	2	2 × (c)	word (RWi) ↔ (eam)	-	-	-	-	-	-	-	-	-	-
MOVL A,ear	2	4	2	0	long (A) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVL A,eam	2+	5 + (a)	0	(d)	long (A) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVL A,#imm32	5	3	0	0	long (A) ← imm32	-	-	-	-	-	*	*	-	-	-
MOVL ear,A	2	4	2	0	long (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVL eam,A	2+	5 + (a)	0	(d)	long(eam) ← (A)	-	-	-	-	-	*	*	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a), (c), and (d) in the table.

Table B.8-3 42 Addition/Subtraction Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD A,#imm8	2	2	0	0	byte (A) ← (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	5	0	(b)	byte (A) ← (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	3	1	0	byte (A) ← (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	3	2	0	byte (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD eam,A	2+	5 + (a)	0	2 × (b)	byte (eam) ← (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	0	byte (A) ← (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	3	1	0	byte (A) ← (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDDC A	1	3	0	0	byte (A) ← (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	0	byte (A) ← (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	5	0	(b)	byte (A) ← (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	3	1	0	byte (A) ← (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	3	2	0	byte (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB eam,A	2+	5 + (a)	0	2 × (b)	byte (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	0	byte (A) ← (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	3	1	0	byte (A) ← (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC A	1	3	0	0	byte (A) ← (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	0	word (A) ← (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	3	1	0	word (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	0	word (A) ← (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	3	2	0	word (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	3	1	0	word (A) ← (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	0	word (A) ← (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	3	1	0	word (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	0	word (A) ← (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	3	2	0	word (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	3	1	0	word (A) ← (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL A,ear	2	6	2	0	long (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	0	long (A) ← (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	6	2	0	long (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	0	long (A) ← (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-4 12 Increment/decrement Instructions (Byte, Word, Long Word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	$2 \times (b)$	byte (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	$2 \times (b)$	byte (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	$2 \times (c)$	word (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	$2 \times (c)$	word (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) \leftarrow (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	$2 \times (d)$	long (eam) \leftarrow (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) \leftarrow (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	$2 \times (d)$	long (eam) \leftarrow (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-5 11 Compare Instructions (Byte, Word, Long Word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-6 11 Unsigned Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU A	1	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW A	1	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0 7: Overflow 15: Normal
 *2: 4: Division by 0 8: Overflow 16: Normal
 *3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal
 *4: 4: Division by 0 7: Overflow 22: Normal
 *5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal
 *6: (b): Division by 0 or overflow 2 × (b): Normal
 *7: (c): Division by 0 or overflow 2 × (c): Normal
 *8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.
 *9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.
 *10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.
 *11: 3: Word (AH) is 0. 11: Word (AH) is not 0.
 *12: 4: Word (ear) is 0. 12: Word (ear) is not 0.
 *13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-7 11 Signed Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV A	2	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL A	2	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW A	2	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal

*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal

*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal

*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal
When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal
When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal

*6: (b): Division by 0 or overflow, 2 × (b): Normal

*7: (c): Division by 0 or overflow, 2 × (c): Normal

*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative

*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative

*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative

*11: 3: Word (AH) is 0, 16: result is positive, 19: result is negative

*12: 4: Word (ear) is 0, 17: result is positive, 20: result is negative

*13: 5+(a): Word (eam) is 0, 18+(a): result is positive, 21+(a): result is negative

Notes:

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-8 39 Logic 1 Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A,#imm8	2	2	0	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	3	1	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	2	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	3	1	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	2	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	3	1	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	2	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	3	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOT eam	2+	5+(a)	0	2 × (b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	3	1	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	2	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	3	1	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	2	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	3	1	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	2	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	3	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW eam	2+	5+(a)	0	2 × (c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-9 6 Logic 2 Instructions (Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	6	2	0	long (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	long (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	long (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (d) in the table.

Table B.8-10 6 Sign Inversion Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	0	byte (A) ← 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	byte (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 × (b)	byte (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	word (A) ← 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 × (c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-11 1 Normalization Instruction (Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A,R0	2	*1	1	0	long (A) ← Shift left to the position where '1' is set for the first time. byte (R0) ← Shift count at that time	-	-	-	-	-	-	*	-	-	-

*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

Table B.8-12 18 Shift Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	3	2	0	byte (ear) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	*
ASR A,R0	2	*1	1	0	byte (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSR A,R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSL A,R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW A/SHRW A	1	2	0	0	word (A) ← Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW A/SHLW A	1	2	0	0	word (A) ← Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW A,R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW A,R0	2	*1	1	0	word (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL A,R0	2	*2	1	0	long (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL A,R0	2	*2	1	0	long (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL A,R0	2	*2	1	0	long (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

*1: 6 when R0 is 0; otherwise, 5 + (R0)

*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-13 31 Branch 1 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/ BNE	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/ BHS	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT rel	2	*1	0	0	Branch on (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE rel	2	*1	0	0	Branch on (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP @A	1	2	0	0	word (PC) ← (A)	-	-	-	-	-	-	-	-	-	-
JMP addr16	3	3	0	0	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
JMP @ear	2	3	1	0	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
JMP @eam	2+	4+(a)	0	(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
JMPP @ear *3	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP @eam *3	2+	6+(a)	0	(d)	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP addr24	4	4	0	0	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL @ear *4	2	6	1	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
CALL @eam *4	2+	7+(a)	0	2 × (c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
CALL addr16 *5	3	6	0	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
CALLV #vct4 *5	1	7	0	2 × (c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP @ear *6	2	10	2	2 × (c)	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
CALLP @eam *6	2+	11+(a)	0	*2	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
CALLP addr24 *7	4	10	0	2 × (c)	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-

*1: 4 when a branch is made; otherwise, 3

*2: 3 × (c) + (b)

*3: Read (word) of branch destination address

*4: W: Save to stack (word) R: Read (word) of branch destination address

*5: Save to stack (word)

*6: W: Save to stack (long word), R: Read (long word) of branch destination address

*7: Save to stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-14 19 Branch 2 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE eam,#imm16,rel*9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ ear,rel	3	*5	2	0	byte (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ eam,rel	3+	*6	2	2 × (b)	byte (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*5	2	0	word (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ eam,rel	3+	*6	2	2 × (c)	word (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	16	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	17	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI	1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK #imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK	1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET *10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP *11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

*1: 5 when a branch is made; otherwise, 4

*2: 13 when a branch is made; otherwise, 12

*3: 7+(a) when a branch is made; otherwise, 6+(a)

*4: 8 when a branch is made; otherwise, 7

*5: 7 when a branch is made; otherwise, 6

*6: 8+(a) when a branch is made; otherwise, 7+(a)

*7: 3 × (b) + 2 × (c) when jumping to the next interruption request; 6 × (c) when returning from the current interruption

*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption

*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.

*10: Return from stack (word)

*11: Return from stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-15 28 Other Control Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	4	0	(c)	word (SP) \leftarrow (SP) - 2, ((SP)) \leftarrow (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*5	*4	(SP) \leftarrow (SP) - 2n, ((SP)) \leftarrow (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	0	(c)	word (A) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	0	(c)	word (AH) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	4	0	(c)	word (PS) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*5	*4	(rlst) \leftarrow ((SP)), (SP) \leftarrow (SP) + 2n	-	-	-	-	-	-	-	-	-	-
JCTX @A	1	14	0	6 \times (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	0	byte (CCR) \leftarrow (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	0	byte (CCR) \leftarrow (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	0	byte (RP) \leftarrow imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	0	byte (ILM) \leftarrow imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	1	0	word (RWi) \leftarrow ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	1	0	word (RWi) \leftarrow eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	1	0	0	word (A) \leftarrow ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	0	word (A) \leftarrow eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	0	word (SP) \leftarrow (SP) + ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	0	word (SP) \leftarrow (SP) + imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	0	byte (A) \leftarrow (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	0	byte (brg2) \leftarrow (A)	-	-	-	-	-	*	*	-	-	-
NOP	1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB	1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB	1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB	1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB	1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC	1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR	1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2

*2: $7 + 3 \times (\text{POP count}) + 2 \times (\text{POP last register number})$, 7 when RLST = 0 (no transfer register)*3: $29 + 3 \times (\text{PUSH count}) - 3 \times (\text{PUSH last register number})$, 8 when RLST = 0 (no transfer register)*4: (POP count) \times (c) or (PUSH count) \times (c)

*5: (POP count) or (PUSH count)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (c) in the table.

Table B.8-16 21 Bit Operand Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A,dir:bp	3	5	0	(b)	byte (A) ← (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 × (b)	bit (dir:bp)b ← (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	7	0	2 × (b)	bit (addr16:bp)b ← (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	6	0	2 × (b)	bit (io:bp)b ← (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	7	0	2 × (b)	bit (dir:bp)b ← 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	7	0	2 × (b)	bit (addr16:bp)b ← 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	7	0	2 × (b)	bit (io:bp)b ← 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	7	0	2 × (b)	bit (dir:bp)b ← 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	7	0	2 × (b)	bit (addr16:bp)b ← 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	7	0	2 × (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	*2	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*3	0	2 × (b)	Branch on (addr16:bp) b = 1, bit (addr16:bp) b ← 1	-	-	-	-	-	-	*	-	-	*
WBTS io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1: 8 when a branch is made; otherwise, 7

*2: 7 when a branch is made; otherwise, 6

*3: 10 when the condition is met; otherwise, 9

*4: Undefined count

*5: Until the condition is met

Note:

See Table B.5-1 and Table B.5-2 for information on (b) in the table.

Table B.8-17 6 Accumulator Operation Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 ↔ (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW	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	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*5	*3	byte transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*5	*3	byte transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*8	*4	byte search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*8	*4	byte search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*8	*3	byte fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*5	*6	word transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*5	*6	word transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*8	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*8	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*8	*6	word fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, $4 + 7 \times (RW0)$ when the counter expires, or $7n + 5$ when a match occurs

*2: 5 when RW0 is 0; otherwise, $4 + 8 \times (RW0)$

*3: $(b) \times (RW0) + (b) \times (RW0)$ When the source and destination access different areas, calculate the (b) item individually.

*4: $(b) \times n$

*5: $2 \times (b) \times (RW0)$

*6: $(c) \times (RW0) + (c) \times (RW0)$ When the source and destination access different areas, calculate the (c) item individually.

*7: $(c) \times n$

*8: $(b) \times (RW0)$

Note:

m: RW0 value (counter value), n: Loop count

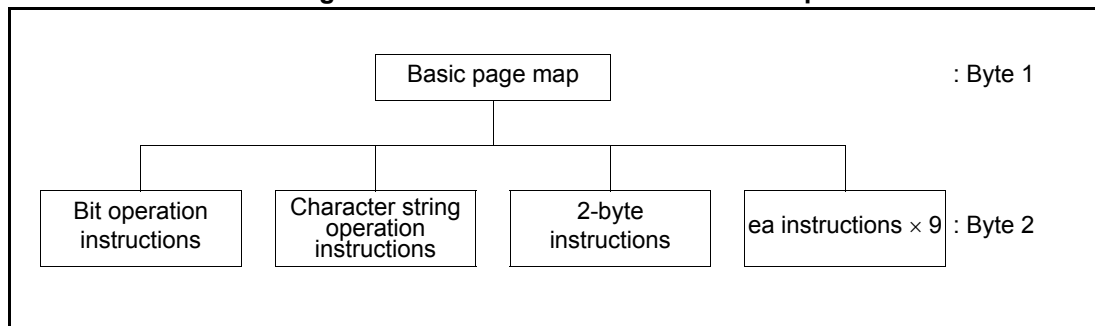
See Table B.5-1 and Table B.5-2 for information on (b) and (c) in the table.

B.9 Instruction Map

Each F²MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 to Table B.9-21 summarize the F²MC-16LX instruction map.

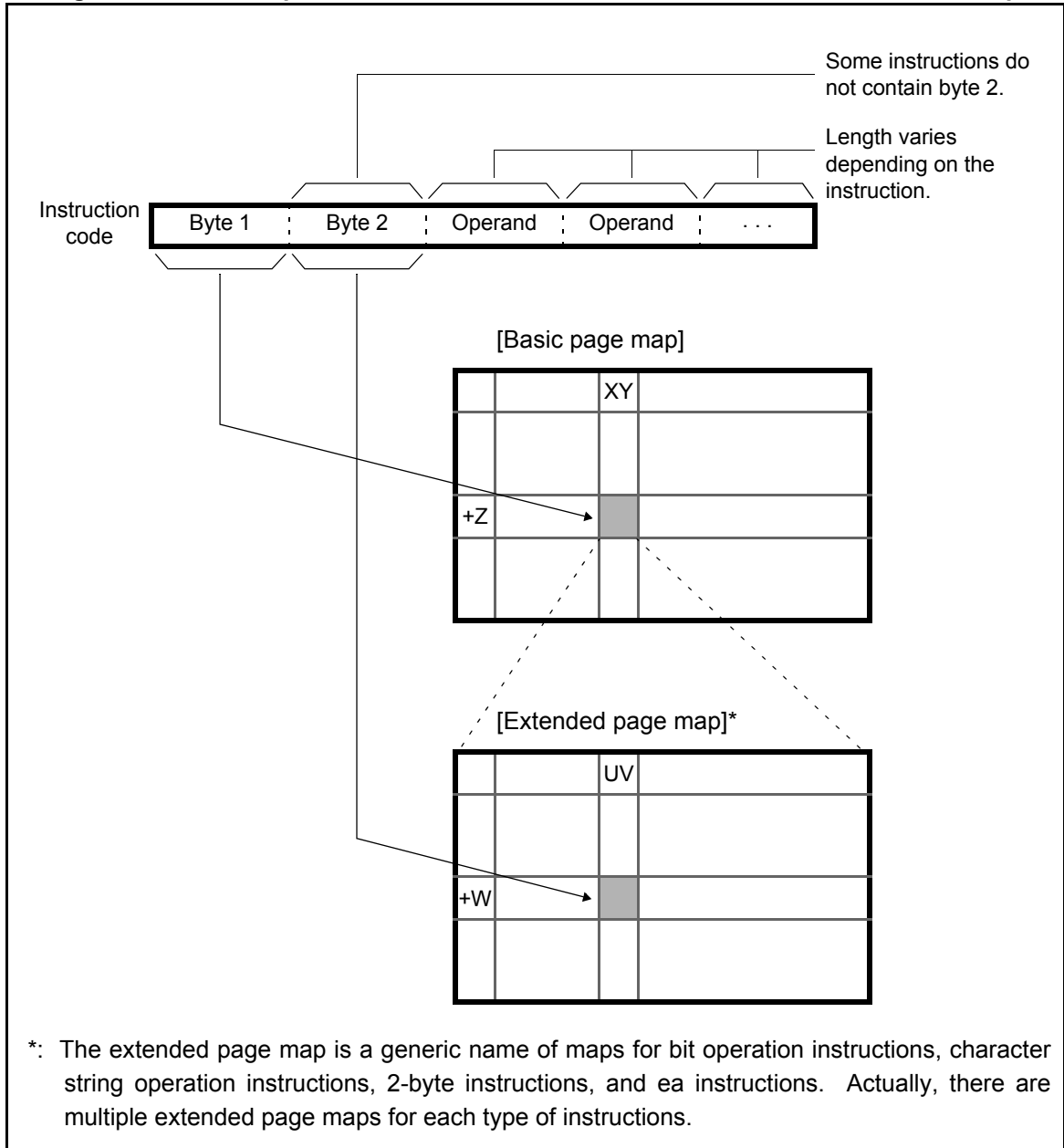
■ Structure of Instruction Map

Figure B.9-1 Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 shows the correspondence between an actual instruction code and instruction map.

Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map



An example of an instruction code is shown in Table B.9-1.

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
MOV A, ADB	60 +F=6F	00 +0=00
CBNE @RW2+d8, #8, rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	NOP	CMR	ADD A, dir	ADD A, #8	MOV A, dir	MOV A, io	BRA rel	ea instruction 1	MOV A, Ri	MOV Ri, A	MOV Ri, #8	MOVX A, Ri	MOVX A, @RWi+d8	MOVN A, #4	CALL #vct4	BZ/BEQ rel
+1	INT9	NCC	SUB A, dir	SUB A, #8	MOV dir, A	MOV io, A	JMP @A	ea instruction 2								BNZ/BNE rel
+2	ADDDC A	SUBDC	ADDC A	SUBC A	MOV A, #8	MOV A, addr16	JMP addr16	ea instruction 3								BC/BLO rel
+3	NEG A	JCTX @A	CMP A	CMP A, #8	MOVX A, #8	MOV addr16, A	JMPP addr24	ea instruction 4								BNC/BHS rel
+4	PCB	EXT	AND CCR, #8	AND A, #8	MOV dir, #8	MOV io, #8	CALL addr16	ea instruction 5								BN rel
+5	DTB	ZEXT	OR CCR, #8	OR A, #8	MOVX A, dir	MOVX A, io	CALLP addr24	ea instruction 6								BP rel
+6	ADB	SWAP	DIVU A	XOR A, #8	MOVW A, SP	MOVW io, #16	RETP	ea instruction 7								BV rel
+7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP, A	MOVW A, addr16	RET	ea instruction 8								BNV rel
+8	LINK #imm8	ADDL A, #32	ADDW A	ADDW A, #16	MOVW A, dir	MOVW A, dir	INT #vct8	ea instruction 9	MOVW A, RWi	MOVW RWi, A	MOVW RWi, #16	MOVW A, @RWi+d8	MOVW @RWi+d8, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT addr16	MOVEA RWi, ea								BNT rel
+A	MOV RP, #8	MOV ILM, #8	CBNE A, #8, rel	CWBNE A, #16, rel	MOVW A, #16	MOVW A, addr16	INTP addr24	MOV Ri, ea								BLT rel
+B	NEGW A	CMPL A, #32	CMPL A	CMPL A, #16	MOVL A, #32	MOVW addr16, A	RETI	MOVW RWi, ea								BGE rel
+C	LSLW A	EXTW	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 operation instruction	XCH Ri, ea								BLS rel
+F	LSRW A	ADDSP #16	MULW A	NOTW A	PUSHW r1st	POPW r1st	2-byte instruction	XCHW RWi, ea								BHI rel

Table B.9-3 Bit Operation Instruction Map (First Byte = 6C_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV B, A, io:bp		MOV B, io:bp, A		CLRB, io:bp		SETB, io:bp		BBC, io:bp, rel		BBS, io:bp, rel		WBTS, io:bp		WBTC, io:bp	
+1																
+2																
+3																
+4																
+5																
+6																
+7																
+8	MOV B, A, dir:bp	MOV B, A, addr16:bp	MOV B, dir:bp, A	MOV B, addr16:bp, A	CLRB, dir:bp	CLRB, addr16:bp	SETB, dir:bp	SETB, addr16:bp	BBC, dir:bp, rel	BBC, addr16:bp, rel	BBS, dir:bp, rel	BBS, addr16:bp, rel				SBBS, addr16:bp
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table B.9-4 Character String Operation Instruction Map (First Byte = 6E_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI, PCB, PCB, MOVSD, MOVSWI, MOVSWD								SCEQI, PCB	SCEQD, PCB	SCWEQI, PCB	SCWEQD, PCB	FILSI, PCB	FILSWI, PCB		
+1	PCB, DTB								DTB	DTB	DTB	DTB	DTB	DTB	DTB	
+2	PCB, ADB								ADB	ADB	ADB	ADB	ADB	ADB	ADB	
+3	PCB, SPB								SPB	SPB	SPB	SPB	SPB	SPB	SPB	
+4	DTB, PCB															
+5	DTB, DTB															
+6	DTB, ADB															
+7	DTB, SPB															
+8	ADB, PCB															
+9	ADB, DTB															
+A	ADB, ADB															
+B	ADB, SPB															
+C	SPB, PCB															
+D	SPB, DTB															
+E	SPB, ADB															
+F	SPB, SPB															

Table B.9-5 2-byte Instruction Map (First Byte = 6F_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A, DTB	MOV DTB, A	MOVX A, @RL0+d8	MOV @RL0+d8, A	MOV @RL0+d8, A											
+1	MOV A, ADB	MOV ADB, A														
+2	MOV A, SSB	MOV SSB, A	MOVX A, @RL1+d8	MOV @RL1+d8, A	MOV @RL1+d8, A											
+3	MOV A, USB	MOV USB, A														
+4	MOV A, DPR	MOV DPR, A	MOVX A, @RL2+d8	MOV @RL2+d8, A	MOV @RL2+d8, A											
+5	MOV A, @A	MOV @AL, AH														
+6	MOV A, PCB	MOV A, @A	MOVX A, @RL3+d8	MOV @RL3+d8, A	MOV @RL3+d8, A											
+7	ROL A	ROL A														
+8				MOVW @RL0+d8, A	MOVW @RL0+d8, A			MUL A								
+9								MULW A								
+A				MOVW @RL1+d8, A	MOVW @RL1+d8, A			DIV A								
+B																
+C	LSLW A, R0	LSLL A, R0	LSL A, R0	MOVW @RL2+d8, A	MOVW @RL2+d8, A											
+D	MOVW A, @A	MOVW @AL, AH	NRML A, R0													
+E	ASRW A, R0	ASRL A, R0	ASR A, R0	MOVW @RL3+d8, A	MOVW @RL3+d8, A											
+F	LSRW A, R0	LSRL A, R0	LSR A, R0													

Table B.9-6 ea Instruction 1 (First Byte = 70_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBNE ↓, CBWNE ↓										CBNE ↓, CBNE ↓	
+0	ADDL A, RL0', @RW0+d8	ADDL A, RL0', @RW0+d8	SUBL A, RL0', @RW0+d8	SUBL A, RL0', @RW0+d8	RW0', @RW0+d8 #16, rel	CMPL A, RL0', @RW0+d8	CMPL A, RL0', @RW0+d8	CMPL A, RL0', @RW0+d8	ANDL A, RL0', @RW0+d8	ANDL A, RL0', @RW0+d8	ORL A, RL0', @RW0+d8	ORL A, RL0', @RW0+d8	XORL A, RL0', @RW0+d8	XORL A, RL0', @RW0+d8	R0', @RW0+d8 #8, rel	
+1	ADDL A, RL0', @RW1+d8	ADDL A, RL0', @RW1+d8	SUBL A, RL0', @RW1+d8	SUBL A, RL0', @RW1+d8	RW1', @RW1+d8 #16, rel	CMPL A, RL0', @RW1+d8	CMPL A, RL0', @RW1+d8	CMPL A, RL0', @RW1+d8	ANDL A, RL0', @RW1+d8	ANDL A, RL0', @RW1+d8	ORL A, RL0', @RW1+d8	ORL A, RL0', @RW1+d8	XORL A, RL0', @RW1+d8	XORL A, RL0', @RW1+d8	R1', @RW1+d8 #8, rel	
+2	ADDL A, RL1', @RW2+d8	ADDL A, RL1', @RW2+d8	SUBL A, RL1', @RW2+d8	SUBL A, RL1', @RW2+d8	RW2', @RW2+d8 #16, rel	CMPL A, RL1', @RW2+d8	CMPL A, RL1', @RW2+d8	CMPL A, RL1', @RW2+d8	ANDL A, RL1', @RW2+d8	ANDL A, RL1', @RW2+d8	ORL A, RL1', @RW2+d8	ORL A, RL1', @RW2+d8	XORL A, RL1', @RW2+d8	XORL A, RL1', @RW2+d8	R2', @RW2+d8 #8, rel	
+3	ADDL A, RL1', @RW3+d8	ADDL A, RL1', @RW3+d8	SUBL A, RL1', @RW3+d8	SUBL A, RL1', @RW3+d8	RW3', @RW3+d8 #16, rel	CMPL A, RL1', @RW3+d8	CMPL A, RL1', @RW3+d8	CMPL A, RL1', @RW3+d8	ANDL A, RL1', @RW3+d8	ANDL A, RL1', @RW3+d8	ORL A, RL1', @RW3+d8	ORL A, RL1', @RW3+d8	XORL A, RL1', @RW3+d8	XORL A, RL1', @RW3+d8	R3', @RW3+d8 #8, rel	
+4	ADDL A, RL2', @RW4+d8	ADDL A, RL2', @RW4+d8	SUBL A, RL2', @RW4+d8	SUBL A, RL2', @RW4+d8	RW4', @RW4+d8 #16, rel	CMPL A, RL2', @RW4+d8	CMPL A, RL2', @RW4+d8	CMPL A, RL2', @RW4+d8	ANDL A, RL2', @RW4+d8	ANDL A, RL2', @RW4+d8	ORL A, RL2', @RW4+d8	ORL A, RL2', @RW4+d8	XORL A, RL2', @RW4+d8	XORL A, RL2', @RW4+d8	R4', @RW4+d8 #8, rel	
+5	ADDL A, RL2', @RW5+d8	ADDL A, RL2', @RW5+d8	SUBL A, RL2', @RW5+d8	SUBL A, RL2', @RW5+d8	RW5', @RW5+d8 #16, rel	CMPL A, RL2', @RW5+d8	CMPL A, RL2', @RW5+d8	CMPL A, RL2', @RW5+d8	ANDL A, RL2', @RW5+d8	ANDL A, RL2', @RW5+d8	ORL A, RL2', @RW5+d8	ORL A, RL2', @RW5+d8	XORL A, RL2', @RW5+d8	XORL A, RL2', @RW5+d8	R5', @RW5+d8 #8, rel	
+6	ADDL A, RL3', @RW6+d8	ADDL A, RL3', @RW6+d8	SUBL A, RL3', @RW6+d8	SUBL A, RL3', @RW6+d8	RW6', @RW6+d8 #16, rel	CMPL A, RL3', @RW6+d8	CMPL A, RL3', @RW6+d8	CMPL A, RL3', @RW6+d8	ANDL A, RL3', @RW6+d8	ANDL A, RL3', @RW6+d8	ORL A, RL3', @RW6+d8	ORL A, RL3', @RW6+d8	XORL A, RL3', @RW6+d8	XORL A, RL3', @RW6+d8	R6', @RW6+d8 #8, rel	
+7	ADDL A, RL3', @RW7+d8	ADDL A, RL3', @RW7+d8	SUBL A, RL3', @RW7+d8	SUBL A, RL3', @RW7+d8	RW7', @RW7+d8 #16, rel	CMPL A, RL3', @RW7+d8	CMPL A, RL3', @RW7+d8	CMPL A, RL3', @RW7+d8	ANDL A, RL3', @RW7+d8	ANDL A, RL3', @RW7+d8	ORL A, RL3', @RW7+d8	ORL A, RL3', @RW7+d8	XORL A, RL3', @RW7+d8	XORL A, RL3', @RW7+d8	R7', @RW7+d8 #8, rel	
+8	ADDL A, @RW0', @RW0+d16	ADDL A, @RW0', @RW0+d16	SUBL A, @RW0', @RW0+d16	SUBL A, @RW0', @RW0+d16	@RW0', @RW0+d16 #16, rel	CMPL A, @RW0', @RW0+d16	CMPL A, @RW0', @RW0+d16	CMPL A, @RW0', @RW0+d16	ANDL A, @RW0', @RW0+d16	ANDL A, @RW0', @RW0+d16	ORL A, @RW0', @RW0+d16	ORL A, @RW0', @RW0+d16	XORL A, @RW0', @RW0+d16	XORL A, @RW0', @RW0+d16	@RW0', @RW0+d16 #8, rel	
+9	ADDL A, @RW1', @RW1+d16	ADDL A, @RW1', @RW1+d16	SUBL A, @RW1', @RW1+d16	SUBL A, @RW1', @RW1+d16	@RW1', @RW1+d16 #16, rel	CMPL A, @RW1', @RW1+d16	CMPL A, @RW1', @RW1+d16	CMPL A, @RW1', @RW1+d16	ANDL A, @RW1', @RW1+d16	ANDL A, @RW1', @RW1+d16	ORL A, @RW1', @RW1+d16	ORL A, @RW1', @RW1+d16	XORL A, @RW1', @RW1+d16	XORL A, @RW1', @RW1+d16	@RW1', @RW1+d16 #8, rel	
+A	ADDL A, @RW2', @RW2+d16	ADDL A, @RW2', @RW2+d16	SUBL A, @RW2', @RW2+d16	SUBL A, @RW2', @RW2+d16	@RW2', @RW2+d16 #16, rel	CMPL A, @RW2', @RW2+d16	CMPL A, @RW2', @RW2+d16	CMPL A, @RW2', @RW2+d16	ANDL A, @RW2', @RW2+d16	ANDL A, @RW2', @RW2+d16	ORL A, @RW2', @RW2+d16	ORL A, @RW2', @RW2+d16	XORL A, @RW2', @RW2+d16	XORL A, @RW2', @RW2+d16	@RW2', @RW2+d16 #8, rel	
+B	ADDL A, @RW3', @RW3+d16	ADDL A, @RW3', @RW3+d16	SUBL A, @RW3', @RW3+d16	SUBL A, @RW3', @RW3+d16	@RW3', @RW3+d16 #16, rel	CMPL A, @RW3', @RW3+d16	CMPL A, @RW3', @RW3+d16	CMPL A, @RW3', @RW3+d16	ANDL A, @RW3', @RW3+d16	ANDL A, @RW3', @RW3+d16	ORL A, @RW3', @RW3+d16	ORL A, @RW3', @RW3+d16	XORL A, @RW3', @RW3+d16	XORL A, @RW3', @RW3+d16	@RW3', @RW3+d16 #8, rel	
+C	ADDL A, @RW0+', @RW0+RW7	ADDL A, @RW0+', @RW0+RW7	SUBL A, @RW0+', @RW0+RW7	SUBL A, @RW0+', @RW0+RW7	Use prohibited	CMPL A, @RW0+', @RW0+RW7	CMPL A, @RW0+', @RW0+RW7	CMPL A, @RW0+', @RW0+RW7	ANDL A, @RW0+', @RW0+RW7	ANDL A, @RW0+', @RW0+RW7	ORL A, @RW0+', @RW0+RW7	ORL A, @RW0+', @RW0+RW7	XORL A, @RW0+', @RW0+RW7	XORL A, @RW0+', @RW0+RW7	Use prohibited	@RW0+RW7 #8, rel
+D	ADDL A, @RW1+', @RW1+RW7	ADDL A, @RW1+', @RW1+RW7	SUBL A, @RW1+', @RW1+RW7	SUBL A, @RW1+', @RW1+RW7	Use prohibited	CMPL A, @RW1+', @RW1+RW7	CMPL A, @RW1+', @RW1+RW7	CMPL A, @RW1+', @RW1+RW7	ANDL A, @RW1+', @RW1+RW7	ANDL A, @RW1+', @RW1+RW7	ORL A, @RW1+', @RW1+RW7	ORL A, @RW1+', @RW1+RW7	XORL A, @RW1+', @RW1+RW7	XORL A, @RW1+', @RW1+RW7	Use prohibited	@RW1+RW7 #8, rel
+E	ADDL A, @RW2+', @RW2+PC+d16	ADDL A, @RW2+', @RW2+PC+d16	SUBL A, @RW2+', @RW2+PC+d16	SUBL A, @RW2+', @RW2+PC+d16	Use prohibited	CMPL A, @RW2+', @RW2+PC+d16	CMPL A, @RW2+', @RW2+PC+d16	CMPL A, @RW2+', @RW2+PC+d16	ANDL A, @RW2+', @RW2+PC+d16	ANDL A, @RW2+', @RW2+PC+d16	ORL A, @RW2+', @RW2+PC+d16	ORL A, @RW2+', @RW2+PC+d16	XORL A, @RW2+', @RW2+PC+d16	XORL A, @RW2+', @RW2+PC+d16	Use prohibited	@RW2+PC+d16 #8, rel
+F	ADDL A, @RW3+', @RW3+addr16	ADDL A, @RW3+', @RW3+addr16	SUBL A, @RW3+', @RW3+addr16	SUBL A, @RW3+', @RW3+addr16	Use prohibited	CMPL A, @RW3+', @RW3+addr16	CMPL A, @RW3+', @RW3+addr16	CMPL A, @RW3+', @RW3+addr16	ANDL A, @RW3+', @RW3+addr16	ANDL A, @RW3+', @RW3+addr16	ORL A, @RW3+', @RW3+addr16	ORL A, @RW3+', @RW3+addr16	XORL A, @RW3+', @RW3+addr16	XORL A, @RW3+', @RW3+addr16	Use prohibited	@RW3+addr16 #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, @RW0+d8	JMPP @ @RL0, @RW0+d8	CALLP @RL0, @RW0+d8	CALLP @ @RL0, @RW0+d8	INCL RL0, @RW0+d8	INCL RL0, @RW0+d8	DECL RL0, @RW0+d8	DECL RL0, @RW0+d8	MOVL A, RL0, @RW0+d8	MOVL A, @RW0+d8	MOVL RL0, A, @RW0+d8	MOVL RL0, A, @RW0+d8	MOV R0, #8, @RW0+d8	MOV R0, #8, @RW0+d8	MOVEA A, RW0, @RW0+d8	MOVEA A, @RW0+d8
+1	JMPP @RL0, @RW1+d8	JMPP @ @RL0, @RW1+d8	CALLP @RL0, @RW1+d8	CALLP @ @RL0, @RW1+d8	INCL RL0, @RW1+d8	INCL RL0, @RW1+d8	DECL RL0, @RW1+d8	DECL RL0, @RW1+d8	MOVL A, RL0, @RW1+d8	MOVL A, @RW1+d8	MOVL RL0, A, @RW1+d8	MOVL RL0, A, @RW1+d8	MOV R1, #8, @RW1+d8	MOV R1, #8, @RW1+d8	MOVEA A, RW1, @RW1+d8	MOVEA A, @RW1+d8
+2	JMPP @RL1, @RW2+d8	JMPP @ @RL1, @RW2+d8	CALLP @RL1, @RW2+d8	CALLP @ @RL1, @RW2+d8	INCL RL1, @RW2+d8	INCL RL1, @RW2+d8	DECL RL1, @RW2+d8	DECL RL1, @RW2+d8	MOVL A, RL1, @RW2+d8	MOVL A, @RW2+d8	MOVL RL1, A, @RW2+d8	MOVL RL1, A, @RW2+d8	MOV R2, #8, @RW2+d8	MOV R2, #8, @RW2+d8	MOVEA A, RW2, @RW2+d8	MOVEA A, @RW2+d8
+3	JMPP @RL1, @RW3+d8	JMPP @ @RL1, @RW3+d8	CALLP @RL1, @RW3+d8	CALLP @ @RL1, @RW3+d8	INCL RL1, @RW3+d8	INCL RL1, @RW3+d8	DECL RL1, @RW3+d8	DECL RL1, @RW3+d8	MOVL A, RL1, @RW3+d8	MOVL A, @RW3+d8	MOVL RL1, A, @RW3+d8	MOVL RL1, A, @RW3+d8	MOV R3, #8, @RW3+d8	MOV R3, #8, @RW3+d8	MOVEA A, RW3, @RW3+d8	MOVEA A, @RW3+d8
+4	JMPP @RL2, @RW4+d8	JMPP @ @RL2, @RW4+d8	CALLP @RL2, @RW4+d8	CALLP @ @RL2, @RW4+d8	INCL RL2, @RW4+d8	INCL RL2, @RW4+d8	DECL RL2, @RW4+d8	DECL RL2, @RW4+d8	MOVL A, RL2, @RW4+d8	MOVL A, @RW4+d8	MOVL RL2, A, @RW4+d8	MOVL RL2, A, @RW4+d8	MOV R4, #8, @RW4+d8	MOV R4, #8, @RW4+d8	MOVEA A, RW4, @RW4+d8	MOVEA A, @RW4+d8
+5	JMPP @RL2, @RW5+d8	JMPP @ @RL2, @RW5+d8	CALLP @RL2, @RW5+d8	CALLP @ @RL2, @RW5+d8	INCL RL2, @RW5+d8	INCL RL2, @RW5+d8	DECL RL2, @RW5+d8	DECL RL2, @RW5+d8	MOVL A, RL2, @RW5+d8	MOVL A, @RW5+d8	MOVL RL2, A, @RW5+d8	MOVL RL2, A, @RW5+d8	MOV R5, #8, @RW5+d8	MOV R5, #8, @RW5+d8	MOVEA A, RW5, @RW5+d8	MOVEA A, @RW5+d8
+6	JMPP @RL3, @RW6+d8	JMPP @ @RL3, @RW6+d8	CALLP @RL3, @RW6+d8	CALLP @ @RL3, @RW6+d8	INCL RL3, @RW6+d8	INCL RL3, @RW6+d8	DECL RL3, @RW6+d8	DECL RL3, @RW6+d8	MOVL A, RL3, @RW6+d8	MOVL A, @RW6+d8	MOVL RL3, A, @RW6+d8	MOVL RL3, A, @RW6+d8	MOV R6, #8, @RW6+d8	MOV R6, #8, @RW6+d8	MOVEA A, RW6, @RW6+d8	MOVEA A, @RW6+d8
+7	JMPP @RL3, @RW7+d8	JMPP @ @RL3, @RW7+d8	CALLP @RL3, @RW7+d8	CALLP @ @RL3, @RW7+d8	INCL RL3, @RW7+d8	INCL RL3, @RW7+d8	DECL RL3, @RW7+d8	DECL RL3, @RW7+d8	MOVL A, RL3, @RW7+d8	MOVL A, @RW7+d8	MOVL RL3, A, @RW7+d8	MOVL RL3, A, @RW7+d8	MOV R7, #8, @RW7+d8	MOV R7, #8, @RW7+d8	MOVEA A, RW7, @RW7+d8	MOVEA A, @RW7+d8
+8	JMPP @RW0, @RW0+d16	JMPP @ @RW0, @RW0+d16	CALLP @RW0, @RW0+d16	CALLP @ @RW0, @RW0+d16	INCL @RW0, @RW0+d16	INCL @RW0, @RW0+d16	DECL @RW0, @RW0+d16	DECL @RW0, @RW0+d16	MOVL A, @RW0, @RW0+d16	MOVL A, @RW0+d16	MOVL @RW0, A, @RW0+d16	MOVL @RW0, A, @RW0+d16	MOV @RW0, #8, @RW0+d16	MOV @RW0, #8, @RW0+d16	MOVEA A, @RW0, @RW0+d16	MOVEA A, @RW0+d16
+9	JMPP @RW1, @RW1+d16	JMPP @ @RW1, @RW1+d16	CALLP @RW1, @RW1+d16	CALLP @ @RW1, @RW1+d16	INCL @RW1, @RW1+d16	INCL @RW1, @RW1+d16	DECL @RW1, @RW1+d16	DECL @RW1, @RW1+d16	MOVL A, @RW1, @RW1+d16	MOVL A, @RW1+d16	MOVL @RW1, A, @RW1+d16	MOVL @RW1, A, @RW1+d16	MOV @RW1, #8, @RW1+d16	MOV @RW1, #8, @RW1+d16	MOVEA A, @RW1, @RW1+d16	MOVEA A, @RW1+d16
+A	JMPP @RW2, @RW2+d16	JMPP @ @RW2, @RW2+d16	CALLP @RW2, @RW2+d16	CALLP @ @RW2, @RW2+d16	INCL @RW2, @RW2+d16	INCL @RW2, @RW2+d16	DECL @RW2, @RW2+d16	DECL @RW2, @RW2+d16	MOVL A, @RW2, @RW2+d16	MOVL A, @RW2+d16	MOVL @RW2, A, @RW2+d16	MOVL @RW2, A, @RW2+d16	MOV @RW2, #8, @RW2+d16	MOV @RW2, #8, @RW2+d16	MOVEA A, @RW2, @RW2+d16	MOVEA A, @RW2+d16
+B	JMPP @RW3, @RW3+d16	JMPP @ @RW3, @RW3+d16	CALLP @RW3, @RW3+d16	CALLP @ @RW3, @RW3+d16	INCL @RW3, @RW3+d16	INCL @RW3, @RW3+d16	DECL @RW3, @RW3+d16	DECL @RW3, @RW3+d16	MOVL A, @RW3, @RW3+d16	MOVL A, @RW3+d16	MOVL @RW3, A, @RW3+d16	MOVL @RW3, A, @RW3+d16	MOV @RW3, #8, @RW3+d16	MOV @RW3, #8, @RW3+d16	MOVEA A, @RW3, @RW3+d16	MOVEA A, @RW3+d16
+C	JMPP @RW0+, @RW0+RW7	JMPP @ @RW0+, @RW0+RW7	CALLP @RW0+, @RW0+RW7	CALLP @ @RW0+, @RW0+RW7	INCL @RW0+, @RW0+RW7	INCL @RW0+, @RW0+RW7	DECL @RW0+, @RW0+RW7	DECL @RW0+, @RW0+RW7	MOVL A, @RW0+, @RW0+RW7	MOVL A, @RW0+RW7	MOVL @RW0+, A, @RW0+RW7	MOVL @RW0+, A, @RW0+RW7	MOV @RW0+, #8, @RW0+RW7	MOV @RW0+, #8, @RW0+RW7	MOVEA A, @RW0+, @RW0+RW7	MOVEA A, @RW0+RW7
+D	JMPP @RW1+, @RW1+RW7	JMPP @ @RW1+, @RW1+RW7	CALLP @RW1+, @RW1+RW7	CALLP @ @RW1+, @RW1+RW7	INCL @RW1+, @RW1+RW7	INCL @RW1+, @RW1+RW7	DECL @RW1+, @RW1+RW7	DECL @RW1+, @RW1+RW7	MOVL A, @RW1+, @RW1+RW7	MOVL A, @RW1+RW7	MOVL @RW1+, A, @RW1+RW7	MOVL @RW1+, A, @RW1+RW7	MOV @RW1+, #8, @RW1+RW7	MOV @RW1+, #8, @RW1+RW7	MOVEA A, @RW1+, @RW1+RW7	MOVEA A, @RW1+RW7
+E	JMPP @RW2+, @PC+d16	JMPP @ @RW2+, @PC+d16	CALLP @RW2+, @PC+d16	CALLP @ @RW2+, @PC+d16	INCL @RW2+, @PC+d16	INCL @RW2+, @PC+d16	DECL @RW2+, @PC+d16	DECL @RW2+, @PC+d16	MOVL A, @RW2+, @PC+d16	MOVL A, @PC+d16	MOVL @RW2+, A, @PC+d16	MOVL @RW2+, A, @PC+d16	MOV @RW2+, #8, @PC+d16	MOV @RW2+, #8, @PC+d16	MOVEA A, @RW2+, @PC+d16	MOVEA A, @PC+d16
+F	JMPP @RW3+, @addr16	JMPP @ @RW3+, @addr16	CALLP @RW3+, @addr16	CALLP @ @RW3+, @addr16	INCL @RW3+, @addr16	INCL @RW3+, @addr16	DECL @RW3+, @addr16	DECL @RW3+, @addr16	MOVL A, @RW3+, @addr16	MOVL A, @addr16	MOVL @RW3+, A, @addr16	MOVL @RW3+, A, @addr16	MOV @RW3+, #8, @addr16	MOV @RW3+, #8, @addr16	MOVEA A, @RW3+, @addr16	MOVEA A, @addr16

Table B.9-8 ea Instruction 3 (First Byte = 72_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ROL R0, @RW0+d8	ROL R0, @RW0+d8	ROR R0, @RW0+d8	ROR R0, @RW0+d8	INC R0, @RW0+d8	INC R0, @RW0+d8	DEC R0, @RW0+d8	DEC R0, @RW0+d8	MOV A, R0, @RW0+d8	MOV A, R0, @RW0+d8	MOV R0, A, @RW0+d8	MOV R0, A, @RW0+d8	MOVX A, R0, @RW0+d8	MOVX A, R0, @RW0+d8	XCH A, R0, @RW0+d8	XCH A, R0, @RW0+d8
+1	ROL R1, @RW1+d8	ROL R1, @RW1+d8	ROR R1, @RW1+d8	ROR R1, @RW1+d8	INC R1, @RW1+d8	INC R1, @RW1+d8	DEC R1, @RW1+d8	DEC R1, @RW1+d8	MOV A, R1, @RW1+d8	MOV A, R1, @RW1+d8	MOV R1, A, @RW1+d8	MOV R1, A, @RW1+d8	MOVX A, R1, @RW1+d8	MOVX A, R1, @RW1+d8	XCH A, R1, @RW1+d8	XCH A, R1, @RW1+d8
+2	ROL R2, @RW2+d8	ROL R2, @RW2+d8	ROR R2, @RW2+d8	ROR R2, @RW2+d8	INC R2, @RW2+d8	INC R2, @RW2+d8	DEC R2, @RW2+d8	DEC R2, @RW2+d8	MOV A, R2, @RW2+d8	MOV A, R2, @RW2+d8	MOV R2, A, @RW2+d8	MOV R2, A, @RW2+d8	MOVX A, R2, @RW2+d8	MOVX A, R2, @RW2+d8	XCH A, R2, @RW2+d8	XCH A, R2, @RW2+d8
+3	ROL R3, @RW3+d8	ROL R3, @RW3+d8	ROR R3, @RW3+d8	ROR R3, @RW3+d8	INC R3, @RW3+d8	INC R3, @RW3+d8	DEC R3, @RW3+d8	DEC R3, @RW3+d8	MOV A, R3, @RW3+d8	MOV A, R3, @RW3+d8	MOV R3, A, @RW3+d8	MOV R3, A, @RW3+d8	MOVX A, R3, @RW3+d8	MOVX A, R3, @RW3+d8	XCH A, R3, @RW3+d8	XCH A, R3, @RW3+d8
+4	ROL R4, @RW4+d8	ROL R4, @RW4+d8	ROR R4, @RW4+d8	ROR R4, @RW4+d8	INC R4, @RW4+d8	INC R4, @RW4+d8	DEC R4, @RW4+d8	DEC R4, @RW4+d8	MOV A, R4, @RW4+d8	MOV A, R4, @RW4+d8	MOV R4, A, @RW4+d8	MOV R4, A, @RW4+d8	MOVX A, R4, @RW4+d8	MOVX A, R4, @RW4+d8	XCH A, R4, @RW4+d8	XCH A, R4, @RW4+d8
+5	ROL R5, @RW5+d8	ROL R5, @RW5+d8	ROR R5, @RW5+d8	ROR R5, @RW5+d8	INC R5, @RW5+d8	INC R5, @RW5+d8	DEC R5, @RW5+d8	DEC R5, @RW5+d8	MOV A, R5, @RW5+d8	MOV A, R5, @RW5+d8	MOV R5, A, @RW5+d8	MOV R5, A, @RW5+d8	MOVX A, R5, @RW5+d8	MOVX A, R5, @RW5+d8	XCH A, R5, @RW5+d8	XCH A, R5, @RW5+d8
+6	ROL R6, @RW6+d8	ROL R6, @RW6+d8	ROR R6, @RW6+d8	ROR R6, @RW6+d8	INC R6, @RW6+d8	INC R6, @RW6+d8	DEC R6, @RW6+d8	DEC R6, @RW6+d8	MOV A, R6, @RW6+d8	MOV A, R6, @RW6+d8	MOV R6, A, @RW6+d8	MOV R6, A, @RW6+d8	MOVX A, R6, @RW6+d8	MOVX A, R6, @RW6+d8	XCH A, R6, @RW6+d8	XCH A, R6, @RW6+d8
+7	ROL R7, @RW7+d8	ROL R7, @RW7+d8	ROR R7, @RW7+d8	ROR R7, @RW7+d8	INC R7, @RW7+d8	INC R7, @RW7+d8	DEC R7, @RW7+d8	DEC R7, @RW7+d8	MOV A, R7, @RW7+d8	MOV A, R7, @RW7+d8	MOV R7, A, @RW7+d8	MOV R7, A, @RW7+d8	MOVX A, R7, @RW7+d8	MOVX A, R7, @RW7+d8	XCH A, R7, @RW7+d8	XCH A, R7, @RW7+d8
+8	ROL @RW0, @RW0+d16	ROL @RW0, @RW0+d16	ROR @RW0, @RW0+d16	ROR @RW0, @RW0+d16	INC @RW0, @RW0+d16	INC @RW0, @RW0+d16	DEC @RW0, @RW0+d16	DEC @RW0, @RW0+d16	MOV @RW0, A, @RW0+d16	MOV @RW0, A, @RW0+d16	MOV @RW0, A, @RW0+d16	MOV @RW0, A, @RW0+d16	MOVX @RW0, A, @RW0+d16	MOVX @RW0, A, @RW0+d16	XCH @RW0, A, @RW0+d16	XCH @RW0, A, @RW0+d16
+9	ROL @RW1, @RW1+d16	ROL @RW1, @RW1+d16	ROR @RW1, @RW1+d16	ROR @RW1, @RW1+d16	INC @RW1, @RW1+d16	INC @RW1, @RW1+d16	DEC @RW1, @RW1+d16	DEC @RW1, @RW1+d16	MOV @RW1, A, @RW1+d16	MOV @RW1, A, @RW1+d16	MOV @RW1, A, @RW1+d16	MOV @RW1, A, @RW1+d16	MOVX @RW1, A, @RW1+d16	MOVX @RW1, A, @RW1+d16	XCH @RW1, A, @RW1+d16	XCH @RW1, A, @RW1+d16
+A	ROL @RW2, @RW2+d16	ROL @RW2, @RW2+d16	ROR @RW2, @RW2+d16	ROR @RW2, @RW2+d16	INC @RW2, @RW2+d16	INC @RW2, @RW2+d16	DEC @RW2, @RW2+d16	DEC @RW2, @RW2+d16	MOV @RW2, A, @RW2+d16	MOV @RW2, A, @RW2+d16	MOV @RW2, A, @RW2+d16	MOV @RW2, A, @RW2+d16	MOVX @RW2, A, @RW2+d16	MOVX @RW2, A, @RW2+d16	XCH @RW2, A, @RW2+d16	XCH @RW2, A, @RW2+d16
+B	ROL @RW3, @RW3+d16	ROL @RW3, @RW3+d16	ROR @RW3, @RW3+d16	ROR @RW3, @RW3+d16	INC @RW3, @RW3+d16	INC @RW3, @RW3+d16	DEC @RW3, @RW3+d16	DEC @RW3, @RW3+d16	MOV @RW3, A, @RW3+d16	MOV @RW3, A, @RW3+d16	MOV @RW3, A, @RW3+d16	MOV @RW3, A, @RW3+d16	MOVX @RW3, A, @RW3+d16	MOVX @RW3, A, @RW3+d16	XCH @RW3, A, @RW3+d16	XCH @RW3, A, @RW3+d16
+C	ROL @RW0+, @RW0+RW7	ROL @RW0+, @RW0+RW7	ROR @RW0+, @RW0+RW7	ROR @RW0+, @RW0+RW7	INC @RW0+, @RW0+RW7	INC @RW0+, @RW0+RW7	DEC @RW0+, @RW0+RW7	DEC @RW0+, @RW0+RW7	MOV @RW0+, A, @RW0+RW7	MOV @RW0+, A, @RW0+RW7	MOV @RW0+, A, @RW0+RW7	MOV @RW0+, A, @RW0+RW7	MOVX @RW0+, A, @RW0+RW7	MOVX @RW0+, A, @RW0+RW7	XCH @RW0+, A, @RW0+RW7	XCH @RW0+, A, @RW0+RW7
+D	ROL @RW1+, @RW1+RW7	ROL @RW1+, @RW1+RW7	ROR @RW1+, @RW1+RW7	ROR @RW1+, @RW1+RW7	INC @RW1+, @RW1+RW7	INC @RW1+, @RW1+RW7	DEC @RW1+, @RW1+RW7	DEC @RW1+, @RW1+RW7	MOV @RW1+, A, @RW1+RW7	MOV @RW1+, A, @RW1+RW7	MOV @RW1+, A, @RW1+RW7	MOV @RW1+, A, @RW1+RW7	MOVX @RW1+, A, @RW1+RW7	MOVX @RW1+, A, @RW1+RW7	XCH @RW1+, A, @RW1+RW7	XCH @RW1+, A, @RW1+RW7
+E	ROL @RW2+, @PC+d16	ROL @RW2+, @PC+d16	ROR @RW2+, @PC+d16	ROR @RW2+, @PC+d16	INC @RW2+, @PC+d16	INC @RW2+, @PC+d16	DEC @RW2+, @PC+d16	DEC @RW2+, @PC+d16	MOV @RW2+, A, @PC+d16	MOV @RW2+, A, @PC+d16	MOV @RW2+, A, @PC+d16	MOV @RW2+, A, @PC+d16	MOVX @RW2+, A, @PC+d16	MOVX @RW2+, A, @PC+d16	XCH @RW2+, A, @PC+d16	XCH @RW2+, A, @PC+d16
+F	ROL @RW3+, addr16	ROL @RW3+, addr16	ROR @RW3+, addr16	ROR @RW3+, addr16	INC @RW3+, addr16	INC @RW3+, addr16	DEC @RW3+, addr16	DEC @RW3+, addr16	MOV @RW3+, A, addr16	MOV @RW3+, A, addr16	MOV @RW3+, A, addr16	MOV @RW3+, A, addr16	MOVX @RW3+, A, addr16	MOVX @RW3+, A, addr16	XCH @RW3+, A, addr16	XCH @RW3+, A, addr16

Table B.9-9 ea Instruction 4 (First Byte = 73_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMP @RW0; @@RW0+d8	JMP @RW0; @@RW0+d8	CALL @RW0; @@RW0+d8	CALL @RW0; @@RW0+d8	INW RW0; @RW0+d8	INW RW0; @RW0+d8	DECW RW0; @RW0+d8	DECW RW0; @RW0+d8	MOVW A, RW0; @RW0+d8	MOVW A, RW0; @RW0+d8	MOVW RW0, A; @RW0+d8,A	MOVW RW0, A; @RW0+d8,A	MOVW RW0, #16; @RW0+d8, #16	MOVW RW0, #16; @RW0+d8, #16	XCHW A, RW0; @RW0+d8	XCHW A, RW0; @RW0+d8
+1	JMP @RW1; @@RW1+d8	JMP @RW1; @@RW1+d8	CALL @RW1; @@RW1+d8	CALL @RW1; @@RW1+d8	INW RW1; @RW1+d8	INW RW1; @RW1+d8	DECW RW1; @RW1+d8	DECW RW1; @RW1+d8	MOVW A, RW1; @RW1+d8	MOVW A, RW1; @RW1+d8	MOVW RW1, A; @RW1+d8,A	MOVW RW1, A; @RW1+d8,A	MOVW RW1, #16; @RW1+d8, #16	MOVW RW1, #16; @RW1+d8, #16	XCHW A, RW1; @RW1+d8	XCHW A, RW1; @RW1+d8
+2	JMP @RW2; @@RW2+d8	JMP @RW2; @@RW2+d8	CALL @RW2; @@RW2+d8	CALL @RW2; @@RW2+d8	INW RW2; @RW2+d8	INW RW2; @RW2+d8	DECW RW2; @RW2+d8	DECW RW2; @RW2+d8	MOVW A, RW2; @RW2+d8	MOVW A, RW2; @RW2+d8	MOVW RW2, A; @RW2+d8,A	MOVW RW2, A; @RW2+d8,A	MOVW RW2, #16; @RW2+d8, #16	MOVW RW2, #16; @RW2+d8, #16	XCHW A, RW2; @RW2+d8	XCHW A, RW2; @RW2+d8
+3	JMP @RW3; @@RW3+d8	JMP @RW3; @@RW3+d8	CALL @RW3; @@RW3+d8	CALL @RW3; @@RW3+d8	INW RW3; @RW3+d8	INW RW3; @RW3+d8	DECW RW3; @RW3+d8	DECW RW3; @RW3+d8	MOVW A, RW3; @RW3+d8	MOVW A, RW3; @RW3+d8	MOVW RW3, A; @RW3+d8,A	MOVW RW3, A; @RW3+d8,A	MOVW RW3, #16; @RW3+d8, #16	MOVW RW3, #16; @RW3+d8, #16	XCHW A, RW3; @RW3+d8	XCHW A, RW3; @RW3+d8
+4	JMP @RW4; @@RW4+d8	JMP @RW4; @@RW4+d8	CALL @RW4; @@RW4+d8	CALL @RW4; @@RW4+d8	INW RW4; @RW4+d8	INW RW4; @RW4+d8	DECW RW4; @RW4+d8	DECW RW4; @RW4+d8	MOVW A, RW4; @RW4+d8	MOVW A, RW4; @RW4+d8	MOVW RW4, A; @RW4+d8,A	MOVW RW4, A; @RW4+d8,A	MOVW RW4, #16; @RW4+d8, #16	MOVW RW4, #16; @RW4+d8, #16	XCHW A, RW4; @RW4+d8	XCHW A, RW4; @RW4+d8
+5	JMP @RW5; @@RW5+d8	JMP @RW5; @@RW5+d8	CALL @RW5; @@RW5+d8	CALL @RW5; @@RW5+d8	INW RW5; @RW5+d8	INW RW5; @RW5+d8	DECW RW5; @RW5+d8	DECW RW5; @RW5+d8	MOVW A, RW5; @RW5+d8	MOVW A, RW5; @RW5+d8	MOVW RW5, A; @RW5+d8,A	MOVW RW5, A; @RW5+d8,A	MOVW RW5, #16; @RW5+d8, #16	MOVW RW5, #16; @RW5+d8, #16	XCHW A, RW5; @RW5+d8	XCHW A, RW5; @RW5+d8
+6	JMP @RW6; @@RW6+d8	JMP @RW6; @@RW6+d8	CALL @RW6; @@RW6+d8	CALL @RW6; @@RW6+d8	INW RW6; @RW6+d8	INW RW6; @RW6+d8	DECW RW6; @RW6+d8	DECW RW6; @RW6+d8	MOVW A, RW6; @RW6+d8	MOVW A, RW6; @RW6+d8	MOVW RW6, A; @RW6+d8,A	MOVW RW6, A; @RW6+d8,A	MOVW RW6, #16; @RW6+d8, #16	MOVW RW6, #16; @RW6+d8, #16	XCHW A, RW6; @RW6+d8	XCHW A, RW6; @RW6+d8
+7	JMP @RW7; @@RW7+d8	JMP @RW7; @@RW7+d8	CALL @RW7; @@RW7+d8	CALL @RW7; @@RW7+d8	INW RW7; @RW7+d8	INW RW7; @RW7+d8	DECW RW7; @RW7+d8	DECW RW7; @RW7+d8	MOVW A, RW7; @RW7+d8	MOVW A, RW7; @RW7+d8	MOVW RW7, A; @RW7+d8,A	MOVW RW7, A; @RW7+d8,A	MOVW RW7, #16; @RW7+d8, #16	MOVW RW7, #16; @RW7+d8, #16	XCHW A, RW7; @RW7+d8	XCHW A, RW7; @RW7+d8
+8	JMP @@RW0; @RW0+d16	JMP @@RW0; @RW0+d16	CALL @@RW0; @RW0+d16	CALL @@RW0; @RW0+d16	INW @RW0; @RW0+d16	INW @RW0; @RW0+d16	DECW @RW0; @RW0+d16	DECW @RW0; @RW0+d16	MOVW A, @RW0; @RW0+d16	MOVW A, @RW0; @RW0+d16	MOVW @RW0, A; @RW0+d16,A	MOVW @RW0, A; @RW0+d16,A	MOVW @RW0, #16; @RW0+d16, #16	MOVW @RW0, #16; @RW0+d16, #16	XCHW A, @RW0; @RW0+d16	XCHW A, @RW0; @RW0+d16
+9	JMP @@RW1; @RW1+d16	JMP @@RW1; @RW1+d16	CALL @@RW1; @RW1+d16	CALL @@RW1; @RW1+d16	INW @RW1; @RW1+d16	INW @RW1; @RW1+d16	DECW @RW1; @RW1+d16	DECW @RW1; @RW1+d16	MOVW A, @RW1; @RW1+d16	MOVW A, @RW1; @RW1+d16	MOVW @RW1, A; @RW1+d16,A	MOVW @RW1, A; @RW1+d16,A	MOVW @RW1, #16; @RW1+d16, #16	MOVW @RW1, #16; @RW1+d16, #16	XCHW A, @RW1; @RW1+d16	XCHW A, @RW1; @RW1+d16
+A	JMP @@RW2; @RW2+d16	JMP @@RW2; @RW2+d16	CALL @@RW2; @RW2+d16	CALL @@RW2; @RW2+d16	INW @RW2; @RW2+d16	INW @RW2; @RW2+d16	DECW @RW2; @RW2+d16	DECW @RW2; @RW2+d16	MOVW A, @RW2; @RW2+d16	MOVW A, @RW2; @RW2+d16	MOVW @RW2, A; @RW2+d16,A	MOVW @RW2, A; @RW2+d16,A	MOVW @RW2, #16; @RW2+d16, #16	MOVW @RW2, #16; @RW2+d16, #16	XCHW A, @RW2; @RW2+d16	XCHW A, @RW2; @RW2+d16
+B	JMP @@RW3; @RW3+d16	JMP @@RW3; @RW3+d16	CALL @@RW3; @RW3+d16	CALL @@RW3; @RW3+d16	INW @RW3; @RW3+d16	INW @RW3; @RW3+d16	DECW @RW3; @RW3+d16	DECW @RW3; @RW3+d16	MOVW A, @RW3; @RW3+d16	MOVW A, @RW3; @RW3+d16	MOVW @RW3, A; @RW3+d16,A	MOVW @RW3, A; @RW3+d16,A	MOVW @RW3, #16; @RW3+d16, #16	MOVW @RW3, #16; @RW3+d16, #16	XCHW A, @RW3; @RW3+d16	XCHW A, @RW3; @RW3+d16
+C	JMP @@RW0+; @RW0+RW7	JMP @@RW0+; @RW0+RW7	CALL @@RW0+; @RW0+RW7	CALL @@RW0+; @RW0+RW7	INW @RW0+; @RW0+RW7	INW @RW0+; @RW0+RW7	DECW @RW0+; @RW0+RW7	DECW @RW0+; @RW0+RW7	MOVW A, @RW0+; @RW0+RW7	MOVW A, @RW0+; @RW0+RW7	MOVW @RW0+, A; @RW0+RW7,A	MOVW @RW0+, A; @RW0+RW7,A	MOVW @RW0+, #16; @RW0+RW7, #16	MOVW @RW0+, #16; @RW0+RW7, #16	XCHW A, @RW0+; @RW0+RW7	XCHW A, @RW0+; @RW0+RW7
+D	JMP @@RW1+; @RW1+RW7	JMP @@RW1+; @RW1+RW7	CALL @@RW1+; @RW1+RW7	CALL @@RW1+; @RW1+RW7	INW @RW1+; @RW1+RW7	INW @RW1+; @RW1+RW7	DECW @RW1+; @RW1+RW7	DECW @RW1+; @RW1+RW7	MOVW A, @RW1+; @RW1+RW7	MOVW A, @RW1+; @RW1+RW7	MOVW @RW1+, A; @RW1+RW7,A	MOVW @RW1+, A; @RW1+RW7,A	MOVW @RW1+, #16; @RW1+RW7, #16	MOVW @RW1+, #16; @RW1+RW7, #16	XCHW A, @RW1+; @RW1+RW7	XCHW A, @RW1+; @RW1+RW7
+E	JMP @@RW2+; @PC+d16	JMP @@RW2+; @PC+d16	CALL @@RW2+; @PC+d16	CALL @@RW2+; @PC+d16	INW @RW2+; @PC+d16	INW @RW2+; @PC+d16	DECW @RW2+; @PC+d16	DECW @RW2+; @PC+d16	MOVW A, @RW2+; @PC+d16	MOVW A, @RW2+; @PC+d16	MOVW @RW2+, A; @PC+d16, A	MOVW @RW2+, A; @PC+d16, A	MOVW @RW2+, #16; @PC+d16, #16	MOVW @RW2+, #16; @PC+d16, #16	XCHW A, @RW2+; @PC+d16	XCHW A, @RW2+; @PC+d16
+F	JMP @@RW3+; @addr16	JMP @@RW3+; @addr16	CALL @@RW3+; @addr16	CALL @@RW3+; @addr16	INW @RW3+; @addr16	INW @RW3+; @addr16	DECW @RW3+; @addr16	DECW @RW3+; @addr16	MOVW A, @RW3+; @addr16	MOVW A, @RW3+; @addr16	MOVW @RW3+, A; @addr16, A	MOVW @RW3+, A; @addr16, A	MOVW @RW3+, #16; @addr16, #16	MOVW @RW3+, #16; @addr16, #16	XCHW A, @RW3+; @addr16	XCHW A, @RW3+; @addr16

Table B.9-10 ea Instruction 5 (First Byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0', @RW0+d8	SUB A, R0', @RW0+d8	SUB A, R0', @RW0+d8	SUB A, R0', @RW0+d8	ADDC A, R0', @RW0+d8	ADDC A, R0', @RW0+d8	CMP A, R0', @RW0+d8	CMP A, R0', @RW0+d8	AND A, R0', @RW0+d8	AND A, R0', @RW0+d8	OR A, R0', @RW0+d8	OR A, R0', @RW0+d8	XOR A, R0', @RW0+d8	XOR A, R0', @RW0+d8	DBNZ R0, rel, +d8, rel	DBNZ @RW0
+1	ADD A, R1', @RW1+d8	SUB A, R1', @RW1+d8	SUB A, R1', @RW1+d8	SUB A, R1', @RW1+d8	ADDC A, R1', @RW1+d8	ADDC A, R1', @RW1+d8	CMP A, R1', @RW1+d8	CMP A, R1', @RW1+d8	AND A, R1', @RW1+d8	AND A, R1', @RW1+d8	OR A, R1', @RW1+d8	OR A, R1', @RW1+d8	XOR A, R1', @RW1+d8	XOR A, R1', @RW1+d8	DBNZ R1, rel, +d8, rel	DBNZ @RW1
+2	ADD A, R2', @RW2+d8	SUB A, R2', @RW2+d8	SUB A, R2', @RW2+d8	SUB A, R2', @RW2+d8	ADDC A, R2', @RW2+d8	ADDC A, R2', @RW2+d8	CMP A, R2', @RW2+d8	CMP A, R2', @RW2+d8	AND A, R2', @RW2+d8	AND A, R2', @RW2+d8	OR A, R2', @RW2+d8	OR A, R2', @RW2+d8	XOR A, R2', @RW2+d8	XOR A, R2', @RW2+d8	DBNZ R2, rel, +d8, rel	DBNZ @RW2
+3	ADD A, R3', @RW3+d8	SUB A, R3', @RW3+d8	SUB A, R3', @RW3+d8	SUB A, R3', @RW3+d8	ADDC A, R3', @RW3+d8	ADDC A, R3', @RW3+d8	CMP A, R3', @RW3+d8	CMP A, R3', @RW3+d8	AND A, R3', @RW3+d8	AND A, R3', @RW3+d8	OR A, R3', @RW3+d8	OR A, R3', @RW3+d8	XOR A, R3', @RW3+d8	XOR A, R3', @RW3+d8	DBNZ R3, rel, +d8, rel	DBNZ @RW3
+4	ADD A, R4', @RW4+d8	SUB A, R4', @RW4+d8	SUB A, R4', @RW4+d8	SUB A, R4', @RW4+d8	ADDC A, R4', @RW4+d8	ADDC A, R4', @RW4+d8	CMP A, R4', @RW4+d8	CMP A, R4', @RW4+d8	AND A, R4', @RW4+d8	AND A, R4', @RW4+d8	OR A, R4', @RW4+d8	OR A, R4', @RW4+d8	XOR A, R4', @RW4+d8	XOR A, R4', @RW4+d8	DBNZ R4, rel, +d8, rel	DBNZ @RW4
+5	ADD A, R5', @RW5+d8	SUB A, R5', @RW5+d8	SUB A, R5', @RW5+d8	SUB A, R5', @RW5+d8	ADDC A, R5', @RW5+d8	ADDC A, R5', @RW5+d8	CMP A, R5', @RW5+d8	CMP A, R5', @RW5+d8	AND A, R5', @RW5+d8	AND A, R5', @RW5+d8	OR A, R5', @RW5+d8	OR A, R5', @RW5+d8	XOR A, R5', @RW5+d8	XOR A, R5', @RW5+d8	DBNZ R5, rel, +d8, rel	DBNZ @RW5
+6	ADD A, R6', @RW6+d8	SUB A, R6', @RW6+d8	SUB A, R6', @RW6+d8	SUB A, R6', @RW6+d8	ADDC A, R6', @RW6+d8	ADDC A, R6', @RW6+d8	CMP A, R6', @RW6+d8	CMP A, R6', @RW6+d8	AND A, R6', @RW6+d8	AND A, R6', @RW6+d8	OR A, R6', @RW6+d8	OR A, R6', @RW6+d8	XOR A, R6', @RW6+d8	XOR A, R6', @RW6+d8	DBNZ R6, rel, +d8, rel	DBNZ @RW6
+7	ADD A, R7', @RW7+d8	SUB A, R7', @RW7+d8	SUB A, R7', @RW7+d8	SUB A, R7', @RW7+d8	ADDC A, R7', @RW7+d8	ADDC A, R7', @RW7+d8	CMP A, R7', @RW7+d8	CMP A, R7', @RW7+d8	AND A, R7', @RW7+d8	AND A, R7', @RW7+d8	OR A, R7', @RW7+d8	OR A, R7', @RW7+d8	XOR A, R7', @RW7+d8	XOR A, R7', @RW7+d8	DBNZ R7, rel, +d8, rel	DBNZ @RW7
+8	ADD A, @RW0', @RW0+d16	SUB A, @RW0', @RW0+d16	SUB A, @RW0', @RW0+d16	SUB A, @RW0', @RW0+d16	ADDC A, @RW0', @RW0+d16	ADDC A, @RW0', @RW0+d16	CMP A, @RW0', @RW0+d16	CMP A, @RW0', @RW0+d16	AND A, @RW0', @RW0+d16	AND A, @RW0', @RW0+d16	OR A, @RW0', @RW0+d16	OR A, @RW0', @RW0+d16	XOR A, @RW0', @RW0+d16	XOR A, @RW0', @RW0+d16	DBNZ @RW0, rel, +d16, rel	DBNZ @RW0
+9	ADD A, @RW1', @RW1+d16	SUB A, @RW1', @RW1+d16	SUB A, @RW1', @RW1+d16	SUB A, @RW1', @RW1+d16	ADDC A, @RW1', @RW1+d16	ADDC A, @RW1', @RW1+d16	CMP A, @RW1', @RW1+d16	CMP A, @RW1', @RW1+d16	AND A, @RW1', @RW1+d16	AND A, @RW1', @RW1+d16	OR A, @RW1', @RW1+d16	OR A, @RW1', @RW1+d16	XOR A, @RW1', @RW1+d16	XOR A, @RW1', @RW1+d16	DBNZ @RW1, rel, +d16, rel	DBNZ @RW1
+A	ADD A, @RW2', @RW2+d16	SUB A, @RW2', @RW2+d16	SUB A, @RW2', @RW2+d16	SUB A, @RW2', @RW2+d16	ADDC A, @RW2', @RW2+d16	ADDC A, @RW2', @RW2+d16	CMP A, @RW2', @RW2+d16	CMP A, @RW2', @RW2+d16	AND A, @RW2', @RW2+d16	AND A, @RW2', @RW2+d16	OR A, @RW2', @RW2+d16	OR A, @RW2', @RW2+d16	XOR A, @RW2', @RW2+d16	XOR A, @RW2', @RW2+d16	DBNZ @RW2, rel, +d16, rel	DBNZ @RW2
+B	ADD A, @RW3', @RW3+d16	SUB A, @RW3', @RW3+d16	SUB A, @RW3', @RW3+d16	SUB A, @RW3', @RW3+d16	ADDC A, @RW3', @RW3+d16	ADDC A, @RW3', @RW3+d16	CMP A, @RW3', @RW3+d16	CMP A, @RW3', @RW3+d16	AND A, @RW3', @RW3+d16	AND A, @RW3', @RW3+d16	OR A, @RW3', @RW3+d16	OR A, @RW3', @RW3+d16	XOR A, @RW3', @RW3+d16	XOR A, @RW3', @RW3+d16	DBNZ @RW3, rel, +d16, rel	DBNZ @RW3
+C	ADD A, @RW0+', @RW0+RW7	SUB A, @RW0+', @RW0+RW7	SUB A, @RW0+', @RW0+RW7	SUB A, @RW0+', @RW0+RW7	ADDC A, @RW0+', @RW0+RW7	ADDC A, @RW0+', @RW0+RW7	CMP A, @RW0+', @RW0+RW7	CMP A, @RW0+', @RW0+RW7	AND A, @RW0+', @RW0+RW7	AND A, @RW0+', @RW0+RW7	OR A, @RW0+', @RW0+RW7	OR A, @RW0+', @RW0+RW7	XOR A, @RW0+', @RW0+RW7	XOR A, @RW0+', @RW0+RW7	DBNZ @RW0+, rel, +RW7, rel	DBNZ @RW0
+D	ADD A, @RW1+', @RW1+RW7	SUB A, @RW1+', @RW1+RW7	SUB A, @RW1+', @RW1+RW7	SUB A, @RW1+', @RW1+RW7	ADDC A, @RW1+', @RW1+RW7	ADDC A, @RW1+', @RW1+RW7	CMP A, @RW1+', @RW1+RW7	CMP A, @RW1+', @RW1+RW7	AND A, @RW1+', @RW1+RW7	AND A, @RW1+', @RW1+RW7	OR A, @RW1+', @RW1+RW7	OR A, @RW1+', @RW1+RW7	XOR A, @RW1+', @RW1+RW7	XOR A, @RW1+', @RW1+RW7	DBNZ @RW1+, rel, +RW7, rel	DBNZ @RW1
+E	ADD A, @RW2+', @PC+d16	SUB A, @RW2+', @PC+d16	SUB A, @RW2+', @PC+d16	SUB A, @RW2+', @PC+d16	ADDC A, @RW2+', @PC+d16	ADDC A, @RW2+', @PC+d16	CMP A, @RW2+', @PC+d16	CMP A, @RW2+', @PC+d16	AND A, @RW2+', @PC+d16	AND A, @RW2+', @PC+d16	OR A, @RW2+', @PC+d16	OR A, @RW2+', @PC+d16	XOR A, @RW2+', @PC+d16	XOR A, @RW2+', @PC+d16	DBNZ @RW2+, rel, +d16, rel	DBNZ @PC
+F	ADD A, @RW3+', A, addr16	SUB A, @RW3+', A, addr16	SUB A, @RW3+', A, addr16	SUB A, @RW3+', A, addr16	ADDC A, @RW3+', A, addr16	ADDC A, @RW3+', A, addr16	CMP A, @RW3+', A, addr16	CMP A, @RW3+', A, addr16	AND A, @RW3+', A, addr16	AND A, @RW3+', A, addr16	OR A, @RW3+', A, addr16	OR A, @RW3+', A, addr16	XOR A, @RW3+', A, addr16	XOR A, @RW3+', A, addr16	DBNZ @RW3+, rel, +addr16, rel	DBNZ @RW3

Table B.9-11 ea Instruction 6 (First Byte = 75_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUBC A, R0, @RW0+d8	SUBC A, R0, @RW0+d8	NEG R0, @RW0+d8	NEG R0, @RW0+d8	AND R0, A, @RW0+d8, A	AND R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	NOT R0, @RW0+d8	NOT R0, @RW0+d8
+1	ADD R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUBC A, R1, @RW1+d8	SUBC A, R1, @RW1+d8	NEG R1, @RW1+d8	NEG R1, @RW1+d8	AND R1, A, @RW1+d8, A	AND R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	NOT R1, @RW1+d8	NOT R1, @RW1+d8
+2	ADD R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUBC A, R2, @RW2+d8	SUBC A, R2, @RW2+d8	NEG R2, @RW2+d8	NEG R2, @RW2+d8	AND R2, A, @RW2+d8, A	AND R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	NOT R2, @RW2+d8	NOT R2, @RW2+d8
+3	ADD R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUBC A, R3, @RW3+d8	SUBC A, R3, @RW3+d8	NEG R3, @RW3+d8	NEG R3, @RW3+d8	AND R3, A, @RW3+d8, A	AND R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	NOT R3, @RW3+d8	NOT R3, @RW3+d8
+4	ADD R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUBC A, R4, @RW4+d8	SUBC A, R4, @RW4+d8	NEG R4, @RW4+d8	NEG R4, @RW4+d8	AND R4, A, @RW4+d8, A	AND R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	NOT R4, @RW4+d8	NOT R4, @RW4+d8
+5	ADD R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUBC A, R5, @RW5+d8	SUBC A, R5, @RW5+d8	NEG R5, @RW5+d8	NEG R5, @RW5+d8	AND R5, A, @RW5+d8, A	AND R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	NOT R5, @RW5+d8	NOT R5, @RW5+d8
+6	ADD R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUBC A, R6, @RW6+d8	SUBC A, R6, @RW6+d8	NEG R6, @RW6+d8	NEG R6, @RW6+d8	AND R6, A, @RW6+d8, A	AND R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	NOT R6, @RW6+d8	NOT R6, @RW6+d8
+7	ADD R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUBC A, R7, @RW7+d8	SUBC A, R7, @RW7+d8	NEG R7, @RW7+d8	NEG R7, @RW7+d8	AND R7, A, @RW7+d8, A	AND R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	NOT R7, @RW7+d8	NOT R7, @RW7+d8
+8	ADD @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUBC A, @RW0, @RW0+d16	SUBC A, @RW0, @RW0+d16	NEG @RW0, @RW0+d16	NEG @RW0, @RW0+d16	AND @RW0, A, @RW0+d16, A	AND @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	NOT @RW0, @RW0+d16	NOT @RW0, @RW0+d16
+9	ADD @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUBC A, @RW1, @RW1+d16	SUBC A, @RW1, @RW1+d16	NEG @RW1, @RW1+d16	NEG @RW1, @RW1+d16	AND @RW1, A, @RW1+d16, A	AND @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	NOT @RW1, @RW1+d16	NOT @RW1, @RW1+d16
+A	ADD @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUBC A, @RW2, @RW2+d16	SUBC A, @RW2, @RW2+d16	NEG @RW2, @RW2+d16	NEG @RW2, @RW2+d16	AND @RW2, A, @RW2+d16, A	AND @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	NOT @RW2, @RW2+d16	NOT @RW2, @RW2+d16
+B	ADD @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUBC A, @RW3, @RW3+d16	SUBC A, @RW3, @RW3+d16	NEG @RW3, @RW3+d16	NEG @RW3, @RW3+d16	AND @RW3, A, @RW3+d16, A	AND @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	NOT @RW3, @RW3+d16	NOT @RW3, @RW3+d16
+C	ADD @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7	SUBC A, @RW0+, @RW0+RW7	NEG @RW0+, @RW0+RW7	NEG @RW0+, @RW0+RW7	AND @RW0+, A, @RW0+RW7, A	AND @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	NOT @RW0+, @RW0+RW7	NOT @RW0+, @RW0+RW7
+D	ADD @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7	SUBC A, @RW1+, @RW1+RW7	NEG @RW1+, @RW1+RW7	NEG @RW1+, @RW1+RW7	AND @RW1+, A, @RW1+RW7, A	AND @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	NOT @RW1+, @RW1+RW7	NOT @RW1+, @RW1+RW7
+E	ADD @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUBC A, @RW2+, @PC+d16	SUBC A, @RW2+, @PC+d16	NEG @RW2+, @PC+d16	NEG @RW2+, @PC+d16	AND @RW2+, A, @PC+d16, A	AND @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	NOT @RW2+, @PC+d16	NOT @RW2+, @PC+d16
+F	ADD @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUBC A, @RW3+, addr16	SUBC A, @RW3+, addr16	NEG @RW3+, addr16	NEG @RW3+, addr16	AND @RW3+, A, addr16, A	AND @RW3+, A, addr16, A	OR @RW3+, A, addr16, A	OR @RW3+, A, addr16, A	XOR @RW3+, A, addr16, A	XOR @RW3+, A, addr16, A	NOT @RW3+, addr16	NOT @RW3+, 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, RW0 @RW0+d8	ADDW A, RW0 @RW0+d8	SUBW A, RW0 @RW0+d8	SUBW A, RW0 @RW0+d8	ADDCW A, RW0 @RW0+d8	ADDCW A, RW0 @RW0+d8	CMPW A, RW0 @RW0+d8	CMPW A, RW0 @RW0+d8	ANDW A, RW0 @RW0+d8	ANDW A, RW0 @RW0+d8	ORW A, RW0 @RW0+d8	ORW A, RW0 @RW0+d8	XORW A, RW0 @RW0+d8	XORW A, RW0 @RW0+d8	DWBNZ @RW0+d8, rel	DWBNZ @RW0+d8, rel
+1	ADDW A, RW1 @RW1+d8	ADDW A, RW1 @RW1+d8	SUBW A, RW1 @RW1+d8	SUBW A, RW1 @RW1+d8	ADDCW A, RW1 @RW1+d8	ADDCW A, RW1 @RW1+d8	CMPW A, RW1 @RW1+d8	CMPW A, RW1 @RW1+d8	ANDW A, RW1 @RW1+d8	ANDW A, RW1 @RW1+d8	ORW A, RW1 @RW1+d8	ORW A, RW1 @RW1+d8	XORW A, RW1 @RW1+d8	XORW A, RW1 @RW1+d8	DWBNZ @RW1+d8, rel	DWBNZ @RW1+d8, rel
+2	ADDW A, RW2 @RW2+d8	ADDW A, RW2 @RW2+d8	SUBW A, RW2 @RW2+d8	SUBW A, RW2 @RW2+d8	ADDCW A, RW2 @RW2+d8	ADDCW A, RW2 @RW2+d8	CMPW A, RW2 @RW2+d8	CMPW A, RW2 @RW2+d8	ANDW A, RW2 @RW2+d8	ANDW A, RW2 @RW2+d8	ORW A, RW2 @RW2+d8	ORW A, RW2 @RW2+d8	XORW A, RW2 @RW2+d8	XORW A, RW2 @RW2+d8	DWBNZ @RW2+d8, rel	DWBNZ @RW2+d8, rel
+3	ADDW A, RW3 @RW3+d8	ADDW A, RW3 @RW3+d8	SUBW A, RW3 @RW3+d8	SUBW A, RW3 @RW3+d8	ADDCW A, RW3 @RW3+d8	ADDCW A, RW3 @RW3+d8	CMPW A, RW3 @RW3+d8	CMPW A, RW3 @RW3+d8	ANDW A, RW3 @RW3+d8	ANDW A, RW3 @RW3+d8	ORW A, RW3 @RW3+d8	ORW A, RW3 @RW3+d8	XORW A, RW3 @RW3+d8	XORW A, RW3 @RW3+d8	DWBNZ @RW3+d8, rel	DWBNZ @RW3+d8, rel
+4	ADDW A, RW4 @RW4+d8	ADDW A, RW4 @RW4+d8	SUBW A, RW4 @RW4+d8	SUBW A, RW4 @RW4+d8	ADDCW A, RW4 @RW4+d8	ADDCW A, RW4 @RW4+d8	CMPW A, RW4 @RW4+d8	CMPW A, RW4 @RW4+d8	ANDW A, RW4 @RW4+d8	ANDW A, RW4 @RW4+d8	ORW A, RW4 @RW4+d8	ORW A, RW4 @RW4+d8	XORW A, RW4 @RW4+d8	XORW A, RW4 @RW4+d8	DWBNZ @RW4+d8, rel	DWBNZ @RW4+d8, rel
+5	ADDW A, RW5 @RW5+d8	ADDW A, RW5 @RW5+d8	SUBW A, RW5 @RW5+d8	SUBW A, RW5 @RW5+d8	ADDCW A, RW5 @RW5+d8	ADDCW A, RW5 @RW5+d8	CMPW A, RW5 @RW5+d8	CMPW A, RW5 @RW5+d8	ANDW A, RW5 @RW5+d8	ANDW A, RW5 @RW5+d8	ORW A, RW5 @RW5+d8	ORW A, RW5 @RW5+d8	XORW A, RW5 @RW5+d8	XORW A, RW5 @RW5+d8	DWBNZ @RW5+d8, rel	DWBNZ @RW5+d8, rel
+6	ADDW A, RW6 @RW6+d8	ADDW A, RW6 @RW6+d8	SUBW A, RW6 @RW6+d8	SUBW A, RW6 @RW6+d8	ADDCW A, RW6 @RW6+d8	ADDCW A, RW6 @RW6+d8	CMPW A, RW6 @RW6+d8	CMPW A, RW6 @RW6+d8	ANDW A, RW6 @RW6+d8	ANDW A, RW6 @RW6+d8	ORW A, RW6 @RW6+d8	ORW A, RW6 @RW6+d8	XORW A, RW6 @RW6+d8	XORW A, RW6 @RW6+d8	DWBNZ @RW6+d8, rel	DWBNZ @RW6+d8, rel
+7	ADDW A, RW7 @RW7+d8	ADDW A, RW7 @RW7+d8	SUBW A, RW7 @RW7+d8	SUBW A, RW7 @RW7+d8	ADDCW A, RW7 @RW7+d8	ADDCW A, RW7 @RW7+d8	CMPW A, RW7 @RW7+d8	CMPW A, RW7 @RW7+d8	ANDW A, RW7 @RW7+d8	ANDW A, RW7 @RW7+d8	ORW A, RW7 @RW7+d8	ORW A, RW7 @RW7+d8	XORW A, RW7 @RW7+d8	XORW A, RW7 @RW7+d8	DWBNZ @RW7+d8, rel	DWBNZ @RW7+d8, rel
+8	ADDW A, RW0 @RW0+d16	ADDW A, RW0 @RW0+d16	SUBW A, RW0 @RW0+d16	SUBW A, RW0 @RW0+d16	ADDCW A, RW0 @RW0+d16	ADDCW A, RW0 @RW0+d16	CMPW A, RW0 @RW0+d16	CMPW A, RW0 @RW0+d16	ANDW A, RW0 @RW0+d16	ANDW A, RW0 @RW0+d16	ORW A, RW0 @RW0+d16	ORW A, RW0 @RW0+d16	XORW A, RW0 @RW0+d16	XORW A, RW0 @RW0+d16	DWBNZ @RW0+d16, rel	DWBNZ @RW0+d16, rel
+9	ADDW A, RW1 @RW1+d16	ADDW A, RW1 @RW1+d16	SUBW A, RW1 @RW1+d16	SUBW A, RW1 @RW1+d16	ADDCW A, RW1 @RW1+d16	ADDCW A, RW1 @RW1+d16	CMPW A, RW1 @RW1+d16	CMPW A, RW1 @RW1+d16	ANDW A, RW1 @RW1+d16	ANDW A, RW1 @RW1+d16	ORW A, RW1 @RW1+d16	ORW A, RW1 @RW1+d16	XORW A, RW1 @RW1+d16	XORW A, RW1 @RW1+d16	DWBNZ @RW1+d16, rel	DWBNZ @RW1+d16, rel
+A	ADDW A, RW2 @RW2+d16	ADDW A, RW2 @RW2+d16	SUBW A, RW2 @RW2+d16	SUBW A, RW2 @RW2+d16	ADDCW A, RW2 @RW2+d16	ADDCW A, RW2 @RW2+d16	CMPW A, RW2 @RW2+d16	CMPW A, RW2 @RW2+d16	ANDW A, RW2 @RW2+d16	ANDW A, RW2 @RW2+d16	ORW A, RW2 @RW2+d16	ORW A, RW2 @RW2+d16	XORW A, RW2 @RW2+d16	XORW A, RW2 @RW2+d16	DWBNZ @RW2+d16, rel	DWBNZ @RW2+d16, rel
+B	ADDW A, RW3 @RW3+d16	ADDW A, RW3 @RW3+d16	SUBW A, RW3 @RW3+d16	SUBW A, RW3 @RW3+d16	ADDCW A, RW3 @RW3+d16	ADDCW A, RW3 @RW3+d16	CMPW A, RW3 @RW3+d16	CMPW A, RW3 @RW3+d16	ANDW A, RW3 @RW3+d16	ANDW A, RW3 @RW3+d16	ORW A, RW3 @RW3+d16	ORW A, RW3 @RW3+d16	XORW A, RW3 @RW3+d16	XORW A, RW3 @RW3+d16	DWBNZ @RW3+d16, rel	DWBNZ @RW3+d16, rel
+C	ADDW A, RW0 @RW0+R7	ADDW A, RW0 @RW0+R7	SUBW A, RW0 @RW0+R7	SUBW A, RW0 @RW0+R7	ADDCW A, RW0 @RW0+R7	ADDCW A, RW0 @RW0+R7	CMPW A, RW0 @RW0+R7	CMPW A, RW0 @RW0+R7	ANDW A, RW0 @RW0+R7	ANDW A, RW0 @RW0+R7	ORW A, RW0 @RW0+R7	ORW A, RW0 @RW0+R7	XORW A, RW0 @RW0+R7	XORW A, RW0 @RW0+R7	DWBNZ @RW0+R7, rel	DWBNZ @RW0+R7, rel
+D	ADDW A, RW1 @RW1+R7	ADDW A, RW1 @RW1+R7	SUBW A, RW1 @RW1+R7	SUBW A, RW1 @RW1+R7	ADDCW A, RW1 @RW1+R7	ADDCW A, RW1 @RW1+R7	CMPW A, RW1 @RW1+R7	CMPW A, RW1 @RW1+R7	ANDW A, RW1 @RW1+R7	ANDW A, RW1 @RW1+R7	ORW A, RW1 @RW1+R7	ORW A, RW1 @RW1+R7	XORW A, RW1 @RW1+R7	XORW A, RW1 @RW1+R7	DWBNZ @RW1+R7, rel	DWBNZ @RW1+R7, rel
+E	ADDW A, PC+d16	ADDW A, PC+d16	SUBW A, PC+d16	SUBW A, PC+d16	ADDCW A, PC+d16	ADDCW A, PC+d16	CMPW A, PC+d16	CMPW A, PC+d16	ANDW A, PC+d16	ANDW A, PC+d16	ORW A, PC+d16	ORW A, PC+d16	XORW A, PC+d16	XORW A, PC+d16	DWBNZ @PC+d16, rel	DWBNZ @PC+d16, rel
+F	ADDW A, RW3+ addr16	ADDW A, RW3+ addr16	SUBW A, RW3+ addr16	SUBW A, RW3+ addr16	ADDCW A, RW3+ addr16	ADDCW A, RW3+ addr16	CMPW A, RW3+ addr16	CMPW A, RW3+ addr16	ANDW A, RW3+ addr16	ANDW A, RW3+ addr16	ORW A, RW3+ addr16	ORW A, RW3+ addr16	XORW A, RW3+ addr16	XORW A, RW3+ addr16	DWBNZ @RW3+ addr16, rel	DWBNZ @RW3+ addr16, rel

Table B.9-13 ea Instruction 8 (First Byte = 77_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A', @RW0+d8, A	SUBW RW0, A', @RW0+d8, A	SUBW RW0, A', @RW0+d8, A	SUBW RW0, A', @RW0+d8, A	SUBCW A, RW0', @RW0+d8	SUBCW A, RW0', @RW0+d8	NEGW RW0', @RW0+d8	NEGW RW0', @RW0+d8	ANDW RW0, A', @RW0+d8, A	ANDW RW0, A', @RW0+d8, A	ORW RW0, A', @RW0+d8, A	ORW RW0, A', @RW0+d8, A	XORW RW0, A', @RW0+d8, A	XORW RW0, A', @RW0+d8, A	NOTW RW0', @RW0+d8	NOTW RW0', @RW0+d8
+1	ADDW RW1, A', @RW1+d8, A	SUBW RW1, A', @RW1+d8, A	SUBW RW1, A', @RW1+d8, A	SUBW RW1, A', @RW1+d8, A	SUBCW A, RW1', @RW1+d8	SUBCW A, RW1', @RW1+d8	NEGW RW1', @RW1+d8	NEGW RW1', @RW1+d8	ANDW RW1, A', @RW1+d8, A	ANDW RW1, A', @RW1+d8, A	ORW RW1, A', @RW1+d8, A	ORW RW1, A', @RW1+d8, A	XORW RW1, A', @RW1+d8, A	XORW RW1, A', @RW1+d8, A	NOTW RW1', @RW1+d8	NOTW RW1', @RW1+d8
+2	ADDW RW2, A', @RW2+d8, A	SUBW RW2, A', @RW2+d8, A	SUBW RW2, A', @RW2+d8, A	SUBW RW2, A', @RW2+d8, A	SUBCW A, RW2', @RW2+d8	SUBCW A, RW2', @RW2+d8	NEGW RW2', @RW2+d8	NEGW RW2', @RW2+d8	ANDW RW2, A', @RW2+d8, A	ANDW RW2, A', @RW2+d8, A	ORW RW2, A', @RW2+d8, A	ORW RW2, A', @RW2+d8, A	XORW RW2, A', @RW2+d8, A	XORW RW2, A', @RW2+d8, A	NOTW RW2', @RW2+d8	NOTW RW2', @RW2+d8
+3	ADDW RW3, A', @RW3+d8, A	SUBW RW3, A', @RW3+d8, A	SUBW RW3, A', @RW3+d8, A	SUBW RW3, A', @RW3+d8, A	SUBCW A, RW3', @RW3+d8	SUBCW A, RW3', @RW3+d8	NEGW RW3', @RW3+d8	NEGW RW3', @RW3+d8	ANDW RW3, A', @RW3+d8, A	ANDW RW3, A', @RW3+d8, A	ORW RW3, A', @RW3+d8, A	ORW RW3, A', @RW3+d8, A	XORW RW3, A', @RW3+d8, A	XORW RW3, A', @RW3+d8, A	NOTW RW3', @RW3+d8	NOTW RW3', @RW3+d8
+4	ADDW RW4, A', @RW4+d8, A	SUBW RW4, A', @RW4+d8, A	SUBW RW4, A', @RW4+d8, A	SUBW RW4, A', @RW4+d8, A	SUBCW A, RW4', @RW4+d8	SUBCW A, RW4', @RW4+d8	NEGW RW4', @RW4+d8	NEGW RW4', @RW4+d8	ANDW RW4, A', @RW4+d8, A	ANDW RW4, A', @RW4+d8, A	ORW RW4, A', @RW4+d8, A	ORW RW4, A', @RW4+d8, A	XORW RW4, A', @RW4+d8, A	XORW RW4, A', @RW4+d8, A	NOTW RW4', @RW4+d8	NOTW RW4', @RW4+d8
+5	ADDW RW5, A', @RW5+d8, A	SUBW RW5, A', @RW5+d8, A	SUBW RW5, A', @RW5+d8, A	SUBW RW5, A', @RW5+d8, A	SUBCW A, RW5', @RW5+d8	SUBCW A, RW5', @RW5+d8	NEGW RW5', @RW5+d8	NEGW RW5', @RW5+d8	ANDW RW5, A', @RW5+d8, A	ANDW RW5, A', @RW5+d8, A	ORW RW5, A', @RW5+d8, A	ORW RW5, A', @RW5+d8, A	XORW RW5, A', @RW5+d8, A	XORW RW5, A', @RW5+d8, A	NOTW RW5', @RW5+d8	NOTW RW5', @RW5+d8
+6	ADDW RW6, A', @RW6+d8, A	SUBW RW6, A', @RW6+d8, A	SUBW RW6, A', @RW6+d8, A	SUBW RW6, A', @RW6+d8, A	SUBCW A, RW6', @RW6+d8	SUBCW A, RW6', @RW6+d8	NEGW RW6', @RW6+d8	NEGW RW6', @RW6+d8	ANDW RW6, A', @RW6+d8, A	ANDW RW6, A', @RW6+d8, A	ORW RW6, A', @RW6+d8, A	ORW RW6, A', @RW6+d8, A	XORW RW6, A', @RW6+d8, A	XORW RW6, A', @RW6+d8, A	NOTW RW6', @RW6+d8	NOTW RW6', @RW6+d8
+7	ADDW RW7, A', @RW7+d8, A	SUBW RW7, A', @RW7+d8, A	SUBW RW7, A', @RW7+d8, A	SUBW RW7, A', @RW7+d8, A	SUBCW A, RW7', @RW7+d8	SUBCW A, RW7', @RW7+d8	NEGW RW7', @RW7+d8	NEGW RW7', @RW7+d8	ANDW RW7, A', @RW7+d8, A	ANDW RW7, A', @RW7+d8, A	ORW RW7, A', @RW7+d8, A	ORW RW7, A', @RW7+d8, A	XORW RW7, A', @RW7+d8, A	XORW RW7, A', @RW7+d8, A	NOTW RW7', @RW7+d8	NOTW RW7', @RW7+d8
+8	ADDW @RW0, A', @RW0+d16, A	SUBW @RW0, A', @RW0+d16, A	SUBW @RW0, A', @RW0+d16, A	SUBW @RW0, A', @RW0+d16, A	SUBCW A, @RW0', @RW0+d16	SUBCW A, @RW0', @RW0+d16	NEGW @RW0', @RW0+d16	NEGW @RW0', @RW0+d16	ANDW @RW0, A', @RW0+d16, A	ANDW @RW0, A', @RW0+d16, A	ORW @RW0, A', @RW0+d16, A	ORW @RW0, A', @RW0+d16, A	XORW @RW0, A', @RW0+d16, A	XORW @RW0, A', @RW0+d16, A	NOTW @RW0', @RW0+d16	NOTW @RW0', @RW0+d16
+9	ADDW @RW1, A', @RW1+d16, A	SUBW @RW1, A', @RW1+d16, A	SUBW @RW1, A', @RW1+d16, A	SUBW @RW1, A', @RW1+d16, A	SUBCW A, @RW1', @RW1+d16	SUBCW A, @RW1', @RW1+d16	NEGW @RW1', @RW1+d16	NEGW @RW1', @RW1+d16	ANDW @RW1, A', @RW1+d16, A	ANDW @RW1, A', @RW1+d16, A	ORW @RW1, A', @RW1+d16, A	ORW @RW1, A', @RW1+d16, A	XORW @RW1, A', @RW1+d16, A	XORW @RW1, A', @RW1+d16, A	NOTW @RW1', @RW1+d16	NOTW @RW1', @RW1+d16
+A	ADDW @RW2, A', @RW2+d16, A	SUBW @RW2, A', @RW2+d16, A	SUBW @RW2, A', @RW2+d16, A	SUBW @RW2, A', @RW2+d16, A	SUBCW A, @RW2', @RW2+d16	SUBCW A, @RW2', @RW2+d16	NEGW @RW2', @RW2+d16	NEGW @RW2', @RW2+d16	ANDW @RW2, A', @RW2+d16, A	ANDW @RW2, A', @RW2+d16, A	ORW @RW2, A', @RW2+d16, A	ORW @RW2, A', @RW2+d16, A	XORW @RW2, A', @RW2+d16, A	XORW @RW2, A', @RW2+d16, A	NOTW @RW2', @RW2+d16	NOTW @RW2', @RW2+d16
+B	ADDW @RW3, A', @RW3+d16, A	SUBW @RW3, A', @RW3+d16, A	SUBW @RW3, A', @RW3+d16, A	SUBW @RW3, A', @RW3+d16, A	SUBCW A, @RW3', @RW3+d16	SUBCW A, @RW3', @RW3+d16	NEGW @RW3', @RW3+d16	NEGW @RW3', @RW3+d16	ANDW @RW3, A', @RW3+d16, A	ANDW @RW3, A', @RW3+d16, A	ORW @RW3, A', @RW3+d16, A	ORW @RW3, A', @RW3+d16, A	XORW @RW3, A', @RW3+d16, A	XORW @RW3, A', @RW3+d16, A	NOTW @RW3', @RW3+d16	NOTW @RW3', @RW3+d16
+C	ADDW @RW0+, A', @RW0+RW7, A	SUBW @RW0+, A', @RW0+RW7, A	SUBW @RW0+, A', @RW0+RW7, A	SUBW @RW0+, A', @RW0+RW7, A	SUBCW A, @RW0+', @RW0+RW7	SUBCW A, @RW0+', @RW0+RW7	NEGW @RW0+', @RW0+RW7	NEGW @RW0+', @RW0+RW7	ANDW @RW0+, A', @RW0+RW7, A	ANDW @RW0+, A', @RW0+RW7, A	ORW @RW0+, A', @RW0+RW7, A	ORW @RW0+, A', @RW0+RW7, A	XORW @RW0+, A', @RW0+RW7, A	XORW @RW0+, A', @RW0+RW7, A	NOTW @RW0+', @RW0+RW7	NOTW @RW0+', @RW0+RW7
+D	ADDW @RW1+, A', @RW1+RW7, A	SUBW @RW1+, A', @RW1+RW7, A	SUBW @RW1+, A', @RW1+RW7, A	SUBW @RW1+, A', @RW1+RW7, A	SUBCW A, @RW1+', @RW1+RW7	SUBCW A, @RW1+', @RW1+RW7	NEGW @RW1+', @RW1+RW7	NEGW @RW1+', @RW1+RW7	ANDW @RW1+, A', @RW1+RW7, A	ANDW @RW1+, A', @RW1+RW7, A	ORW @RW1+, A', @RW1+RW7, A	ORW @RW1+, A', @RW1+RW7, A	XORW @RW1+, A', @RW1+RW7, A	XORW @RW1+, A', @RW1+RW7, A	NOTW @RW1+', @RW1+RW7	NOTW @RW1+', @RW1+RW7
+E	ADDW @RW2+, A', @PC+d16, A	SUBW @RW2+, A', @PC+d16, A	SUBW @RW2+, A', @PC+d16, A	SUBW @RW2+, A', @PC+d16, A	SUBCW A, @RW2+', @PC+d16	SUBCW A, @RW2+', @PC+d16	NEGW @RW2+', @PC+d16	NEGW @RW2+', @PC+d16	ANDW @RW2+, A', @PC+d16, A	ANDW @RW2+, A', @PC+d16, A	ORW @RW2+, A', @PC+d16, A	ORW @RW2+, A', @PC+d16, A	XORW @RW2+, A', @PC+d16, A	XORW @RW2+, A', @PC+d16, A	NOTW @RW2+', @PC+d16	NOTW @RW2+', @PC+d16
+F	ADDW @RW3+, A', addr16, A	SUBW @RW3+, A', addr16, A	SUBW @RW3+, A', addr16, A	SUBW @RW3+, A', addr16, A	SUBCW A, @RW3+', addr16	SUBCW A, @RW3+', addr16	NEGW @RW3+', addr16	NEGW @RW3+', addr16	ANDW @RW3+, A', addr16, A	ANDW @RW3+, A', addr16, A	ORW @RW3+, A', addr16, A	ORW @RW3+, A', addr16, A	XORW @RW3+, A', addr16, A	XORW @RW3+, A', addr16, A	NOTW @RW3+', addr16	NOTW @RW3+', addr16

Table B.9-14 ea Instruction 9 (First Byte = 78_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

Table B.9-15 MOVEA RWi, ea Instruction (First Byte = 79_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

Table B.9-16 MOV Ri, ea Instruction (First Byte = 7A_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

Table B.9-17 MOVW RWi, ea Instruction (First Byte = 7B_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

Table B.9-18 MOV ea, Ri Instruction (First Byte = 7C_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0, @RW0+d8, R0	MOV R0, R1, @RW0+d8, R1	MOV R0, R2, @RW0+d8, R2	MOV R0, R3, @RW0+d8, R3	MOV R0, R4, @RW0+d8, R4	MOV R0, R5, @RW0+d8, R5	MOV R0, R6, @RW0+d8, R6	MOV R0, R7, @RW0+d8, R7	MOV R0, R8, @RW0+d8, R8	MOV R0, R9, @RW0+d8, R9	MOV R0, R10, @RW0+d8, R10	MOV R0, R11, @RW0+d8, R11	MOV R0, R12, @RW0+d8, R12	MOV R0, R13, @RW0+d8, R13	MOV R0, R14, @RW0+d8, R14	MOV R0, R15, @RW0+d8, R15
+1	MOV R1, R0, @RW1+d8, R0	MOV R1, R1, @RW1+d8, R1	MOV R1, R2, @RW1+d8, R2	MOV R1, R3, @RW1+d8, R3	MOV R1, R4, @RW1+d8, R4	MOV R1, R5, @RW1+d8, R5	MOV R1, R6, @RW1+d8, R6	MOV R1, R7, @RW1+d8, R7	MOV R1, R8, @RW1+d8, R8	MOV R1, R9, @RW1+d8, R9	MOV R1, R10, @RW1+d8, R10	MOV R1, R11, @RW1+d8, R11	MOV R1, R12, @RW1+d8, R12	MOV R1, R13, @RW1+d8, R13	MOV R1, R14, @RW1+d8, R14	MOV R1, R15, @RW1+d8, R15
+2	MOV R2, R0, @RW2+d8, R0	MOV R2, R1, @RW2+d8, R1	MOV R2, R2, @RW2+d8, R2	MOV R2, R3, @RW2+d8, R3	MOV R2, R4, @RW2+d8, R4	MOV R2, R5, @RW2+d8, R5	MOV R2, R6, @RW2+d8, R6	MOV R2, R7, @RW2+d8, R7	MOV R2, R8, @RW2+d8, R8	MOV R2, R9, @RW2+d8, R9	MOV R2, R10, @RW2+d8, R10	MOV R2, R11, @RW2+d8, R11	MOV R2, R12, @RW2+d8, R12	MOV R2, R13, @RW2+d8, R13	MOV R2, R14, @RW2+d8, R14	MOV R2, R15, @RW2+d8, R15
+3	MOV R3, R0, @RW3+d8, R0	MOV R3, R1, @RW3+d8, R1	MOV R3, R2, @RW3+d8, R2	MOV R3, R3, @RW3+d8, R3	MOV R3, R4, @RW3+d8, R4	MOV R3, R5, @RW3+d8, R5	MOV R3, R6, @RW3+d8, R6	MOV R3, R7, @RW3+d8, R7	MOV R3, R8, @RW3+d8, R8	MOV R3, R9, @RW3+d8, R9	MOV R3, R10, @RW3+d8, R10	MOV R3, R11, @RW3+d8, R11	MOV R3, R12, @RW3+d8, R12	MOV R3, R13, @RW3+d8, R13	MOV R3, R14, @RW3+d8, R14	MOV R3, R15, @RW3+d8, R15
+4	MOV R4, R0, @RW4+d8, R0	MOV R4, R1, @RW4+d8, R1	MOV R4, R2, @RW4+d8, R2	MOV R4, R3, @RW4+d8, R3	MOV R4, R4, @RW4+d8, R4	MOV R4, R5, @RW4+d8, R5	MOV R4, R6, @RW4+d8, R6	MOV R4, R7, @RW4+d8, R7	MOV R4, R8, @RW4+d8, R8	MOV R4, R9, @RW4+d8, R9	MOV R4, R10, @RW4+d8, R10	MOV R4, R11, @RW4+d8, R11	MOV R4, R12, @RW4+d8, R12	MOV R4, R13, @RW4+d8, R13	MOV R4, R14, @RW4+d8, R14	MOV R4, R15, @RW4+d8, R15
+5	MOV R5, R0, @RW5+d8, R0	MOV R5, R1, @RW5+d8, R1	MOV R5, R2, @RW5+d8, R2	MOV R5, R3, @RW5+d8, R3	MOV R5, R4, @RW5+d8, R4	MOV R5, R5, @RW5+d8, R5	MOV R5, R6, @RW5+d8, R6	MOV R5, R7, @RW5+d8, R7	MOV R5, R8, @RW5+d8, R8	MOV R5, R9, @RW5+d8, R9	MOV R5, R10, @RW5+d8, R10	MOV R5, R11, @RW5+d8, R11	MOV R5, R12, @RW5+d8, R12	MOV R5, R13, @RW5+d8, R13	MOV R5, R14, @RW5+d8, R14	MOV R5, R15, @RW5+d8, R15
+6	MOV R6, R0, @RW6+d8, R0	MOV R6, R1, @RW6+d8, R1	MOV R6, R2, @RW6+d8, R2	MOV R6, R3, @RW6+d8, R3	MOV R6, R4, @RW6+d8, R4	MOV R6, R5, @RW6+d8, R5	MOV R6, R6, @RW6+d8, R6	MOV R6, R7, @RW6+d8, R7	MOV R6, R8, @RW6+d8, R8	MOV R6, R9, @RW6+d8, R9	MOV R6, R10, @RW6+d8, R10	MOV R6, R11, @RW6+d8, R11	MOV R6, R12, @RW6+d8, R12	MOV R6, R13, @RW6+d8, R13	MOV R6, R14, @RW6+d8, R14	MOV R6, R15, @RW6+d8, R15
+7	MOV R7, R0, @RW7+d8, R0	MOV R7, R1, @RW7+d8, R1	MOV R7, R2, @RW7+d8, R2	MOV R7, R3, @RW7+d8, R3	MOV R7, R4, @RW7+d8, R4	MOV R7, R5, @RW7+d8, R5	MOV R7, R6, @RW7+d8, R6	MOV R7, R7, @RW7+d8, R7	MOV R7, R8, @RW7+d8, R8	MOV R7, R9, @RW7+d8, R9	MOV R7, R10, @RW7+d8, R10	MOV R7, R11, @RW7+d8, R11	MOV R7, R12, @RW7+d8, R12	MOV R7, R13, @RW7+d8, R13	MOV R7, R14, @RW7+d8, R14	MOV R7, R15, @RW7+d8, R15
+8	MOV @RW0, R0, @RW0+d16, R0	MOV @RW0, R1, @RW0+d16, R1	MOV @RW0, R2, @RW0+d16, R2	MOV @RW0, R3, @RW0+d16, R3	MOV @RW0, R4, @RW0+d16, R4	MOV @RW0, R5, @RW0+d16, R5	MOV @RW0, R6, @RW0+d16, R6	MOV @RW0, R7, @RW0+d16, R7	MOV @RW0, R8, @RW0+d16, R8	MOV @RW0, R9, @RW0+d16, R9	MOV @RW0, R10, @RW0+d16, R10	MOV @RW0, R11, @RW0+d16, R11	MOV @RW0, R12, @RW0+d16, R12	MOV @RW0, R13, @RW0+d16, R13	MOV @RW0, R14, @RW0+d16, R14	MOV @RW0, R15, @RW0+d16, R15
+9	MOV @RW1, R0, @RW1+d16, R0	MOV @RW1, R1, @RW1+d16, R1	MOV @RW1, R2, @RW1+d16, R2	MOV @RW1, R3, @RW1+d16, R3	MOV @RW1, R4, @RW1+d16, R4	MOV @RW1, R5, @RW1+d16, R5	MOV @RW1, R6, @RW1+d16, R6	MOV @RW1, R7, @RW1+d16, R7	MOV @RW1, R8, @RW1+d16, R8	MOV @RW1, R9, @RW1+d16, R9	MOV @RW1, R10, @RW1+d16, R10	MOV @RW1, R11, @RW1+d16, R11	MOV @RW1, R12, @RW1+d16, R12	MOV @RW1, R13, @RW1+d16, R13	MOV @RW1, R14, @RW1+d16, R14	MOV @RW1, R15, @RW1+d16, R15
+A	MOV @RW2, R0, @RW2+d16, R0	MOV @RW2, R1, @RW2+d16, R1	MOV @RW2, R2, @RW2+d16, R2	MOV @RW2, R3, @RW2+d16, R3	MOV @RW2, R4, @RW2+d16, R4	MOV @RW2, R5, @RW2+d16, R5	MOV @RW2, R6, @RW2+d16, R6	MOV @RW2, R7, @RW2+d16, R7	MOV @RW2, R8, @RW2+d16, R8	MOV @RW2, R9, @RW2+d16, R9	MOV @RW2, R10, @RW2+d16, R10	MOV @RW2, R11, @RW2+d16, R11	MOV @RW2, R12, @RW2+d16, R12	MOV @RW2, R13, @RW2+d16, R13	MOV @RW2, R14, @RW2+d16, R14	MOV @RW2, R15, @RW2+d16, R15
+B	MOV @RW3, R0, @RW3+d16, R0	MOV @RW3, R1, @RW3+d16, R1	MOV @RW3, R2, @RW3+d16, R2	MOV @RW3, R3, @RW3+d16, R3	MOV @RW3, R4, @RW3+d16, R4	MOV @RW3, R5, @RW3+d16, R5	MOV @RW3, R6, @RW3+d16, R6	MOV @RW3, R7, @RW3+d16, R7	MOV @RW3, R8, @RW3+d16, R8	MOV @RW3, R9, @RW3+d16, R9	MOV @RW3, R10, @RW3+d16, R10	MOV @RW3, R11, @RW3+d16, R11	MOV @RW3, R12, @RW3+d16, R12	MOV @RW3, R13, @RW3+d16, R13	MOV @RW3, R14, @RW3+d16, R14	MOV @RW3, R15, @RW3+d16, R15
+C	MOV @RW0+, R0, @RW0+RW7, R0	MOV @RW0+, R1, @RW0+RW7, R1	MOV @RW0+, R2, @RW0+RW7, R2	MOV @RW0+, R3, @RW0+RW7, R3	MOV @RW0+, R4, @RW0+RW7, R4	MOV @RW0+, R5, @RW0+RW7, R5	MOV @RW0+, R6, @RW0+RW7, R6	MOV @RW0+, R7, @RW0+RW7, R7	MOV @RW0+, R8, @RW0+RW7, R8	MOV @RW0+, R9, @RW0+RW7, R9	MOV @RW0+, R10, @RW0+RW7, R10	MOV @RW0+, R11, @RW0+RW7, R11	MOV @RW0+, R12, @RW0+RW7, R12	MOV @RW0+, R13, @RW0+RW7, R13	MOV @RW0+, R14, @RW0+RW7, R14	MOV @RW0+, R15, @RW0+RW7, R15
+D	MOV @RW1+, R0, @RW1+RW7, R0	MOV @RW1+, R1, @RW1+RW7, R1	MOV @RW1+, R2, @RW1+RW7, R2	MOV @RW1+, R3, @RW1+RW7, R3	MOV @RW1+, R4, @RW1+RW7, R4	MOV @RW1+, R5, @RW1+RW7, R5	MOV @RW1+, R6, @RW1+RW7, R6	MOV @RW1+, R7, @RW1+RW7, R7	MOV @RW1+, R8, @RW1+RW7, R8	MOV @RW1+, R9, @RW1+RW7, R9	MOV @RW1+, R10, @RW1+RW7, R10	MOV @RW1+, R11, @RW1+RW7, R11	MOV @RW1+, R12, @RW1+RW7, R12	MOV @RW1+, R13, @RW1+RW7, R13	MOV @RW1+, R14, @RW1+RW7, R14	MOV @RW1+, R15, @RW1+RW7, R15
+E	MOV @RW2+, R0, @PC+d16, R0	MOV @RW2+, R1, @PC+d16, R1	MOV @RW2+, R2, @PC+d16, R2	MOV @RW2+, R3, @PC+d16, R3	MOV @RW2+, R4, @PC+d16, R4	MOV @RW2+, R5, @PC+d16, R5	MOV @RW2+, R6, @PC+d16, R6	MOV @RW2+, R7, @PC+d16, R7	MOV @RW2+, R8, @PC+d16, R8	MOV @RW2+, R9, @PC+d16, R9	MOV @RW2+, R10, @PC+d16, R10	MOV @RW2+, R11, @PC+d16, R11	MOV @RW2+, R12, @PC+d16, R12	MOV @RW2+, R13, @PC+d16, R13	MOV @RW2+, R14, @PC+d16, R14	MOV @RW2+, R15, @PC+d16, R15
+F	MOV @RW3+, R0, addr16, R0	MOV @RW3+, R1, addr16, R1	MOV @RW3+, R2, addr16, R2	MOV @RW3+, R3, addr16, R3	MOV @RW3+, R4, addr16, R4	MOV @RW3+, R5, addr16, R5	MOV @RW3+, R6, addr16, R6	MOV @RW3+, R7, addr16, R7	MOV @RW3+, R8, addr16, R8	MOV @RW3+, R9, addr16, R9	MOV @RW3+, R10, addr16, R10	MOV @RW3+, R11, addr16, R11	MOV @RW3+, R12, addr16, R12	MOV @RW3+, R13, addr16, R13	MOV @RW3+, R14, addr16, R14	MOV @RW3+, R15, addr16, R15

Table B.9-19 MOVW ea, Rwi Instruction (First Byte = 7D_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

Table B.9-20 XCH Ri, ea Instruction (First Byte = 7E_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

Table B.9-21 XCHW RWi, ea Instruction (First Byte = 7F_H)

F²MC-16LX, MB90990 Series Hardware Manual, Doc. # 002-04564 Rev. *A

APPENDIX C List of MB90990 Series 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 interrupt instructions.

■ List of MB90990 Series Interrupt Vectors

Table C-1 lists the interrupt vectors for the MB90990 series.

Table C-1 Interrupt Vectors (Sheet 1 of 2)

Interrupt request	Interrupt source	Interrupt control register		Vector address Low	Vector address Middle	Vector address High	Mode register
		Number	Address				
INT 1*	—	—	—	FFFFFC _H	FFFFFD _H	FFFFFE _H	Unused
INT 2*	—	—	—	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	Reserved	ICR00	0000B0 _H	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Unused
INT 12	Reserved			FFFFCC _H	FFFFCD _H	FFFFCE _H	Unused
INT 13	CAN1 reception	ICR01	0000B1 _H	FFFFC8 _H	FFFFC9 _H	FFFFCA _H	Unused
INT 14	CAN1 transmission/ Node status			FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Unused
INT 15	Reserved	ICR02	0000B2 _H	FFFFC0 _H	FFFFC1 _H	FFFFC2 _H	Unused
INT 16	Clock Calibration Unit			FFFFBC _H	FFFFBD _H	FFFFBE _H	Unused
INT 17	Reserved	ICR03	0000B3 _H	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Unused
INT 18	Reserved			FFFFB4 _H	FFFFB5 _H	FFFFB6 _H	Unused
INT 19	16-bit reload timer2	ICR04	0000B4 _H	FFFFB0 _H	FFFFB1 _H	FFFFB2 _H	Unused
INT 20	16-bit reload timer3			FFFFAC _H	FFFFAD _H	FFFFAE _H	Unused
INT 21	Reserved	ICR05	0000B5 _H	FFFFA8 _H	FFFFA9 _H	FFFFAA _H	Unused
INT 22	Reserved			FFFFA4 _H	FFFFA5 _H	FFFFA6 _H	Unused
INT 23	PPGC/PPGD	ICR06	0000B6 _H	FFFFA0 _H	FFFFA1 _H	FFFFA2 _H	Unused
INT 24	PPGA/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 8 to 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 12 to 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			FFFF84 _H	FFFF85 _H	FFFF86 _H	Unused
INT 31	Reserved	ICR10	0000BA _H	FFFF80 _H	FFFF81 _H	FFFF82 _H	Unused
INT 32	Reserved			FFFF7C _H	FFFF7D _H	FFFF7E _H	Unused

Table C-1 Interrupt Vectors (Sheet 2 of 2)

Interrupt request	Interrupt source	Interrupt control register		Vector address Low	Vector address Middle	Vector address High	Mode register
		Number	Address				
INT 33	Input capture 0 to 3	ICR11	0000BB _H	FFFF78 _H	FFFF79 _H	FFFF7A _H	Unused
INT 34	Reserved			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	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	Unused
INT 38	UART 1 TX			FFFF64 _H	FFFF65 _H	FFFF66 _H	Unused
INT 39	UART 2 RX	ICR14	0000BE _H	FFFF60 _H	FFFF61 _H	FFFF62 _H	Unused
INT 40	UART 2 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 generation module			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.							

■ Interrupt Sources, Interrupt Vectors, and Interrupt Control Registers

Table C-2 summarizes the relationships among the interrupt sources, interrupt vectors, and interrupt control registers of the MB90990 series.

Table C-2 Interrupt Sources, Interrupt Vectors, and Interrupt Control Registers (Sheet 1 of 2)

Interrupt source	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	ICR	Address
Reset	N	#08	FFFFDC _H	—	—
INT9 instruction	N	#09	FFFFD8 _H	—	—
Exception	N	#10	FFFFD4 _H	—	—
Reserved	N	#11	FFFFD0 _H	ICR00	0000B0 _H
Reserved	N	#12	FFFFCC _H		
CAN 1 RX	N	#13	FFFC08 _H	ICR01	0000B1 _H
CAN 1 TX/NS	N	#14	FFFC04 _H		
Reserved	N	#15	FFFC00 _H	ICR02	0000B2 _H
Clock Calibration Unit	N	#16	FFFFBC _H		
Reserved	N	#17	FFFFB8 _H	ICR03	0000B3 _H
Reserved	N	#18	FFFFB4 _H		

Table C-2 Interrupt Sources, Interrupt Vectors, and Interrupt Control Registers (Sheet 2 of 2)

Interrupt source	EI ² OS clear	Interrupt vector		Interrupt control register	
		Number	Address	ICR	Address
16-bit reload timer 2	Y1	#19	FFFFB0 _H	ICR04	0000B4 _H
16-bit reload timer 3	Y1	#20	FFFFAC _H		
Reserved	N	#21	FFFFA8 _H	ICR05	0000B5 _H
Reserved	N	#22	FFFFA4 _H		
PPGC/D	N	#23	FFFFA0 _H	ICR06	0000B6 _H
PPGA/B/E/F	N	#24	FFFF9C _H		
Time-base timer	N	#25	FFFF98 _H	ICR07	0000B7 _H
External interrupt 8 to 11	Y1	#26	FFFF94 _H		
Watch timer	N	#27	FFFF90 _H	ICR08	0000B8 _H
External interrupt 12 to 15	Y1	#28	FFFF8C _H		
A/D converter	Y1	#29	FFFF88 _H	ICR09	0000B9 _H
I/O timer 0	N	#30	FFFF84 _H		
Reserved	N	#31	FFFF80 _H	ICR10	0000BA _H
Reserved	N	#32	FFFF7C _H		
Input capture 0 to 3	Y1	#33	FFFF78 _H	ICR11	0000BB _H
Reserved	N	#34	FFFF74 _H		
UART 0 RX	Y2	#35	FFFF70 _H	ICR12	0000BC _H
UART 0 TX	Y1	#36	FFFF6C _H		
UART 1 RX	Y2	#37	FFFF68 _H	ICR13	0000BD _H
UART 1 TX	Y1	#38	FFFF64 _H		
UART 2 RX	Y2	#39	FFFF60 _H	ICR14	0000BE _H
UART 2 TX	Y1	#40	FFFF5C _H		
Flash memory	N	#41	FFFF58 _H	ICR15	0000BF _H
Delayed interrupt generation module	N	#42	FFFF54 _H		

Y1: An EI²OS interrupt clear signal or EI²OS register read access clears the interrupt source.

Y2: An EI²OS interrupt clear signal or EI²OS register read access clears the interrupt source. 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 source for one interrupt number, an EI²OS interrupt request clear signal clears both interrupt request flags.

When EI²OS transfer ends, an EI²OS interrupt clear signal is sent to every interrupt source 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 is shared by two interrupts. Therefore, while one interrupt is enabled, the other interrupt must be disabled.

Index



Numerics

16-bit Free-run Timer	
Block Diagram of 16-bit Free-run Timer	211
Explanation of Operation of 16-bit Free-run Timer	227
16-bit I/O Timer	
16-bit I/O Timer Interrupt and EI ² OS	226
Block Diagram of 16-bit I/O Timer	209
Functions of 16-bit I/O Timer	208
Generation of Interrupt Request from 16-bit I/O Timer	214
Interrupts of 16-bit I/O Timer	225
Module Configuration of 16-bit I/O Timer	208
Pins of 16-bit I/O Timer	214
Program Example of 16-bit I/O Timer	232
16-bit PPG Output Operation Mode	
Setting for 16-bit PPG Output Operation Mode	305
16-bit Reload Registers	
16-bit Reload Registers (TMRLR)	248
16-bit Reload Timer	
Block Diagram of 16-bit Reload Timer	238
Correspondence between 16-bit Reload Timer Interrupt and EI ² OS	249
EI ² OS Function of 16-bit Reload Timer	249
Generation of Interrupt Request from 16-bit Reload Timer	242
Interrupts of 16-bit Reload Timer	249
List of 16-bit Reload Timer Registers and Initial Value	241
Notes on Using 16-bit Reload Timer	260
Operation Modes of 16-bit Reload Timer	236
Pins of 16-bit Reload Timer	240
Setting of 16-bit Reload Timer	250
16-bit Timer Register	
16-bit Timer Registers (TMR)	247
Operating State of 16-bit Timer Register	251
Operation as 16-bit Timer Register Underflows	253, 258
24-bit Operand	
24-bit Operand Specification	31
8+8-bit PPG	
Setting for 8+8-bit PPG Output Operation Mode	308
8-/10-bit A/D Converter	
Block Diagram of 8-/10-bit A/D Converter	342
Conversion Modes of 8-/10-bit A/D Converter	341
EI ² OS of 8-/10-bit A/D Converter	362
Explanation of A/D Conversion Data Protection Function of 8-/10-bit A/D Converter	372
Functions of 8-/10-bit A/D Converter	340
Interrupts, EI ² OS of 8-/10-bit A/D Converter	362
List of Registers and Initial Values of 8-/10-bit A/D Converter	346
Notes on Using 8-/10-bit A/D Converter	376
Pins of 8-/10-bit A/D Converter	345
8-/16-bit PPG Timer	
Block Diagram of 8-/16-bit PPG Timer C	286
Block Diagram of 8-/16-bit PPG Timer D	289
Functions of 8-/16-bit PPG Timer	282
Generation of Interrupt Request from 8-/16-bit PPG Timer	292
Interrupts of 8-/16-bit PPG Timer	300
List of Registers and Initial Values of 8-/16-bit PPG Timer	292
Operation Modes of 8-/16-bit PPG Timer	283
Operation of 8-/16-bit PPG Timer	301
Pins of 8-/16-bit PPG Timer	291
8-bit PPG Output 2-channel Independent Operation Mode	
Setting for 8-bit PPG Output 2-channel Independent Operation Mode	302

A	
A	
Accumulator (A).....	38
A/D Control Status Register	
A/D Control Status Register 0 (ADCS0).....	351
A/D Control Status Register 1 (ADCS1).....	347
A/D Converter	
Interrupts of A/D Converter	362
A/D Data Registers	
A/D Data Registers (ADCR0/ADCR1).....	354
A/D Setting Registers	
A/D Setting Registers (ADSR0/ADSR1).....	355
Acceptance Filter	
Setting Acceptance Filter	504
Acceptance Filtering	
Acceptance Filtering.....	500
Accessing	
Accessing Multi-byte Data	34
Accumulator	
Accumulator (A).....	38
ADCR	
A/D Data Registers (ADCR0/ADCR1).....	354
ADCS	
A/D Control Status Register 0 (ADCS0).....	351
A/D Control Status Register 1 (ADCS1).....	347
Continuous Conversion Mode (ADCS: MD1 and MD0=10 _B)	363
Single Conversion Mode (ADCS: MD1 and MD0=00 _B or 01 _B).....	363
Stop Conversion Mode (ADCS: MD1 and MD0=11 _B)	363
Address Detection Control Register	
Address Detection Control Register 0 (PACSR0)	519
Address Detection Control Register 1 (PACSR1)	521
Address Match Detection	
Block Diagram of Address Match Detection Function	517
List of Registers and Initial Values of Address Match Detection Function	518
Operation of Address Match Detection Function	526
Operation of Address Match Detection Function at Storing Patch Program in E ² PROM.....	530
Overview of Address Match Detection Function	516
Program Example for Address Match Detection Function	532
Addressing	
Addressing	616
Direct Addressing	618
Indirect Addressing	624
ADER	
Analog Input Enable Registers (ADER)	173
Analog Input Enable Registers (ADER5, ADER6).....	360
ADSR	
A/D Setting Registers (ADSR0/ADSR1).....	355
Alternative Mode	
Alternative Mode.....	544
Analog Input Enable Register	
Analog Input Enable Registers (ADER).....	173
Analog Input Enable Registers (ADER5, ADER6).....	360
Asynchronous LIN Mode	
Operation in Asynchronous LIN Mode.....	439
Asynchronous Mode	
Operation in Asynchronous Mode	432
B	
Bank Addressing	
Bank Addressing Types.....	32
Bank Select Prefix	
Bank Select Prefix	46
BAP	
Buffer Address Pointer (BAP).....	73
Basic Configuration	
Basic Configuration of Serial Programming Connection	572
Baud Rate	
Calculating the Baud Rate	425
LIN-UART Baud Rate Selection	423
BGR	
Bit Configuration of Baud Rate Generator Register (BGR0/BGR1)	415
Bidirectional Communication	
Bidirectional Communication Function	443
Bit Configuration of Baud Rate Generator Register	
Bit Configuration of Baud Rate Generator Register (BGR0/BGR1)	415
Bit Timing	
Setting Bit Timing	504
Block Diagram	
Block Diagram of 16-bit Free-run Timer	211
Block Diagram of 16-bit I/O Timer	209
Block Diagram of 16-bit Reload Timer.....	238
Block Diagram of 8-/10-bit A/D Converter	342
Block Diagram of 8-/16-bit PPG Timer C.....	286
Block Diagram of 8-/16-bit PPG Timer D.....	289
Block Diagram of Address Match Detection Function	517
Block Diagram of CAN Controller	455
Block Diagram of Clock Supervisor	107
Block Diagram of D/A Converter	597
Block Diagram of Delayed Interrupt Generation Module	81
Block Diagram of DTP/External Interrupt	315
Block Diagram of Evaluation Product	8
Block Diagram of Flash Memory Product	10
Block Diagram of Input Capture	212

Block Diagram of LIN-UART.....	396	Chip Erase	
Block Diagram of LIN-UART Pins.....	401	Notes on Chip Erase.....	564
Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit	380	CKSCR	
Block Diagram of Pull-up Control Register (PUCR2)	172	Configuration of the Clock Selection Register (CKSCR)	93
Block Diagram of ROM Mirroring Function Select Module	536	Clock Calibration Unit	
Block Diagram of the Clock Calibration Unit.....	585	Block Diagram of the Clock Calibration Unit.....	585
Block Diagram of the Clock Generation Block	90	Clock Timing of Clock Calibration Unit	586
Block Diagram of the D/A Converter Pin	598	Timing of Clock Calibration Unit	586
Block Diagram of the Entire Flash Memory	542	Clock Frequency	
Block Diagram of the Low-power Consumption Circuit	135	Oscillating Clock Frequency and Serial Clock Input Frequency	574
Block Diagram of Time-base Timer	180	Clock Generation Block	
Block Diagram of Watch Timer	270	Block Diagram of the Clock Generation Block.....	90
Block Diagram of Watchdog Timer	196	List of Register of Clock Generation Block and Initial Value	92
Block Diagrams of the External Reset Pin.....	123	Clock Mode	
Buffer Address Pointer		Clock Mode	98
Buffer Address Pointer (BAP).....	73	Clock Mode Switching	156
Bus Mode		Clock Mode Transition	98
Memory Space in Each Bus Mode	163	Internal Clock Mode	236
Bus Operation Stop		Operation in Internal Clock Mode	253
Conditions for Canceling Bus Operation Stop (HALT=0)	468	Program Example in Internal Clock Mode	264
Conditions for Setting Bus Operation Stop (HALT=1)	468	Setting of Internal Clock Mode.....	252
State during Bus Operation Stop (HALT=1).....	468	Clock Selection Register	
BVAL		Configuration of the Clock Selection Register (CKSCR)	93
Note for Disabling Message Buffers by BVAL Bits	513	Clock Supervisor	
		Block Diagram of Clock Supervisor	107
C		Clock Supervisor Control Register (CSVCR)	110
Calculating		Clock Supervisor Register	109
Calculating the Execution Cycle Count	633	Example of Operation Flowchart for the Clock Supervisor	113
Calibration Unit		Example of Startup Flowchart when using the Clock Supervisor	114
Calibration Unit Control Register (CUCR)	590	Notes on using the Clock Supervisor.....	115
Features of the Calibration Unit.....	584	Operations of Clock Supervisor	112
Register List of the Calibration Unit.....	587	Overview of Clock Supervisor.....	106
Calibration Unit Control Register		Clock Supply	
Calibration Unit Control Register (CUCR)	590	Cycle of Clock Supply	269
CAN Controller		Clocks	
Block Diagram of CAN Controller	455	Clocks	88
Canceling Transmission Request from CAN Controller	498	CMR	
Features of CAN Controller	454	Common Register Bank Prefix (CMR).....	47
Reception Flowchart of the CAN Controller.....	503	Command	
Starting Transmission of CAN Controller.....	498	Notes on Issuing Commands.....	553
Transmission Flowchart of CAN Controller	499	Command Sequence	
CAN Direct Mode Register		Command Sequence Table	552
CAN Direct Mode Register (CDMR).....	512	Common Register Bank Prefix	
CCR		Common Register Bank Prefix (CMR).....	47
Condition Code Register (CCR)	40	Communication	
CDMR		Bidirectional Communication Function	443
CAN Direct Mode Register (CDMR).....	512	LIN-master-slave Communication Function.....	448
		Master-slave Communication Function	445

Compare Time		Control Status Register (CSR) (Upper)	463
Compare Time Setup (Bits CT2 to CT0)	359	Control Status Register (CSR-lower)	464
Condition Code Register		CSVCR	
Condition Code Register (CCR)	40	Clock Supervisor Control Register	
Configuration of the PLL/Sub clock Control Register		(CSVCR)	110
Configuration of the PLL/Sub clock Control Register		CT	
(PSCCR)	96	Compare Time Setup (Bits CT2 to CT0)	359
Continuous Conversion Mode		CUCR	
Continuous Conversion Mode (ADCS: MD1 and		Calibration Unit Control Register (CUCR)	590
MD0=10 _B)	363	Current Dissipation	
Continuous Conversion Mode Setup	367	CPU Operating Modes and	
Operations and Applications of Continuous		Current Dissipation	132
Conversion Mode	367	CUTD	
Control Status Register		CR Oscillation Timer Data Register(16-bit)	
Control Status Register (CSR) (Lower)	463	(CUTD)	592
Control Status Register (CSR) (Upper)	463	CUTR	
Control Status Register (CSR-lower)	464	Main Oscillation Timer Data Register(24-bit)	
Conversion Modes		(CUTR)	593
Conversion Modes of 8-/10-bit A/D Converter			
.....	341	D	
CPU		D/A Control Register	
Overview of CPU Memory Space	27	D/A Control Register 0 (DACR0)	601
Overview of the CPU	26	D/A Converter	
CPU Intermittent Operating Mode		Block Diagram of D/A Converter	597
CPU Intermittent Operating Mode	133	Block Diagram of the D/A Converter Pin	598
CPU Intermittent Operation Mode		D/A Converter Pins	598
CPU Intermittent Operation Mode	140	D/A Converter Registers	599
CPU Operating Detection Reset Circuit		Function and Operation of the D/A Converter	596
Block Diagram of Low Voltage/CPU Operating		Sample Program for the D/A Converter	602
Detection Reset Circuit	380	D/A Converter Register	
CPU Operating Detection Reset Circuit	379	D/A Converter Register 0 (DAT0)	600
Notes on Using CPU Operating Detection Reset		DACR	
Circuit	388	D/A Control Register 0 (DACR0)	601
Operating of CPU Operating Detection Reset		DAT	
Circuit	386	D/A Converter Register 0 (DAT0)	600
Sample Program for Low Voltage/CPU Operating		Data Counter	
Detection Reset Circuit	389	Data Counter (DCT)	72
CPU Operating Modes		Data Frame	
CPU Operating Modes and Current Dissipation		Processing for Reception of Data Frame and	
.....	132	Remote Frame	501
CPU Operation Detection		Data Polling Flag	
Low-voltage/CPU Operation Detection Reset Setting		Data Polling Flag (DQ7)	556
Register (LVRS)	384	Data Protection Function	
CR Oscillation Timer Data Register		Explanation of A/D Conversion Data Protection	
Configuration of CR Oscillation Timer Data		Function of 8-/10-bit A/D Converter	372
Register	594	Data Register	
CR Oscillation Timer Data Register(16-bit)		List of Message Buffers (Data Registers)	461
(CUTD)	592	List of Message Buffers (DLC Registers)	460
CR Oscillation Trimming Setting Register		DCT	
CR Oscillation Trimming Setting Register		Data Counter (DCT)	72
(CRTR)	589	DDR	
CRTR		Port Direction Register (DDR)	170
CR Oscillation Trimming Setting Register		Delayed Interrupt Generation Module	
(CRTR)	589	Block Diagram of Delayed Interrupt Generation	
CSR		Module	81
Control Status Register (CSR) (Lower)	463		

Explanation of Operation of Delayed Interrupt Generation Module	84	Pins of DTP/External Interrupt	317
Notes on Using Delayed Interrupt Generation Module	85	Program Example of DTP/External Interrupt Function.....	335
Overview of Delayed Interrupt Generation Module	80	Setting of DTP/External Interrupt.....	327
Program Example of Delayed Interrupt Generation Module	86	DTP/External Interrupt Enable Register 1	
Delayed Interrupt Request Generate/Cancel Register		DTP/External Interrupt Enable Register 1 (ENIR1)	321
Delayed Interrupt Request Generate/Cancel Register (DIRR)	83	DTP/External Interrupt Source Register 1	
Description		DTP/External Interrupt Source Register 1 (EIRR1)	319
Description of Instruction Presentation Items and Symbols	636	E	
Descriptor		E ² PROM	
Extended Intelligent I/O Service Descriptor (ISD).....	72	E ² PROM Memory Map	528
Detect Address		Operation of Address Match Detection Function at Storing Patch Program in E ² PROM	530
Setting Detect Address.....	526	System Configuration and E ² PROM Memory Map.....	527
Detect Address Setting Registers		ECCR	
Detect Address Setting Registers (PADR0 to PADR5).....	523	Bit Configuration of Extended Communication Control Register (ECCR).....	413
Functions of Detect Address Setting Registers.....	524	Effective Address Field	
Detection Level Setting Register		Effective Address Field	617, 635
Detection Level Setting Register 1 (ELVR1)	323	EI ² OS	
Detection Reset Circuit		16-bit I/O Timer Interrupt and EI ² OS	226
Operating of Low Voltage Detection Reset Circuit.....	386	Conversion Operation by EI ² OS Function.....	371
Device		Correspondence between 16-bit Reload Timer Interrupt and EI ² OS.....	249
Notes on Handling the Device	21	Correspondence Between Time-base Timer Interrupt and EI ² OS	185
Direct Addressing		Correspondence to EI ² OS Function	226
Direct Addressing	618	EI ² OS Function of 16-bit Reload Timer.....	249
DIRR		EI ² OS of 8-/10-bit A/D Converter.....	362
Delayed Interrupt Request Generate/Cancel Register (DIRR).....	83	EI ² OS Operation Flow.....	75
DLC Registers		Extended Intelligent I/O Service (EI ² OS).....	53, 70
List of Message Buffers (DLC Registers)	460	Interrupts, EI ² OS of 8-/10-bit A/D Converter	362
DQ3		LIN-UART Interrupts and EI ² OS.....	418
Sector Erase Timer Flag (DQ3).....	559	Structure of Extended Intelligent I/O Service(EI ² OS).....	71
DQ5		EI ² OS Status Register	
Timing Limit Exceeded Flag (DQ5)	558	EI ² OS Status Register (ISCS).....	73
DQ6		EIRR	
Toggle Bit Flag (DQ6)	557	DTP/External Interrupt Source Register 1 (EIRR1)	319
DQ7		ELVR	
Data Polling Flag (DQ7)	556	Detection Level Setting Register 1 (ELVR1).....	323
DTP		ENIR	
DTP Function	332	DTP/External Interrupt Enable Register 1 (ENIR1)	321
Program Example of DTP Function.....	336	Erase	
DTP/External Interrupt		Suspending Sector Erase of Flash Memory	567
Block Diagram of DTP/External Interrupt	315	Erasing	
DTP/External Interrupt Function	314	Detailed Explanation of Writing to and Erasing Flash Memory	560
DTP/External Interrupt Operation	329	Erasing All Data in the Flash Memory	
List of Registers and Initial Values in DTP/External Interrupt.....	318		
Notes on Using DTP/External Interrupt	333		

(Erasing Chips)	564
Erasing Optional Data in the Flash Memory (Erasing Sectors)	565
Erasing Chip	
Erasing All Data in the Flash Memory (Erasing Chips)	564
ESCR	
Bit Configuration of Extended Status/Control Register (ESCR)	411
Evaluation Product	
Block Diagram of Evaluation Product	8
Event Count Mode	
Event Count Mode	236
Operation in Event Count Mode	259
Program Example in Event Count Mode	265
Setting of Event Count Mode	257
Exceptions	
Exceptions processing	54
Execution Cycle Count	
Calculating the Execution Cycle Count	633
Execution Cycle Count	632
Explanation	
Explanation of A/D Conversion Data Protection Function of 8-/10-bit A/D Converter	372
Extended Communication Control Register	
Bit Configuration of Extended Communication Control Register (ECCR)	413
Extended Intelligent I/O Service	
Extended Intelligent I/O Service (EI ² OS)	53, 70
Extended Intelligent I/O Service Descriptor (ISD)	72
Structure of Extended Intelligent I/O Service (EI ² OS)	71
Extended Status/Control Register	
Bit Configuration of Extended Status/Control Register (ESCR)	411
External Interrupt	
Block Diagram of DTP/External Interrupt	315
DTP/External Interrupt Enable Register 1 (ENIR1)	321
DTP/External Interrupt Function	314
DTP/External Interrupt Operation	329
DTP/External Interrupt Source Register 1 (EIRR1)	319
External Interrupt Function	331
List of Registers and Initial Values in DTP/External Interrupt	318
Notes on Using DTP/External Interrupt	333
Pins of DTP/External Interrupt	317
Program Example of DTP/External Interrupt Function	335
Selection of External Interrupt Source	325
Setting of DTP/External Interrupt	327
External Reset	
Block Diagrams of the External Reset Pin	123

F

F ² MC-16LX Instruction List	
F ² MC-16LX Instruction List	639
Flag Change Disable Prefix	
Flag Change Disable Prefix (NCC)	47
Flag Set Timing	
Reception Interrupt Generation and Flag Set Timing	419
Transmission Interrupt Generation and Flag Set Timing	421
Flash Memory	
Block Diagram of Flash Memory Product	10
Block Diagram of the Entire Flash Memory	542
Detailed Explanation of Writing to and Erasing Flash Memory	560
Erasing All Data in the Flash Memory (Erasing Chips)	564
Erasing Optional Data in the Flash Memory (Erasing Sectors)	565
Features of Flash Memory	540
Notes on Using Flash Memory	569
Restarting Sector Erase of Flash Memory	568
Sector Configuration of the Flash Memory	543
Sector Erase Procedure for Flash Memory	565
Setting the Flash Memory to the Read/reset State	561
Suspending Sector Erase of Flash Memory	567
Writing Data to the Flash Memory	562
Writing to the Flash Memory	562
Writing to/Erasing Flash Memory	540
Flash Memory Control Status Register	
Automatic Algorithm End Timing	547
Flash Memory Control Status Register (FMCS)	545
Flash Memory Mode	
Flash Memory Mode	544
Flash Memory Write Control Register	
Flash Memory Write Control Register (FWR0/FWR1)	548
Setup Flow of Flash Memory Write Control Register (FWR0/FWR1)	550
Flash Microcontroller Programmer	
Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)	581
Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)	579
Flash Security	
How to Disable the Flash Security Feature	570
How to Enable the Flash Security Feature	570
FMCS	
Flash Memory Control Status Register (FMCS)	545, 547
Setting FMCS:WE	551
Frame Format	

Setting Frame Format	504	List of Message Buffers (ID Registers)	458
Free-run Timer		ILSR	
Block Diagram of 16-bit Free-run Timer	211	Input Level Select Register (ILSR0, ILSR1)	174
Explanation of Operation of 16-bit		Indirect Addressing	
Free-run Timer	227	Indirect Addressing	624
FWR		Initial Value	
Flash Memory Write Control Register		List of Registers and Initial Values in DTP/External	
(FWR0/FWR1)	548	Interrupt	318
Setup Flow of Flash Memory Write Control Register		Input Capture	
(FWR0/FWR1)	550	Block Diagram of Input Capture	212
		Explanation of Operation of Input Capture	229
H		Input Capture Control Status Registers	
HALT		Input Capture Control Status Registers (ICS)	219
Conditions for Canceling Bus Operation Stop		Input Capture Edge Register	
(HALT=0)	468	Input Capture Edge Register (ICE)	222
Conditions for Setting Bus Operation Stop		Input Capture Register	
(HALT=1)	468	Input Capture Register (IPCP)	221
State during Bus Operation Stop (HALT=1)	468	Input Level Select Register	
Hardware Interrupt		Input Level Select Register (ILSR0, ILSR1)	174
Hardware Interrupt Operation	64	Input-output Circuits	
Hardware Interrupts	52, 63	Input-output Circuits	16
Occurrence and Release of Hardware Interrupt	65	Instruction	
Structure of Hardware Interrupt	63	Description of Instruction Presentation Items and	
Hardware Sequence Flags		Symbols	636
Hardware Sequence Flags	554	Exception due to Execution of an Undefined	
		Instruction	78
I		Execution of an Undefined Instruction	78
I/O		F2MC-16LX Instruction List	639
I/O Area	28	Instruction Types	615
I/O Maps		Interrupt Disable Instructions	49
I/O Maps (Addresses:000000 _H to 0000FF _H)	604	Restrictions on Interrupt Disable Instructions and	
I/O Pins		Prefix Instructions	49
Status of I/O Pins (Single-chip Mode)	154	Structure of Instruction Map	653
I/O Port		Instruction Presentation Items and Symbols	
I/O Port Registers	167	Description of Instruction Presentation Items and	
Overview of I/O Ports	166	Symbols	636
I/O Timer		Inter-CPU Connection	
16-bit I/O Timer Interrupt and EI ² OS	226	Inter-CPU Connection Method	431
Block Diagram of 16-bit I/O Timer	209	Internal Clock Mode	
Functions of 16-bit I/O Timer	208	Internal Clock Mode	236
Generation of Interrupt Request from 16-bit		Operation in Internal Clock Mode	253
I/O Timer	214	Program Example in Internal Clock Mode	264
Interrupts of 16-bit I/O Timer	225	Setting of Internal Clock Mode	252
Module Configuration of 16-bit I/O Timer	208	Interrupt	
Pins of 16-bit I/O Timer	214	16-bit I/O Timer Interrupt and EI ² OS	226
Program Example of 16-bit I/O Timer	232	Cancellation of Standby Mode by Interrupt	155
ICE		Correspondence between 16-bit Reload Timer	
Input Capture Edge Register (ICE)	222	Interrupt and EI ² OS	249
ICR		Correspondence Between Time-base Timer Interrupt	
Interrupt Control Register (ICR00 to ICR15)	57	and EI ² OS	185
ICS		Hardware Interrupt Operation	64
Input Capture Control Status Registers (ICS)	219	Hardware Interrupts	52, 63
ID		Interrupt Disable Instructions	49
Setting ID	504	Interrupt Flow	61
ID Registers		Interrupt Number	81
		Interrupt of Time-base Timer	185
		Interrupts of 16-bit Reload Timer	249

Interrupts of 8-/16-bit PPG Timer	300	Input Capture Register (IPCP).....	221
Interrupts of A/D Converter	362	ISCS	
Interrupts, EI ² OS of 8-/10-bit A/D Converter	362	EI ² OS Status Register (ISCS).....	73
LIN-UART Interrupts.....	416	ISD	
LIN-UART Interrupts and EI ² OS	418	Extended Intelligent I/O Service Descriptor	
Multiple Interrupts.....	67	(ISD).....	72
Occurrence and Release of Hardware Interrupt.....	65		
Reception Interrupt Generation and Flag Set		L	
Timing	419	Last Event Indicator Register	
Restrictions on Interrupt Disable Instructions and		Register Configuration	469
Prefix Instructions	49	LEIR	
Software Interrupt Operation.....	68	Register Configuration	469
Software Interrupts	53, 68	LIN Master Device	
Structure of Hardware Interrupt.....	63	LIN-UART as LIN Master Device	449
Structure of Software Interrupts	68	LIN-master-slave Communication	
Transmission Interrupt Generation and Flag Set		LIN-master-slave Communication Function	448
Timing	421	LIN-UART	
Transmission Interrupt Request Generation		Block Diagram of LIN-UART	396
Timing	422	Block Diagram of LIN-UART Pins.....	401
Watch Timer Interrupt.....	275	Configuration of LIN-UART.....	395
Watch Timer Interrupt and EI ² OS Transfer		LIN-UART as LIN Master Device.....	449
Function	275	LIN-UART Baud Rate Selection	423
Interrupt Control Register		LIN-UART Functions	392
Interrupt Control Register (ICR00 to ICR15)	57	LIN-UART Interrupts	416
Interrupt Sources, Interrupt Vectors, and Interrupt		LIN-UART Interrupts and EI ² OS	418
Control Registers	677	LIN-UART Pin Direct Access	442
Interrupt Disable Instructions		LIN-UART Pins	400
Interrupt Disable Instructions.....	49	LIN-UART Registers.....	402
Restrictions on Interrupt Disable Instructions and		Notes on Using LIN-UART	451
Prefix Instructions	49	Operation of LIN-UART	430
Interrupt Number		LIN-UART Serial Mode Register	
Details of Pins and Interrupt Number	210	LIN-UART Serial Mode Register (SMR)	405
Details of Pins and Interrupt Numbers	316	Low Voltage	
Interrupt Number	81	Block Diagram of Low Voltage/CPU Operating	
Interrupt Request		Detection Reset Circuit	380
Generation of Interrupt Request from 16-bit		Operating of Low Voltage Detection Reset	
I/O Timer	214	Circuit	386
Generation of Interrupt Request from 16-bit		Sample Program for Low Voltage/CPU Operating	
Reload Timer	242	Detection Reset Circuit	389
Generation of Interrupt Request from 8-/16-bit		Low Voltage Detection	
PPG Timer	292	Status of Reset Source Bit and Low Voltage Detection	
Generation of Interrupt Request from		Bit	128
Time-base Timer	182	Low Voltage Detection Reset Circuit	
Generation of Interrupt Request from		Low Voltage Detection Reset Circuit	378
Watch Timer	272	Notes on Using Low Voltage Detection Reset	
Interrupt Sources		Circuit	387
Interrupt Sources, Interrupt Vectors, and Interrupt		Low Voltage/CPU Operating Detection Reset Control	
Control Registers	677	Register	
Interrupt Vector		Low Voltage/CPU Operating Detection Reset Control	
Interrupt Sources, Interrupt Vectors, and Interrupt		Register (LVRC).....	382
Control Registers	677	Low-power Consumption	
Interrupt Vector	55	Block Diagram of the Low-power Consumption	
List of Interrupt Vectors	68	Circuit	135
List of MB90990 Series Interrupt Vectors.....	675	Setting Low-power Consumption Mode.....	505
Interval Timer		Low-power Consumption Mode Control Register	
Interval Timer Function.....	178, 186, 268, 277	Low-power Consumption Mode Control Register	
IPCP			

(LPMCR)	137	(x)	506
Notes on Accessing the Low-power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode	157	Setting Configuration of Multi-level Message Buffer	510
Low-voltage		Message Buffer Control Registers	
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)	384	Message Buffer Control Registers	462
Low-voltage/CPU Operation Detection Reset Setting Register		Microcontroller	
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)	384	Connection of an Oscillator to the Microcontroller	103
LPMCR		Minimum Connection	
Low-power Consumption Mode Control Register (LPMCR)	137	Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)	581
Notes on Accessing the Low-power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode	157	Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)	579
LVRC		Mode Data	
Low Voltage/CPU Operating Detection Reset Control Register (LVRC)	382	Mode Data	162
LVRS		Status of Pins after Mode Data is Read	130
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)	384	Mode Fetch	
		Mode Fetch	125
M		Mode Pins	
Machine Clock		Mode Pins	124, 161
Machine Clock	99	Module Configuration	
Main Oscillation Timer Data Register		Module Configuration of 16-bit I/O Timer	208
Main Oscillation Timer Data Register(24-bit) (CUTR)	593	Multi-byte Data	
Master-slave Communication		Accessing Multi-byte Data	34
Master-slave Communication Function	445	Multi-byte Data Allocation	
MB90990 Series		Multi-byte Data Allocation in Memory Space	34
Features of MB90990 Series	2	Multi-level Message Buffer	
List of MB90990 Series Interrupt Vectors	675	Setting Configuration of Multi-level Message Buffer	510
Memory Access Modes		Multiple Interrupts	
Outline of Memory Access Modes	160	Multiple Interrupts	67
Memory Map		Multiplier	
E ² PROM Memory Map	528	Selection of a PLL Clock Multiplier	99
Memory Map	30		
System Configuration and E ² PROM Memory Map	527	N	
Memory Space		NCC	
Memory Space in Each Bus Mode	163	Flag Change Disable Prefix (NCC)	47
Multi-byte Data Allocation in Memory Space	34	Node Status	
Overview of CPU Memory Space	27	Correspondence between Node Status Bits (NS1 and NS0) and Node Status	467
Message Buffer		O	
List of Message Buffers (Data Registers)	461	Operating Detection Reset Circuit	
List of Message Buffers (DLC Registers)	460	Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit	380
List of Message Buffers (ID Registers)	458	Sample Program for Low Voltage/CPU Operating Detection Reset Circuit	389
Message Buffers	462, 491	Operating Mode	
Note for Disabling Message Buffers by BVAL Bits	513	CPU Intermittent Operating Mode	133
Procedure for Reception by Message Buffer (x)	508	CPU Operating Modes and Current Dissipation	132
Procedure for Transmission by Message Buffer		Operation Clock	
		Supply of Operation Clock	189

Operation Mode	
CPU Intermittent Operation Mode	140
Operation in Synchronous Mode	
(Operation Mode 2)	436
Operation Modes of 16-bit Reload Timer	236
Setting for 16-bit PPG Output Operation	
Mode	305
Setting for 8+8-bit PPG Output	
Operation Mode	308
Setting for 8-bit PPG Output 2-channel Independent	
Operation Mode	302
Operation Status	
Type of Standby Mode and Operation Status	141
Oscillating Clock Frequency	
Oscillating Clock Frequency and Serial Clock Input	
Frequency	574
Oscillation Stabilization Wait	
Operation of Oscillation Stabilization Wait Time	
.....	102
Oscillation Stabilization Wait Reset State	122
Oscillation Stabilization Wait Time	
Oscillation Stabilization Wait Time	156
Oscillation Stabilization Wait Time Timer of	
Sub clock	277
Reset Sources and Oscillation Stabilization Wait	
Times	121
Oscillator	
Connection of an Oscillator to the	
Microcontroller	103
Others	
Others	69
Overall Control Registers	
List of Overall Control Registers	456
Overall Control Registers	462
Overview	
Overview	570
P	
Package Dimensions	
Package Dimensions (LQFP-48)	11
PACSR	
Address Detection Control Register 0	
(PACSR0)	519
Address Detection Control Register 1	
(PACSR1)	521
PADR	
Detect Address Setting Registers	
(PADR0 to PADR5)	523
Patch Processing	
Flow of Patch Processing for Patch Program	531
Patch Program	
Flow of Patch Processing for Patch Program	531
PC	
Program Counter (PC)	43
PDR	
Port Data Register (PDR)	168

Pin	
Pin Functions	13
Pin Direct access	
LIN-UART Pin Direct Access	442
Pins	
Details of Pins and Interrupt Number	210
Mode Pins	161
Pins of 16-bit I/O Timer	214
Status of I/O Pins (Single-chip Mode)	154
Status of Pins after Mode Data is Read	130
Status of Pins during Reset	130
PLL Clock Multiplier	
Selection of a PLL Clock Multiplier	99
Port Data Register	
Port Data Register (PDR)	168
Port Direction Register	
Port Direction Register (DDR)	170
Power Supplied From Programmer	
Example of Serial Programming Connection	
(Power Supplied from Programmer)	577
PPG	
Channels and PPG Pins of PPG Timers	285
Setting for 16-bit PPG Output Operation	
Mode	305
Setting for 8+8-bit PPG Output	
Operation Mode	308
Setting for 8-bit PPG Output 2-channel Independent	
Operation Mode	302
PPG Reload Registers	
PPG Reload Registers (PRLLC/PRLHC,	
PRLLD/PRLHD)	299
PPG Timer	
Block Diagram of 8-/16-bit PPG Timer C	286
Block Diagram of 8-/16-bit PPG Timer D	289
Channels and PPG Pins of PPG Timers	285
Functions of 8-/16-bit PPG Timer	282
Generation of Interrupt Request from 8-/16-bit	
PPG Timer	292
Interrupts of 8-/16-bit PPG Timer	300
List of Registers and Initial Values of 8-/16-bit	
PPG Timer	292
Operation Modes of 8-/16-bit PPG Timer	283
Operation of 8-/16-bit PPG Timer	301
Pins of 8-/16-bit PPG Timer	291
PPGC Operation Mode Control Register	
PPGC Operation Mode Control Register	
(PPGCC)	293
PPGC/PPGD Count Clock Select Register	
PPGC/PPGD Count Clock Select Register	
(PPGCD)	297
PPGCC	
PPGC Operation Mode Control Register	
(PPGCC)	293
PPGCD	
PPGC/PPGD Count Clock Select Register	
(PPGCD)	297
PPGD Operation Mode Control Register	

(PPGCD).....	295	Read/reset State	561
PPGD Operation Mode Control Register		Receive Overrun	
PPGD Operation Mode Control Register		Receive Overrun	501
(PPGCD).....	295	Received Message	
Prefix		Storing Received Message	500
Bank Select Prefix	46	Reception	
Common Register Bank Prefix (CMR).....	47	Completing Reception	502
Flag Change Disable Prefix (NCC).....	47	Permission of Transmission and Reception	431
Restrictions on Interrupt Disable Instructions and		Procedure for Reception by Message Buffer	
Prefix Instructions	49	(x)	508
Prefix Instructions		Processing for Reception of Data Frame and	
Restrictions on Interrupt Disable Instructions and		Remote Frame	501
Prefix Instructions	49	Reception Flowchart of the CAN Controller	503
Prescaler Settings		Reception Data Register	
Prescaler Settings	473	Reception Data Register (RDR)	409
Processor Status		Reception Interrupt	
Processor Status (PS)	40	Reception Interrupt Generation and Flag Set	
Program Counter		Timing.....	419
Program Counter (PC).....	43	Register	
Program Example		Bit Configuration of Extended Communication	
Program Example for Address Match Detection		Control Register (ECCR).....	413
Function	532	Bit Configuration of Extended Status/Control Register	
Program Example in Event Count Mode	265	(ESCR)	411
Program Example in Internal Clock Mode	264	Calibration Unit Control Register (CUCR)	590
Program Example of 16-bit I/O Timer.....	232	Configuration of CR Oscillation Timer Data	
Program Example of Delayed Interrupt Generation		Register	594
Module	86	CR Oscillation Timer Data Register(16-bit)	
Program Example of DTP Function.....	336	(CUTD)	592
Program Example of DTP/External Interrupt		CR Oscillation Trimming Setting Register	
Function	335	(CRTR)	589
Program Example of Time-base Timer.....	191	D/A Control Register 0 (DACR0)	601
Program Example of Watch Timer	278	D/A Converter Register 0 (DAT0)	600
Program Example of Watchdog Timer	206	Flash Memory Write Control Register	
Program Execution		(FWR0/FWR1).....	548
Program Execution	526	Input Level Select Register (ILSR0, ILSR1)	174
PS		Interrupt Control Register (ICR00 to ICR15).....	57
Processor Status (PS)	40	Low-voltage/CPU Operation Detection Reset Setting	
PSCCR		Register (LVRS)	384
Configuration of the PLL/Sub clock Control Register		Main Oscillation Timer Data Register(24-bit)	
(PSCCR)	96	(CUTR)	593
PUCR2		Setup Flow of Flash Memory Write Control Register	
Block Diagram of Pull-up Control Register		(FWR0/FWR1).....	550
(PUCR2)	172	Register Bank	
Pull-up Control Register (PUCR2).....	172	Register Bank	44
Pull-up Control Register		Register Bank Pointer	
Block Diagram of Pull-up Control Register		Register Bank Pointer (RP)	41
(PUCR2)	172	Reload Counter	
Pull-up Control Register (PUCR2).....	172	Function of Reload Counter	428
R		Reload Counters	
RAM		Operation of Reload Counters	427
RAM Area	28	Reload Timer	
RDR		Block Diagram of 16-bit Reload Timer	238
Reception Data Register (RDR)	409	Correspondence between 16-bit Reload Timer	
Read		Interrupt and EI ² OS.....	249
Setting the Flash Memory to the		EI ² OS Function of 16-bit Reload Timer.....	249
		Generation of Interrupt Request from 16-bit	
		Reload Timer	242

Interrupts of 16-bit Reload Timer.....	249	Sampling Time	
List of 16-bit Reload Timer Registers		Sampling Time Setup (Bits ST2 to ST0).....	359
and Initial Value	241	SCR	
Notes on Using 16-bit Reload Timer	260	Serial Control Register (SCR)	403
Operation Modes of 16-bit Reload Timer	236	Sector Configuration	
Pins of 16-bit Reload Timer.....	240	Sector Configuration of the Flash Memory	543
Setting of 16-bit Reload Timer.....	250	Sector Erase	
Remote Frame		Restarting Sector Erase of Flash Memory.....	568
Processing for Reception of Data Frame and		Sector Erase Procedure for Flash Memory	565
Remote Frame.....	501	Suspending Sector Erase of Flash Memory	567
Reset		Sector Erase Timer Flag	
Block Diagrams of the External Reset Pin	123	Sector Erase Timer Flag (DQ3).....	559
Factors of Reset.....	118	Serial Clock	
List of 16-bit Reload Timer Registers		Oscillating Clock Frequency and Serial Clock Input	
and Initial Value	241	Frequency	574
List of Registers and Initial Values of 8-/10-bit		Serial Control Register	
A/D Converter	346	Serial Control Register (SCR)	403
Oscillation Stabilization Wait Reset State	122	Serial Programming Connection	
Overview of Reset Operation	124	Basic Configuration of Serial Programming	
Status of Pins during Reset.....	130	Connection	572
Reset Source		Example of Serial Programming Connection	
Correspondence between Reset Source Bits and Reset		(Power Supplied from Programmer)	577
Sources.....	127	Example of Serial Programming Connection	
Notes about Reset Source Bits	129	(User Power Supply Used).....	575
Reset Source Bits	126	Serial Status Register	
Reset Sources and Oscillation Stabilization Wait		Serial Status Register (SSR)	407
Times	121	Setting	
Status of Reset Source Bit and Low Voltage Detection		Setting for 16-bit PPG Output Operation	
Bit.....	128	Mode	305
Reset State		Signal Mode	
Setting the Flash Memory to the		Signal Mode.....	431
Read/reset State	561	Single Conversion Mode	
Restart		Operations and Applications of Single Conversion	
Restarting Sector Erase of Flash Memory	568	Mode	366
ROM Mirroring		Single Conversion Mode (ADCS: MD1 and	
Block Diagram of ROM Mirroring Function Select		MD0=00 _B or 01 _B).....	363
Module	536	Single Conversion Mode Setup	365
ROM Mirroring Function Select Module		Single-chip Mode	
Block Diagram of ROM Mirroring Function Select		Status of I/O Pins (Single-chip Mode)	154
Module	536	Sleep Mode	
List of Registers and Initial Values of ROM Mirroring		Return from Sleep Mode	144
Function Select Module	537	Switching to Sleep Mode	143
ROM Mirroring Function Select Register		SMR	
ROM Mirroring Function Select Register		LIN-UART Serial Mode Register (SMR)	405
(ROMM)	538	Software Interrupt	
ROMM		Software Interrupt Operation	68
ROM Mirroring Function Select Register		Software Interrupts	53, 68
(ROMM)	538	Structure of Software Interrupts.....	68
RP		Special Registers	
Register Bank Pointer (RP).....	41	Special Registers.....	35
S		SSP	
Sample Program		User Stack Pointer (USP) and System Stack Pointer	
Sample Program for Low Voltage/CPU Operating		(SSP).....	39
Detection Reset Circuit	389	SSR	
Sample Program for the D/A Converter	602	Serial Status Register (SSR)	407
		ST	

Sampling Time Setup (Bits ST2 to ST0).....	359	(TCCSL0)	216
Standby Mode		TCDT0	
Cancellation of Standby Mode by Interrupt	155	Timer Data Register (TCDT0).....	218
Note on Canceling Standby Mode.....	155	TDR	
Notes on Accessing the Low-power Consumption		Transmission Data Register (TDR).....	410
Mode Control Register (LPMCR) to Enter		Time-base Timer	
the Standby Mode	157	Block Diagram of Time-base Timer	180
Notes on the Transition to Standby Mode	155	Correspondence Between Time-base Timer Interrupt	
Standby Mode	133	and EI ² OS	185
Transition to Standby Mode	155	Generation of Interrupt Request from	
Type of Standby Mode and Operation Status	141	Time-base Timer	182
State		Interrupt of Time-base Timer	185
Setting the Flash Memory to the		List of Registers and Initial Values of	
Read/reset State	561	Time-base Timer	182
Status Bit		Notes on Using Time-base Timer	190
Correspondence between Node Status Bits (NS1 and		Program Example of Time-base Timer	191
NS0) and Node Status	467	Time-base Timer Control Register	
Status Change		Time-base Timer Control Register (TBTC).....	183
Status Change Diagram	153	Time-base Timer Mode	
Stop Conversion Mode		Return from Time-base Timer Mode.....	149
Operations and Applications of Stop Conversion		Switching to the Time-base Timer Mode	148
Mode	369	Timer Control Status Register	
Stop Conversion Mode (ADCS:MD1 and		Timer Control Status Register (Upper)	
MD0=11 _B)	363	(TCCSH0).....	215
Stop Conversion Mode Setup	369	Timer Control Status Register (Lower)	
Stop Mode		(TCCSL0)	216
Stop Mode	150	Timer Control Status Registers (Lower)	
Storing Patch Program		(TMCSR: L)	245
Operation of Address Match Detection Function		Timer Control Status Registers (Upper)	
at Storing Patch Program in E ² PROM		(TMCSR:H).....	243
.....	530	Timer Data Register	
Structure		Timer Data Register (TCDT0).....	218
Structure of Instruction Map	653	Timer Register	
Sub clock		Operating State of 16-bit Timer Register	251
Oscillation Stabilization Wait Time Timer of		Timing Limit Exceeded Flag	
Sub clock	277	Timing Limit Exceeded Flag (DQ5).....	558
Synchronization Methods		TMCSR	
Synchronization Methods	431	Timer Control Status Registers (Lower)	
Synchronous Mode		(TMCSR: L)	245
Operation in Synchronous Mode		Timer Control Status Registers (Upper)	
(Operation Mode 2).....	436	(TMCSR:H).....	243
System Configuration		TMR	
System Configuration and E ² PROM		16-bit Timer Registers (TMR)	247
Memory Map	527	TMRLR	
System Stack Pointer		16-bit Reload Registers (TMRLR)	248
User Stack Pointer (USP) and System Stack Pointer		Toggle Bit Flag	
(SSP)	39	Toggle Bit Flag (DQ6).....	557
T		Transition	
TBTC		Clock Mode Transition	98
Time-base Timer Control Register (TBTC).....	183	Notes on the Transition to Standby Mode	155
TCCSH0		Transition to Standby Mode	155
Timer Control Status Register (Upper)		Transmission	
(TCCSH0)	215	Canceling Transmission Request from	
TCCSL0		CAN Controller	498
Timer Control Status Register (Lower)		Permission of Transmission and Reception	431
		Procedure for Transmission by Message Buffer	

(x).....	506	Watch Timer Counter	276
Starting Transmission of CAN Controller	498	Watch Timer Interrupt.....	275
Transmission Flowchart of CAN Controller	499	Watch Timer Interrupt and EI ² OS Transfer Function	275
Transmission Data Register		Watch Timer Control Register	
Transmission Data Register (TDR)	410	Watch Timer Control Register (WTC).....	273
Transmission Interrupt		Watchdog Timer	
Transmission Interrupt Generation and Flag Set		Block Diagram of Watchdog Timer	196
Timing	421	Functions of Watchdog Timer	194
Transmission Interrupt Request Generation		List of Registers and Initial Values of Watchdog Timer	198
Timing	422	Notes on Using Watchdog Timer	204
		Operations of Watchdog Timer	202
		Program Example of Watchdog Timer	206
		Setting Operation Clock of Watchdog Timer	277
U		Watchdog timer control register	
UART		Watchdog Timer Control Register (WDTC)	199
Block Diagram of LIN-UART	396	WDTC	
Block Diagram of LIN-UART Pins	401	Watchdog Timer Control Register (WDTC)	199
LIN-UART as LIN Master Device	449	Writing	
LIN-UART Baud Rate Selection	423	Detailed Explanation of Writing to and Erasing Flash Memory	560
LIN-UART Functions	392	Writing to/Erasing Flash Memory	
LIN-UART Interrupts.....	416	Writing to/Erasing Flash Memory	540
LIN-UART Interrupts and EI ² OS	418	WTC	
LIN-UART Pin Direct Access.....	442	Watch Timer Control Register (WTC).....	273
LIN-UART Pins.....	400		
LIN-UART Registers.....	402		
LIN-UART Serial Mode Register (SMR).....	405		
Notes on Using LIN-UART	451		
Operation of LIN-UART	430		
Undefined Instruction			
Exception due to Execution of an Undefined Instruction	78		
Execution of an Undefined Instruction.....	78		
Underflow			
Operation as 16-bit Timer Register			
Underflows	253, 258		
Operation at Underflow	237		
User Power Supply			
Example of Serial Programming Connection (User Power Supply Used)	575		
User Stack Pointer			
User Stack Pointer (USP) and System Stack Pointer (SSP)	39		
USP			
User Stack Pointer (USP) and System Stack Pointer (SSP)	39		
W			
Watch Mode			
Return from Watch Mode	147		
Switching to the Watch Mode.....	146		
Watch Timer			
Block Diagram of Watch Timer.....	270		
Generation of Interrupt Request from Watch Timer	272		
List of Registers and Initial Values of Watch Timer	272		
Program Example of Watch Timer	278		

Numerics



16-bit Free-run Timer		Block Diagram of 8-/10-bit A/D Converter	344
Block Diagram of 16-bit Free-run Timer	213	Conversion Modes of 8-/10-bit A/D Converter	343
Explanation of Operation of 16-bit Free-run Timer	229	El ² OS of 8-/10-bit A/D Converter	366
16-bit I/O Timer		Explanation of A/D Conversion Data Protection Function of 8-/10-bit A/D Converter	376
16-bit I/O Timer Interrupt and El ² OS	228	Functions of 8-/10-bit A/D Converter	342
Block Diagram of 16-bit I/O Timer	211	Interrupts, El ² OS of 8-/10-bit A/D Converter	366
Functions of 16-bit I/O Timer	210	List of Registers and Initial Values of 8-/10-bit A/D Converter	348
Generation of Interrupt Request from 16-bit I/O Timer	216	Notes on Using 8-/10-bit A/D Converter	380
Interrupts of 16-bit I/O Timer	227	Pins of 8-/10-bit A/D Converter	347
Module Configuration of 16-bit I/O Timer	210	8-/16-bit PPG Timer	
Pins of 16-bit I/O Timer	216	Block Diagram of 8-/16-bit PPG Timer C	288
Program Example of 16-bit I/O Timer	234	Block Diagram of 8-/16-bit PPG Timer D	291
16-bit PPG Output Operation Mode		Functions of 8-/16-bit PPG Timer	284
Setting for 16-bit PPG Output Operation Mode	307	Generation of Interrupt Request from 8-/16-bit PPG Timer	294
16-bit Reload Registers		Interrupts of 8-/16-bit PPG Timer	302
16-bit Reload Registers (TMRLR)	250	List of Registers and Initial Values of 8-/16-bit PPG Timer	294
16-bit Reload Timer		Operation Modes of 8-/16-bit PPG Timer	285
Block Diagram of 16-bit Reload Timer	240	Operation of 8-/16-bit PPG Timer	303
Correspondence between 16-bit Reload Timer Interrupt and El ² OS	251	Pins of 8-/16-bit PPG Timer	293
El ² OS Function of 16-bit Reload Timer	251	8-bit PPG Output 2-channel Independent Operation Mode	
Generation of Interrupt Request from 16-bit Reload Timer	244	Setting for 8-bit PPG Output 2-channel Independent Operation Mode	304
Interrupts of 16-bit Reload Timer	251		
List of 16-bit Reload Timer Registers and Initial Value	243		
Notes on Using 16-bit Reload Timer	262		
Operation Modes of 16-bit Reload Timer	238		
Pins of 16-bit Reload Timer	242		
Setting of 16-bit Reload Timer	252		
16-bit Timer Register			
16-bit Timer Registers (TMR)	249		
Operating State of 16-bit Timer Register	253		
Operation as 16-bit Timer Register Underflows	255, 260		
24-bit Operand			
24-bit Operand Specification	33		
512K-bit Flash Memory			
Features of Flash Memory	544		
8+8-bit PPG			
Setting for 8+8-bit PPG Output Operation Mode	310		
8-/10-bit A/D Converter			

A		
A		
Accumulator (A).....	40	
A/D Control Status Register		
A/D Control Status Register 0 (ADCS0).....	354	
A/D Control Status Register 1 (ADCS1).....	350	
A/D Converter		
Interrupts of A/D Converter	366	
A/D Data Registers		
A/D Data Registers (ADCR0/ADCR1).....	357	
A/D Setting Registers		
A/D Setting Registers (ADSR0/ADSR1).....	358	
Acceptance Filter		
Setting Acceptance Filter	508	
Acceptance Filtering		
Acceptance Filtering.....	504	
Accessing		
Accessing Multi-byte Data	36	
Accumulator		
Accumulator (A).....	40	
ADCR		
A/D Data Registers (ADCR0/ADCR1).....	357	
ADCS		
A/D Control Status Register 0 (ADCS0).....	354	
A/D Control Status Register 1 (ADCS1).....	350	
Continuous Conversion Mode (ADCS: MD1 and MD0=10 _B)	367	
Single Conversion Mode (ADCS: MD1 and MD0=00 _B or 01 _B).....	367	
Stop Conversion Mode (ADCS: MD1 and MD0=11 _B)	367	
Address Detection Control Register		
Address Detection Control Register 0 (PACSR0)	523	
Address Detection Control Register 1 (PACSR1)	525	
Address Match Detection		
Block Diagram of Address Match Detection Function	521	
List of Registers and Initial Values of Address Match Detection Function	522	
Operation of Address Match Detection Function	530	
Operation of Address Match Detection Function at Storing Patch Program in E ² PROM	534	
Overview of Address Match Detection Function	520	
Program Example for Address Match Detection Function	536	
Addressing 624		
Indirect Addressing	632	
ADER		
Analog Input Enable Registers (ADER)	175	
Analog Input Enable Registers (ADER5, ADER6)	364	
ADSR		
A/D Setting Registers (ADSR0/ADSR1).....	358	
Alternative Mode		
Alternative Mode.....	548	
Analog Input Enable Register		
Analog Input Enable Registers (ADER).....	175	
Analog Input Enable Registers (ADER5, ADER6)	364	
Asynchronous LIN Mode		
Operation in Asynchronous LIN Mode.....	443	
Asynchronous Mode		
Operation in Asynchronous Mode	436	
B		
Bank Addressing		
Bank Addressing Types.....	34	
Bank Select Prefix		
Bank Select Prefix	48	
BAP		
Buffer Address Pointer (BAP)	75	
Basic Configuration		
Basic Configuration of Serial Programming Connection	578	
Baud Rate		
Calculating the Baud Rate	429	
LIN-UART Baud Rate Selection	427	
Baud Rate Generator Register		
Baud Rate Generator Register (BGR0/BGR1)	419	
BGR		
Baud Rate Generator Register (BGR0/BGR1)	419	
Bidirectional Communication		
Bidirectional Communication Function	447	
Bit Timing		
Setting Bit Timing	508	
Block Diagram		
Block Diagram of 16-bit Free-run Timer	213	
Block Diagram of 16-bit I/O Timer	211	
Block Diagram of 16-bit Reload Timer.....	240	
Block Diagram of 8-/10-bit A/D Converter	344	
Block Diagram of 8-/16-bit PPG Timer C.....	288	
Block Diagram of 8-/16-bit PPG Timer D.....	291	
Block Diagram of Address Match Detection Function	521	
Block Diagram of CAN Controller	459	
Block Diagram of Clock Supervisor	109	
Block Diagram of D/A Converter	605	
Block Diagram of Delayed Interrupt Generation Module	83	
Block Diagram of DTP/External Interrupt	317	
Block Diagram of Evaluation Chip	8	
Block Diagram of Flash/Mask ROM Version	10	
Block Diagram of Input Capture	214	
Block Diagram of LIN-UART	400	
Block Diagram of LIN-UART Pins.....	405	

Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit	384	Configuration of the Clock Selection Register (CKSCR)	96
Block Diagram of Pull-up Control Register (PUCR2)	174	Clock Calibration Unit	
Block Diagram of ROM Mirroring Function Select Module	540	Block Diagram of the Clock Calibration Unit	593
Block Diagram of the Clock Calibration Unit	593	Clock Timing of Clock Calibration Unit	594
Block Diagram of the Clock Generation Block	93	Timing of Clock Calibration Unit	594
Block Diagram of the D/A Converter Pin	606	Clock Frequency	
Block Diagram of the Entire Flash Memory	546	Oscillating Clock Frequency and Serial Clock Input Frequency	580
Block Diagram of the Low-power Consumption Circuit	137	Clock Generation Block	
Block Diagram of Time-base Timer	182	Block Diagram of the Clock Generation Block	93
Block Diagram of Watch Timer	272	List of Register of Clock Generation Block and Initial Value	95
Block Diagram of Watchdog Timer	198	Clock Mode	
Block Diagrams of the External Reset Pin	125	Clock Mode	101
Buffer Address Pointer		Clock Mode Switching	158
Buffer Address Pointer (BAP)	75	Clock Mode Transition	101
Bus Mode		Internal Clock Mode	238
Memory Space in Each Bus Mode	165	Operation in Internal Clock Mode	255
Bus Operation Stop		Program Example in Internal Clock Mode	266
Conditions for Canceling Bus Operation Stop (HALT=0)	472	Setting of Internal Clock Mode	254
Conditions for Setting Bus Operation Stop (HALT=1)	472	Clock Selection Register	
State during Bus Operation Stop (HALT=1)	472	Configuration of the Clock Selection Register (CKSCR)	96
BVAL		Clock Supervisor	
Note for Disabling Message Buffers by BVAL Bits	517	Block Diagram of Clock Supervisor	109
C		Clock Supervisor Control Register (CSVCR)	112
Calculating the Execution Cycle Count	641	Clock Supervisor Register	111
Calibration Unit		Example of Operation Flowchart for the Clock Supervisor	115
Calibration Unit Control Register (CUCR)	598	Example of Startup Flowchart when using the Clock Supervisor	116
Features of the Calibration Unit	592	Notes on using the Clock Supervisor	117
Register List of the Calibration Unit	595	Operations of Clock Supervisor	114
Calibration Unit Control Register		Overview of Clock Supervisor	108
Calibration Unit Control Register (CUCR)	598	Clock Supply	
CAN Controller		Cycle of Clock Supply	271
Block Diagram of CAN Controller	459	Clocks	
Canceling Transmission Request from CAN Controller	502	Clocks	90
Features of CAN Controller	458	CMR	
Reception Flowchart of the CAN Controller	507	Common Register Bank Prefix (CMR)	49
Starting Transmission of CAN Controller	502	Command	
Transmission Flowchart of CAN Controller	503	Notes on Issuing Commands	557
CAN Direct Mode Register		Command Sequence	
CAN Direct Mode Register (CDMR)	516	Command Sequence Table	556
CCR		Common Register Bank Prefix	
Condition Code Register (CCR)	42	Common Register Bank Prefix (CMR)	49
CDMR		Communication	
CAN Direct Mode Register (CDMR)	516	Bidirectional Communication Function	447
Chip Erase		LIN-master-slave Communication Function	452
Notes on Chip Erase	570	Master-slave Communication Function	449
CKSCR		Compare Time	

Compare Time Setup (Bits CT2 to CT0)	362
Condition Code Register	
Condition Code Register (CCR)	42
Configuration of the PLL/Sub clock Control Register	
Configuration of the PLL/Sub clock Control Register (PSCCR)	99
Continuous Conversion Mode	
Continuous Conversion Mode (ADCS: MD1 and MD0=10 _B)	367
Continuous Conversion Mode Setup	371
Operations and Applications of Continuous Conversion Mode	371
Control Status Register	
Control Status Register (CSR) (Lower)	467
Control Status Register (CSR) (Upper)	467
Control Status Register (CSR-lower)	468
Conversion Modes	
Conversion Modes of 8-/10-bit A/D Converter	343
CPU	
Overview of CPU Memory Space	29
Overview of the CPU	28
CPU Intermittent Operating Mode	
CPU Intermittent Operating Mode	135
CPU Intermittent Operation Mode	
CPU Intermittent Operation Mode	142
CPU Operating Detection Reset Circuit	
Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit	384
CPU Operating Detection Reset Circuit	383
Notes on Using CPU Operating Detection Reset Circuit	392
Operating of CPU Operating Detection Reset Circuit	390
Sample Program for Low Voltage/CPU Operating Detection Reset Circuit	393
CPU Operating Modes	
CPU Operating Modes and Current Dissipation	134
CPU Operation Detection	
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)	388
CR Oscillation Timer Data Register	
Configuration of CR Oscillation Timer Data Register	602
CR Oscillation Timer Data Register(16-bit) (CUTD)	600
CR Oscillation Trimming Setting Register	
CR Oscillation Trimming Setting Register (CRTR)	597
CRTR	
CR Oscillation Trimming Setting Register (CRTR)	597
CSR	
Control Status Register (CSR) (Lower)	467
Control Status Register (CSR) (Upper)	467

Control Status Register (CSR-lower)	468
CSVCR	
Clock Supervisor Control Register (CSVCR)	112
CT	
Compare Time Setup (Bits CT2 to CT0)	362
CUCRr	
Calibration Unit Control Register (CUCR)	598
Current Dissipation	
CPU Operating Modes and Current Dissipation	134
CUTD	
CR Oscillation Timer Data Register(16-bit) (CUTD)	600
CUTR	
Main Oscillation Timer Data Register(24-bit) (CUTR)	601

D

D/A Control Register	
D/A Control Register 0 (DACR0)	609
D/A Converter	
Block Diagram of D/A Converter	605
Block Diagram of the D/A Converter Pin	606
D/A Converter Pins	606
D/A Converter Registers	607
Function and Operation of the D/A Converter	604
Sample Program for the D/A Converter	610
D/A Converter Register	
D/A Converter Register 0 (DAT0)	608
DACR	
D/A Control Register 0 (DACR0)	609
DAT	
D/A Converter Register 0 (DAT0)	608
Data Counter	
Data Counter (DCT)	74
Data Frame	
Processing for Reception of Data Frame and Remote Frame	505
Data Polling Flag	
Data Polling Flag (DQ7)	560
Data Protection Function	
Explanation of A/D Conversion Data Protection Function of 8-/10-bit A/D Converter	376
Data Register	
List of Message Buffers (Data Registers)	465
List of Message Buffers (DLC Registers)	464
DCT	
Data Counter (DCT)	74
DDR	
Port Direction Register (DDR)	172

Delayed Interrupt Generation Module	
Block Diagram of Delayed Interrupt Generation Module	83
Explanation of Operation of Delayed Interrupt Generation Module	86
Notes on Using Delayed Interrupt Generation Module	87
Overview of Delayed Interrupt Generation Module	82
Program Example of Delayed Interrupt Generation Module	88
Delayed Interrupt Request Generate/Cancel Register	
Delayed Interrupt Request Generate/Cancel Register (DIRR)	85
Description of Instruction Presentation Items and Symbols	644
Descriptor	
Extended Intelligent I/O Service Descriptor (ISD)	74
Detect Address	
Setting Detect Address	530
Detect Address Setting Registers	
Detect Address Setting Registers (PADR0 to PADR5)	527
Functions of Detect Address Setting Registers	528
Detection Level Setting Register	
Detection Level Setting Register 1 (ELVR1)	325
Detection Reset Circuit	
Operating of Low Voltage Detection Reset Circuit	390
Device	
Notes on Handling the Device	21
Direct Addressing	626
Direct Pin access	
LIN-UART Direct Pin Access	446
DIRR	
Delayed Interrupt Request Generate/Cancel Register (DIRR)	85
DLC Registers	
List of Message Buffers (DLC Registers)	464
DQ2	
Toggle Bit-2 Flag (DQ2)	564
DQ3	
Sector Erase Timer Flag (DQ3)	563
DQ5	
Timing Limit Exceeded Flag (DQ5)	562
DQ6	
Toggle Bit Flag (DQ6)	561
DQ7	
Data Polling Flag (DQ7)	560
DTP	
DTP Function	334
Program Example of DTP Function	338
DTP/External Interrupt	
Block Diagram of DTP/External Interrupt	317
DTP/External Interrupt Function	316
DTP/External Interrupt Operation	331
List of Registers and Initial Values in DTP/External Interrupt	320
Notes on Using DTP/External Interrupt	335
Pins of DTP/External Interrupt	319
Program Example of DTP/External Interrupt Function	337
Setting of DTP/External Interrupt	329
DTP/External Interrupt Enable Register 1	
DTP/External Interrupt Enable Register 1 (ENIR1)	323
DTP/External Interrupt Source Register 1	
DTP/External Interrupt Source Register 1 (EIRR1)	321
E	
E ² PROM	
E ² PROM Memory Map	532
Operation of Address Match Detection Function at Storing Patch Program in E ² PROM	534
System Configuration and E ² PROM Memory Map	531
ECCR	
Bit Configuration of Extended Communication Control Register (ECCR)	417
Effective Address Field	625, 643
EI ² OS	
16-bit I/O Timer Interrupt and EI ² OS	228
Conversion Operation by EI ² OS Function	375
Correspondence between 16-bit Reload Timer Interrupt and EI ² OS	251
Correspondence Between Time-base Timer Interrupt and EI ² OS	187
Correspondence to EI ² OS Function	228
EI ² OS Function of 16-bit Reload Timer	251
EI ² OS of 8-/10-bit	
A/D Converter	366
EI ² OS Operation Flow	77
Extended Intelligent I/O Service (EI ² OS)	55, 72
Interrupts, EI ² OS of 8-/10-bit A/D Converter	366
LIN-UART Interrupts and EI ² OS	422
Structure of Extended Intelligent I/O Service (EI ² OS)	73
EI ² OS Status Register	
EI ² OS Status Register (ISCS)	75
EIRR	
DTP/External Interrupt Source Register 1 (EIRR1)	321
ELVR	
Detection Level Setting Register 1 (ELVR1)	325
ENIR	
DTP/External Interrupt Enable Register 1 (ENIR1)	323

Erase	
Suspending Sector Erase of Flash Memory	573
Erasing	
Detailed Explanation of Writing to and Erasing Flash Memory	566
Erasing All Data in the Flash Memory (Erasing Chips)	570
Erasing Optional Data in the Flash Memory (Erasing Sectors)	571
Erasing Chip	
Erasing All Data in the Flash Memory (Erasing Chips)	570
ESCR	
Bit Configuration of Extended Status/Control Register (ESCR)	415
Evaluation Chip	
Block Diagram of Evaluation Chip	8
Event Count Mode	
Event Count Mode	238
Operation in Event Count Mode	261
Program Example in Event Count Mode	267
Setting of Event Count Mode	259
Exceptions	
Exceptions	56
Execution Cycle Count	640
Explanation	
Explanation of A/D Conversion Data Protection Function of 8-/10-bit A/D Converter	376
Extended Communication Control Register	
Bit Configuration of Extended Communication Control Register (ECCR)	417
Extended Intelligent I/O Service	
Extended Intelligent I/O Service (EI ² OS)	55, 72
Extended Intelligent I/O Service Descriptor (ISD)	74
Structure of Extended Intelligent I/O Service (EI ² OS)	73
Extended Status/Control Register	
Bit Configuration of Extended Status/Control Register (ESCR)	415
External Clock	
Connection of an Oscillator or an External Clock to the Microcontroller	106
External Interrupt	
Block Diagram of DTP/External Interrupt	317
DTP/External Interrupt Enable Register 1 (ENIR1)	323
DTP/External Interrupt Function	316
DTP/External Interrupt Operation	331
DTP/External Interrupt Source Register 1 (EIRR1)	321
External Interrupt Function	333
List of Registers and Initial Values in DTP/External Interrupt	320
Notes on Using DTP/External	

Interrupt	335
Pins of DTP/External Interrupt	319
Program Example of DTP/External Interrupt Function	337
Selection of External Interrupt Source	327
Setting of DTP/External Interrupt	329
External Reset	
Block Diagrams of the External Reset Pin	125

F

F ² MC-16LX Instruction List	647
Flag Change Disable Prefix	
Flag Change Disable Prefix (NCC)	49
Flag Set Timing	
Reception Interrupt Generation and Flag Set Timing	423
Transmission Interrupt Generation and Flag Set Timing	425
Flash	
Block Diagram of Flash/Mask ROM Version	10
Flash Memory	
Block Diagram of the Entire Flash Memory	546
Detailed Explanation of Writing to and Erasing Flash Memory	566
Erasing All Data in the Flash Memory (Erasing Chips)	570
Erasing Optional Data in the Flash Memory (Erasing Sectors)	571
Features of Flash Memory	544
Notes on Using Flash Memory	575
Restarting Sector Erase of Flash Memory	574
Sector Configuration of the Flash Memory	547
Sector Erase Procedure for Flash Memory	571
Setting the Flash Memory to the Read/reset State	567
Suspending Sector Erase of Flash Memory	573
Writing Data to the Flash Memory	568
Writing to the Flash Memory	568
Writing to/Erasing Flash Memory	544
Flash Memory Control Status Register	
Flash Memory Control Status Register (FMCS)	549
Flash Memory Mode	
Flash Memory Mode	548
Flash Memory Write Control Register	
Flash Memory Write Control Register (FWR0/FWR1)	552
Setup Flow of Flash Memory Write Control Register (FWR0/FWR1)	554
Flash Microcontroller Programmer	
Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)	587
Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)	585

Flash Security		ICE	
How to Disable the Flash Security Feature	576	Input Capture Edge Register (ICE)	224
How to Enable the Flash Security Feature	576	ICR	
FMCS		Interrupt Control Register (ICR00 to ICR15)	59
Flash Memory Control Status Register (FMCS)		ICS	
.....	549	Input Capture Control Status Registers (ICS)	
Setting FMCS:WE	555	221
Frame Format		ID	
Setting Frame Format	508	Setting ID	508
Free-run Timer		ID Registers	
Block Diagram of 16-bit Free-run Timer	213	List of Message Buffers (ID Registers)	462
Explanation of Operation of 16-bit Free-run Timer		ILSR	
.....	229	Input Level Select Register (ILSR0, ILSR1)	176
FWR		Indirect Addressing632	
Flash Memory Write Control Register (FWR0/FWR1)		Indirect Addressing	632
.....	552	Initial Value	
Setup Flow of Flash Memory Write Control Register		List of Registers and Initial Values in DTP/External	
(FWR0/FWR1)	554	Interrupt	320
H		Input Capture	
HALT		Block Diagram of Input Capture	214
Conditions for Canceling Bus Operation Stop		Explanation of Operation of Input Capture	231
(HALT=0)	472	Input Capture Control Status Registers	
Conditions for Setting Bus Operation Stop (HALT=1)		Input Capture Control Status Registers (ICS)	
.....	472	221
State during Bus Operation Stop (HALT=1)	472	Input Capture Edge Register	
Hardware Interrupt		Input Capture Edge Register (ICE)	224
Hardware Interrupt Operation	66	Input Capture Register	
Hardware Interrupts	54, 65	Input Capture Register (IPCP)	223
Occurrence and Release of Hardware Interrupt		Input Level Select Register	
.....	67	Input Level Select Register (ILSR0, ILSR1)	176
Structure of Hardware Interrupt	65	Input-output Circuits	
Hardware Sequence Flags		Input-output Circuits	16
Hardware Sequence Flags	558	Instruction	
I		Description of Instruction Presentation Items and	
I/O		Symbols	644
I/O Area	30	Exception due to Execution of an Undefined	
I/O Maps		Instruction	80
I/O Maps (Addresses:000000 _H to 0000FF _H)	612	Execution of an Undefined Instruction	80
I/O Pins		Interrupt Disable Instructions	51
Status of I/O Pins (Single-chip Mode)	156	Restrictions on Interrupt Disable Instructions and	
I/O Port		Prefix Instructions	51
I/O Port Registers	169	Instruction Types623	
Overview of I/O Ports	168	Inter-CPU Connection	
I/O Timer		Inter-CPU Connection Method	435
16-bit I/O Timer Interrupt and EI ² OS	228	Internal Clock Mode	
Block Diagram of 16-bit I/O Timer	211	Internal Clock Mode	238
Functions of 16-bit I/O Timer	210	Operation in Internal Clock Mode	255
Generation of Interrupt Request from 16-bit		Program Example in Internal Clock Mode	266
I/O Timer	216	Setting of Internal Clock Mode	254
Interrupts of 16-bit I/O Timer	227	Interrupt	
Module Configuration of 16-bit I/O Timer	210	16-bit I/O Timer Interrupt and EI ² OS	228
Pins of 16-bit I/O Timer	216	Cancellation of Standby Mode by Interrupt	157
Program Example of 16-bit I/O Timer	234	Correspondence between 16-bit Reload Timer	
		Interrupt and EI ² OS	251
		Correspondence Between Time-base Timer Interrupt	

and EI ² OS	187	Interrupt Vector	
Hardware Interrupt Operation.....	66	Interrupt Sources, Interrupt Vectors, and Interrupt	
Hardware Interrupts.....	54, 65	Control Registers	685
Interrupt Disable Instructions.....	51	Interrupt Vector.....	57
Interrupt Flow	63	List of Interrupt Vectors	70
Interrupt Number	83	List of MB90990 Series Interrupt Vectors.....	683
Interrupt of Time-base Timer.....	187	Interval Timer	
Interrupts of 16-bit Reload Timer.....	251	Interval Timer Function.....	180, 188, 270, 279
Interrupts of 8-/16-bit PPG Timer	302	IPCP	
Interrupts of A/D Converter	366	Input Capture Register (IPCP).....	223
Interrupts, EI ² OS of 8-/10-bit A/D Converter.....	366	ISCS	
LIN-UART Interrupts.....	420	EI ² OS Status Register (ISCS).....	75
LIN-UART Interrupts and EI ² OS.....	422	ISD	
Multiple Interrupts.....	69	Extended Intelligent I/O Service Descriptor (ISD)	
Occurrence and Release of Hardware Interrupt.....	67	74
Reception Interrupt Generation and Flag Set Timing			
.....	423	L	
Restrictions on Interrupt Disable Instructions and		Last Event Indicator Register	
Prefix Instructions	51	Register Configuration	473
Software Interrupt Operation.....	70	LEIR	
Software Interrupts	55, 70	Register Configuration	473
Structure of Hardware Interrupt.....	65	LIN Master Device	
Structure of Software Interrupts	70	LIN-UART as LIN Master Device.....	453
Transmission Interrupt Generation and Flag Set		LIN-master-slave Communication	
Timing	425	LIN-master-slave Communication Function	452
Transmission Interrupt Request Generation Timing		LIN-UART	
.....	426	Block Diagram of LIN-UART.....	400
Watch Timer Interrupt.....	277	Block Diagram of LIN-UART Pins.....	405
Watch Timer Interrupt and EI ² OS Transfer		Configuration of LIN-UART.....	399
Function	277	LIN-UART as LIN Master Device.....	453
Interrupt Control Register		LIN-UART Baud Rate Selection	427
Interrupt Control Register (ICR00 to ICR15).....	59	LIN-UART Direct Pin Access.....	446
		LIN-UART Functions	396
Interrupt Sources, Interrupt Vectors, and Interrupt		LIN-UART Interrupts.....	420
Control Registers	685	LIN-UART Interrupts and EI ² OS	422
Interrupt Disable Instructions		LIN-UART Pins	404
Interrupt Disable Instructions.....	51	LIN-UART Registers.....	406
Restrictions on Interrupt Disable Instructions and		Notes on Using LIN-UART	455
Prefix Instructions	51	Operation of LIN-UART	434
Interrupt Number		LIN-UART Serial Mode Register	
Details of Pins and Interrupt Number	212	LIN-UART Serial Mode Register (SMR)	409
Details of Pins and Interrupt Numbers	318	Low Voltage	
Interrupt Number	83	Block Diagram of Low Voltage/CPU Operating	
Interrupt Request		Detection Reset Circuit	384
Generation of Interrupt Request from 16-bit I/O Timer		Operating of Low Voltage Detection Reset Circuit	
.....	216	390
Generation of Interrupt Request from 16-bit		Sample Program for Low Voltage/CPU Operating	
Reload Timer	244	Detection Reset Circuit	393
Generation of Interrupt Request from 8-/16-bit		Low Voltage Detection	
PPG Timer	294	Status of Reset Source Bit and Low Voltage Detection	
Generation of Interrupt Request from Time-base		Bit.....	130
Timer.....	184	Low Voltage Detection Reset Circuit	
Generation of Interrupt Request from Watch Timer		Low Voltage Detection Reset Circuit.....	382
.....	274	Notes on Using Low Voltage Detection Reset Circuit	
Interrupt Sources		391
Interrupt Sources, Interrupt Vectors, and Interrupt			
Control Registers	685		

Low Voltage/CPU Operating Detection Reset Control Register	Memory Map.....32
Low Voltage/CPU Operating Detection Reset Control Register (LVRC).....386	System Configuration and E ² PROM Memory Map.....531
Low-power Consumption	Memory Space
Block Diagram of the Low-power Consumption Circuit.....137	Memory Space in Each Bus Mode165
Setting Low-power Consumption Mode.....509	Multi-byte Data Allocation in Memory Space36
Low-power Consumption Mode Control Register	Overview of CPU Memory Space29
Low-power Consumption Mode Control Register (LPMCR)139	Message Buffer
Notes on Accessing the Low-power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode159	List of Message Buffers (Data Registers)465
Low-voltage	List of Message Buffers (DLC Registers)464
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS).....388	List of Message Buffers (ID Registers)462
Low-voltage/CPU Operation Detection Reset Setting Register	Message Buffers466, 495
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS).....388	Note for Disabling Message Buffers by BVAL Bits517
LPMCR	Procedure for Reception by Message Buffer (x)512
Low-power Consumption Mode Control Register (LPMCR)139	Procedure for Transmission by Message Buffer (x)510
Notes on Accessing the Low-power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode159	Setting Configuration of Multi-level Message Buffer514
LVRC	Message Buffer Control Registers
Low Voltage/CPU Operating Detection Reset Control Register (LVRC).....386	Message Buffer Control Registers466
LVRS	Microcontroller
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS).....388	Connection of an Oscillator or an External Clock to the Microcontroller106
M	Minimum Connection
Machine Clock	Example of Minimum Connection to Flash
Machine Clock102	Microcontroller Programmer
Main Oscillation Timer Data Register	(Power Supplied from Programmer).....587
Main Oscillation Timer Data Register(24-bit) (CUTR)601	Example of Minimum Connection to Flash
Mask ROM	Microcontroller Programmer
Block Diagram of Flash/Mask ROM Version10	(User Power Supply Used).....585
Master-slave Communication	Mode Data
Master-slave Communication Function449	Mode Data164
MB90990 Series	Status of Pins after Mode Data is Read.....132
Features of MB90990 Series.....2	Mode Fetch
List of MB90990 Series Interrupt Vectors.....683	Mode Fetch.....127
MB90F362	Mode Pins
Basic Configuration of Serial Programming	Mode Pins126, 163
Connection.....578	Module Configuration
MB90V340E	Module Configuration of 16-bit I/O Timer.....210
CAN Direct Mode Register (CDMR).....516	Multi-byte Data
Memory Access Modes	Accessing Multi-byte Data36
Outline of Memory Access Modes.....162	Multi-byte Data Allocation
Memory Map	Multi-byte Data Allocation in Memory Space36
E ² PROM Memory Map.....532	Multi-level Message Buffer
	Setting Configuration of Multi-level Message Buffer514
	Multiple Interrupts
	Multiple Interrupts69
	Multiplier
	Selection of a PLL Clock Multiplier102

N		
NCC		
Flag Change Disable Prefix (NCC)	49	
Node Status		
Correspondence between Node Status Bits (NS1 and NS0) and Node Status.....	471	
O		
Operating Detection Reset Circuit		
Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit	384	
Sample Program for Low Voltage/CPU Operating Detection Reset Circuit	393	
Operating Mode		
CPU Intermittent Operating Mode	135	
CPU Operating Modes and Current Dissipation	134	
Operation Clock		
Supply of Operation Clock.....	191	
Operation Mode		
CPU Intermittent Operation Mode	142	
Operation in Synchronous Mode (Operation Mode 2).....	440	
Operation Modes of 16-bit Reload Timer	238	
Setting for 16-bit PPG Output Operation Mode	307	
Setting for 8+8-bit PPG Output Operation Mode	310	
Setting for 8-bit PPG Output 2-channel Independent Operation Mode	304	
Operation Status		
Type of Standby Mode and Operation Status	143	
Oscillating Clock Frequency		
Oscillating Clock Frequency and Serial Clock Input Frequency.....	580	
Oscillation Stabilization Wait		
Operation of Oscillation Stabilization Wait Time	105	
Oscillation Stabilization Wait Reset State	124	
Oscillation Stabilization Wait Time		
Oscillation Stabilization Wait Time	158	
Oscillation Stabilization Wait Time Timer of Sub clock	279	
Reset Sources and Oscillation Stabilization Wait Times	123	
Oscillator		
Connection of an Oscillator or an External Clock to the Microcontroller	106	
Others		
Others	71	
Overall Control Registers		
List of Overall Control Registers.....	460	
		Overall Control Registers
		466
		Overview
		Overview.....
		576
P		
Package Dimensions		
Package Dimensions (LQFP-48).....	11	
PACSR		
Address Detection Control Register 0 (PACSR0)	523	
Address Detection Control Register 1 (PACSR1)	525	
PADR		
Detect Address Setting Registers (PADR0 to PADR5)	527	
Patch Processing		
Flow of Patch Processing for Patch Program.....	535	
Patch Program		
Flow of Patch Processing for Patch Program.....	535	
PC		
Program Counter (PC).....	45	
PDR		
Port Data Register (PDR)	170	
Pin		
Pin Functions.....	13	
Pins		
Details of Pins and Interrupt Number	212	
Mode Pins.....	163	
Pins of 16-bit I/O Timer.....	216	
Status of I/O Pins (Single-chip Mode)	156	
Status of Pins after Mode Data is Read	132	
Status of Pins during Reset.....	132	
PLL Clock Multiplier		
Selection of a PLL Clock Multiplier	102	
Port Data Register		
Port Data Register (PDR)	170	
Port Direction Register		
Port Direction Register (DDR)	172	
Power Supplied From Programmer		
Example of Serial Programming Connection (Power Supplied from Programmer)	583	
PPG		
Channels and PPG Pins of PPG Timers	287	
Setting for 16-bit PPG Output Operation Mode	307	
Setting for 8+8-bit PPG Output Operation Mode	310	
Setting for 8-bit PPG Output 2-channel Independent Operation Mode	304	
PPG Reload Registers		
PPG Reload Registers (PRLLC/PRLHC,PRLLD/PRLHD)	301	
PPG Timer		
Block Diagram of 8-/16-bit PPG Timer C.....	288	

Block Diagram of 8-/16-bit PPG Timer D.....	291	Function.....	337
Channels and PPG Pins of PPG Timers	287	Program Example of Time-base Timer	193
Functions of 8-/16-bit PPG Timer	284	Program Example of Watch Timer.....	280
Generation of Interrupt Request from 8-/16-bit PPG Timer	294	Program Example of Watchdog Timer.....	208
Interrupts of 8-/16-bit PPG Timer	302	Program Execution	530
List of Registers and Initial Values of 8-/16-bit PPG Timer	294	PS	42
Operation Modes of 8-/16-bit PPG Timer	285	Processor Status (PS)	42
Operation of 8-/16-bit PPG Timer.....	303	PSCCR	99
Pins of 8-/16-bit PPG Timer.....	293	Configuration of the PLL/Sub clock Control Register (PSCCR)	99
PPGC Operation Mode Control Register	295	PUCR2	174
PPGC Operation Mode Control Register (PPGCC)	295	Block Diagram of Pull-up Control Register (PUCR2)	174
PPGC/PPGD Count Clock Select Register	299	Pull-up Control Register (PUCR2)	174
PPGC/PPGD Count Clock Select Register (PPGCD)	299	Pull-up Control Register	174
PPGCC	295	Block Diagram of Pull-up Control Register (PUCR2)	174
PPGC Operation Mode Control Register (PPGCC)	295	Pull-up Control Register (PUCR2)	174
PPGCD	299		
PPGC/PPGD Count Clock Select Register (PPGCD)	299	R	
PPGD Operation Mode Control Register (PPGCD)	297	RAM	30
PPGD Operation Mode Control Register	297	RAM Area	30
PPGD Operation Mode Control Register (PPGCD)	297	RDR	413
Prefix	48	Reception Data Register (RDR)	413
Bank Select Prefix	48	Read	567
Common Register Bank Prefix (CMR).....	49	Setting the Flash Memory to the Read/reset State	567
Flag Change Disable Prefix (NCC).....	49	Receive Overrun	505
Restrictions on Interrupt Disable Instructions and Prefix Instructions	51	Receive Overrun	505
Prefix Instructions	51	Received Message	504
Restrictions on Interrupt Disable Instructions and Prefix Instructions	51	Storing Received Message	504
Prescaler Settings	477	Reception	506
Prescaler Settings	477	Completing Reception	506
Presentation Item	644	Permission of Transmission and Reception	435
Description of Instruction Presentation Items and Symbols	644	Procedure for Reception by Message Buffer (x)	512
Processor Status	42	Processing for Reception of Data Frame and Remote Frame	505
Processor Status (PS)	42	Reception Flowchart of the CAN Controller	507
Program Counter	45	Reception Data Register	413
Program Counter (PC).....	45	Reception Data Register (RDR)	413
Program Example	536	Reception Interrupt	423
Program Example for Address Match Detection	536	Reception Interrupt Generation and Flag Set Timing	423
Function	536	Register	417
Program Example in Event Count Mode	267	Bit Configuration of Extended Communication Control Register (ECCR)	417
Program Example in Internal Clock Mode	266	Bit Configuration of Extended Status/Control Register (ESCR)	415
Program Example of 16-bit I/O Timer.....	234	Calibration Unit Control Register (CUCR)	598
Program Example of Delayed Interrupt Generation Module	88	Configuration of CR Oscillation Timer Data Register	602
Program Example of DTP Function.....	338		
Program Example of DTP/External Interrupt			

CR Oscillation Timer Data Register(16-bit) (CUTD)	600, 601	Status of Reset Source Bit and Low Voltage Detection Bit	130
CR Oscillation Trimming Setting Register (CRTR)	597	Reset State	
D/A Control Register 0 (DACR0)	609	Setting the Flash Memory to the Read/reset State	567
D/A Converter Register 0 (DAT0)	608	Restart	
Flash Memory Write Control Register (FWR0/FWR1)	552	Restarting Sector Erase of Flash Memory	574
Input Level Select Register (ILSR0, ILSR1)	176	ROM Mirroring	
Interrupt Control Register (ICR00 to ICR15)	59	Block Diagram of ROM Mirroring Function Select Module	540
Low-voltage/CPU Operation Detection Reset Setting Register (LVRS)	388	ROM Mirroring Function Select Module	
Setup Flow of Flash Memory Write Control Register (FWR0/FWR1)	554	Block Diagram of ROM Mirroring Function Select Module	540
Register Bank		List of Registers and Initial Values of ROM Mirroring Function Select Module	541
Register Bank	46	ROM Mirroring Function Select Register	
Register Bank Pointer		ROM Mirroring Function Select Register (ROMM)	542
Register Bank Pointer (RP)	43	ROMM	
Reload Counter		ROM Mirroring Function Select Register (ROMM)	542
Function of Reload Counter	432	ROM Security Function	
Reload Counters		Overview of ROM Security Function	590
Operation of Reload Counters	431	RP	
Reload Timer		Register Bank Pointer (RP)	43
Block Diagram of 16-bit Reload Timer	240	S	
Correspondence between 16-bit Reload Timer Interrupt and EI ² OS	251	Sample Program	
EI ² OS Function of 16-bit Reload Timer	251	Sample Program for Low Voltage/CPU Operating Detection Reset Circuit	393
Generation of Interrupt Request from 16-bit Reload Timer	244	Sample Program for the D/A Converter	610
Interrupts of 16-bit Reload Timer	251	Sampling Time	
List of 16-bit Reload Timer Registers and Initial Value	243	Sampling Time Setup (Bits ST2 to ST0)	362
Notes on Using 16-bit Reload Timer	262	SCR	
Operation Modes of 16-bit Reload Timer	238	Serial Control Register (SCR)	407
Pins of 16-bit Reload Timer	242	Sector Configuration	
Setting of 16-bit Reload Timer	252	Sector Configuration of the Flash Memory	547
Remote Frame		Sector Erase	
Processing for Reception of Data Frame and Remote Frame	505	Restarting Sector Erase of Flash Memory	574
Reset		Sector Erase Procedure for Flash Memory	571
Block Diagrams of the External Reset Pin	125	Suspending Sector Erase of Flash Memory	573
Factors of Reset	120	Sector Erase Timer Flag	
List of 16-bit Reload Timer Registers and Initial Value	243	Sector Erase Timer Flag (DQ3)	563
List of Registers and Initial Values of 8-/10-bit A/D Converter	348	Serial Clock	
Oscillation Stabilization Wait Reset State	124	Oscillating Clock Frequency and Serial Clock Input Frequency	580
Overview of Reset Operation	126	Serial Control Register	
Status of Pins during Reset	132	Serial Control Register (SCR)	407
Reset Source		Serial Programming Connection	
Correspondence between Reset Source Bits and Reset Sources	129	Basic Configuration of Serial Programming Connection	578
Notes about Reset Source Bits	131	Example of Serial Programming Connection (Power Supplied from Programmer)	583
Reset Source Bits	128	Example of Serial Programming Connection	
Reset Sources and Oscillation Stabilization Wait Times	123		

(User Power Supply Used)	581	Mode	373
Serial Status Register		Stop Conversion Mode (ADCS:MD1 and MD0=11 _B)	367
Serial Status Register (SSR)	411	Stop Conversion Mode Setup	373
Setting		Stop Mode	
Setting for 16-bit PPG Output Operation Mode	307	Stop Mode	152
Signal Mode		Storing Patch Program	
Signal Mode	435	Operation of Address Match Detection Function at Storing Patch Program in E ² PROM	534
Single Conversion Mode		Structure of Instruction Map661	
Operations and Applications of Single Conversion Mode 370		Sub clock	
Single Conversion Mode (ADCS: MD1 and MD0=00 _B or 01 _B)	367	Oscillation Stabilization Wait Time Timer of Sub clock	279
Single Conversion Mode Setup	369	Symbol	
Single-chip Mode		Description of Instruction Presentation Items and Symbols	644
Status of I/O Pins (Single-chip Mode)	156	Synchronization Methods	
Sleep Mode		Synchronization Methods	435
Return from Sleep Mode	146	Synchronous Mode	
Switching to Sleep Mode	145	Operation in Synchronous Mode (Operation Mode 2)	440
SMR		System Configuration	
LIN-UART Serial Mode Register (SMR)	409	System Configuration and E ² PROM Memory Map	531
Software Interrupt		System Stack Pointer	
Software Interrupt Operation	70	User Stack Pointer (USP) and System Stack Pointer (SSP)	41
Software Interrupts	55, 70	T	
Structure of Software Interrupts	70	TBTC	
Special Registers		Time-base Timer Control Register (TBTC)	185
Special Registers	37	TCCSH0	
SSP		Timer Control Status Register (Upper) (TCCSH0)	217
User Stack Pointer (USP) and System Stack Pointer (SSP)	41	TCCSL0	
SSR		Timer Control Status Register (Lower) (TCCSL0)	218
Serial Status Register (SSR)	411	TCDT0	
ST		Timer Data Register (TCDT0)	220
Sampling Time Setup (Bits ST2 to ST0)	362	TDR	
Standby Mode		Transmission Data Register (TDR)	414
Cancellation of Standby Mode by Interrupt	157	Time-base Timer	
Note on Canceling Standby Mode	157	Block Diagram of Time-base Timer	182
Notes on Accessing the Low-power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode	159	Correspondence Between Time-base Timer Interrupt and EI ² OS	187
Notes on the Transition to Standby Mode	157	Generation of Interrupt Request from Time-base Timer	184
Standby Mode	135	Interrupt of Time-base Timer	187
Transition to Standby Mode	157	List of Registers and Initial Values of Time-base Timer	184
Type of Standby Mode and Operation Status	143	Notes on Using Time-base Timer	192
State		Program Example of Time-base Timer	193
Setting the Flash Memory to the Read/reset State	567	Time-base Timer Control Register	
Status Bit			
Correspondence between Node Status Bits (NS1 and NS0) and Node Status	471		
Status Change			
Status Change Diagram	155		
Stop Conversion Mode			
Operations and Applications of Stop Conversion			

Time-base Timer Control Register (TBTC)	185	Block Diagram of LIN-UART Pins.....	405
Time-base Timer Mode		LIN-UART as LIN Master Device.....	453
Return from Time-base Timer Mode	151	LIN-UART Baud Rate Selection	427
Switching to the Time-base Timer Mode.....	150	LIN-UART Direct Pin Access	446
Timer Control Status Register		LIN-UART Functions	396
Timer Control Status Register (Upper) (TCCSH0)		LIN-UART Interrupts	420
.....	217	LIN-UART Interrupts and EI ² OS	422
Timer Control Status Register (Lower) (TCCSL0)		LIN-UART Pins	404
.....	218	LIN-UART Registers	406
Timer Control Status Registers (Lower) (TMCSR: L)		LIN-UART Serial Mode Register (SMR)	409
.....	247	Notes on Using LIN-UART	455
Timer Control Status Registers (Upper) (TMCSR:H)		Operation of LIN-UART	434
.....	245	Undefined Instruction	
Timer Data Register		Exception due to Execution of an Undefined	
Timer Data Register (TCDT0)	220	Instruction.....	80
Timer Register		Execution of an Undefined Instruction	80
Operating State of 16-bit Timer Register.....	253	Underflow	
Timing Limit Exceeded Flag		Operation as 16-bit Timer Register Underflows	
Timing Limit Exceeded Flag (DQ5)	562	255, 260
TMCSR		Operation at Underflow.....	239
Timer Control Status Registers (Lower)		User Power Supply	
(TMCSR: L).....	247	Example of Serial Programming Connection	
Timer Control Status Registers (Upper)		(User Power Supply Used).....	581
(TMCSR:H)	245	User Stack Pointer	
TMR		User Stack Pointer (USP) and System Stack Pointer	
16-bit Timer Registers (TMR).....	249	(SSP).....	41
TMRLR		USP	
16-bit Reload Registers (TMRLR)	250	User Stack Pointer (USP) and System Stack Pointer	
Toggle Bit Flag		(SSP).....	41
Toggle Bit Flag (DQ6)	561	W	
Toggle Bit-2 Flag		Watch Mode	
Toggle Bit-2 Flag (DQ2)	564	Return from Watch Mode	149
Transition		Switching to the Watch Mode	148
Clock Mode Transition.....	101	Watch Timer	
Notes on the Transition to Standby Mode	157	Block Diagram of Watch Timer	272
Transition to Standby Mode	157	Generation of Interrupt Request from Watch Timer	
Transmission		274
Canceling Transmission Request from		List of Registers and Initial Values of Watch Timer	
CAN Controller.....	502	274
Permission of Transmission and Reception	435	Program Example of Watch Timer	280
Procedure for Transmission by Message Buffer (x)		Watch Timer Counter	278
.....	510	Watch Timer Interrupt.....	277
Starting Transmission of CAN Controller	502	Watch Timer Interrupt and EI ² OS Transfer	
Transmission Flowchart of CAN Controller	503	Function	277
Transmission Data Register		Watch Timer Control Register	
Transmission Data Register (TDR)	414	Watch Timer Control Register (WTC).....	275
Transmission Interrupt		Watchdog Timer	
Transmission Interrupt Generation and Flag Set		Block Diagram of Watchdog Timer	198
Timing	425	Functions of Watchdog Timer.....	196
Transmission Interrupt Request Generation Timing		List of Registers and Initial Values of	
.....	426	Watchdog Timer	200
U		Notes on Using Watchdog Timer.....	206
UART		Operations of Watchdog Timer.....	204
Block Diagram of LIN-UART	400	Program Example of Watchdog Timer	208

Numerics

Setting Operation Clock of Watchdog Timer	279
Watchdog timer control register	
Watchdog Timer Control Register (WDTC)	201
WDTC	
Watchdog Timer Control Register (WDTC)	201
Writing	
Detailed Explanation of Writing to and Erasing Flash	
Memory.....	566
Writing to/Erasing Flash Memory	
Writing to/Erasing Flash Memory.....	544
WTC	
Watch Timer Control Register (WTC).....	275

Revision History



Major Changes

Revisions	
—	Changed the series name. MB95560H Series → MB95560H/570H/580H Series
	Added information on the MB95570H Series.
	Added information on the MB95580H Series.
CHAPTER 2 NOTES ON DEVICE HANDLING 2.1 Notes on Device Handling ■ Pin Connection • C pin	Corrected the following statement. The decoupling capacitor for the V_{CC} pin must have a capacitance larger than C_S . ® The decoupling capacitor for the V_{CC} pin must have a capacitance equal to or larger than the capacitance of C_S .
CHAPTER 7 RESET 7.1 Reset Operation ■ Reset Sources ● Low-voltage detection reset (optional)	Added the following statement. However, the LVD reset voltage selection ID register (LVDR) of the low-voltage detection reset circuit is not reset by the low-voltage detection reset.
CHAPTER 20 DUAL OPERATION FLASH MEMORY 20.4 Invoking Flash Memory Automatic Algorithm ■ Command Sequence Table Table 20.4-1	Revised the following address. $UAA8_H \rightarrow UAAA_H/UAA8_H$
APPENDIX APPENDIX A I/O Map ■ I/O Map Table A-2	Corrected the R/W attribute of the CMDR register. R/W → R
	Corrected the R/W attribute of the WDTL register. R/W → R
	Corrected the R/W attribute of the WDTL register. R/W → R
APPENDIX A I/O Map ■ I/O Map Table A-3	Corrected the R/W attribute of the CMDR register. R/W → R
	Corrected the R/W attribute of the WDTL register. R/W → R
	Corrected the R/W attribute of the WDTL register. R/W → R

Revisions	
CHAPTER 6 CLOCK CONTROLLER 6.1 Overview of Clock Controller ■ Block Diagram of Clock Controller Figure 6.1-1	Corrected the connection between the main CR PLL clock oscillator and the PLLC control register (PLLC).
CHAPTER 10 TIME-BASE TIMER 10.3.1 Time-base Timer Control Register (TBTC) ■ Time-base Timer Control Register (TBTC) Figure 10.3-2	Corrected the interval time (main CR clock multiplied by a PLL multiplication rate of 2, $F_{MCRPLL} = 8 \text{ MHz}$) for the setting TBC[3:0] = 1101. $2^{22} \times 1/F_{MCRPLL} (524.288 \text{ s})$ ® $2^{22} \times 1/F_{MCRPLL} (524.288 \text{ ms})$
CHAPTER 12 WATCH PRESCALER 12.3.1 Watch Prescaler Control Register (WPCR) ■ Watch Prescaler Control Register (WPCR) Figure 12.3-2	Corrected the R/W attribute of the WTC2 bit. R/X → R/W
CHAPTER 17 8/10-BIT A/D CONVERTER 17.2 Configuration of 8/10-bit A/D Converter	Corrected the register name of the ADDH and ADDL registers. A/D converter data registers (ADDH, ADDL) ® 8/10-bit A/D converter data registers (ADDH, ADDL)
	Corrected the register name of the ADC1 register. A/D converter control register 1 (ADC1) ® 8/10-bit A/D converter control register 1 (ADC1)
	Corrected the register name of the ADC2 register. A/D converter control register 2 (ADC2) ® 8/10-bit A/D converter control register 2 (ADC2)
17.2 Configuration of 8/10-bit A/D Converter ■ Block Diagram of 8/10-bit A/D converter Figure 17.2-1	Revised the register name of the ADDH and ADDL registers. A/D converter data registers (ADDH, ADDL) ® 8/10-bit A/D converter data registers (ADDH, ADDL)
	Revised the register name of the ADC1 register. A/D converter control register 1 (ADC1) ® 8/10-bit A/D converter control register 1 (ADC1)
	Revised the register name of the ADC2 register. A/D converter control register 2 (ADC2) ® 8/10-bit A/D converter control register 2 (ADC2)
17.2 Configuration of 8/10-bit A/D Converter ■ Block Diagram of 8/10-bit A/D converter	Renamed the section "● A/D converter data registers (ADDH/ADDL)" to "● 8/10-bit A/D converter data registers (ADDH, ADDL)".
	Renamed the section "● A/D converter control register 1 (ADC1)" to "● 8/10-bit A/D converter control register 1 (ADC1)".
	Renamed the section "● A/D converter control register 2 (ADC2)" to "● 8/10-bit A/D converter control register 2 (ADC2)".

Revisions	
17.6 Operations of 8/10-bit A/D Converter and Setting Procedure Example ■ Operations of 8/10-bit A/D Converter Conversion Function ● Continuous activation Figure 17.6-2	Added the settings of the ADDL register.
APPENDIX APPENDIX A I/O Map ■ I/O Map Table A-1	Corrected the R/W attribute of the CMDR register. R/W → R
	Corrected the R/W attribute of the WDTL register. R/W → R
	Corrected the R/W attribute of the WDTL register. R/W → R
APPENDIX D Pin States ■ State of Pins in Each Mode Table D-1	Deleted HCLK1 from P04/INT04/AN04/SIN/HCLK1/EC0.
	Deleted HCLK2 from P05/INT05/AN05/TO00/HCLK2.

Document History

Document Title: F2MC-16LX 16-Bit Microcontroller MB90990 Series Hardware Manual				
Document Number: 002-04564				
Revision	ECN#	Issue Date	Origin of Change	Description of Change
**	-	06/30/2009	AKIH	Initial release
*A	5405169	08/22/2016	AKIH	Migrated Spansion document to Cypress format.